INJECT3D: RE-ACCESSING DENSE GEOMETRIC FEATURES WITH COMPACT TOKENS FOR FINE-GRAINED 3D UNDERSTANDING

Anonymous authors
Paper under double-blind review

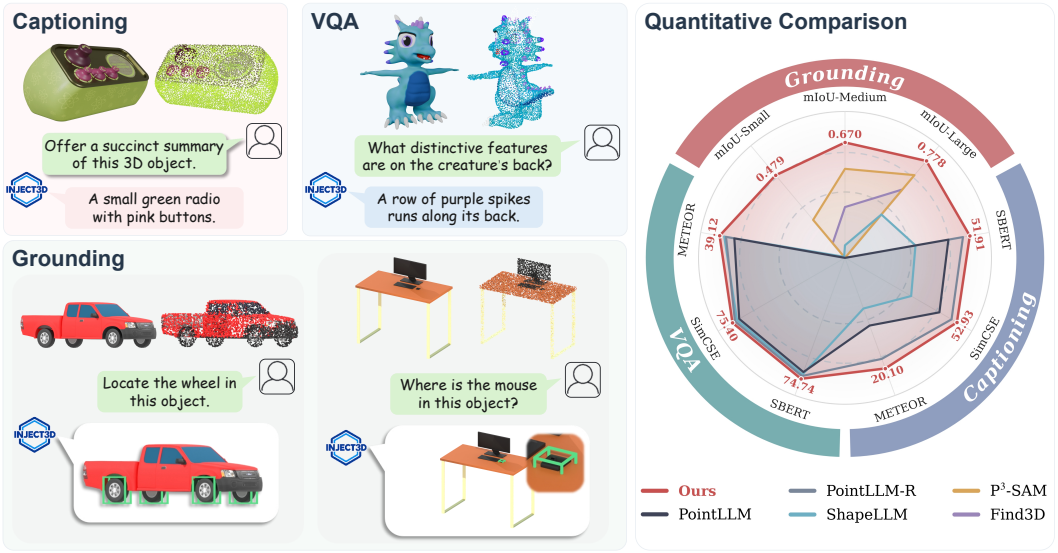

Figure 1: **Fine-grained 3D understanding.** Inject3D captures part-level details in 3D captioning, VQA, and grounding (left) and outperforms existing methods across these tasks (right).

ABSTRACT

Fine-grained 3D understanding requires access to local geometric evidence, yet existing 3D multimodal large language models typically compress dense point clouds into a small set of point tokens before language modeling. This compression can discard geometric details that subsequent LLM layers can no longer directly access, while increasing the point-token budget adds computational and memory overhead and complicates optimization. We introduce *Inject3D*, which feeds a compact set of spatially anchored point tokens into the LLM and retains high-resolution point-wise features as external geometric memory. At selected LLM layers, each anchor token retrieves features from its local 3D neighborhood and integrates them into its hidden state via cross-attention. To encourage effective use of the retrieved geometric evidence, we further introduce task-specific reinforcement post-training across captioning, visual question answering, and grounding. Experiments on object- and part-level tasks show that *Inject3D* effectively enhances fine-grained 3D understanding with a compact point-token sequence and substantially lower computational cost than increasing the input token budget.

1 INTRODUCTION

Fine-grained 3D understanding goes beyond recognizing an object as a whole to identifying its geometric details, distinguishing its semantic parts, and reasoning about their spatial relationships. Such capabilities are important for part-aware description (Xue et al., 2026; Wang et al., 2026a), 3D

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

grounding (Qi et al., 2024a; Kareem et al., 2024; Ahmed et al., 2025a), interactive editing (Wang et al., 2026a; Slim et al., 2026; Liao et al., 2026), and embodied interaction (Yu et al., 2026). Recent 3D multimodal large language models (MLLMs) have shown promising progress by connecting point-cloud representations with language models, enabling object captioning, visual question answering, and spatial reasoning (Guo et al., 2023; Qi et al., 2024a; Xu et al., 2024; Qi et al., 2024b). Extending these models from holistic recognition to fine-grained understanding, however, requires preserving substantially richer local geometric evidence.

A major obstacle is the *point-token bottleneck*: compressing a dense point cloud into a small set of input tokens can discard fine-grained geometry (Zhang et al., 2022; Chang et al., 2026). Increasing the token budget preserves more local detail but raises computation and memory costs throughout the LLM and can make learning from longer point-token sequences more difficult, making token scaling alone a costly and challenging route to geometric fidelity.

When geometry is provided only at the LLM input, later layers cannot directly retrieve evidence discarded during compression. This raises a question: *must all useful geometric evidence be carried by the input point tokens?* We observe that compact tokens can instead serve as spatial anchors to a retained dense representation, keeping local geometry accessible during language processing without lengthening the input sequence.

Based on this idea, we introduce *Inject3D*, which enables compact point tokens to *re-access dense 3D geometry* during language processing. Inject3D retains a high-resolution point-wise representation outside the LLM and projects only a compact subset of spatially anchored features into input tokens. At selected LLM layers, our *Anchor-Indexed Feature Re-Access (AFRA)* module retrieves each anchor's local neighborhood using its spatial location, aggregates complementary geometric evidence through lightweight cross-attention, and injects it into the anchor's hidden state. By maintaining the same neighborhood across injection layers, each anchor preserves a stable correspondence with the dense point cloud and provides repeated access to fine-grained local geometry without adding dense features to the LLM sequence.

Providing access to richer geometric evidence, however, does not by itself guarantee that an MLLM will exploit it effectively (Su et al., 2026; Chen et al., 2026). Fine-grained 3D understanding involves heterogeneous objectives: captioning emphasizes semantic completeness and fidelity, visual question answering emphasizes answer correctness, while grounding requires accurate geometric localization. Standard supervised fine-tuning optimizes these tasks through the same token-level likelihood objective, which only indirectly reflects their task-level goals. We therefore complement AFRA with task-specific reinforcement post-training (Shao et al., 2024). Dedicated rewards are constructed according to the target task, encouraging the model to use the available geometric evidence in ways that are directly aligned with semantic and geometric evaluation objectives. Experiments on 3D captioning, visual question answering, and grounding show that Inject3D improves fine-grained 3D understanding and computational efficiency while maintaining a compact point-token sequence.

Our main contributions are summarized as follows:

- We revisit the point-token bottleneck in 3D MLLMs from the perspective of *geometric accessibility*: fine-grained 3D evidence need not be fully encoded into the input token sequence as long as it remains accessible during language processing.
- We propose *Inject3D*, which retains a dense point-wise representation outside the LLM and enables compact spatial anchor tokens to re-access their corresponding local 3D evidence at selected LLM layers. Our AFRA module realizes this re-access through geometry-indexed neighborhood retrieval and cross-attention without increasing the point-token sequence length.
- We complement geometric re-access with task-specific reinforcement post-training and evaluate the resulting framework across 3D captioning, visual question answering, and grounding, demonstrating a favorable trade-off between fine-grained 3D understanding and computational efficiency.

2 Related Work

3D Understanding with MLLMs Recent advances in point-cloud MLLMs have enabled language-based understanding of 3D objects. Prior work has primarily focused on holistic object-level understanding (Xu et al., 2024; Qi et al., 2024a; Xue et al., 2023; Luo et al., 2023; Guo et al., 2023).

To improve the granularity and quality of 3D understanding, some studies focus on preserving geometric information and improving reasoning within the LLM. PointAlign (Su et al., 2026) aligns intermediate point representations to mitigate geometric information degradation, while PointLLM-R (Chen et al., 2026) introduces chain-of-thought reasoning to improve the quality of 3D reasoning.

Other studies develop specialized architectures and introduce part-level data for fine-grained understanding. PARIS3D (Kareem et al., 2024) and Kestrel (Ahmed et al., 2025a) integrate LLMs with segmentation heads for 3D part segmentation. 3D-PLOT-LLM (Xue et al., 2026) introduces point-referential tokens to support localized querying and description. Part-X-MLLM (Wang et al., 2026a) introduces native part-level instruction data and employs dual point-cloud encoders with higher-resolution inputs and longer point-token sequences, at the cost of substantially increased computation. Compared with these methods, our method injects complementary information from the dense geometric features into the LLM layers to achieve fine-grained understanding more efficiently and effectively than increasing the token count.

Efficient Perception in MLLMs Existing MLLMs preserve fine-grained visual information through high-resolution inputs, complementary visual encoders, and visual-token compression (Guo et al., 2024; Zheng et al., 2026; Wu et al., 2026; Li et al., 2025). Reconstruction and auxiliary feature-alignment objectives further reduce information loss during modality alignment (Wang et al., 2025; Huang et al., 2026; Li et al., 2026). Other methods extend visual conditioning beyond the input layer. DeepStack (Meng et al., 2024) injects groups of high-resolution visual tokens into successive early LLM layers, while Cambrian-1 (Tong et al., 2024) repeatedly queries features from multiple visual encoders through spatially localized cross-attention. Their cross-layer interactions operate on image features and are not structured by 3D coordinates or point-cloud representation.

Recent 3D MLLMs have explored preserving fine-grained geometric information during language modeling. PointAlign (Su et al., 2026) aligns point representations with geometric features, while ENEL (Tang et al., 2026) performs hierarchical geometry aggregation within LLM layers. HCC-3D (Zhang et al., 2026a), Efficient3D (Lin et al., 2026), and SAGE (Paul et al., 2026) improve geometric detail preservation and token efficiency through adaptive detail mining, token pruning, and neighborhood aggregation. In contrast, our Inject3D uses compact spatial anchor tokens to repeatedly access dense local geometric features across LLM layers without expanding the input sequence.

Reinforcement Post-Training for MLLMs Reinforcement post-training improves multimodal perception and reasoning through verifiable feedback. Visual-RFT (Liu et al., 2025) introduces GRPO-based visual reinforcement fine-tuning with task-specific rewards, while Perception-R1 (Yu et al., 2025) extends this paradigm to diverse perception tasks. For 3D understanding, SAGE (Paul et al., 2026) employs semantic-reward preference optimization for point-cloud reasoning. 3D-R1 (Huang et al., 2025) applies GRPO across multiple 3D scene-understanding tasks using a generic reward set. Our method instead adopts task-aware reward compositions tailored to the distinct outputs and evaluation criteria of captioning, question answering, and grounding, enabling effective reinforcement post-training across heterogeneous 3D understanding tasks.

3 METHOD

Inject3D provides LLMs with fine-grained geometric evidence while keeping the visual token sequence compact. Our key insight is that dense geometric details need not all enter the LLM as input tokens: compact spatial anchors provide the initial representation, while a dense geometric memory supplies local evidence at selected LLM layers. Figure 2 illustrates the framework. We first describe the construction of anchors and memory, then introduce Anchor-Indexed Feature Re-Access, followed by task-specific reinforcement post-training and the three-stage training recipe.

3.1 GEOMETRIC MEMORY AND COMPACT ANCHORS

Geometric Memory Construction. Given a point cloud $\mathbf{X} \in \mathbb{R}^{N\times 6}$ with xyz and rgb channels, we extract dense point-wise features $\mathbf{F} \in \mathbb{R}^{N\times d}$ using Utonia (Zhang et al., 2026b) and concatenate them with sinusoidal positional encodings $\mathrm{PE}(\mathbf{X}_{xyz})$ of the coordinates:

$$\mathbf{M} = \mathbf{F} \oplus \mathrm{PE}(\mathbf{X}_{xyz}) \in \mathbb{R}^{N\times(d+d_{\mathrm{PE}})}, \tag{1}$$

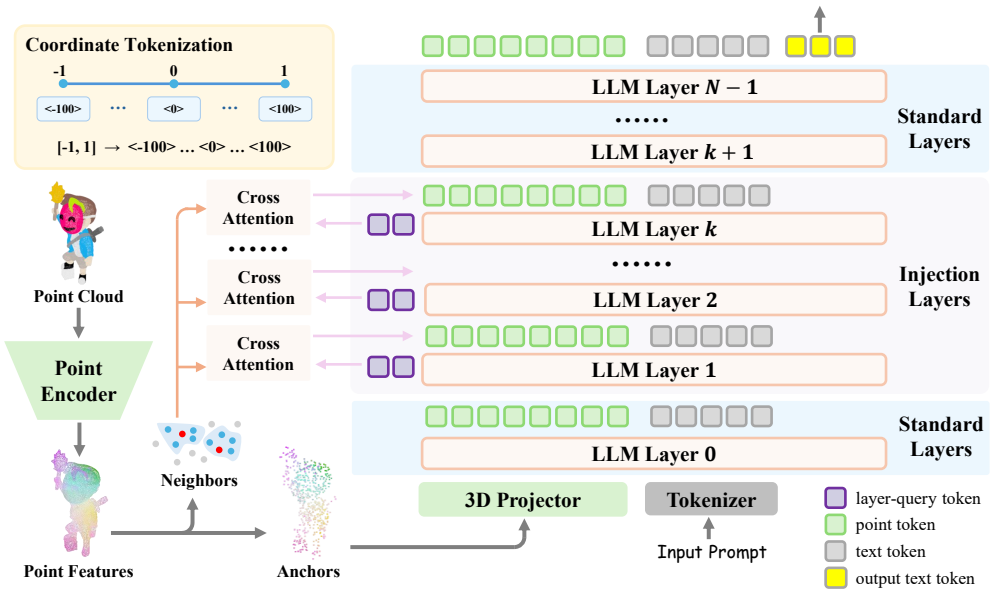

Figure 2: The *Inject3D* framework. Inject3D retains dense point-wise features as geometric memory and projects compact spatial anchors into the LLM input. At selected layers, Anchor-indexed feature re-access (AFRA) retrieves each anchor's fixed local neighborhood, aggregates geometric evidence through cross-attention with learnable queries, and updates the corresponding anchor hidden state.

where $\oplus$ denotes feature concatenation, and d and d_{PE} are the Utonia feature dimension and positional encoding dimension, respectively. The resulting geometric memory $\mathbf{M}$ retains a position-aware feature for every input point and is reused across the LLM layers.

Compact Anchor Tokenization. To keep the LLM input compact, we select $A \ll N$ anchor points using farthest point sampling (FPS). Since a point cloud is unordered whereas the LLM consumes a sequence, we arrange the anchors in Morton order, which tends to place spatially nearby anchors close together in the sequence. Their features are then projected into the LLM embedding space:

$$\mathbf{T} = \text{MLP}_{\text{proj}}\left(\text{SortMorton}(\mathbf{M}_{\text{anchor}})\right) \in \mathbb{R}^{A \times d_{\text{LLM}}}, \tag{2}$$

where $\mathbf{M}_{\text{anchor}}$ contains the memory features of the sampled anchors, and SortMorton orders these features by the Morton codes of their corresponding coordinates without changing their dimensions. The MLP maps each anchor feature from $d + d_{\text{PE}}$ dimensions to the LLM hidden dimension d_{LLM}. Each resulting token remains associated with its anchor point throughout the forward pass. Only the anchor tokens $\mathbf{T}$ enter the LLM sequence, and the memory $\mathbf{M}$ is retained for subsequent re-access.

3.2 Anchor-Indexed Feature Re-Access (AFRA)

The anchor tokens provide a compact representation, while the memory $\mathbf{M}$ retains geometric details. To make these details available within the LLM, the AFRA module supplements anchor hidden states at selected layers through fixed spatial neighborhoods and layer-specific learnable queries.

Anchor-Indexed Memory Association. For each anchor a, we find its k nearest input points using Euclidean distance in 3D coordinate space. We denote their point indices by $\mathcal{N}_a$, so that $\mathbf{M}[\mathcal{N}_a]$ gives the corresponding local memory features. The neighborhood is computed once per input and reused at every injection layer. Thus, each anchor accesses the same local region across layers, with both its neighborhood and memory features fixed within a forward pass.

Layer-Specific Evidence Aggregation. At injection layer ℓ, a bank of learnable queries $\mathbf{Q}^\ell$ aggregates features from each anchor's local memory through cross-attention:

$$\mathbf{Z}_a^\ell = \text{CrossAttn}^\ell\left(\mathbf{Q}^\ell, \mathbf{M}[\mathcal{N}_a]\right), \tag{3}$$

where $\mathbf{Q}^\ell$ provides the queries and $\mathbf{M}[\mathcal{N}_a]$ provides the keys and values. The query bank is shared across anchors within a layer and learned independently for each injection layer. It is independent

of the current anchor hidden state. Different layers can therefore learn different aggregations of the same local geometry. We consistently use the subscript a for anchors and the superscript ℓ for layers.

Anchor-Wise Residual Injection. We flatten the aggregated query features $\mathbf{Z}_a^\ell$, project them to the LLM hidden dimension through a layer-specific linear layer, and add the result to the corresponding anchor hidden state $\mathbf{h}_a^\ell$:

$$\mathbf{h}_a^\ell \leftarrow \mathbf{h}_a^\ell + \mathrm{Linear}^\ell\left(\mathrm{Flatten}\left(\mathbf{Z}_a^\ell\right)\right). \tag{4}$$

Here, $\mathbf{h}_a^\ell$ denotes the state at the anchor's sequence position immediately before injection. The query features remain internal to AFRA and are not appended to the LLM sequence. This provides repeated access to dense local geometry while keeping the number of visual tokens fixed at A.

3.3 Task-Specific Reinforcement Post-Training

Supervised fine-tuning (SFT) optimizes token-level likelihood and does not directly reflect answer-level objectives. While AFRA makes fine-grained geometric evidence accessible to the LLM, this token-level supervision may fail to fully exploit it (Qiu et al., 2025). This can be particularly limiting in our multi-task setting, where captioning, VQA, and grounding require different forms of answer-level correctness. To address this, we apply task-specific reinforcement post-training to encourage the model to better exploit retrieved geometric evidence through direct answer-level optimization. We therefore design task-specific rewards to reflect the answer-level objectives of different tasks.

Captioning and VQA Rewards. Captioning and VQA both require the model to produce responses that are consistent with the reference text in both content and semantics. We therefore use CIDEr (Vedantam et al., 2015) and CLIP-T (Radford et al., 2021) to assess consistency with the reference text, where the former measures content-level consistency and the latter captures semantic similarity. We adopt different aggregation strategies for these two tasks.

For captioning, multiple descriptions can validly express the same 3D content despite differences in wording, so the additive formulation provides greater tolerance to such variations:

$$R_{\mathrm{cap}} = \alpha_{\mathrm{cap}}\hat{R}_{\mathrm{CIDEr}} + (1-\alpha_{\mathrm{cap}})\hat{R}_{\mathrm{CLIP\text{-}T}}, \tag{5}$$

For VQA, both content consistency and semantic similarity are required, so we adopt the geometric mean to encourage joint satisfaction and penalize poor performance on either dimension:

$$R_{\mathrm{vqa}} = \sqrt{\hat{R}_{\mathrm{CIDEr}}\hat{R}_{\mathrm{CLIP\text{-}T}}}. \tag{6}$$

Part Grounding Rewards. For part grounding, the model predicts 3D bounding boxes given a part-name query. To reduce the number of box tokens, we represent normalized coordinates in $[-1, 1]$ using special tokens ranging from `<-100>` to `<100>`, as illustrated in the top-left of Figure 2. We design a grounding reward to jointly evaluates localization accuracy and geometric validity.

Given the ground-truth boxes $\mathcal{B}_{\mathrm{gt}}$ and predicted boxes $\mathcal{B}_{\mathrm{pred}}$, we establish one-to-one correspondences using greedy matching based on pairwise IoU, from which we compute IoU-based precision and recall and their F1 score: $F_{\mathrm{IoU}} = 2P_{\mathrm{IoU}}R_{\mathrm{IoU}}/(P_{\mathrm{IoU}}+R_{\mathrm{IoU}})$. This F1 score serves as a localization accuracy measure. To discourage invalid box predictions, we further introduce a format reward r_{fmt} for valid box predictions and an empty-box penalty r_{empty} for predicted boxes containing no input points. The final reward is

$$R_{\mathrm{ground}} = \max\left(s_{\mathrm{floor}}, \alpha F_{\mathrm{IoU}} + \beta r_{\mathrm{fmt}} - \lambda r_{\mathrm{empty}}\right), \tag{7}$$

where α, β, and λ are reward weights, and s_{floor} is the reward floor.

These task-specific rewards jointly guide the model to better exploit the fine-grained geometric information provided by AFRA across different understanding tasks.

3.4 Training

As illustrated in Figure 3, Inject3D follows a three-stage training pipeline:

Stage 1: Feature Alignment. This stage aims to establish a connection between the 3D representation and the language space. We freeze the point cloud encoder and the MLLM, and train only the 3D

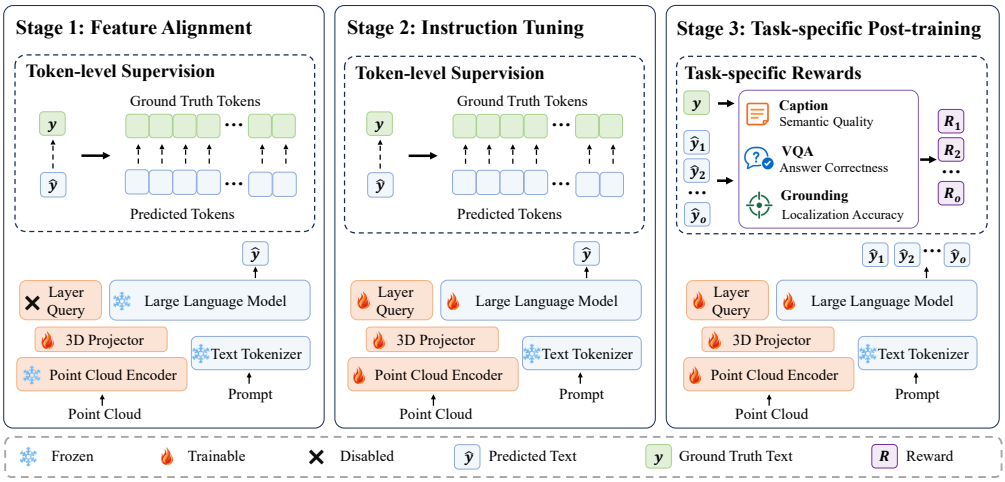

Figure 3: Three-stage training of Inject3D with module training status and supervision signals.

projector on a small subset of short-caption data. AFRA is disabled at this stage to avoid premature geometric injection and excessive information flow during the initial 3D-language alignment.

Stage 2: Instruction Tuning. We jointly train the model on object captioning, part captioning, VQA, and part grounding data. AFRA is initialized from scratch and all modules except the text tokenizer are optimized in this stage. This stage enables the model to learn how to use geometric evidence for fine-grained understanding under supervised instruction tuning.

Stage 3: Task-Specific Reinforcement Post-Training. Following instruction tuning, we apply GRPO (Shao et al., 2024) to a small subset of multi-task training data with task-specific rewards. By optimizing at the answer level rather than the token level, this stage helps the model better exploit fine-grained geometric evidence and meet task-specific requirements.

4 Experiments

4.1 Experiment Setup

Training Data. We curate our training data from public 3D datasets, including PointLLM (Xu et al., 2024), PartVerse (Ding et al., 2026), PartNeXt (Wang et al., 2026b), and 3DCoMPaT200 (Ahmed et al., 2025b). PointLLM contains 660K point cloud-caption pairs and 70K question-answer pairs with holistic object-level supervision, while the other three datasets provide fine-grained part-level supervision. Detailed data construction procedures are provided in the Appendix.

Implementation Details. Inject3D is initialized from Qwen3-VL-4B (Bai et al., 2025) with Utonia (Zhang et al., 2026b) as the point-cloud encoder. AFRA is injected into four consecutive LLM layers starting from the second. In each injection layer, 8 learnable query tokens attend to each anchor's 16 nearest neighbors to extract geometric features and add them to the corresponding anchor hidden states. See the Appendix for more implementation and training details.

4.2 Evaluation on Object Captioning and VQA

Object Captioning. We evaluate Inject3D on 3K test objects from PointLLM (Xu et al., 2024), using short reference captions. We compare against PointLLM (Xu et al., 2024), ShapeLLM (Qi et al., 2024a), PointAlign (Su et al., 2026), and PointLLM-R (Chen et al., 2026), using metrics including BLEU-1 (Papineni et al., 2002), ROUGE-L (Lin, 2004), METEOR (Banerjee & Lavie, 2005), SBERT (Reimers & Gurevych, 2019), and SimCSE (Gao et al., 2021). As shown in Table 1, Inject3D-SFT outperforms existing methods across all metrics, validating AFRA for precise point-cloud perception. GRPO further improves SFT by enabling better use of retrieved geometric evidence. Qualitative comparisons in Figure 4 show that Inject3D better captures object details and reduces incorrect or hallucinated descriptions.

Table 1: Quantitative comparison on object captioning and VQA.

Model	Object Captioning					VQA				
	BLEU-1	ROUGE-L	METEOR	SBERT	SIMCSE	BLEU-1	ROUGE-L	METEOR	SBERT	SIMCSE
PointLLM-7B (Xu et al., 2024)	5.44	8.50	12.30	43.01	44.66	33.81	39.72	34.52	70.57	70.87
ShapeLLM-7B (Qi et al., 2024a)	4.45	5.97	9.21	29.32	31.29	37.04	42.90	37.68	72.06	72.32
PointAlign (Su et al., 2026)	12.42	17.21	17.87	44.08	44.36	35.61	43.37	36.50	71.93	72.07
PointLLM-R (Chen et al., 2026)	15.15	19.58	18.36	49.07	50.40	35.32	40.54	37.12	73.02	73.73
Inject3D-SFT (Ours)	16.66	21.98	19.81	51.47	52.51	37.62	45.16	38.90	74.55	75.28
Inject3D-RL (Ours)	**17.47**	**23.22**	**20.10**	**51.91**	**52.93**	**37.89**	**45.20**	**39.12**	**74.74**	**75.40**

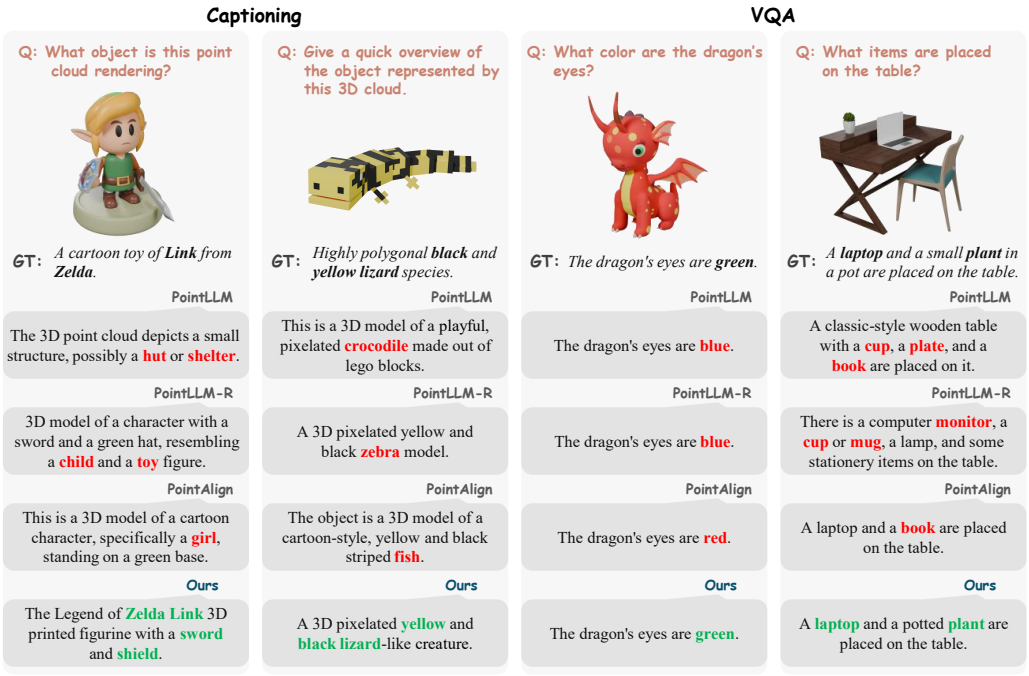

Figure 4: Qualitative comparison on object captioning and visual question answering.

Visual Question Answering. We construct a held-out VQA subset from PointLLM-v2 (Xu et al., 2025) using the same unseen object IDs, resulting in 1.7k QA pairs. We use the same metrics and baselines as in object captioning. As shown in Table 1, Inject3D-SFT outperforms existing methods, with further gains from GRPO. Qualitative examples in Figure 4 further show improved accuracy on detail-oriented questions.

4.3 Evaluation on Part Grounding

We evaluate Inject3D on the test splits of PartVerse, PartNeXt, and 3DCoMPaT200, and further assess cross-dataset and open-vocabulary generalization on PartObjaverse-Tiny (Yang et al., 2024) and HY3D-Bench (Hunyuan3D et al., 2026). We compare against LLM-based methods, including ShapeLLM (Qi et al., 2024a) and PARIS3D (Kareem et al., 2024), and promptable segmentation models, including Find3D (Ma et al., 2025b), CoSMo3D (Jin et al., 2026), and P^3-SAM (Ma et al., 2025a). For unified box-based evaluation, we convert the masks produced by segmentation-based methods to bounding boxes. We further stratify parts into small, medium, and large groups based on bounding-box size for scale-wise analysis. Details are provided in the Appendix.

As shown in Table 2, Inject3D consistently outperforms existing methods, especially on small parts, with gains from AFRA and task-specific post-training. Figure 5 illustrates that Inject3D achieves tighter boxes, better unseen-category generalization, and more reliable multi-box grounding, while baselines often miss target parts or produce noisy boxes.

4.4 Ablation Study

We conduct ablations to examine the contributions of key design choices. To reduce training cost, we train each variant on half of the full training data and evaluate it on the validation set.

Table 2: Comparison with existing methods on part grounding.

Method	PartVerse	HY3D-Bench	PartObj.Tiny	PartNeXt	3DCoMPaT200		Average			
					Coarse	Fine	Small	Medium	Large	Total
ShapeLLM (Qi et al., 2024a)	0.142	0.100	0.093	0.263	0.185	0.098	0.006	0.073	0.348	0.147
PARIS3D (Kareem et al., 2024)	0.327	0.295	0.258	0.316	0.320	0.188	0.067	0.255	0.468	0.283
Find3D (Ma et al., 2025b)	0.386	0.382	0.308	0.363	0.367	0.213	0.085	0.295	0.549	0.332
CoSMo3D (Jin et al., 2026)	0.311	0.291	0.232	0.357	0.448	0.238	0.079	0.276	0.537	0.316
P^3-SAM (Ma et al., 2025a)	0.580	0.587	0.506	0.515	0.523	0.352	0.220	0.517	0.664	0.502
Inject3D-SFT (Ours)	0.659	0.693	0.565	0.650	0.688	0.570	0.461	0.659	0.775	0.638
Inject3D-RL (Ours)	**0.680**	**0.699**	**0.572**	**0.653**	**0.703**	**0.586**	**0.479**	**0.670**	**0.778**	**0.649**

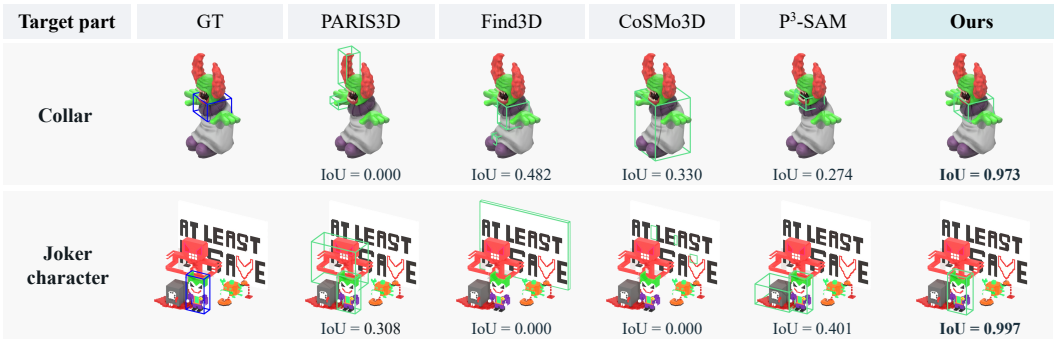

Figure 5: Qualitative comparison on part grounding.

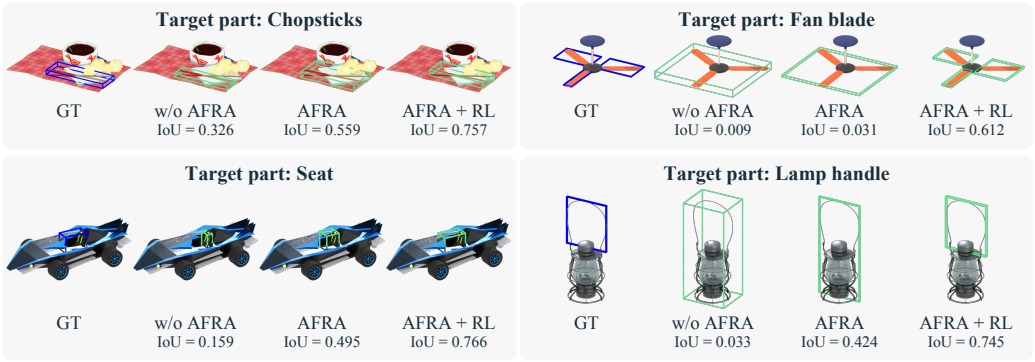

Figure 6: Qualitative analysis of the contributions of AFRA and reinforcement learning (RL) to part grounding.

Contribution of Anchor-Indexed Feature Re-Access (AFRA). Comparing IDs 0 and 3 in Table 3, adding AFRA improves overall grounding mIoU from 0.514 to 0.603, with gains across all part sizes. Captioning also improves, with only a marginal increase in prefill FLOPs. Figure 6 shows that AFRA effectively improves the localization of small and thin parts.

Comparison of Geometric Injection Strategies. To justify our design, we compare two alternatives that use anchor hidden states as cross-attention queries or inject query-aggregated geometric features at the LLM input. Details are provided in the Appendix. Comparing IDs 1, 2, and 3 in Table 3, AFRA achieves the highest grounding mIoU across all part sizes, while maintaining comparable captioning performance. These results support the effectiveness of our design for grounding, which requires fine-grained spatial understanding.

Token Budget and Computational Efficiency. At matched token budgets, AFRA consistently improves grounding mIoU with only a modest increase in prefill FLOPs (Table 4). With the same LLM backbone, point encoder, training data, and optimization recipe, AFRA with only 512 point tokens also outperforms the 4096-token baseline in grounding mIoU (0.603 vs. 0.597), while requiring 4.24T rather than 30.24T prefill FLOPs. Using one-eighth as many point tokens and 86% fewer prefill FLOPs, local geometric re-access offers a more efficient alternative to token scaling.

4.5 Discussion of AFRA Design

We examine how the placement and number of injection layers and the size of each anchor's local neighborhood affect part grounding. Figure 7 summarizes these design ablations.

Table 3: **Geometric injection strategies.** All variants use 512 point tokens without RL.

ID	Injection Strategy	Grounding (mIoU) ↑				Captioning		Efficiency		
		Small	Medium	Large	Total	SBERT ↑	SimCSE ↑	Prefill (T) ↓	TTFT (ms) ↓	Time/Batch (s) ↓
0	Base model (Ours w/o AFRA)	0.322	0.485	0.699	0.514	50.78	51.91	**4.07**	**136.86**	**5.96**
1	+ Hidden-state queries	0.364	0.565	0.766	0.572	**51.16**	**52.18**	4.28	157.47	6.29
2	+ Query features at input	0.406	0.569	0.768	0.589	51.01	52.05	4.24	156.82	6.33
3	+ AFRA	**0.412**	**0.593**	**0.785**	**0.603**	51.00	52.06	4.24	154.76	6.25

Table 4: **3D understanding performance and efficiency across token budgets.**

Configuration		Grounding (mIoU)				Captioning		VQA	Efficiency		
# Tokens	AFRA	Small ↑	Medium ↑	Large ↑	Total ↑	SBERT ↑	SimCSE ↑	SBERT ↑	Prefill (T) ↓	TTFT (ms) ↓	Time/Batch (s) ↓
256	–	0.241	0.470	0.710	0.483	50.67	51.84	73.88	**2.21**	**116.80**	**4.71**
256	✓	**0.343**	**0.563**	**0.762**	**0.568**	**50.77**	**51.85**	**74.03**	2.29	134.67	4.84
512	–	0.311	0.512	0.714	0.520	50.70	51.82	**74.27**	4.07	136.86	5.96
512	✓	**0.412**	**0.593**	**0.785**	**0.603**	**51.00**	**52.06**	74.24	4.24	154.76	6.25
1024	–	0.346	0.560	0.749	0.563	50.94	51.99	74.07	7.81	175.74	8.85
1024	✓	**0.435**	**0.610**	**0.760**	**0.612**	**50.96**	**52.08**	**74.40**	8.15	208.46	9.33
2048	–	0.415	0.572	0.759	0.589	50.98	52.05	74.25	15.29	270.06	15.00
2048	✓	**0.445**	**0.617**	**0.771**	**0.618**	**51.13**	**52.12**	**74.53**	15.96	310.61	15.66
4096	–	0.408	0.589	0.775	0.597	50.95	52.02	74.17	30.24	469.42	28.61

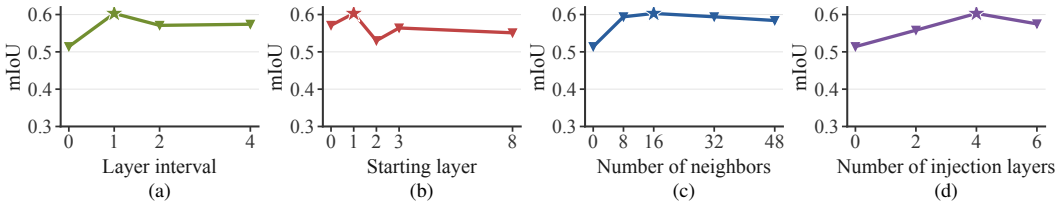

Figure 7: **AFRA design ablations.** Stars mark selected settings.

Injection Layer Placement. Figure 7 (a) shows that a layer interval of 1 performs best among the evaluated settings; increasing the interval to 2 or 4 yields lower mIoU. Figure 7 (b) varies the starting layer and achieves the highest mIoU at layer 1. Although the trend is not monotonic, later starting layers perform worse than this setting. Together, these results favor introducing geometric evidence early and using closely spaced injection layers.

Local Neighborhood Size. Figure 7 (c) varies the number of neighboring points retrieved for each anchor. Performance improves up to 16 neighbors, while using 32 or 48 neighbors brings no further gain. This supports a compact local neighborhood for evidence aggregation in the evaluated setting. A larger neighborhood may include points less relevant to the anchor's local geometry, potentially weakening the evidence used for grounding.

Number of Injection Layers. Figure 7 (d) shows that increasing the number of injection layers from 2 to 4 improves grounding, whereas using 6 layers slightly reduces mIoU. We therefore use four injection layers, providing repeated access to local geometry without adding further modules that do not improve performance in this ablation.

5 Conclusion

We presented Inject3D, a 3D multimodal LLM that addresses the point-token bottleneck by retaining high-resolution point-wise features as external geometric memory. A compact set of spatially anchored point tokens retrieves local geometric evidence through cross-attention at selected LLM layers, preserving access to fine-grained details without lengthening the input sequence. Task-specific reinforcement post-training further encourages effective use of this evidence across captioning, VQA, and grounding. Experiments on object- and part-level tasks demonstrate improved fine-grained 3D understanding at substantially lower computational cost than increasing the point-token budget. These results highlight geometric accessibility during language processing as an effective approach to balancing fine-grained understanding and computational efficiency.

Limitations and Future Work. Our study focuses on object-centric point clouds and their constituent parts; extending the framework to complete multi-object scenes remains unexplored. Future work will investigate how to organize geometric memory across multiple objects and support reasoning about their spatial relationships while retaining a compact point-token sequence.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with manuscript drafting, grammar and language polishing, LaTeX formatting, and code organization and formatting. Generative AI tools were also used to assist with synthetic data generation. All AI-assisted text and code were reviewed by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies fine-grained 3D understanding using existing public 3D datasets and derived annotations. The model may inherit biases from these datasets and its pretrained components, and may produce inaccurate descriptions, answers, or spatial localizations. Its outputs should therefore be independently validated before use in safety-critical applications, including embodied interaction.

REPRODUCIBILITY STATEMENT

The model architecture, geometric memory access mechanism, and training procedure are described in Section 3. The datasets, implementation details, and evaluation protocols are presented in Section 4 and Appendix.

REFERENCES

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*, pp. 8973–8983, 2025a.

Mahmoud Ahmed, Xiang Li, Arpit Prajapati, and Mohamed Elhoseiny. 3DCoMPaT200: Language-grounded compositional understanding of parts and materials of 3d shapes. *arXiv preprint arXiv:2501.06785*, 2025b.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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*, pp. 65–72, 2005.

Junlin Chang, Longhao Zou, and Rui Li. Point ladder tuning: Parameter-efficient hierarchical adaptation for 3d point cloud understanding. *arXiv preprint arXiv:2607.19171*, 2026.

Chaoqi Chen, Qile Xu, Wenjun Zhou, and Hui Huang. Pointllm-r: Enhancing 3d point cloud reasoning via chain-of-thought. *arXiv preprint arXiv:2605.22013*, 2026.

Lihe Ding, Shaocong Dong, Yaokun Li, Chenjian Gao, Xiao Chen, Rui Han, Yihao Kuang, Hong Zhang, Bo Huang, Zhanpeng Huang, et al. Fullpart: Generating each 3d part at full resolution. In *International Conference on Learning Representations*, volume 2026, pp. 45856–45876, 2026.

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*, pp. 6894–6910, 2021.

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.

Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *European Conference on Computer Vision*, pp. 390–406. Springer, 2024.

Ting Huang, Zeyu Zhang, and Hao Tang. 3d-r1: Enhancing reasoning in 3d vlms for unified scene understanding. *arXiv preprint arXiv:2507.23478*, 2025.

Xiaohu Huang, Jingjing Wu, Qunyi Xie, and Kai Han. 3drs: Mllms need 3d-aware representation supervision for scene understanding. *Advances in Neural Information Processing Systems*, 38: 67961–67988, 2026.

Team Hunyuan3D, Bowen Zhang, Chunchao Guo, Dongyuan Guo, Haolin Liu, Hongyu Yan, Huiwen Shi, Jiaao Yu, Jiachen Xu, Jingwei Huang, et al. Hy3d-bench: Generation of 3d assets. *arXiv preprint arXiv:2602.03907*, 2026.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25472–25483, 2026.

Li Jin, Weikai Chen, Yujie Wang, Yingda Yin, Zeyu Hu, Runze Zhang, Keyang Luo, Shengju Qian, Xin Wang, and Xueying Qin. Cosmo3d: Open-world promptable 3d semantic segmentation through llm-guided canonical spatial modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14325–14334, 2026.

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

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 132324–132345, 2026.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. TokenPacker: Efficient visual projector for multimodal LLM. *International Journal of Computer Vision*, 133(10):6794–6812, 2025.

Manwen Liao, Xinyu Lian, Jian Mao, Kaixu Chen, Li Luo, Jinghao Yan, Wanshui Gan, Qiao Yu, Weitian Zhang, Chunhua Shen, Guang Chen, Bo Dai, Xudong Xu, and Zhaoyang Lyu. Megaparts: Scaling part-aware 3d object generation to 300 parts via token-efficient autoregressive modeling. *arXiv preprint arXiv:2608.14783*, 2026.

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

Yuhui Lin, Siyue Yu, Yuxing Yang, Guangliang Cheng, and Jimin Xiao. Efficient3D: A unified framework for adaptive and debiased token reduction in 3D MLLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Findings*, pp. 8941–8951, 2026.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-RFT: Visual reinforcement fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2034–2044, 2025.

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.

Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025a.

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

Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms. *Advances in Neural Information Processing Systems*, 37:23464–23487, 2024.

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*, pp. 311–318, 2002.

Sneha Paul, Zachary Patterson, and Nizar Bouguila. Point cloud as a foreign language for multi-modal large language model. *arXiv preprint arXiv:2603.09173*, 2026.

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*, pp. 214–238. Springer, 2024a.

Zhangyang Qi, Ye Fang, Zeyi Sun, Xiaoyang Wu, Tong Wu, Jiaqi Wang, Dahua Lin, and Hengshuang Zhao. Gpt4point: A unified framework for point-language understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26417–26427, 2024b.

Haibo Qiu, Xiaohan Lan, Fanfan Liu, Xiaohu Sun, Delian Ruan, Peng Shi, and Lin Ma. Metis-rise: Rl incentivizes and sft enhances multimodal reasoning model learning. *arXiv preprint arXiv:2506.13056*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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)*, pp. 3982–3992, 2019.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Habib Slim, Shariq Farooq Bhat, Mohamed Elhoseiny, Yifan Wang, and Mike Roberts. Compose: Compositional synthesis and editing of 3d shapes via part-aware control. *arXiv preprint arXiv:2605.19350*, 2026.

Yuanhao Su, Shaofeng Zhang, Xiaosong Jia, and Qi Fan. Pointalign: Feature-level alignment regularization for 3d vision-language models. *arXiv preprint arXiv:2603.00412*, 2026.

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, pp. 150063–150084, 2026.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

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. In *International Conference on Learning Representations*, volume 2026, pp. 150386–150417, 2026a.

Haochen Wang, Anlin Zheng, Yucheng Zhao, Tiancai Wang, Zheng Ge, Xiangyu Zhang, and Zhaoxiang Zhang. Reconstructive visual instruction tuning. In *International Conference on Learning Representations*, volume 2025, pp. 14374–14399, 2025.

Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding. *Advances in Neural Information Processing Systems*, 38, 2026b.

Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. Spatial-mllm: Boosting mllm capabilities in visual-based spatial intelligence. *Advances in neural information processing systems*, 38: 13569–13597, 2026.

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*, pp. 131–147. Springer, 2024.

Runsen Xu, Shuai Yang, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm-v2: Empowering large language models to better understand point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 2025.

Jintang Xue, Xinyu Wang, Yixing Wu, Jingwen Chen, and C-C Jay Kuo. 3d-plot-llm: Part-level object tokens for 3d large language models. *arXiv preprint arXiv:2606.19828*, 2026.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1189. IEEE, 2023.

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

En Yu, Kangheng Lin, Liang Zhao, Jisheng Yin, Yana Wei, Yuang Peng, Haoran Wei, Jianjian Sun, Chunrui Han, Zheng Ge, Xiangyu Zhang, Daxin Jiang, Jingyu Wang, and Wenbing Tao. Perception-r1: Pioneering perception policy with reinforcement learning. *arXiv preprint arXiv:2504.07954*, 2025.

Xinqiang Yu, Zekun Qi, Jiawei He, Wenyao Zhang, Xuchuan Chen, Guaocai Yao, Li Yi, Zhaoxiang Zhang, and He Wang. Unipart: Towards zero-shot language-grounded 3d part segmentation for embodied interaction. *arXiv preprint arXiv:2609.12898*, 2026.

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*, pp. 19313–19322, 2022.

Liheng Zhang, Jin Wang, Hui Li, Bingfeng Zhang, and Weifeng Liu. Hcc-3d: Hierarchical compensatory compression for 98% 3d token reduction in vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12618–12626, 2026a.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022.

Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. Utonia: Toward one encoder for all point clouds. *arXiv preprint arXiv:2603.03283*, 2026b.

Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors. *Advances in neural information processing systems*, 38:20560–20586, 2026.

APPENDIX

A Data Construction and Evaluation Protocols **14**

A.1 Source Datasets 14

A.2 Part Label and Bounding Box Annotation 14

A.3 Part-Level Task Construction 16

A.4 Training and Test Splits 16

A.5 Part Grounding Evaluation Protocol 17

A.6 Computational Cost Measurement Protocol 18

B Model Implementation and Training **18**

B.1 Coordinate Tokenization 18

B.2 Architecture and Geometric Injection 19

B.3 Reward Hyperparameters in Reinforcement Post-Training. 19

B.4 Three-Stage Training Configuration 19

B.5 Alternative Geometric Injection Designs 20

C More Experimental Analyses **20**

C.1 Point Encoder Comparison (Utonia vs. Point-BERT) 20

C.2 Qualitative Effects of AFRA and Reinforcement Post-Training 21

C.3 Qualitative Comparisons Across Tasks 21

A DATA CONSTRUCTION AND EVALUATION PROTOCOLS

This section describes the construction of our training and evaluation data. We convert multiple public 3D object- and part-level datasets into a unified point-cloud question-answering format. All point clouds are represented as 8192×6 arrays with xyz coordinates and rgb values, where the xyz coordinates are normalized to $[-1, 1]$ under a Y-up coordinate system. The resulting data covers five task types: **object-level captioning, visual question answering, part naming, part grounding, and part captioning**.

A.1 SOURCE DATASETS

We use both object-level and part-level public datasets. For object-level data, we use PointLLM (Xu et al., 2024) curated brief object caption data, together with the PointLLM complex-instruction data. The caption data provides large-scale brief object descriptions, while the complex-instruction data provides open-ended instruction-following samples over 3D point clouds. For part-level data, we use PartVerse (Ding et al., 2026), PartNeXt (Wang et al., 2026b), 3DCoMPaT200 (Ahmed et al., 2025b), PartObjaverse-Tiny (Yang et al., 2024), and a small subset of HY3D-Bench (Hunyuan3D et al., 2026). The first three datasets are curated for both training and testing, whereas PartObjaverse-Tiny and HY3D-Bench are used for cross-dataset validation. Collectively, these sources form a unified corpus for fine-grained point-cloud understanding.

A.2 PART LABEL AND BOUNDING BOX ANNOTATION

Different datasets provide part semantics in different forms. We therefore first convert all original annotations into unified natural-language part labels.

PartVerse. PartVerse (Ding et al., 2026) is a large-scale part-annotated 3D dataset constructed from Objaverse, containing approximately 12K objects. It provides mesh face labels together with

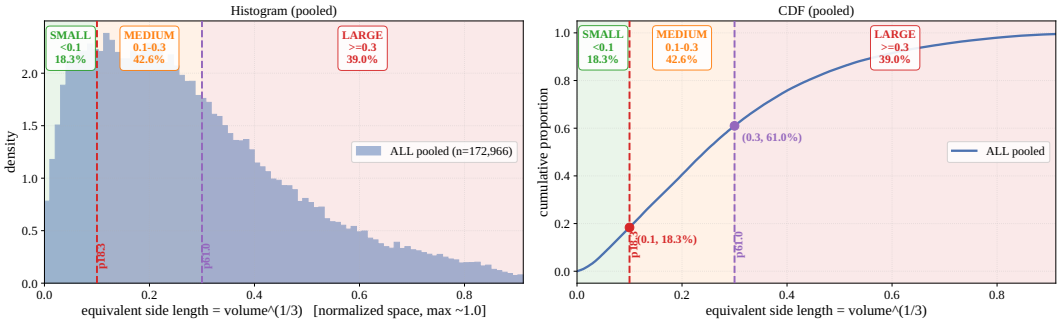

Figure 8: Distribution of part sizes in the constructed part-level triplets.

textual descriptions for object parts. We use Qwen3.5-27B (Bai et al., 2025) to convert the textual descriptions into concise part labels, and further merge semantically equivalent labels (e.g., "wheel" and "tire", or "arm" and "robot arm"), ensuring consistent naming of semantically related parts across objects.

PartNeXt. PartNeXt (Wang et al., 2026b) contains 23,519 high-quality textured 3D models from 50 object categories, with 350,187 annotated part instances organized in fine-grained hierarchical structures. It provides mesh face labels together with hierarchical segmentation annotations. We flatten the official hierarchy into readable part labels and remove redundant category prefixes when necessary. We also discard meaningless or overly generic labels (e.g., "other"), to ensure that each retained label corresponds to a semantically meaningful object part.

3DCoMPaT200. 3DCoMPaT200 (Ahmed et al., 2025b) consists of 19,051 3D shapes from 200 object categories. Each shape is annotated with hierarchical part labels spanning multiple levels. We manually consolidate these annotations by merging labels that refer to the same semantic part, while preserving meaningful distinctions between different parts. This yields a more consistent part-label space without losing meaningful semantic distinctions.

PartObjaverse-Tiny and HY3D-Bench. PartObjaverse-Tiny (Yang et al., 2024) and HY3D-Bench (Hunyuan3D et al., 2026) are used only for evaluation. PartObjaverse-Tiny directly provides face-to-part-name annotations, so no additional label conversion is required. In contrast, HY3D-Bench provides only face-level part segmentation without corresponding textual labels. We therefore render multi-view images of each object, where the segmented parts are converted into 3D bounding boxes and highlighted in the views, and use Qwen3.5-27B (Bai et al., 2025) to generate concise textual labels for the highlighted parts.

After part-level text normalization, we obtain a mapping from mesh faces to part names. Since point clouds are sampled from the meshes, this mapping further assigns a part label to each sampled point. We then derive box-level part-name annotations by computing the bounding box of each part point cluster. In addition, we consider mesh-face connectivity when clustering part points, so the same part label within an object may correspond to multiple boxes. In such cases, the boxes are deterministically ordered by their quantized spatial coordinates, proceeding from bottom to top, then from back to front, and finally from left to right, with the minimum corner used for tie-breaking.

Through these steps, we obtain unified relation triplets of object ID, part name, and bounding box across all part-level datasets. We further apply rule-based filtering based on bounding-box size and count to remove noisy triplets caused by face-level segmentation artifacts, including extremely small boxes and cases with an implausibly large number of boxes for a single part label. We then manually inspect the remaining annotations and remove erroneous cases, yielding 175K object-id–part-name–bounding-box triplets.

Part Scale Stratification. We further analyze the size distribution of the constructed part-level triplets based on the equivalent side length of each bounding box, defined as $\sqrt[3]{V}$, where V denotes the box volume in the normalized coordinate space. For a triplet associated with multiple boxes, we use the largest box for this analysis. As shown in Figure 8, we divide parts into small, medium, and large groups based on two size thresholds corresponding to approximately the 20th and 60th percentiles of the empirical size distribution, with equivalent edge lengths of 0.1 and 0.3, respectively.

Table 5: Prompt templates used for part-level data construction.

Task	User prompt template

Part grounding	`Give me the location of the {part} in this object.`
	`Identify the {part} in this object.`
	`Locate the {part} in this object.`
	`Find the {part} in this object.`
	`Where is the {part} in this object?`

Part naming	`What is in <bbox_aabb> in this object?`

Part caption	`Give a short description of the part in <bbox_aabb>.`
	`Provide a short description of the part inside <bbox_aabb>.`
	`Briefly describe the part indicated by <bbox_aabb> in this object.`

We report part grounding performance separately for these groups, since smaller parts generally require more fine-grained spatial perception and thus provide an important indicator of fine-grained part-level understanding.

A.3 Part-Level Task Construction

We construct task-specific question-answering pairs from the unified triplets for three part-level tasks: part grounding, part naming, and part captioning.

Part Grounding. The grounding task requires the model to localize a queried part in a point cloud. The input consists of a point cloud and a textual part query, and the output is one or multiple 3D bounding boxes. For each object-id–part-name–bounding-box triplet, we generate a corresponding part-grounding question-answer pair, following the prompt templates in Table 5. To improve robustness to spatial transformations, we apply right-angle rotation augmentation to the grounding training data, expanding the original part-grounding corpus by a factor of four.

Part Naming. The part naming task asks the model to recognize the semantic label of a given part box. We randomly sample 20% of the triplets to construct the part naming data, using the prompt template shown in Table 5. When a part label corresponds to multiple boxes within the same object, we randomly select one box as the query region. The part naming data are constructed only from the training-set sources.

Part Captioning. The part captioning task requires the model to generate a short natural-language description for a specified part box. To obtain part-level descriptions, we randomly sample 20% of the triplets and render multi-view images with the target 3D box highlighted. We then use Qwen3.5-27B to generate concise descriptions of the target parts from the rendered views, followed by an additional Qwen3.5-27B-based filtering step. The filtering focuses on inaccurate descriptions, captions inconsistent with the target part, overly vague descriptions, and descriptions biased by the color of the rendered 3D box. After filtering, we construct question-answering pairs using the prompt templates shown in Table 5. The part captioning data are constructed only from the training-set sources.

A.4 Training and Test Splits

We construct evaluation data to measure generalization to unseen objects. For each public part-level dataset, we construct approximately/up to 600 QA pairs per dataset whose object IDs are excluded from the training set and whose queried parts cover as many distinct part categories as possible, ensuring that evaluation is performed on unseen objects with diverse part semantics.

For object-level captioning, we use the 3,000 held-out objects from PointLLM (Xu et al., 2024) as the test set. For visual question answering, we construct a held-out VQA test subset from PointLLM-v2 (Xu et al., 2025) training split using the same unseen object IDs, resulting in 1.7K QA pairs. The final training and test statistics for each task are summarized in Table 6 and Table 7.

Table 6: Training set statistics by task and data source.

Task	Public dataset	Unique IDs	QA pairs	Ratio
Object captioning	PointLLM	658,575	658,575	42.0%
VQA	PointLLM	15,098	70,274	4.5%
Part naming	PartVerse	4,598	8,029	0.5%
	PartNeXt	13,348	19,739	1.3%
	3DCoMPaT200	12,186	26,620	1.7%
Subtotal	–	*30,038*	*54,388*	*3.5%*
Part grounding	PartVerse	4,598	32,116	2.1%
	PartNeXt	22,538	393,192	25.1%
	3DCoMPaT200	14,349	267,296	17.1%
Subtotal	–	*41,352*	*692,604*	*44.2%*
Part caption	PartVerse	7,061	11,977	0.8%
	PartNeXt	21,625	69,159	4.4%
	3DCoMPaT200	7,089	9,665	0.6%
Subtotal	–	*35,560*	*90,801*	*5.8%*
Total		**690,345**	**1,566,642**	**100.0%**

Table 7: Main test set statistics by task and data source.

Task	Public dataset	Unique IDs	QA pairs	Ratio
Object captioning	PointLLM	3,000	3,000	37.2%
VQA	PointLLM-v2	434	1,773	22.0%
Part grounding	PartVerse	276	628	7.8%
	PartNeXt	93	600	7.4%
	3DCoMPaT200-coarse	510	600	7.4%
	3DCoMPaT200-fine	325	600	7.4%
	PartObjaverse-Tiny	160	582	7.2%
	HY3D-Bench-part	247	287	3.6%
Total		**4,440**	**8,070**	**100.0%**

For ablation studies and AFRA design analysis, we use half of the constructed training data to train Inject3D and randomly sample 600 instances from the other half as the validation set.

A.5 Part Grounding Evaluation Protocol

We evaluate part grounding using mean 3D IoU (mIoU). For each query, our model predicts a set of one or more axis-aligned 3D bounding boxes, while the ground truth may also contain multiple boxes for the queried part. We first compute the pairwise 3D IoU between all predicted boxes and ground-truth boxes and apply Hungarian matching to obtain the one-to-one assignment that maximizes the total IoU. Matched pairs contribute their corresponding IoU scores, while unmatched predicted or ground-truth boxes are assigned an IoU of zero. Given N_p predicted boxes and N_g ground-truth boxes, the sample-level IoU is computed as

$$\mathrm{IoU}_{\mathrm{sample}} = \frac{\sum_{(i,j)\in\mathcal{M}} \mathrm{IoU}(\hat{B}_i, B_j)}{\max(N_p, N_g)}, \tag{8}$$

where $\hat{B}_i$ and B_j denote predicted and ground-truth boxes, respectively, and $\mathcal{M}$ denotes the Hungarian-matched box pairs. The final mIoU is obtained by averaging $\mathrm{IoU}_{\mathrm{sample}}$ over all evaluation samples. This protocol evaluates both localization accuracy and whether the model predicts the correct number of part boxes.

We apply the above protocol to all compared methods, adapting the evaluation procedure when necessary to accommodate their output formats. ShapeLLM (Qi et al., 2024a) supports box-formatted outputs, so we directly parse its predicted boxes for evaluation. However, its box generation is not always stable. When it fails to produce a valid target-part box, we treat the prediction as the whole object and use the object-level bounding box for mIoU computation.

For methods that do not directly output bounding boxes, we convert their predictions into boxes for unified evaluation. PARIS3D (Kareem et al., 2024), Find3D (Ma et al., 2025b), and CoSMo3D (Jin et al., 2026) produce point-level relevance scores or segmentation masks for the queried part. We threshold the predicted point scores to obtain a binary mask and take the tight axis-aligned bounding box of all selected points as the predicted box. When no point is selected, the prediction is treated as empty and receives zero IoU.

For multi-box ground truth, we define the target region as the union of all ground-truth boxes rather than their enclosing box, which may include large irrelevant empty spaces between disconnected part instances. For these methods, the pairwise IoU between a predicted box $\hat{B}_i$ and a ground-truth box B_j is approximated using the input point cloud. Specifically, let $\mathcal{P}_{\hat{B}_i}$ denote the set of input points inside $\hat{B}_i$, and let $\mathcal{P}_{B_j}$ denote the set of input points inside B_j. The pairwise IoU is computed as

$$\mathrm{IoU}(\hat{B}_i, B_j) = \frac{|\mathcal{P}_{\hat{B}_i} \cap \mathcal{P}_{B_j}|}{|\mathcal{P}_{\hat{B}_i} \cup \mathcal{P}_{B_j}|}. \tag{9}$$

For multi-box ground truth, $\mathcal{P}_{B_j}$ is defined over the union of all ground-truth boxes. These pairwise IoU values are then used in the same Hungarian matching and sample-level mIoU computation described above.

P^3-SAM (Ma et al., 2025a) requires a different evaluation setup, as it supports point prompts but not text prompts. Following its prompt-based formulation, we provide point prompts sampled from the ground-truth part region to obtain positive responses, and additionally sample negative prompts outside all ground-truth boxes. Specifically, we sample positive points inside each ground-truth box and negative points outside all ground-truth boxes, and use these point prompts to generate a binary mask. The resulting binary mask is converted into a bounding box in the same way as other mask-based baselines and evaluated using the same point-based mIoU protocol. We report P^3-SAM as a strong promptable segmentation reference.

A.6 Computational Cost Measurement Protocol

We use three metrics to measure computational cost:

Time to First Token (TTFT, ms). The latency from submitting an input prompt to receiving the first generated token, measured using wall-clock time during inference, including prompt processing (prefill) and first-token generation.

Prefill FLOPs. The number of floating-point operations required to process the input prompt and compute the initial KV cache before autoregressive decoding. It is estimated from the model's forward-pass operations during prefill, excluding token-by-token decoding.

Time per Batch (s). The wall-clock time required to complete one training batch, measured during model training, including forward and backward passes and parameter updates.

We use TTFT and prefill FLOPs to evaluate inference efficiency, with measurements conducted on a single NVIDIA H20 GPU. Time per batch measures training throughput and is evaluated on 8 NVIDIA H200 GPUs.

B Model Implementation and Training

B.1 Coordinate Tokenization

Part-level tasks require the model to represent 3D spatial regions, usually in the form of 3D bounding boxes. However, standard LLM tokenizers tend to split numbers, especially decimals, into multiple separate digit or subword tokens. This increases token count unnecessarily and makes it difficult for the model to learn and predict accurate spatial coordinates.

To address this, we use **Coordinate Tokenization** (Jiang et al., 2026). We evenly divide the normalized $[-1, 1]$ range on each axis into 201 discrete intervals, and map each interval to a unique special token. The resulting coordinate vocabulary runs from `<-100>` to `<100>`. This way, each single-axis coordinate is encoded as one compact token, reducing token overhead for spatial descriptions.

Each token also maps to a clear physical position, which helps the model learn spatial patterns and predict coordinates for part-level tasks.

B.2 Architecture and Geometric Injection

Inject3D is initialized from Qwen3-VL-4B (Bai et al., 2025) with Utonia (Zhang et al., 2026b) as the point-cloud encoder. The 3D projector follows the same architectural design as PointLLM (Xu et al., 2024).

The input point cloud is first encoded into point-wise features of dimension 1386, then concatenated with 60-dimensional sinusoidal positional encoding, resulting in a final feature dimension of 1446.

AFRA is injected across four consecutive LLM layers, from the second to the fifth. In each injection layer, for each anchor point token, 8 layer-specific learnable query tokens of dimension 512 extract geometric evidence from the 16 nearest point features of the anchor via cross-attention. The extracted features are flattened and projected to the hidden state dimension of 2560, then added to the corresponding anchor hidden states.

B.3 Reward Hyperparameters in Reinforcement Post-Training.

For captioning, we set $\alpha_{\text{cap}} = 0.5$, assigning equal weights to the CIDEr and CLIP-T terms. For grounding, we set $\alpha = 0.9$ and $\beta = 0.1$ for the localization and format terms, respectively. The empty-box penalty is weighted by $\lambda = 1.0$ and is applied when a predicted box contains no input points. The grounding reward is lower-bounded by $s_{\text{floor}} = -1.0$.

B.4 Three-Stage Training Configuration

We introduce the three-stage training pipeline in Section 3.4 and illustrate the overall procedure in Figure 3. Here, we provide implementation details for each training stage.

Stage 1: Feature Alignment. We sample 40% of the object-captioning data, resulting in approximately 260K short-caption samples for training. During this stage, the point-cloud encoder and the MLLM are frozen, and only the 3D projector is optimized. AFRA is disabled to avoid premature geometric injection and excessive information flow before the basic 3D-language alignment is established.

Stage 2: Instruction Tuning. We use all training data except for part naming, for which we randomly sample 60% of the available data. AFRA is initialized from scratch in this stage. All model components are optimized while the text tokenizer remains fixed.

Stage 3: Task-Specific Reinforcement Post-Training. The reinforcement data contains object captioning, VQA, part captioning, and part grounding samples. At each optimization step, we sample 16 unique prompts with a task mixture of 10 grounding prompts, 3 VQA prompts, 2 object-captioning prompts, and 1 part-captioning prompt, assigning a larger proportion to grounding to emphasize fine-grained localization. For each prompt, we sample 8 responses, resulting in a batch size of 128 generated responses per step and 768K generated responses over 6K training steps.

More detailed training configurations are provided in Table 8.

Table 8: Task composition and training hyperparameters for the three-stage training procedure.

Stage	Training tasks / data	Data size	LR	Batch	Steps	Warmup / Weight decay
Feature alignment	object captioning	260K	1×10^{-4}	512	1K	3% / 0.01
Instruction tuning	object captioning, VQA, part grounding, part captioning, part naming	1.5M	2×10^{-5}	512	6K	3% / 0.01
GRPO post-training	object captioning, VQA, part grounding, part captioning	768K	1×10^{-6}	128	6K	5% / 0

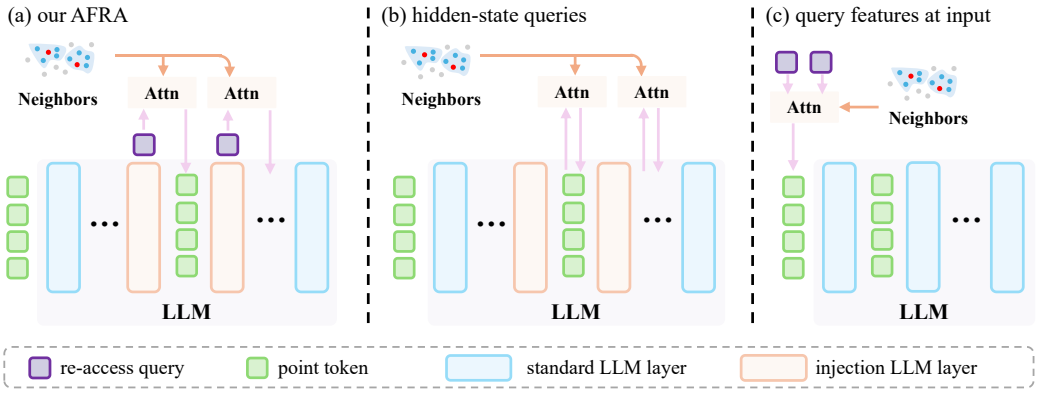

Figure 9: Architectural comparison of AFRA and two alternative designs. (a) Our AFRA uses layer-specific learnable queries and injects the aggregated geometric features into selected LLM layers. (b) The hidden-state query variant re-accesses local geometric memory using anchor hidden states as queries. (c) The input injection variant aggregates geometric features only once before the LLM, without intermediate-layer re-access.

B.5 Alternative Geometric Injection Designs

To better understand the design choices of AFRA, we compare it with two alternative injection designs that vary the query representation and the stage of geometric information injection. Figure 9 illustrates the architectures of different variants compared in Table 3 of the main paper.

The hidden-state query variant uses the current anchor hidden states as cross-attention queries instead of the learnable queries in AFRA. In implementation, each anchor hidden state is projected through a linear layer and split into 8 query tokens, which attend to neighboring geometric features and are then added back to the corresponding hidden state.

The input injection variant supplies query-aggregated geometric features at the LLM input instead of injecting them at intermediate layers. In implementation, a single set of learnable queries aggregates local neighbor geometric features once before the LLM backbone, and the aggregated features are merged with point tokens to form the input sequence, with no further geometric re-access in intermediate layers.

Our design uses layer-specific learnable queries to aggregate each anchor's local geometric memory and adds the projected features to the corresponding anchor hidden state at selected LLM layers. As demonstrated in Table 3, this design provides more effective geometric information injection than the alternative designs.

C More Experimental Analyses

Table 9: Effect of different point encoders.

Configuration	Grounding (mIoU)				Captioning		VQA	Efficiency		
Point Encoder	Small ↑	Medium ↑	Large ↑	Total ↑	SBERT ↑	SimCSE ↑	SBERT ↑	Prefill (T) ↓	TTFT (ms) ↓	Time/Batch (s) ↓
Point-BERT	0.263	0.462	0.700	0.482	50.55	51.54	73.79	4.14	**98.22**	**4.69**
Utonia	**0.311**	**0.512**	**0.714**	**0.520**	**50.70**	**51.82**	**74.27**	**4.07**	136.86	5.96

C.1 Point Encoder Comparison (Utonia vs. Point-BERT)

We compare our adopted point encoder Utonia (Zhang et al., 2026b) with Point-BERT (Yu et al., 2022), which adopts patch-level representations, in contrast to Utonia's per-point features. This comparison examines how different point representation strategies affect downstream 3D understanding. Given an input point cloud, Point-BERT produces 512 patch tokens, whereas Utonia extracts per-point features through an encoder-decoder architecture, from which 512 tokens are sampled via farthest point sampling (FPS). We replace Utonia with Point-BERT under the same training setup and evaluate both encoders across tasks and in terms of training and inference efficiency.

As shown in Table 9, Utonia achieves better overall performance than Point-BERT, particularly on part grounding. We attribute this advantage to two factors: Utonia is pretrained on more extensive data, providing stronger priors, and its per-point features preserve more fine-grained geometric information, whereas Point-BERT-style architectures tend to discard such details during feature downsampling.

In terms of computational efficiency, Point-BERT achieves lower time per batch and TTFT than Utonia, owing to its Transformer encoder architecture, despite requiring more computation during prefill, as indicated by its higher prefill FLOPs.

C.2 Qualitative Effects of AFRA and Reinforcement Post-Training

Figure 10 provides additional qualitative evidence for the contributions of AFRA and reinforcement learning. Without AFRA, the model often fails to reliably focus on the queried part region. Introducing AFRA improves part-aware localization by injecting fine-grained geometric evidence, while the reinforcement post-training stage further refines the predictions, leading to more compact, cleaner, and better-formatted bounding boxes.

C.3 Qualitative Comparisons Across Tasks

Figures 11, 12, and 13 present additional qualitative comparisons between Inject3D and existing methods. Overall, Inject3D demonstrates stronger fine-grained 3D understanding. It captures detailed object attributes in captioning, accurately localizes relevant regions for detail-oriented visual questions and produces correct answers, and generates compact bounding boxes for part grounding. Moreover, Inject3D also shows strong localization ability in multi-box grounding scenarios.

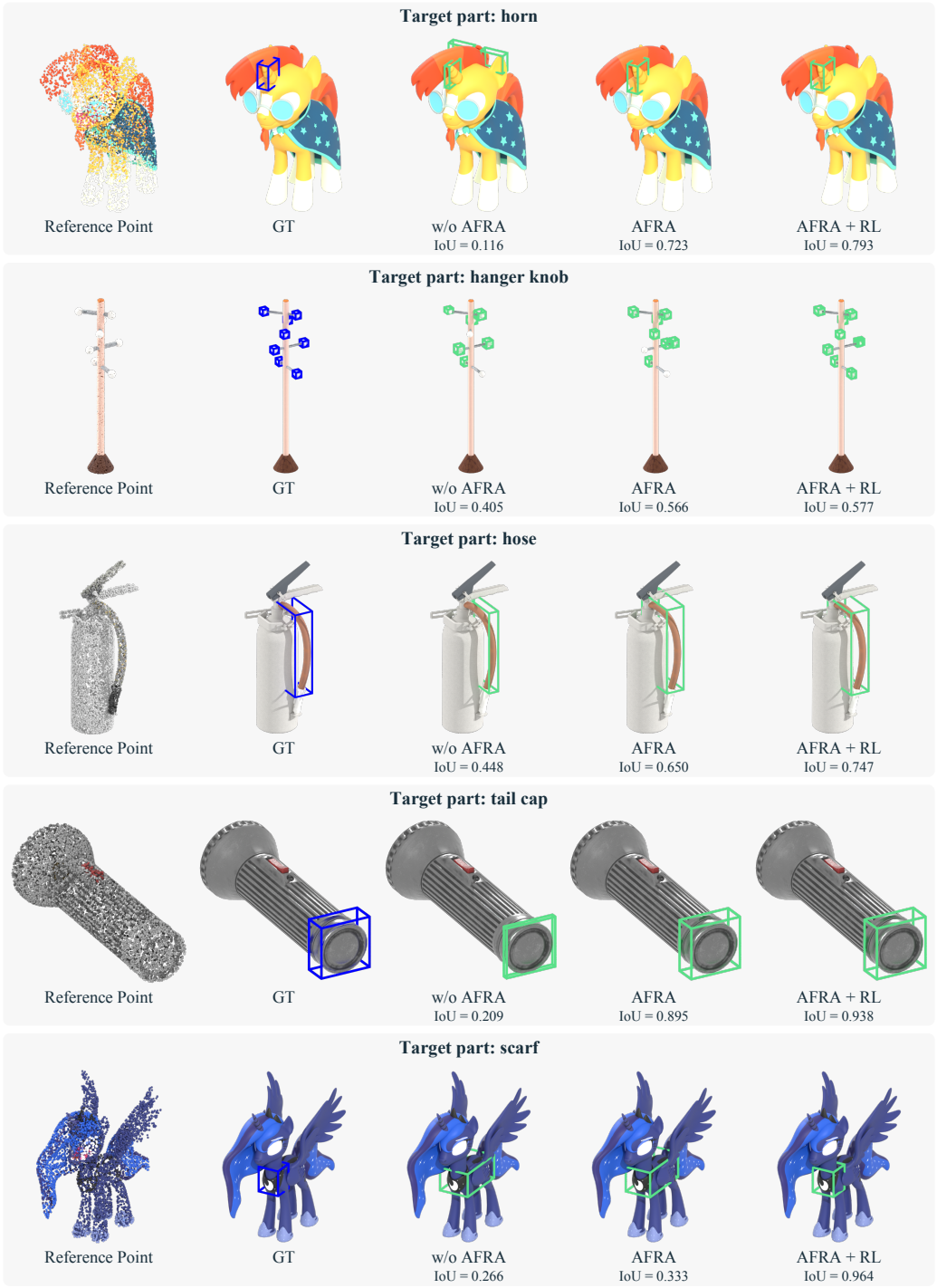

Figure 10: Additional qualitative analysis of the contributions of AFRA and reinforcement learning to part grounding. AFRA helps the model localize the queried part regions, while reinforcement learning further encourages the model to generate compact and well-formatted boxes.

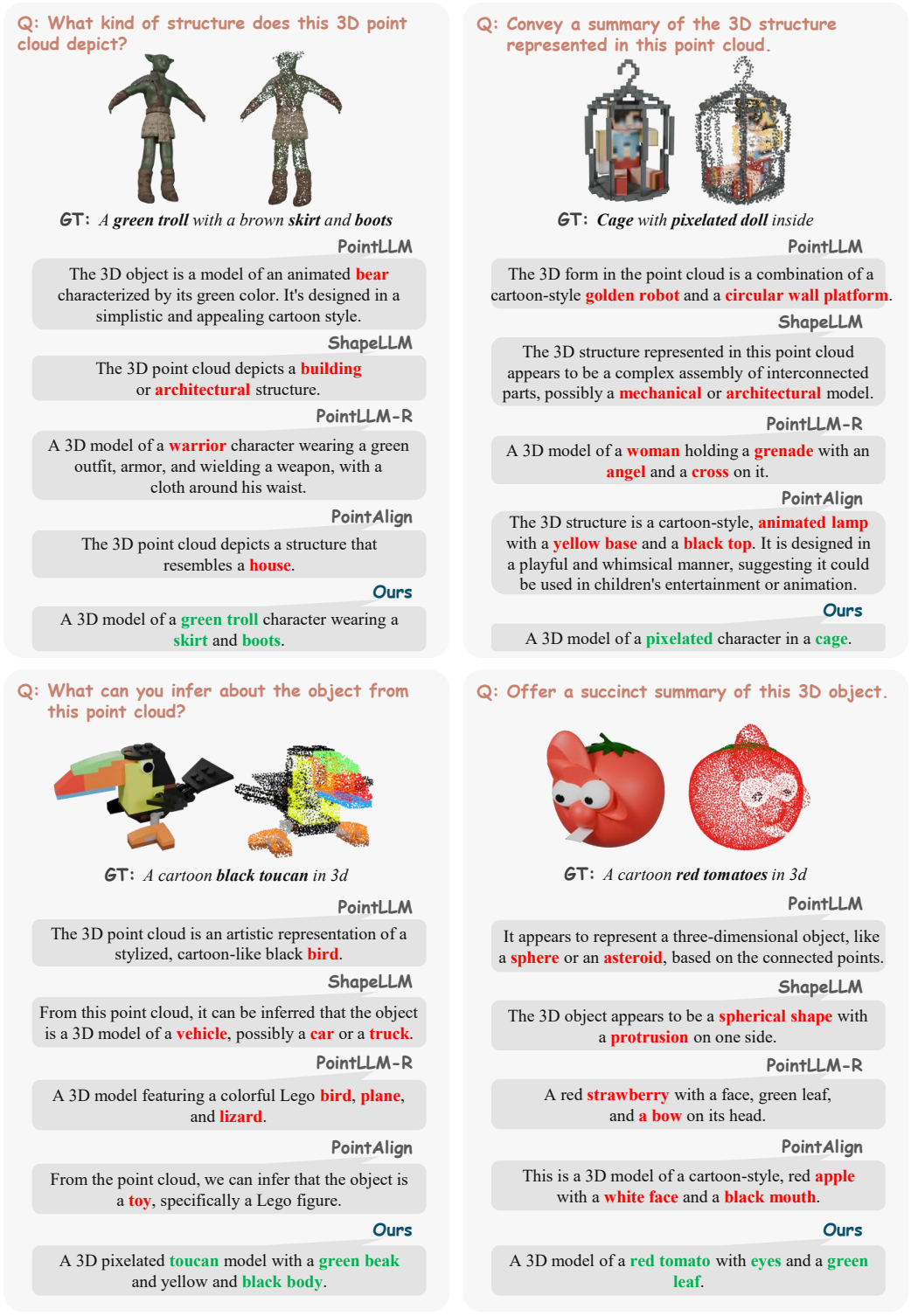

Figure 11: Additional qualitative comparison on object *captioning*. Inject3D provides more detailed and accurate descriptions compared with baselines.

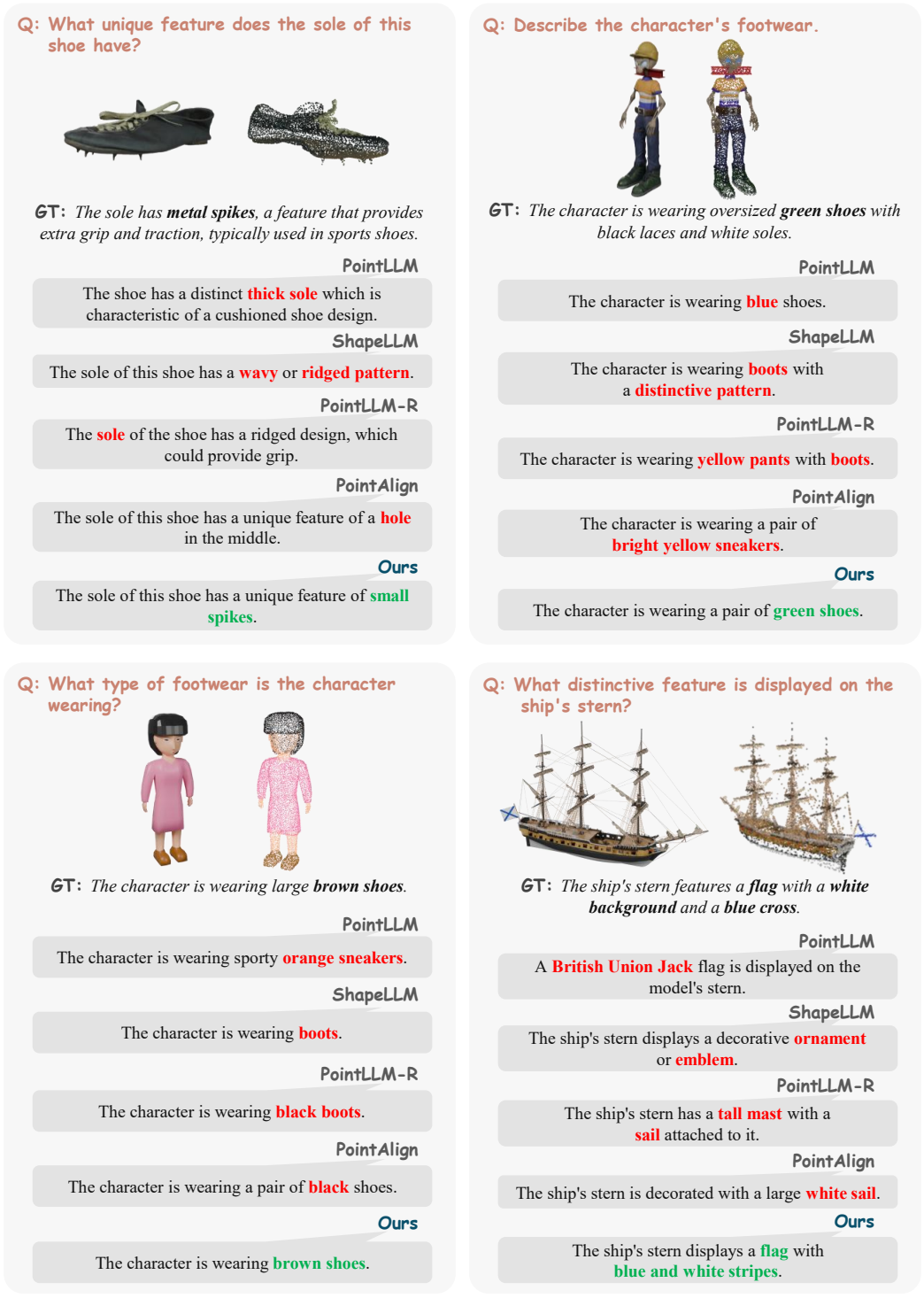

Figure 12: Additional qualitative comparison on object *visual question answering*. For detail-oriented questions, Inject3D can accurately localize the queried fine-grained details and generate precise answers.

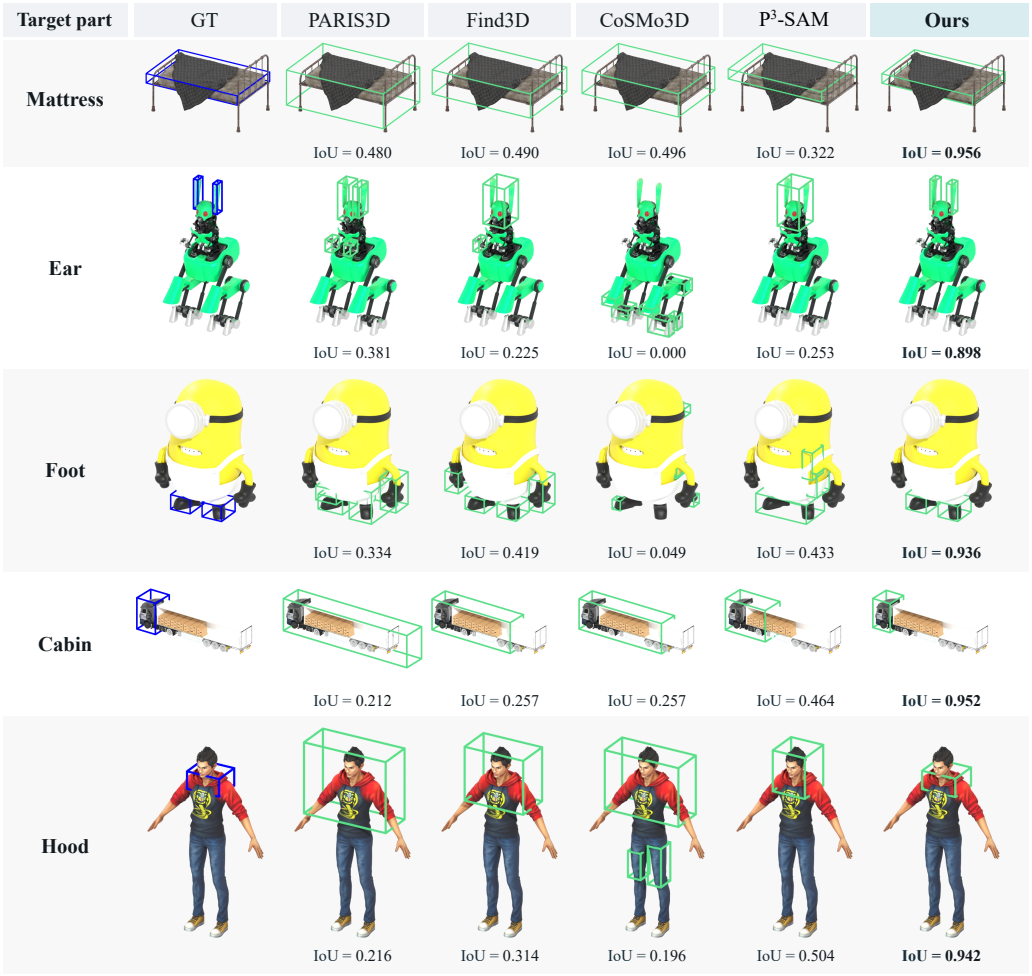

Figure 13: Additional qualitative comparison on *part grounding*. Compared with existing methods, Inject3D localizes target parts more accurately, generates compact bounding boxes, and rarely produces spurious boxes.