UNLEASHING THE VALUE OF RAW POINT CLOUDS FOR 3D UNDERSTANDING WITH LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard 3D point cloud language models typically rely on pretrained point encoders to extract 3D features and project them into representations processable by large language models (LLMs). However, these encoders are often originally optimized for pre-defined tasks, resulting in a *representation bottleneck* that may lead to a substantial mismatch with downstream 3D understanding. Moreover, existing models largely follow a *single pass closed-book* generation, limiting their ability to actively inspect task-relevant 3D regions or retrieve necessary external knowledge. In this work, we present POINTTOOL, an encoder-free and tool-augmented framework for 3D reasoning. POINTTOOL introduces Alignment-Free Patch Tokenization (AFPT) to locally aggregate raw point clouds while preserving fine-grained geometric structure, and Decoupled Point-Text LoRA (DPL) to facilitate global aggregation and 3D knowledge internalization within a frozen LLM. Building upon these designs, Program-Supervised Tool Reasoning (PSTR) enables the model to explicitly operate on raw 3D data and external resources through executable tools, supporting interactive 3D reasoning grounded in spatial evidence. We further introduce PointCloud-TR, a benchmark comprising 13K 3D objects, 39.8K questions, and 35.6K tool trajectories to facilitate training and evaluation of complex 3D reasoning. Experiments demonstrate that POINTTOOL achieves strong performance on both standard and complex 3D understanding tasks, validating the effectiveness of the proposed framework.



Figure 1: POINTTOOL achieves fine-grained reasoning in an encoder-free manner and leverages tool calls to access external knowledge and task-relevant evidence from raw data for 3D understanding.

1 INTRODUCTION

3D visual understanding extracts semantic, geometric, and spatial information from three-dimensional data for tasks such as object classification and detection (Qi et al., 2017a; Zhou & Tuzel, 2018), language-based object localisation and dense captioning (Chen et al., 2020; 2021), and 3D question answering (Azuma et al., 2022). Multimodal LLMs combine visual perception with linguistic knowledge (Wu et al., 2023); recent 3D LMMs extend this paradigm to point clouds and reconstructed scenes (Xu et al., 2024; Hong et al., 2023; Chen et al., 2024). Within this line

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.

of work, point clouds are a common native 3D input (Xu et al., 2024; Qi et al., 2024a; Tang et al., 2024), encoding geometry and, when available, colour.

Existing mainstream 3D LMM paradigms predominantly employ pretrained 3D point encoders to extract point cloud features (Xu et al., 2024; Qi et al., 2024a; Tang et al., 2024). Representative encoders are often pretrained with self-supervised objectives on point clouds (Yu et al., 2022; Pang et al., 2022). Because these objectives do not directly use language supervision, their outputs are not explicitly optimized for language-conditioned downstream tasks. ULIP and ULIP-2 align point-cloud, image, and language representations during pre-training (Xue et al., 2023; 2024), while Point-Bind aligns point clouds with images, language, audio, and video in a shared embedding space (Guo et al., 2023). These approaches are effective, but a pretrained encoder limits the information exposed to the LLM and can therefore form a **representation bottleneck** when a downstream question depends on signals not emphasized by pre-training.

Recent encoder-free 3D LMMs learn point representations more directly with the LLM, but their tokenizers still introduce substantial structural or representational constraints. ENEL (Tang et al., 2026) relies on architecture-specific hierarchical aggregation and propagation, which repeatedly compress and reconstruct point features and may obscure fine-grained local information. SAGE (Paul et al., 2026) further discretizes continuous point features through a finite codebook, introducing a quantization bottleneck that may collapse subtle geometric and appearance variations. Moreover, both methods remain closed-book, single-pass generators: they cannot invoke tools to dynamically inspect the underlying point cloud or access external knowledge beyond the LLM's parametric memory. In Fig. 1, baseline is prone to errors in fine-grained spatial reasoning (a) and fails to answer knowledge-intensive questions that require information beyond the point cloud (b).

Motivated by these limitations, we present **POINTTOOL**, an encoder-free framework that exposes fine-grained point-cloud information directly to the LLM. POINTTOOL introduces Alignment-Free Patch Tokenization (AFPT), which groups raw points into local patches and represents each patch as a continuous, position-aware token. By performing only local aggregation, AFPT preserves fine-grained geometric and color information without architecture-specific hierarchical compression or discrete codebook quantization. We further introduce Decoupled Point-Text LoRA (DPL), which provides separate low-rank adaptation paths for point and text tokens while allowing the LLM to model global interactions across patches. Together, AFPT and DPL establish a direct and information-preserving point-language interface, enabling task-dependent 3D modeling without relying on a pretrained point encoder or a restrictive tokenization pipeline.

To further exploit raw point-cloud information and extend reasoning beyond the LLM's parametric knowledge, POINTTOOL introduces Program-Supervised Tool Reasoning (PSTR), which decomposes complex 3D questions into executable programs for point-cloud inspection, numerical computation, and external knowledge retrieval. Through iterative tool interaction, POINTTOOL dynamically acquires task-relevant information and localizes the point-cloud regions supporting its predictions. To train and evaluate this capability, we construct **PointCloud-TR**, a benchmark for tool-augmented 3D reasoning comprising 13K 3D objects, 39.8K question-answer pairs, and 35.6K executable tool trajectories across point-cloud perception, external-knowledge, and no-tool reasoning tasks. As illustrated in Fig. 1, this framework enables fine-grained, knowledge-intensive, and evidence-grounded 3D question answering. Our key contributions are summarized as follows:

- We introduce POINTTOOL, an encoder-free, tool-augmented 3D language model that unifies direct point-cloud modeling with interactive tool reasoning.
- We develop Alignment-Free Patch Tokenization and Decoupled Point-Text LoRA. AFPT preserves fine-grained geometric, positional, and color information through continuous local patch tokens, while DPL enables a frozen LLM to adaptively model point and text tokens through modality-specific low-rank branches.
- We propose Program-Supervised Tool Reasoning, which teaches the model to decompose complex 3D questions into executable programs and iteratively invoke point-cloud operations, numerical computation, and external knowledge retrieval.
- We construct PointCloud-TR, a benchmark for tool-augmented 3D reasoning comprising 13K 3D objects, 39.8K question-answer pairs, and 35.6K executable tool trajectories across point-cloud perception, external-knowledge, and no-tool reasoning tasks, providing a comprehensive testbed for evaluating interactive and evidence-grounded 3D reasoning.

2 Related Work

3D Large Multimodal Models. Early works on 3D LMMs focused on establishing unified 3D-language alignment. Point-Bind aligned point clouds with multiple modalities in a shared embedding space (Guo et al., 2023). PointLLM coupled a Point-BERT encoder with multimodal pretraining and two-stage instruction tuning (Xu et al., 2024; Yu et al., 2022; Xue et al., 2024), while ShapeLLM built its encoder on ReCon and added multi-view distillation (Qi et al., 2024a; 2023). MiniGPT-3D exploited 2D visual priors for efficient 3D–LLM alignment (Tang et al., 2024), and GPT4Point unified object-level point-language understanding and generation (Qi et al., 2024b). Other systems extended 3D LMMs to scene-level understanding (Hong et al., 2023; Wang et al., 2025; Fu et al., 2025; Chen et al., 2024). These methods rely on pretrained feature encoders as the interface to the LLM, making the information available to the LLM dependent on the encoder.

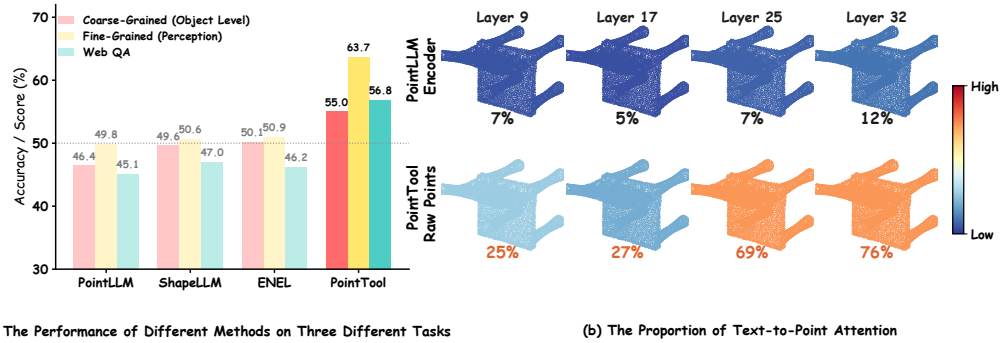
Encoder-Free Vision-Language Models. To overcome the representational limitations of pretrained 3D encoders and further reduce computational costs, recent studies have explored encoder-free 3D LMMs. ENEL (Tang et al., 2026) showed that encoder-free architectures can match encoder-based methods on multiple 3D understanding tasks. SAGE (Paul et al., 2026) introduced a lightweight 3D tokenizer and semantic alignment rewards to improve LLMs' processing of raw point clouds. Fase3D (Mei et al., 2026) further employed Fourier transforms to approximate self-attention, reducing both computational and parameter costs.

Tool-Augmented Language Models. Tool use provides an effective paradigm for extending the capabilities of language models. Toolformer learned when and how to invoke APIs through self-supervised training over unlabeled text, seeded by a handful of demonstrations for each API (Schick et al., 2023), while Gorilla focused on accurate API invocation over large tool collections (Patil et al., 2024). ViperGPT and VisProg generate executable programs over visual modules (Surís et al., 2023; Gupta & Kembhavi, 2023), whereas MM-ReAct prompts ChatGPT to call vision experts and integrate their outputs (Yang et al., 2023). In contrast to prior works, we aim to leverage this paradigm to enhance LLMs' 3D understanding.

3 Preliminary Analysis

As shown in Fig. 2(a), the pretrained-encoder-based PointLLM and ShapeLLM and the encoder-free ENEL remain close on fine-grained perception. In contrast, PointTool improves fine-grained perception while also strengthening coarse-grained performance. Its gain on Web QA further shows the value of combining point-cloud evidence with external knowledge.

To further examine this difference, we measure text-to-point attention across LLM layers. As shown in Fig. 2(b), PointLLM keeps this share low after the earliest layers, whereas PointTool increases it above 70% in the final layer. Appendix G provides the full layer-wise curve and object-level visualisations.



(a) The Performance of Different Methods on Three Different Tasks (b) The Proportion of Text-to-Point Attention

Figure 2: **Preliminary analysis of 3D representation bottleneck.** (a) The performance of different methods on three different tasks. (b) The proportion of text-to-point attention.

4 METHODOLOGY

In this section, we present POINTTOOL, an encoder-free and tool-augmented framework designed to fully exploit information inherent in raw point clouds for complex 3D reasoning. The overall architecture of POINTTOOL is illustrated in Fig. 3, which consists of three key components.

4.1 ALIGNMENT-FREE PATCH TOKENIZATION (AFPT)

Unlike conventional pretrained 3D encoders that compress the entire point cloud into a compact feature representation, AFPT performs only local aggregation, preserving geometric details from the raw point cloud before LLM processing. It keeps each patch as an independent point token and defers cross-patch interaction to the language model for downstream task-dependent modeling.

Formally, given a raw point cloud $P = \left\{\left(p_n^{xyz}, p_n^{rgb}\right)\right\}_{n=1}^{N}$, where p_n^{xyz}, p_n^{rgb} denote the 3D coordinates and color attributes of the n-th point, respectively. AFPT first employs the iterative *Farthest Point Sampling* (FPS) routine used by PointNet++ (Qi et al., 2017b) to select M center points: $C = \{c_i\}_{i=1}^{M}$. For each center point c_i, its k nearest neighbors are retrieved from the raw point cloud to construct a local point patch:

$$\mathcal{N}_i = \mathrm{kNN}(c_i, P; k). \tag{1}$$

For each local patch, we represent point coordinates in a center-relative coordinate system while retaining the original color attributes: $\tilde{p}_{ij} = p_{ij}^{xyz} - c_i$, where p_{ij} denotes the j-th neighbor in $\mathcal{N}_i$.

We then apply a shared point-wise mapping f_θ to each point and aggregate the points within each local patch using a symmetric max-pooling operation:

$$e_i = \mathrm{MaxPool}_j \, f_\theta(x_{ij}), \tag{2}$$

where $x_{ij} = \left[\tilde{p}_{ij}, p_{ij}^{rgb}\right]$ is the input feature of the j-th neighbor in the i-th patch. e_i denotes the feature representation of the i-th local patch.

To preserve the spatial location of each local patch within the entire object, we further introduce a positional encoding $\phi(c_i) \in \mathbb{R}^{d_p}$ and obtain the position-aware patch representation: $\bar{e}_i = e_i + \phi(c_i)$.

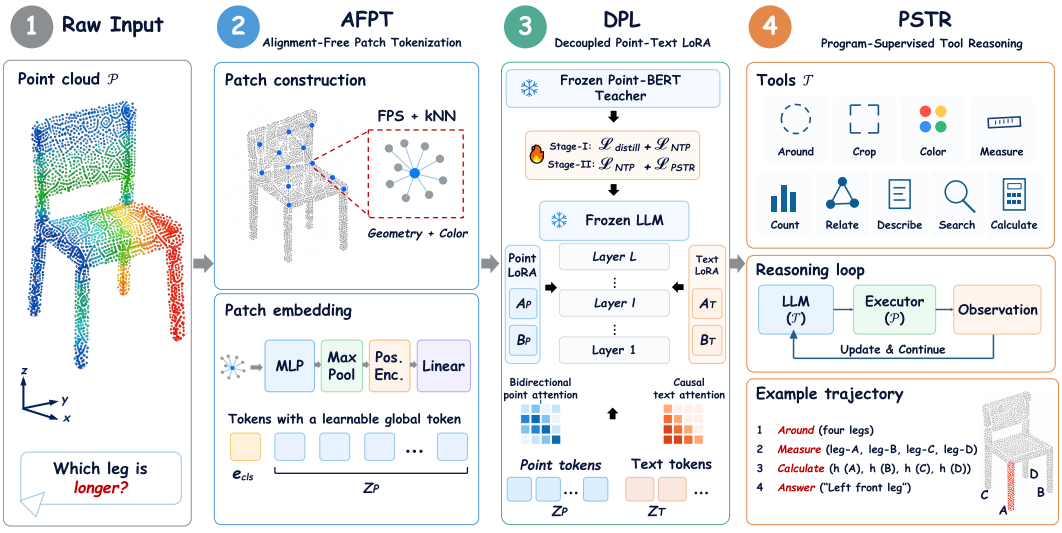


Figure 3: **Overview of POINTTOOL**: : (1) **AFPT** converts point coordinates and colors into position-aware point tokens Z_P via farthest point sampling (FPS) and kNN grouping. (2) **DPL** adapts the frozen LLM with decoupled Point/Text LoRA and bidirectional point attention, plus feature distillation for 3D structure internalization produced by AFPT. (3) **PSTR** performs tool with raw point clouds and external resources through an action–observation loop, enabling interactive 3D reasoning grounded in spatial evidence.

Finally, AFPT introduce a learnable global token e_{cls} and project the resulting point cloud representations into the hidden dimension of the language model:

$$Z_P = E_P W_P = [e_{\text{cls}}; \bar{e}_1; \ldots; \bar{e}_M]\, W_P, \qquad W_P \in \mathbb{R}^{d_p \times d_{\text{LM}}}, \tag{3}$$

where E_P and Z_P denote the pre-projection and projected point-token sequences, respectively. Z_P is then inserted into the LLM input at the position corresponding to the referenced point cloud.

4.2 Decoupled Point-Text LoRA (DPL)

AFPT effectively preserves local geometry in the resulting point tokens. However, enabling the LLM to further internalize these 3D representations still poses challenges. First, point and text tokens are heterogeneous, requiring effective modality adaptation within the shared LLM. Second, AFPT defers global aggregation, requiring the LLM to recover global 3D structure via cross-region interactions. Third, a mechanism is needed to enable the LLM to acquire sufficient 3D structural knowledge. To this end, we propose DPL, which integrates three complementary components.

Modality-Specific Adaptation. To accommodate the point-text heterogeneity, DPL assigns separate Low-Rank Adaptation (LoRA) branches (Hu et al., 2022) to each modality while keeping the LLM frozen. Formally, consider a frozen linear layer with weight matrix: $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. For the j-th token position, let h_j and h'_j denote its input and output hidden states, respectively. The adapted transformation is $h'_j = Wh_j + \Delta W_j h_j$, where DPL selects the low-rank update according to the token type:

$$\Delta W_j = \frac{\alpha}{r} \begin{cases} B_P A_P, & j \in S_P, \\ B_T A_T, & j \in S_T, \end{cases} \tag{4}$$

S_P and S_T denote the sets of point-token and text-token positions, respectively. The corresponding low-rank parameters are $A_P, A_T \in \mathbb{R}^{r \times d_{\text{in}}}$, $B_P, B_T \in \mathbb{R}^{d_{\text{out}} \times r}$.

Global Aggregation. To enable the LLM to model relationships among different 3D regions, we modify the attention mask for point tokens. Let Γ denote the attention mask, for any two point-token positions $i, j \in S_P$, we set $\Gamma_{ij} = 1$, allowing bidirectional attention among point tokens such that each local region can access the complete set of point-token representations. In contrast, text tokens retain the standard causal attention constraint and cannot attend to future text positions: $\Gamma_{ij} = 0\ (j > i,\ i, j \in S_T)$.

Feature Distillation. We introduce a Point-BERT (Yu et al., 2022) as a training-time teacher to further facilitate point LoRA's internalization of 3D structural knowledge. Specifically,

$$\mathcal{L}_{\text{distill}} = \frac{1}{|L|M} \sum_{l \in L} \sum_{i=1}^{M} \left[1 - \cos\left(\hat{f}_i^{(l)}, f_i^{\text{T}} \right) \right], \tag{5}$$

where L is the set of LLM layers used for distillation. $\hat{f}_i^{(l)}$ denotes the student feature of patch i at layer l, and f_i^{T} the corresponding teacher feature. Teacher network remains frozen during training.

4.3 Program-Supervised Tool Reasoning (PSTR)

Although AFPT and DPL proposed in PointTool alleviate the representation bottleneck, fully exploiting the information inherent in raw point clouds, particularly for complex 3D tasks, often requires explicit operations such as local region inspection, geometric measurement, spatial relation comparison, and even external knowledge retrieval. To address this, we propose PSTR, which formulates 3D understanding as an interactive reasoning process.

Tool Space. PSTR defines a unified tool set:

$$\mathcal{T} = \{\text{Locate}, \text{Crop}, \text{Color}, \text{Measure}, \text{Count}, \text{Relate}, \text{Describe}, \text{Search}, \text{Calculate}\}, \tag{6}$$

Together with a terminating action ANSWER, these tools form the complete action space $\mathcal{A} = \mathcal{T} \cup \{\text{ANSWER}\}$. Details of each tool are provided in Appendix B.

Interactive Execution. Given the point-token sequence Z_P, a question q, and the interaction history $\mathcal{H}_{t-1} = \{(a_i, o_i)\}_{i=1}^{t-1}$, where $a_i \in \mathcal{A}$ and o_i denote the action and observation at step i, respectively. The next action is predicted as $a_t \sim \pi_\Theta(a_t \mid Z_P, q, \mathcal{H}_{t-1})$, where π_Θ denotes the policy implemented by the LLM. For a tool call, the executor applies a_t to the environment, returning $o_t = \mathcal{E}(a_t; P, \mathcal{D})$, P is the raw point cloud and $\mathcal{D}$ denotes external data sources when required.

The resulting observation is then appended to the interaction history: $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{(a_t, o_t)\}$, and fed back to the model as part of the updated context, allowing subsequent actions to be conditioned on the newly acquired evidence. Region-producing tools additionally return persistent region references, enabling selected areas to be reused and progressively refined across multiple operations.The process terminates when the model generates ANSWER, yielding an interaction trajectory ending with the final answer y:

$$\tau = (a_1, o_1, \ldots, a_K, o_K, y), \qquad K \leq 6, \tag{7}$$

Through this action–observation loop, the model can actively gather task-relevant evidence from the 3D data, enabling it to dynamically adjust subsequent operations according to the returned tool results and thereby benefit downstream 3D understanding tasks.

Program-Supervised Trajectories. The above interaction requires the model to learn not only the final answer, but also when to invoke a tool, which operation to perform, and where to perform it. Supervising only the final answer cannot directly constrain these intermediate decisions. We therefore construct complete supervision trajectories from executable reference programs.

For each training question with known scene structure, we first construct a reference program $\pi^* = (a_1^*, \ldots, a_K^*)$, where each action is executable within the PSTR tool space. The program is then executed on the raw point cloud to obtain the corresponding observations: $o_t^* = \mathcal{E}(a_t^*; P, \mathcal{D})$, yielding the reference trajectory:

$$\tau^* = (a_1^*, o_1^*, \ldots, a_K^*, o_K^*, \mathrm{ANSWER}(y^*)). \tag{8}$$

The model is trained on the executed trajectory, learning each tool invocation and the final answer autoregressively conditioned on the preceding observations. Specifically, the training objective is

$$\mathcal{L}_{\mathrm{PSTR}} = -\sum_{t=1}^{K} \log \pi_\Theta \left(a_t^* \mid Z_P, q, \mathcal{H}_{t-1}^*\right) - \log \pi_\Theta \left(\mathrm{ANSWER}(y^*) \mid Z_P, q, \mathcal{H}_K^*\right), \tag{9}$$

where $\mathcal{H}_{t-1}^*$ contains the previously executed actions and their corresponding observations.

4.4 Training Strategy

PointTool is trained in two stages. Stage I jointly trains AFPT and DPL to establish and internalize 3D representations within the LLM, while Stage II further incorporates PSTR to learn instruction following and interactive tool reasoning. Throughout both stages, the LLM backbone remains frozen, while the AFPT tokenizer and Point/Text LoRA branches are optimized.

Next-Token Prediction. Both stages adopt a unified autoregressive objective. Given a sequence $X = (x_1, \ldots, x_T)$ and a binary mask $m_t \in \{0, 1\}$ indicating supervised positions, we define:

$$\mathcal{L}_{\mathrm{NTP}} = -\frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \log p_\Theta(x_t \mid Z_P, x_{<t}), \tag{10}$$

For regular instructions, only response tokens are supervised. For PSTR trajectories, the model-generated tool calls and final answer are supervised, so that Eq. 10 becomes the trajectory objective $\mathcal{L}_{\mathrm{PSTR}}$ in Eq. 9.

Stage I: Following PointLLM, Stage I uses 660K brief object descriptions from Objaverse (Deitke et al., 2023; Xu et al., 2024) and trains AFPT and DPL with the feature distillation objective, providing additional structural supervision for 3D representation learning within the LLM:

$$\mathcal{L}_{\mathrm{stage1}} = \mathcal{L}_{\mathrm{NTP}} + \lambda_{\mathrm{dist}} \mathcal{L}_{\mathrm{distill}}, \tag{11}$$

where $\lambda_{\mathrm{dist}} = 1$. This stage enables AFPT to preserve and expose fine-grained point cloud information, while DPL progressively adapts and organizes these representations within the LLM.

Stage II: At this stage, we remove the Point-BERT teacher and distillation and jointly train on 70K complex instructions from Objaverse (Deitke et al., 2023; Xu et al., 2024) and the training split of PointCloud-TR.

Instruction data are supervised with Eq. 10 and tool trajectories with Eq. 9, which is the same objective under the trajectory mask, so the two sources are optimized jointly:

$$\mathcal{L}_{\text{stage2}} = \mathbb{E}_{X\sim\mathcal{D}_{\text{inst}}}\left[\mathcal{L}_{\text{NTP}}(X)\right] + \mathbb{E}_{\tau\sim\mathcal{D}_{\text{tool}}}\left[\mathcal{L}_{\text{PSTR}}(\tau)\right]. \tag{12}$$

Following the representation-learning-to-instruction-tuning progression used in prior 3D LMMs (Xu et al., 2024; Tang et al., 2026), this two-stage design first focuses on 3D knowledge internalization, followed by instruction tuning for both instruction following and tool reasoning.

5 Experiments

5.1 Experimental Setup

Evaluation Benchmarks. We conduct experiments on two complementary suites: standard 3D understanding and tool reasoning. Following the protocol established in prior work (Xu et al., 2024; Tang et al., 2026), the former covers captioning and classification on Objaverse (Deitke et al., 2023; Xu et al., 2024) and multimodal VQA on 3D MM-Vet (Qi et al., 2024a). The latter is evaluated on the held-out split of PointCloud-TR Dataset (Section 5.2), comprising Perception QA, Web QA, and No-tool QA.

Implementation Details. Following the setup of the previous work Xu et al. (2024), we adopt Vicuna-7B v1.1 as the frozen LLM backbone. For AFPT, each raw point cloud ($N = 8192$) is sampled into $M = 512$ local patches with $k = 32$ nearest neighbors per patch, projecting the local patch representations to $d_p = 384$ before LLM projection. For DPL, both Point and Text LoRA adapters are configured with rank $r = 64$ and scaling factor $\alpha = 128$, applied to all linear projection matrices in every Transformer layer. Feature distillation is performed at LLM layers $L = \{8, 16\}$ using a frozen Point-BERT teacher. Detailed training settings are provided in Appendix C.

Dataset	Obj.	QA	Fine	Ext. Know.	Traj.	Avg. Calls
3D-LLM	N/A	300K	✗	✗	✗	–
PointLLM-B	660K	660K	✗	✗	✗	–
PointLLM-C	15K	70K	✗	✗	✗	–
ShapeLLM	47K	45K	✗	✗	✗	–
PointCloud-TR	13K	39.8K	✓	✓	✓	2.2

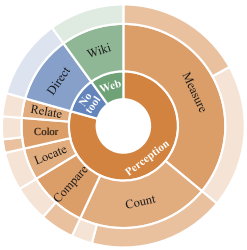


Figure 4: Composition of the Proposed PointCloud-TR Dataset

5.2 PointCloud-TR Dataset for Tool Reasoning

To support the training and evaluation of tool-mediated 3D reasoning for challenging tasks, we construct PointCloud-TR, a benchmark comprising diverse types of 3D reasoning data (Fig. 4). It contains approximately 13K 3D objects and 39.8K question-answer pairs, accompanied by 35.6K executable tool trajectories with an average of 2.2 tool calls per trajectory.

Existing 3D datasets such as 3D-LLM (Hong et al., 2023), PointLLM (Xu et al., 2024), and ShapeLLM (Qi et al., 2024a) do not jointly provide fine-grained part queries, retrieval-grounded external-knowledge questions, and executable reasoning trajectories. PointCloud-TR is organized into three key task categories: **Perception**, which covers geometric and spatial queries, including Measurement (14.3K), Count (8.3K), Comparison (3.7K), Localisation (2.0K), Colour (1.9K), and Relation (1.2K); **Web**, which contains 4.0K Wikipedia-grounded questions; and **No-tool**, which contains 4.2K questions that can be answered directly without tool use. Together, these categories evaluate not only 3D reasoning, but also the ability to acquire task-relevant evidence and determine when explicit tool use is necessary, providing a dedicated benchmark for 3D understanding.

Table 1: Performance on standard 3D understanding benchmarks: Objaverse and 3D MM-Vet. †: same training data as baselines; full POINTTOOL additionally trained on PointCloud-TR.

	Captioning			Cls.	VQA
Method	GPT-4	S-BERT	SimCSE	GPT-4	GPT-4
Pretrained-encoder-based					
3D-LLM	33.42	44.48	43.68	45.25	–
PointLLM-7B	44.85	47.47	48.55	53.00	41.20
PointLLM-13B	48.15	47.91	49.12	54.00	46.60
ShapeLLM-7B	46.92	48.20	49.23	54.50	47.40
ShapeLLM-13B	48.94	48.52	49.98	54.00	53.10
MiniGPT-3D	52.49	48.73	49.26	54.50	43.60
Encoder-free					
ENEL-7B	51.03	48.79	49.52	55.55	43.80
SAGE-7B	50.98	50.11	49.70	57.11	49.53
POINTTOOL-7B†	52.8	50.3	**50.3**	57.3	51.9
POINTTOOL-7B	**53.6**	**50.6**	50.1	**57.9**	**53.4**

5.3 PERFORMANCE ON STANDARD 3D UNDERSTANDING BENCHMARKS

Table 1 reports performance on standard 3D understanding benchmarks. While prior encoder-free methods largely match encoder-based counterparts, our POINTTOOL leads four of the five reported metrics and remains close on SimCSE, supporting the benefit of internalizing raw point clouds in the LLM. Specifically, it improves over the strongest prior results by 1.1, 0.8, and 0.3 points on GPT-4 captioning, classification, and VQA, respectively. Notably, the strongest competing VQA result comes from 13B ShapeLLM, whereas ours uses a standard 7B LLM.

We further compare two training variants of POINTTOOL. The † variant is already competitive when trained with the same data as the competing baselines, while incorporating PointCloud-TR consistently improves GPT-4 captioning ($+0.8$), Sentence-BERT ($+0.3$), classification ($+0.6$), and VQA ($+1.5$), with only a marginal change in SimCSE (-0.2). These results show that the additional supervision provided by PointCloud-TR, particularly its substantial collection of executable tool trajectories, complements instruction tuning without compromising standard 3D understanding.

5.4 PERFORMANCE ON THE PROPOSED POINTCLOUD-TR BENCHMARK

We evaluate POINTTOOL and representative baselines on the proposed PointCloud-TR benchmark in Table 2. In the one-pass setting, existing methods and POINTTOOL achieve only around 50 $\text{Overall}_{\text{P+W}}$ accuracy, highlighting the difficulty of the benchmark without explicit tool interaction. In contrast, enabling PSTR improves POINTTOOL's $\text{Overall}_{\text{P+W}}$ accuracy from 50.9 to 63.2, with

Table 2: Performance on PointCloud-TR. $\text{Overall}_{\text{P+W}}$ is the question-weighted Perception–Web average; Routing covers all three routes, including No-tool.

	Answer ↑				
Model	Perception	Web	$\text{Overall}_{\text{P+W}}$	Routing ↑	Execution ↑
One-pass setting					
PointLLM-7B	49.8	45.1	49.5	–	–
ShapeLLM-7B	50.6	47.0	50.4	–	–
ENEL-7B	50.9	46.2	50.6	–	–
POINTTOOL-7B	51.2	47.5	50.9	–	–
POINTTOOL-7B, Answer-only	54.6	51.2	54.4	–	–
Tool-reasoning setting					
PointLLM-7B + PSTR	57.6	52.9	57.3	80.1	89.3
POINTTOOL-7B + PSTR	**63.7**	**56.8**	**63.2**	**83.2**	**90.1**

substantial gains on both perception and external-knowledge questions, demonstrating the significant benefit of interactive tools for 3D understanding.

To isolate the source of these gains, the answer-only variant, trained on the same questions but without tool trajectories, reaches only 54.4 Overall$_{\mathrm{P+W}}$, far below POINTTOOL with PSTR (63.2). Moreover, PointLLM-7B + PSTR reaches 57.3 Overall$_{\mathrm{P+W}}$, while POINTTOOL + PSTR achieves 63.2, further demonstrating the effectiveness of our encoder-free design given the same tool space and program supervision. Appendix E provides the corresponding breakdown by question type, reference-call ceiling, and localisation quality.

5.5 ABLATION STUDY

Component Ablation. Table 3 presents the ablation results of the proposed components. Replacing the pretrained Point-BERT encoder(*Original*) with AFPT consistently improves all four metrics, demonstrating the benefits of the encoder-free design. Adding DPL without distillation yields a further 0.5–0.6 point improvement, validating the effectiveness of modality-specific adaptation and in-LLM global aggregation. Finally, feature distillation brings an additional 0.4–0.6 point gain across all metrics, supporting its complementary role in 3D structural representation learning. Additional

Table 3: The ablation results on the proposed components from POINTTOOL.

	Captioning		Cls.	VQA
Component	GPT-4	S-BERT	GPT-4	GPT-4
Original	50.9	48.8	55.5	49.6
+ AFPT	51.7	49.4	56.2	50.7
+ DPL, w/o FD	<u>52.2</u>	<u>49.9</u>	<u>56.7</u>	<u>51.3</u>
+ FD (POINTTOOL-7B†)	**52.8**	**50.3**	**57.3**	**51.9**

robustness checks and representation diagnostics are reported in Appendices F and G; runtime and visual front-end cost are reported in Appendix H.

Analysis of Interactive Tool Reasoning. Beyond overall performance, we analyze the effect of multi-step tool interaction in Table 4. Compared with direct answering, predicted tool execution consistently improves accuracy across different trajectory lengths, with gains of 15.3, 8.3, and 4.4 points for 2-, 3-, and 5-call trajectories, respectively. The performance gap to oracle execution also increases for longer trajectories, indicating that errors accumulate as the interaction becomes more involved. Table 5 further shows that PSTR learns to select appropriate initial actions, achieving 83.9%, 77.0%, and 80.5% accuracy for Perception, Web, and No-tool questions, respectively. These results demonstrate that PSTR supports both multi-step tool interaction and adaptive tool selection.

Table 4: Multi-step reasoning analysis.

Calls	Answer Accuracy (%)		
	No tools	Reference	Predicted
2 Calls	51.9	82.1	67.2
3 Calls	50.6	77.5	58.9
5 Calls	49.0	72.8	53.4

Table 5: Tool routing behavior.

Question Type	First Action (%)		
	Perception	Web	Direct
Perception	83.9	4.3	11.8
Web	8.6	77.0	14.4
No Tool	13.1	6.4	80.5

6 CONCLUSION

In this work, we present POINTTOOL, an encoder-free, tool-augmented framework for 3D reasoning. AFPT and DPL directly connect raw point clouds to an LLM, preserving fine-grained 3D information while enabling its internalization. PSTR adds interactive reasoning via executable point-cloud operations and external resources, letting the model actively gather task-relevant evidence. We also build PointCloud-TR, a benchmark with executable trajectories for tool-augmented 3D understanding. Experiments show POINTTOOL achieves superior performance on most metrics across standard benchmarks and challenging PointCloud-TR tasks.

AI USE STATEMENT

Generative AI tools were used to assist with language and presentation, with migrating the manuscript to the ICLR 2027 template, and with checking internal consistency and bibliographic metadata. A large language model was also used inside the data pipeline, as described in Appendix A: it drafts the web questions from retrieved passages, and every draft is kept only if it passes the automatic quote and name-leakage checks. GPT-4 serves as the automatic judge for captioning, classification and visual question answering, following the protocol of the benchmarks we report on. All experiments, results and claims are the authors' own; the authors reviewed every suggested change and take responsibility for the final content.

ETHICS STATEMENT

This work uses publicly released research datasets, drawing objects from Objaverse and ShapeNet-Part (Deitke et al., 2023; Yi et al., 2016). The generated questions concern object geometry, appearance, and category rather than personal or identifying information. External evidence is retrieved from Wikipedia and stored with its article title and revision date, which both credits the source and keeps the evaluation reproducible. The released model inherits the biases of its frozen language backbone and of the retrieved text, and the tools return measurements of the input point cloud rather than assertions about people.

REPRODUCIBILITY STATEMENT

The supplementary material contains the model implementation, the benchmark construction pipeline, the tool definitions and the executor; the generated data and the checkpoints will be released. Appendix A documents the question templates, the generation and filtering procedure and the splits; Appendix B gives the tool signatures, argument ranges and the executor protocol; Appendix C lists the architecture, the two training stages and every hyperparameter, together with the provenance of the instruction corpora. Evaluation follows the public protocols of Xu et al. (2024) and Qi et al. (2024a), and web questions are scored against the stored Wikipedia snapshots so that the numbers do not depend on when the benchmark is run.

REFERENCES

Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. ScanQA: 3D question answering for spatial scene understanding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19107–19117. IEEE, 2022.

Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. ScanRefer: 3D object localization in RGB-D scans using natural language. In *European conference on computer vision*, pp. 202–221. Springer, 2020.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X Chang. Scan2Cap: Context-aware dense captioning in RGB-D scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3193–3203, 2021.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-LLM: Extending language model for 3D visual reasoning. In *Winter Conference on Applications of Computer Vision (WACV)*, pp. 2195–2206, 2025.

Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, and Pheng-Ann Heng. Point-Bind & Point-LLM: Aligning point cloud with multi-modality for 3D understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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

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

Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3D captioning with pretrained models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Guofeng Mei, Wei Lin, Luigi Riz, Yujiao Wu, Yiming Wang, and Fabio Poiesi. Efficient encoder-free Fourier-based 3D large multimodal model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

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

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sneha Paul, Zachary Patterson, and Nizar Bouguila. Point cloud as a foreign language for multi-modal large language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017b.

Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. Contrast with reconstruct: Contrastive 3D representation learning guided by generative pretraining. In *International Conference on Machine Learning (ICML)*, 2023.

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

Zhangyang Qi, Ye Fang, Zeyi Sun, Xiaoyang Wu, Tong Wu, Jiaqi Wang, Dahua Lin, and Hengshuang Zhao. GPT4Point: A unified framework for point-language understanding and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

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 (ICLR)*, 2026.

Yuan Tang, Xu Han, Xianzhi Li, Qiao Yu, Yixue Hao, Long Hu, and Min Chen. MiniGPT-3D: Efficiently aligning 3D point clouds with large language models using 2D priors. In *ACM International Conference on Multimedia (ACM MM)*, 2024.

Zehan Wang, Haifeng Huang, Yang Zhao, Ziang Zhang, Tao Jin, and Zhou Zhao. Data-efficiently learn large language model for universal 3D scene perception. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 313–333, 2025. doi: 10.18653/v1/2025.findings-naacl.18.

Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S Yu. Multimodal large language models: A survey. In *2023 IEEE international conference on big data (bigdata)*, pp. 2247–2256. IEEE, 2023.

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 (ECCV)*, 2024.

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

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. ULIP-2: Towards scalable multimodal pre-training for 3D understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. MM-REACT: Prompting ChatGPT for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics*, 35(6):210:1–210:12, 2016. doi: 10.1145/2980179.2980238.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499. IEEE, 2018.

The appendix mirrors the evidence chain of the main paper. We first document PointCloud-TR and the PSTR executor (Appendices A–B), then give implementation and evaluation details (Appendices C–D), and finally provide the diagnostic evidence for PSTR and the encoder-free AFPT–DPL interface (Appendices E–H).

A POINTCLOUD-TR: CONSTRUCTION AND VALIDATION

Table 6 reports the final split sizes used throughout the paper.

Table 6: Final PointCloud-TR question counts by split and route. Web counts are reported after shortcut filtering.

	Questions			
Split	Perception	Web	No-tool	Total
Train	24,861	3,426	3,624	31,911
Validation	2,273	229	273	2,775
Test	4,451	324	344	5,119

A.1 PERCEPTION QUESTION TEMPLATES

Perception templates. Each perception question comes from one of ten template variants spanning six evaluation types over the annotated parts of a ShapeNet-Part object. Binary templates compare two parts (comparison), ask whether a region has more than a given number of separate pieces (count), whether an extent exceeds a threshold (measurement) or whether one part lies in a given direction from another (relation); open templates ask for a count, an extent, the ratio of two extents, or the name of the part with the largest extent (localisation). Colour questions use the same forms on the procedurally built objects. Each template measures the object before choosing what to ask, and a running tally per template steers each new question toward the answer it has produced less often. The reference calls are authored from the part annotation: an Locate call takes the centroid of the part and the radius of its farthest point, and a part is used only if that ball holds at least 60 percent part points and covers at least 80 percent of the part.

Table 7: Perception template variants and their reference programs. The final split counts sum to the Perception totals in Table 6. Binary and open variants form the six types in the main paper, with Ratio grouped under Comparison; r0 and r1 denote regions returned by Locate.

Template	Answer	Train	Val	Test	Reference program
One region					
Count	binary	5,642	538	1,052	Locate, Count
Count, open	integer	858	94	154	Locate, Count
Measurement	binary	5,382	442	918	Locate, Measure
Measurement, open	scalar	5,883	601	1,110	Locate, Measure
Colour	binary	606	58	52	Locate, Color
Colour, open	name	911	138	158	Locate, Color
Localisation	part name	1,563	154	315	Locate, Describe
Two regions					
Relation	binary	1,050	34	142	Locate×2, Relate
Comparison	binary	1,460	92	268	(Locate, Measure)×2, Calculate
Ratio	scalar	1,506	122	282	(Locate, Measure)×2, Calculate

Table 7 lists the templates. Seven act on a single region and three on a pair, so trajectories are two, three or five calls long; across the training split the perception trajectories average 2.4 calls, and together with the single-call web trajectories this gives the mean of $K = 2.2$ reported in Section 5.2. The calls are distributed as Locate 28,877, Measure 17,197, Count 6,500, Calculate 2,966, Describe 1,563, Color 1,517 and Relate 1,050. The objects are the 14 ShapeNet-Part classes

whose part names are unambiguous, listed in Table 8, together with procedurally built objects that carry colour, which ShapeNet-Part does not.

Table 8: The 14 ShapeNet-Part classes used and their part names. Part annotations are used only to generate questions and reference programs and are never shown to the model.

Class	Parts
Airplane	body, wing, tail, engine
Bag	handle, body
Cap	crown, brim
Car	roof, hood, wheel, body
Chair	back, seat, leg, armrest
Guitar	head, neck, body
Knife	blade, handle
Lamp	base, shade, canopy, tube
Laptop	keyboard, screen
Mug	handle, body
Pistol	barrel, grip, trigger
Rocket	body, fin, nose
Skateboard	wheel, deck, truck
Table	top, leg

Reference-program examples. A one-region measurement question such as "Is the body wider than 0.37?" executes Locate followed by Measure. A relation question executes two Locate calls followed by Relate, while a comparison or ratio question executes two location–measurement pairs followed by Calculate. Concrete coordinates are expressed in the unit cube centred at the origin; the full program skeleton for every variant is given in Table 7.

A.2 Web Question Generation

Web drafting. For each Objaverse-LVIS category the article of that category on Wikipedia is retrieved and stored with its title and revision date. A language model (Qwen3-Max, through an API) receives the passage and returns a question about the object, its answer, a verbatim quote that supports the answer and the query that would retrieve the passage. A draft is kept only if the quote is an exact substring of the passage and neither the question nor the query names the object or its category. Questions that the backbone answers correctly with neither point cloud nor passage are removed.

A kept draft has the form

> *Question:* Besides sports and general work, what other fields use this protective equipment?
> *Answer:* Trades and military use.
> *Quote:* "Knee pads are also . . . "
> *Query:* knee pads trades military use

The question refers to the object only as *this protective equipment*, so the object has to be recognised from the point cloud before the query can be formed. After shortcut filtering, 3,979 Web questions remain: 3,426 for training, 229 for validation, and 324 for testing, as in Table 6.

A.3 No-Tool Questions

Ordinary questions about the same object categories are taken from the instruction data of Xu et al. (2024) and are stored in the same format with an empty call sequence, so that the decision not to invoke a tool is supervised alongside the decision of which tool to invoke. They are split by category in the same way as the web questions.

A.4 Answer Balancing

Each Perception template measures the relevant object property before the question is instantiated. A running tally then favours less frequent answer values; Table 7 reports the resulting split counts.

A.5 Splits and Statistics

Table 6 reports the size of each split. Perception questions are split by object on the ShapeNet-Part benchmark splits and web and no-tool questions by category, so no held-out item shares an object or an entity with training. The Web counts are reported after shortcut filtering.

Representative prompts. The three routes ask complementary questions without naming the object: No-tool asks, for example, "What object is represented, and what is its typical function?"; Perception asks "Is the widest part of this object taller than its narrowest part?"; and Web asks "When was the species represented by this object first scientifically described?"

B PSTR Tool Space and Executor

B.1 Tool Interface

Table 9 lists the tools with their argument ranges and return values. Point clouds are normalised to the unit cube centred at the origin, so coordinates lie in $[-0.5, 0.5]$ and the radius of Locate in $(0, 1]$. Measure returns the extent of a region along its longest axis (length), along z (height) or along its middle axis (width), its bounding volume, or the ratio of longest to shortest extent (elongation). Count links points within a radius of 0.06 and returns the number of connected components with at least $\max(8, n/200)$ points, where n is the size of the region. Relate returns the direction along the axis of largest separation between the centroids of two regions and the distance between them. Describe returns a description of the region's points produced by the model itself; Search returns a passage with its source and date; Calculate evaluates an expression restricted to numbers, parentheses and the four arithmetic operators. The executor identifiers are `around`, `crop`, `color_of`, `measure`, `count`, `relate`, `describe`, `web_search`, and `calc`.

Table 9: Tool interface shared by training and evaluation. Spatial arguments use unit-cube coordinates, and region handles refer to outputs of preceding calls.

Tool	Arguments and ranges	Returns
Locate	$x, y, z \in [-0.5, 0.5]$, $r \in (0, 1]$	region handle, `n_points`
Crop	region, axis $\in \{x, y, z\}$, side matching the axis	region handle, `n_points`
Color	region	colour name from a palette of eight, mean RGB
Measure	region, quantity $\in$ {length, height, width, volume, elongation}	scalar, four decimals
Count	region	integer
Relate	two region handles	direction name, distance
Describe	region	short description produced by the model
Search	query string	passage, source title, revision date
Calculate	expression over digits, brackets and $+ - \times \div$	scalar, four decimals

B.2 Action–Observation Protocol

The model emits one call per turn as a JSON object with the tool name and its arguments, and the executor returns the result as a JSON object on the next turn. A call that produces a region returns a handle, `r0`, `r1` and so on, which later calls refer to. A trajectory may contain at most six calls. During training the executor results come from running the reference program; at evaluation the same executor runs the model's own calls. For the reported numbers Search resolves against

the Wikipedia snapshot stored with each question, identified by its article title and revision date, so that the benchmark is deterministic; querying Wikipedia live is supported and used only for the qualitative examples.

B.3 Execution Failures

A call that fails, because the selected region is empty, an argument lies outside its range, the tool name is unknown or the call cap has been reached, returns an error message in the same JSON format rather than terminating the trajectory, and the model continues from it. The Execution metric of Table 2 is the share of predicted trajectories in which every emitted call executes without error, so a trajectory containing any failed call does not count towards it.

C AFPT, DPL, and PSTR Implementation Details

C.1 Notation and Hyperparameters

Table 10 lists the key default architecture and interaction settings; sensitivity experiments state their overrides separately.

Table 10: Default architecture and interaction settings.

Symbol	Meaning	Value
N	points per object	8,192
M	patch centres sampled by FPS, and point tokens	512
k	neighbours aggregated into each patch	32
d_p	patch embedding dimension	384
d_{LM}	hidden size of the language model	4,096
r	rank of the Point and Text LoRA branches	64
α	LoRA scaling	128
L	layers at which the point branch is distilled	$\{8, 16\}$
λ_{dist}	weight of the distillation loss in Stage I	1
$K_{\max}$	calls allowed in one trajectory	6

The language model is Vicuna-7B v1.1 with 32 transformer layers and stays frozen throughout. LoRA branches are attached to the query, key, value and output projections of attention and to the three projections of the feed-forward block in every layer, with dropout 0.05. The distillation heads are one linear map per distilled layer, from d_{LM} to d_p.

C.2 Instruction Corpora

Both stages reuse the instruction data released by Xu et al. (2024), which is built on Cap3D captions of Objaverse objects (Deitke et al., 2023; Luo et al., 2023) and therefore overlaps in source domain with the web and no-tool routes of PointCloud-TR. The perception route instead uses ShapeNet-Part objects.

The Stage-I corpus pairs each object with one short description. It contains 658,577 examples after the authors' filtering step, one per object, with a mean response length of 13 words; the prompt is drawn from a fixed set of paraphrases such as *Deliver a quick description of the object represented here*. We use the filtered release and sample no further, so Stage I sees every object exactly once per epoch.

The Stage-II corpus contains 70,274 examples over 15,098 objects, a subset of the objects seen in Stage I, generated by prompting GPT-4 with the captions of Xu et al. (2024). It is a mixture of three conversation types: 15,055 detailed descriptions, 40,122 single-round question-answer pairs and 15,097 multi-round dialogues, the last of which are three rounds long in all but six cases. We keep the mixture unchanged and add the training split of PointCloud-TR to it.

Neither corpus contains any of the 3,000 objects held out for the Objaverse captioning and classification benchmarks, so the evaluation objects of Table 1 are unseen in both stages. The perception questions of PointCloud-TR come from ShapeNet-Part rather than Objaverse and therefore share no object with either corpus.

C.3 Stage I

The brief descriptions are trained for three epochs with an effective batch of 128 (two GPUs, 32 per device, two accumulation steps), learning rate 2×10^{-4}, cosine schedule with 3 percent warm-up and bf16. The language model is frozen; the tokenizer, the two LoRA branches (rank 64, α=128, on the query, key, value and output projections and the three feed-forward projections of all 32 layers) and the distillation heads are trained. The distillation regresses the point-position hidden states at layers 8 and 16 through one linear head per layer onto the 384-dimensional patch features that the Point-BERT encoder of PointLLM produces for the same patches, with a cosine loss of weight 1 added to the next-token loss. The stage takes about 87 hours on two H20 GPUs.

C.4 Stage II

The complex instructions and the 31,911 training examples of PointCloud-TR are mixed and trained for three epochs with an effective batch of 32 (8 per device, two accumulation steps), learning rate 5×10^{-5}, cosine schedule with 3 percent warm-up, maximum sequence length 2048 and bf16. Training continues from the stage-one tokenizer and branches; the distillation heads are dropped. Within a tool trajectory the loss is applied to model turns only.

C.5 Inference Settings

Point tokens attend bidirectionally within the point block and text keeps the causal mask. Decoding is greedy, one turn at a time, with the executor cap of six calls.

D Evaluation Protocol and Metrics

We compare with representative encoder-based 3D LMMs, including PointLLM, ShapeLLM, and MiniGPT-3D, and encoder-free models, including ENEL and SAGE. Published results are taken from the compilations in Tang et al. (2026); Paul et al. (2026). We also reproduce PointLLM-7B and ENEL-7B using their released implementations, obtaining results consistent with the reported values.

For standard 3D understanding, we follow the evaluation protocols of the corresponding benchmarks. For tool reasoning, existing baselines are evaluated in a one-pass setting because they do not support tool calls. PointTool is evaluated under two settings: *one-pass*, where PSTR is disabled, and *tool reasoning*, where the model can interact with PSTR. During evaluation, Search resolves against the stored Wikipedia snapshot of each question, identified by its article title and revision date, so that scores are reproducible.

PointCloud-TR metrics. *Answer* scores the final response, using exact match for binary and short-form answers, a fixed relative tolerance for numerical answers, and normalised token overlap for free-form responses. *Routing* is the accuracy of the first action across the Perception, Web, and No-tool routes: the model must select a perception tool, Search, or direct answering, respectively. *Execution* is the fraction of predicted trajectories that emit at least one call and in which every emitted call can be parsed and executed without error. *Overall (P+W)* is the question-weighted average of Perception and Web answer accuracy; No-tool is reported in the routing analysis and is not included in this aggregate.

To isolate the contributions of tool use, program supervision, and the proposed encoder-free interface, we further evaluate two controlled variants. *Answer-only* uses the same model and training recipe as PointTool, but replaces each tool trajectory with its final answer, so that the model sees the same questions without tool supervision. *PointLLM-7B + PSTR* equips PointLLM-7B with the same PSTR tools and program-supervised trajectories, isolating the effect of the encoder-free interface from tool use. We additionally evaluate a *Reference-call* setting, where the reference tool

calls are provided as context, to quantify the effect of tool routing; detailed results are reported in Appendix E.

E PSTR DIAGNOSTICS ON POINTCLOUD-TR

This section complements Table 2 with the reference-call ceiling and localisation diagnostics. Question types and split counts are defined once in Tables 7 and 6; depth and routing results remain in the main paper.

E.1 REFERENCE-CALL CEILING

Supplied with the reference calls as context, POINTTOOL-7B answers 79.8 percent of Perception and 72.1 percent of Web questions, reaching 79.3 Overall$_{\mathrm{P+W}}$. Since the calls are supplied rather than generated, the predicted-trajectory *Execution* metric does not apply. This setting estimates answer accuracy when routing and arguments are fixed to their references; final-answer generation is still evaluated normally.

E.2 REGION LOCALISATION

A predicted Locate region counts as correct when it covers at least 80 percent of the points of the part that the reference call targets and at least 60 percent of the selected points belong to that part, the same criterion used to author the reference calls (Appendix A). Where the reference contains two Locate calls, both regions must be correct. In predicted trajectories 70 percent of Locate calls meet this criterion. Answer accuracy is 76 percent when localisation succeeds and 36 percent when it fails. Locate coordinates receive autoregressive sequence supervision through the reference program, but no separate geometric localisation loss.

F AFPT HYPERPARAMETER SENSITIVITY

Table 11 varies k and distillation depth one at a time around the default in Table 3. All rows use $M{=}512$ point tokens and the PointLLM instruction data of Xu et al. (2024); stars mark defaults.

Table 11: One-factor sensitivity to neighbourhood size and distillation depth. Stars mark defaults; best results are bold and second best are underlined.

		Captioning		Cls.	VQA
Setting	Value	GPT-4	S-BERT	GPT-4	GPT-4
Neighbours k	8	52.4	49.9	56.9	51.5
	16	<u>52.6</u>	<u>50.1</u>	<u>57.1</u>	<u>51.7</u>
	32*	**52.8**	**50.3**	**57.3**	**51.9**
	64	52.5	50.0	57.0	51.6
Distillation layers	8	52.5	50.0	57.0	51.6
	8, 16*	**52.8**	**50.3**	**57.3**	**51.9**
	8, 16, 24	<u>52.7</u>	<u>50.2</u>	<u>57.2</u>	<u>51.8</u>

All displayed differences are within 0.4 points. Without run-to-run variability, we treat them as descriptive rather than statistically significant. The default is best on all four metrics, and adding layer 24 gives no further improvement.

G AFPT–DPL REPRESENTATION ANALYSIS

G.1 INFORMATION AT THE POINT-TOKEN INPUT

A linear probe is fitted on the tokens that enter the language model, from the aligned Point-BERT encoder of PointLLM and from the tokenizer of POINTTOOL, to predict the dominant colour of a

patch and the part it belongs to on ShapeNet-Part objects; the number reported is held-out accuracy against the majority baseline. Tokens are the tokenizer output for POINTTOOL and the encoder output for PointLLM; 48 procedurally built objects give 24,576 patches, with eight colour classes (majority 26.0 percent) and nine part classes (majority 40.6 percent), and probes are fitted on some objects and scored on the others. Table 12 reports the result. Alignment pre-training keeps stronger part identity but substantially less colour, whereas the tokenizer trained with the language model keeps colour almost perfectly and carries less part structure at the token level.

Table 12: Information linearly decodable from point representations. (a) Patch-level held-out accuracy at the LLM input. (b) Part-segmentation mIoU at the input and selected LLM layers.

(a) Patch-level probes

Target	Classes	PointLLM-7B	POINTTOOL-7B
Patch colour	8 (majority 26.0)	79.4	**99.7**
Part identity	9 (majority 40.6)	**90.9**	66.9

(b) Part segmentation across depth

Representation	PointLLM-7B	POINTTOOL-7B
Tokens entering the language model	**61.8**	47.5
Layer 8	60.4	55.2
Layer 16	59.1	60.7
Layer 24	58.3	**64.1**
Layer 32	56.9	62.8

G.2 EMERGENCE OF STRUCTURE ACROSS LLM DEPTH

Linear decodability measures what a single patch carries. To measure what the representation carries about the object as a whole, a linear head is trained on the point tokens to predict the part label of every point of a held-out ShapeNet-Part object, and accuracy is reported as mean intersection over union across the annotated parts. Table 12(b) tracks this quantity from the input tokens through the LLM depth.

The two models carry part structure at different places. With an aligned encoder it is strongest at the tokens that enter the language model and decays with depth, since the encoder has already organised the object and the later layers work on language. With AFPT the tokens start lower and part structure builds up inside the language model, peaking in the upper middle layers, which is where Figure 5 shows the text tokens beginning to read the point block.

G.3 DEPENDENCE ON THE INPUT AND SELECTED REGION

For each item the likelihood of the gold answer is computed under the object's own point cloud and under the cloud of a random other object; the statistic is the share of items on which the own cloud is likelier. Captions use 200 objects of the Objaverse captioning set, and the fine-grained questions are part-colour questions. Table 13 reports the result: both models depend heavily on the point cloud when captioning and are at chance on the fine-grained questions. The same part-colour questions are answered from the whole cloud, from only the queried region, and from the same region of a different object, scored by the likelihood of the gold answer, on 200 questions about procedurally built objects whose part colours are known exactly. Both models gain about thirteen points when the queried region is isolated and fall back to chance when the region comes from another object, so the evidence is usable once it is separated from the rest of the cloud.

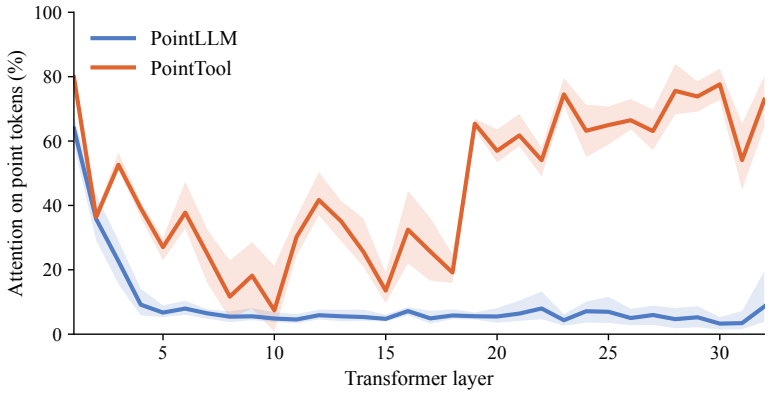


Figure 5: Share of the text tokens' attention that falls on the point block, layer by layer. Lines are means over the 28 objects and bands span the minimum and maximum.

Table 13: Dependence on the point input and selected region. Upper values are the percentage of items for which the matched cloud gives higher gold-answer likelihood; lower values are mean gold-answer likelihoods (%) under the indicated input restriction.

Setting	n	PointLLM-7B	POINTTOOL-7B
Cloud sensitivity			
Captions	200	93.5	95.5
Fine-grained questions	200	48.4	48.7
Restricting the input			
Whole cloud	200	53.5	55.1
Queried region only	200	**67.0**	**68.4**
Same region, other object	200	48.0	48.6

G.4 TEXT-TO-POINT ATTENTION ACROSS DEPTH

Inspired by the text-to-point attention visualisation of Tang et al. (2026), we extend the measurement across all layers: for the caption prompt we average the attention from every text token onto the point tokens over heads and text positions at each layer. The measurement covers 28 Objaverse objects drawn from eleven categories.

Figure 5 shows that the two models diverge after the first few layers. With an aligned encoder the share settles below ten percent from layer four onwards and stays there, so the language model reads the point block only briefly and relies on what the encoder has already summarised. With AFPT the share falls in the early layers, then rises sharply through the middle and late layers, reaching 72.8 percent at layer 32 (versus 8.6 percent for PointLLM). Figure 6 shows the same measurement object by object. Attention allocation is diagnostic rather than causal evidence, but its depth-wise trend is consistent with aggregation being transferred into the LLM.

G.5 ATTENTION MAPS

Figure 6 shows the last layer for individual objects. The pattern holds across categories: POINTTOOL raises the share to between 64 and 80 percent on every object, while PointLLM stays below twenty percent throughout.

H EFFICIENCY AND RUNTIME

Table 14 compares the visual front-end size and measured single-object inference cost under a shared hardware setting.

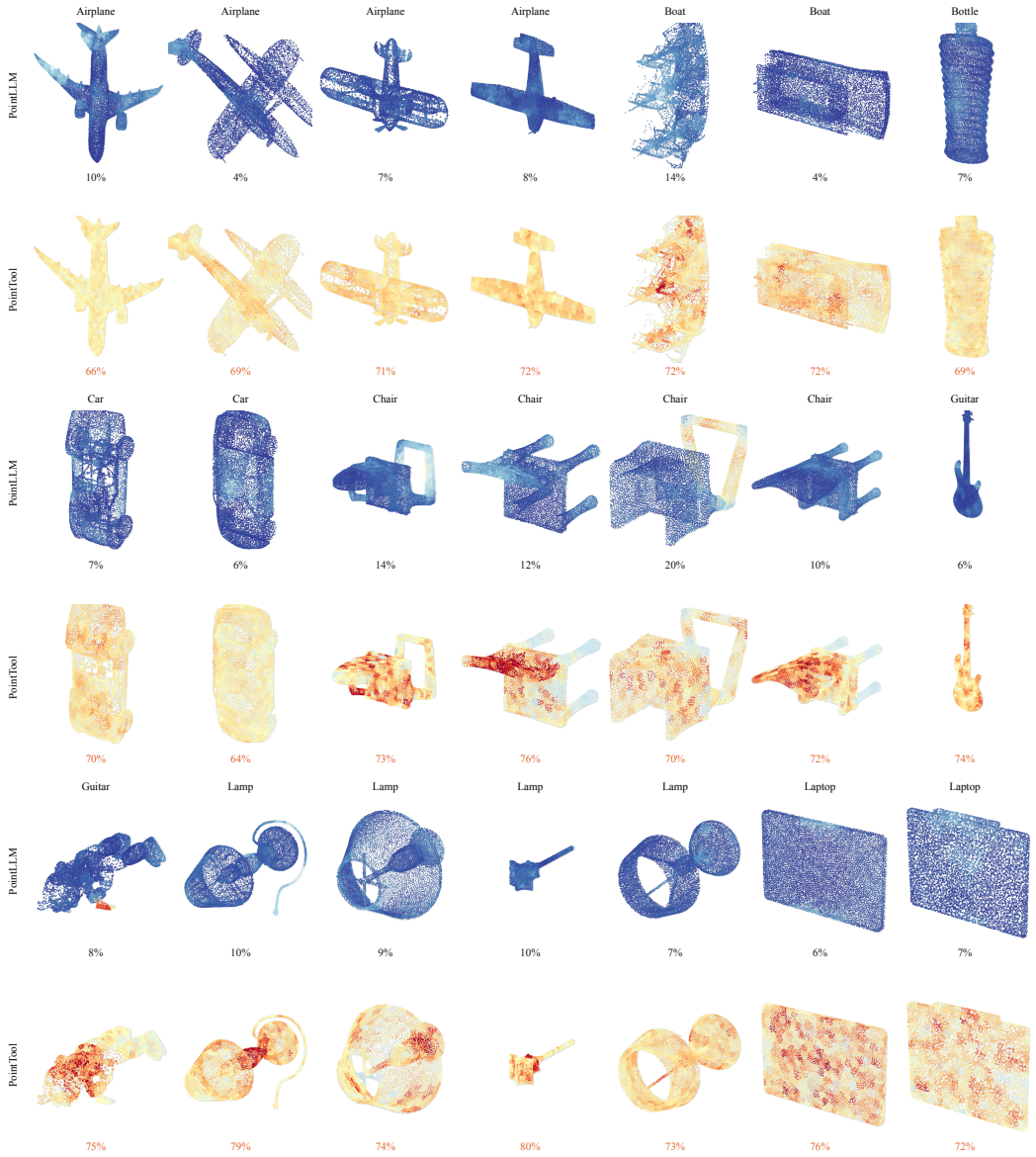


Figure 6: Last-layer attention onto the point tokens for 21 held-out objects, PointLLM above and POINTTOOL below in each pair, on one colour scale from 0 to 1.2 times the uniform value. The number under each object is the share of the text tokens' attention that falls on the point block.

Table 14: Inference cost on 8K-point inputs, one H20 GPU, batch size 1. Latency and throughput are measured independently under the same harness; visual parameters count the encoder or tokenizer only. Best in bold, second underlined.

Model	Visual params (M) ↓	Latency (ms) ↓	Throughput (obj/s) ↑
PointLLM-7B	22.1	318	3.0
SAGE-7B	**4.2**	<u>301</u>	<u>3.2</u>
POINTTOOL-7B	<u>8.0</u>	**296**	**3.3**

The tokenizer of POINTTOOL-7B has 8.0M parameters against the 22.1M of the Point-BERT encoder of PointLLM-7B. End-to-end latency is comparable across the three systems in this single-GPU setting; the clearest efficiency gain is therefore the smaller visual front end rather than the small timing differences.