ANYPOINT: DYNAMIC EVALUATION FOR POINT-CLOUD MULTIMODAL LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Point-cloud multimodal large language models (MLLMs) are advancing rapidly in 3D scene understanding, yet their evaluation relies almost exclusively on static benchmarks that quickly saturate as models improve and fail to locate individual model weaknesses. Transitioning to dynamic evaluation requires an expansive pool of physically plausible 3D scenes with verified ground-truth answers, a dual challenge that neither manual annotation nor pure language model generation can satisfy. To resolve this dilemma, we present ANYPOINT, a dynamic evaluation framework that couples programmatic 3D scene generation with adaptive failure discovery for point-cloud MLLMs. ANYPOINT decouples scene creation into semantic layout generation via a language model and spatial coordinate solving via a deterministic constraint solver, producing physically valid 3D scenes alongside grounded 3D scene graphs. To avert saturation, evaluation tasks are derived compositionally as programs over each graph, allowing ground-truth answers to be computed directly from the geometry without human annotation. Within this generated task space, a budget-aware dynamic evaluation algorithm adaptively probes each model's failure boundary by balancing error exploitation and novelty exploration. Across six representative point-cloud MLLMs, ANYPOINT raises the failure discovery rate within a 1,000-query budget from 65.80% under uniform sampling to 83.73%. The recovered failure profiles reveal distinct model-specific weaknesses alongside shared capability bottlenecks that single aggregate scores obscure. Finally, we demonstrate that discovered failures can drive targeted model improvement, while preserving task diversity is essential for overall generalization.

1 INTRODUCTION

Point-cloud multimodal large language models (MLLMs) are advancing rapidly in 3D scene understanding and spatial reasoning (Xu et al., 2024; Qi et al., 2024a; Han et al., 2024; Tang et al., 2024; Su et al., 2026; Hong et al., 2023; Chen et al., 2024; Qi et al., 2024b; Huang et al., 2025b; Ahmed et al., 2025; Chen et al., 2026), yet their evaluation has emerged as a critical bottleneck for continued progress. Despite rapid advances in model capabilities, standard evaluation continues to rely almost exclusively on static benchmark suites (Azuma et al., 2022; Ma et al., 2022; Zhang et al., 2025a), where models are repeatedly evaluated against the same frozen set of questions. This static paradigm suffers from two fundamental limitations: such benchmarks rapidly saturate as models improve (Kiela et al., 2021), and under a finite evaluation budget, testing models against fixed questions squanders queries on already-mastered capabilities while failing to locate model-specific weaknesses. Overcoming these limitations requires transitioning to dynamic evaluation for point-cloud MLLMs, synthesizing verified evaluation tasks at scale to avert benchmark saturation, while adaptively concentrating query budgets along each model's capability frontier.

Realizing such dynamic evaluation, however, presents three key technical challenges. First, grounding evaluation tasks in physically plausible 3D scenes at scale faces an intrinsic impasse: real-world scans are scarce and costly to collect; generative language models propose rich 3D scene layouts (Feng et al., 2023; Yang et al., 2024; Huang et al., 2025a) but lack continuous spatial grounding, frequently causing object collisions and floating artifacts; and procedural engines (Deitke et al., 2022) enforce physical validity only through rigid rules that restrict diversity. Second, deriving evaluation tasks with certified ground truth is difficult without human intervention: prompting language models risks hallucinated spatial assertions, while manual annotation cannot scale to large task volumes. Third,

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.

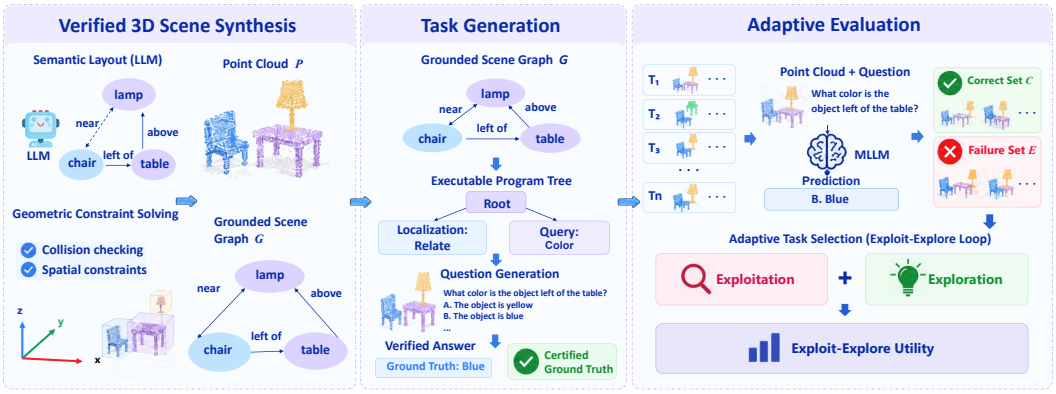

Figure 1: Overview of the ANYPOINT framework. Left (§2.2): **Verified 3D Scene Synthesis** grounds LLM-proposed layouts into physically valid 3D scenes and scene graphs G via deterministic constraint solving. Center (§2.3): **Programmatic Task Generation** executes compositional programs over G to certify ground-truth QA pairs directly from geometry. Right (§2.4): **Budget-Aware Dynamic Evaluation** adaptively probes model failure boundaries via an exploit–explore utility loop.

discovering failure boundaries under a constrained query budget requires navigating an immense combinatorial task space: uniform testing squanders queries on already-mastered capabilities, while each model's vulnerabilities are sparse, varied, and unknown prior to testing.

To address these challenges, we introduce ANYPOINT, a dynamic evaluation framework that couples scalable 3D task generation with budget-aware failure discovery for point-cloud MLLMs (Figure 1). ANYPOINT implements a principled division of labor between generative language models and deterministic programs: first, for scene creation, a language model proposes a semantic scene graph of spatial relationships, which a deterministic constraint solver grounds into collision-free 3D coordinates to construct a physically valid scene and grounded 3D scene graph; second, for task generation, questions are formalized as compositional programs over each grounded scene graph, allowing certified ground-truth answers to be computed directly by executing on the underlying geometry without manual annotation. By deriving tasks through program composition rather than manual enumeration, this framework yields an effectively unbounded task space with certified labels, averting benchmark saturation.

Operating over this synthesized task space, ANYPOINT dynamically steers a constrained query budget toward each model's failure boundary via an exploit–explore utility. By encoding candidate tasks into joint representations spanning scene graph topology, question semantics, and answer targets, the utility prioritizes queries that exploit known failure clusters while exploring untested regions distant from mastered capabilities, enabling real-time probing over million-scale candidate pools within fixed testing budgets.

Extensive experiments across six representative point-cloud MLLMs demonstrate that ANYPOINT raises failure discovery within a $1{,}000$-query budget from 65.80% under uniform sampling to 83.73%, substantially outperforming existing adaptive testing strategies. An additive gain decomposition confirms that this advantage is overwhelmingly driven by isolating challenging boundary instances within task types rather than coarse category reallocation, verifying true instance-level diagnostic precision. Furthermore, the recovered failure profiles reveal distinct model-specific weaknesses alongside shared capability bottlenecks that single aggregate scores obscure. Finally, we show that discovered failures can effectively guide targeted model improvement, where training on a balanced mixture of error-guided and uniform data outperforms uniform retrieval, while preserving task diversity is essential to prevent catastrophic forgetting.

2 THE ANYPOINT FRAMEWORK

Given a catalog of 3D object assets, ANYPOINT automatically synthesizes evaluation tasks with certified ground truth at scale and adaptively allocates testing queries to locate failure boundaries (Figure 1). After formalizing the evaluation objective (§2.1), our framework operates across three core

stages: (1) decoupled 3D scene synthesis (§2.2); (2) programmatic task generation over grounded scene graphs (§2.3); and (3) budget-aware dynamic evaluation via an exploit–explore utility (§2.4).

2.1 Problem Formulation

We consider the problem of evaluating a point-cloud multimodal large language model $\mathcal{M}$ on 3D scene understanding under a constrained query budget B.

Definition 1 (Budget-Constrained Dynamic Evaluation Problem)**.** *Let an evaluation task be formalized as a tuple $t = (\mathbf{P}, Q, y^\star)$, where $\mathbf{P} \in \mathbb{R}^{N\times 6}$ denotes an input scene point cloud of N points carrying continuous 3D coordinates and normalized RGB color channels, Q is a natural-language question, and $y^\star \in \mathcal{A}$ is the certified ground-truth answer within a discrete answer space $\mathcal{A}$. Given point cloud $\mathbf{P}$ and question Q, model $\mathcal{M}$ predicts answer $\hat{y} = \mathcal{M}(\mathbf{P}, Q)$; the task is resolved correctly if prediction matches ground truth ($\hat{y} = y^\star$) and recorded as a failure otherwise. Given an expansive candidate task pool $\mathcal{P}$ and a finite query budget B, dynamic evaluation adaptively queries a subset of tasks $\mathcal{S} \subseteq \mathcal{P}$ subject to $|\mathcal{S}| \le B$. The resulting evaluation history $H = C \cup E$ partitions into disjoint sets of correctly resolved tasks C and discovered failures E, satisfying $C \cap E = \emptyset$ and $|C| + |E| \le B$. The dynamic evaluation objective is to actively maximize the discovery of informative model failures within budget B:*

$$\max_{\mathcal{S}\subseteq\mathcal{P},\, |\mathcal{S}|\le B} \; |E(\mathcal{S})|, \tag{1}$$

thereby efficiently tracing the capability boundary of $\mathcal{M}$.

Solving the dynamic evaluation problem in Definition 1 requires addressing two interdependent demands: synthesizing candidate task pool $\mathcal{P}$ at scale with verified ground truth (§2.2 – §2.3), and adaptively steering queries along each model's failure boundary under budget B (§2.4).

2.2 Decoupled 3D Scene Synthesis

To generate 3D scenes at scale for the candidate task pool $\mathcal{P}$, AnyPoint decouples scene synthesis into semantic layout proposal and deterministic geometric solving.

Semantic Layout Proposal. Given a sampled subset of object categories from an asset catalog (curated from TexVerse (Zhang et al., 2025c)), a language model (DeepSeek-V4-Pro) proposes a layout structured as a semantic scene graph of spatial relationships. Each node in this graph specifies an object category, an initial canonical rotation, and one of five discrete scale tiers, with the full generation prompt and scale definitions documented in Appendix G. Each directed edge represents a spatial relation triplet linking a subject object to a reference object, spanning directional, vertical, and proximity relations illustrated in Figure 1 (left) and formalized in Appendix E. To avoid querying the language model redundantly for every scene, AnyPoint prompts the model with generic category names rather than specific 3D object assets. This separates the abstract layout topology from concrete physical assets, allowing each generated layout to serve as a reusable template that is populated with diverse asset combinations to multiply scene variety.

Geometric Constraint Solving. While the semantic scene graph specifies spatial relationships, it lacks continuous coordinates and physical boundaries, risking collisions and floating artifacts. To ground the layout into physically valid 3D space, our deterministic solver executes three stages: (1) mapping discrete scale tiers to canonical physical dimensions to sample realistic bounding boxes; (2) topologically ordering placements along a directed acyclic dependency graph so reference anchors precede dependent entities; and (3) decoupling placement into vertical elevation (resolving contact and ground support) and horizontal coordinates within directional constraint cones, accepting coordinates only under collision-free clearance (formalized in Algorithm 1, with automated physical and answerability audits in Appendix D).

Point-Cloud and Scene Graph Generation. With 3D bounding boxes and orientations resolved, corresponding object assets are retrieved from the curated TexVerse catalog (Zhang et al., 2025c), scaled to instantiated dimensions, and placed at their target coordinates (filtering in Appendix C). Surface points are uniformly sampled with normalized RGB colors, producing scene point cloud $\mathbf{P} \in \mathbb{R}^{N\times 6}$. Concurrently, the scene is structured as a grounded 3D scene graph $G = (V, \mathcal{E})$: each vertex $v \in V$ carries an object instance's category, bounding box, spatial center, and visual attributes (color and shape); each directed edge $(u, r, v) \in \mathcal{E}$ asserts a verified geometric relation r,

using vocabularies defined in Appendix E. This delivers both the input point cloud $\mathbf{P}$ and structural ground-truth graph G for task generation (§2.3).

2.3 PROGRAMMATIC TASK GENERATION OVER 3D SCENE GRAPHS

Given the grounded 3D scene graph $G = (V, \mathcal{E})$, handcrafted templates restrict diversity while open-ended language prompting introduces spatial hallucinations. Intuitively, spatial questioning in a 3D scene decomposes into two fundamental steps: first identifying target entities through spatial relationships, and then querying their specific properties or counts. Formally, ANYPOINT represents each evaluation task as an executable program tree $q = (\pi, \ell)$ over the grounded 3D scene graph $G = (V, \mathcal{E})$ (Johnson et al., 2017; Hudson & Manning, 2019), comprising a recursive localization selector ℓ and a query operator π. Answers are certified by construction via deterministic execution on the underlying geometry:

$$S = [\![\ell]\!]_G \subseteq V, \qquad y^\star = [\![\pi]\!]_G(S) \in \mathcal{A}, \tag{2}$$

where evaluating ℓ first resolves to target entity set $S \subseteq V$ (e.g., all chairs adjacent to a table), over which π extracts the target property or cardinality (e.g., identity, color, or count) to yield certified ground truth $y^\star \in \mathcal{A}$.

Localization selectors ℓ are recursively sampled via a typed grammar:

$$\ell \rightarrow V \mid V_c \mid \textsc{Extremum}_m(\ell) \mid \textsc{Relate}_r(\ell) \mid \textsc{Region}_r(\ell) \mid \textsc{Filter}_{a=\text{val}}(\ell), \tag{3}$$

spanning category filters V_c, metric extrema $\textsc{Extremum}_m$ (volume, distance), spatial relations $\textsc{Relate}_r$, directional regions $\textsc{Region}_r$, and attribute filters $\textsc{Filter}_a$, followed by query operator π extracting identities, attributes, cardinalities (counting), or directions (semantics in Appendix F).

To guarantee unambiguous evaluation, candidate programs are executed over scene graph G, retaining only tasks that resolve to unique, well-defined ground-truth answers (certification criteria detailed in Appendix F). Deterministic templates then translate validated program trees into fluent natural-language questions Q, with multiple-choice distractors sampled from other entities situated in the same scene. Crucially, sampling distractors strictly from co-occurring in-scene objects prevents language-only shortcuts: since candidate entities physically exist in the scene, questions cannot be resolved via text priors alone without inspecting 3D point-cloud geometry. We formalize the complete task synthesis and verification procedure in Algorithm 2 (Appendix F). Every synthesized task $t = (\mathbf{P}, Q, y^\star)$ is thus structurally characterized along three canonical axes: its spatial localization cue ℓ, query type π, and reasoning depth $|\ell|$ (qualitative examples in Figure 5, Appendix H).

2.4 BUDGET-AWARE DYNAMIC EVALUATION

Given candidate task pool $\mathcal{P}$ (Sections 2.2 – 2.3), our dynamic evaluation algorithm adaptively concentrates testing queries along each model's failure boundary under budget B (Figure 1, right).

Joint Task Representation. To quantify affinities across evaluation items, each task $t = (\mathbf{P}, Q, y^\star)$ is encoded into a joint three-component vector representation:

$$\mathbf{v}_t = \frac{1}{\sqrt{3}}\big[\,\mathbf{e}_G\,;\,\mathbf{e}_Q\,;\,\mathbf{e}_y\,\big] \in \mathbb{R}^{3d}, \tag{4}$$

where $\mathbf{e}_G = f_{\text{emb}}(T_G)$ encodes a serialized scene graph T_G of situated categories and spatial relations, while $\mathbf{e}_Q = f_{\text{emb}}(Q)$ and $\mathbf{e}_y = f_{\text{emb}}(y^\star)$ encode question intent and certified answers (using Qwen3-Embedding-8B, component ablation in Appendix I). This unified space distinguishes tasks sharing geometry but querying different relations, or identical questions situated in distinct scenes. Crucially, this joint representation is used strictly by the external dynamic scheduler to navigate candidate diversity; evaluated point-cloud MLLMs receive only the raw point cloud $\mathbf{P}$ and natural-language question Q, with zero access to symbolic graphs, embeddings, or ground-truth labels.

Exploit–Explore Utility Function. To actively probe capability boundaries under budget B, each candidate task $t \in \mathcal{P}$ is scored by balancing semantic proximity to discovered failures E (exploitation) against distance from resolved successes C (exploration). Using the non-negative similarity $s(\mathbf{v}, \mathbf{v}') = \max(0, \langle \mathbf{v}, \mathbf{v}' \rangle) \in [0, 1]$, each candidate is evaluated via:

$$U(t) = \underbrace{\Big(\max_{e \in E} s(\mathbf{v}_t, \mathbf{v}_e)\Big)^{1-\lambda}}_{\text{exploit}} \cdot \underbrace{\Big(1 - \max_{c \in C} s(\mathbf{v}_t, \mathbf{v}_c)\Big)^{\lambda}}_{\text{explore}}, \tag{5}$$

where $\lambda \in [0, 1]$ governs the trade-off. At cold start before any failure is observed ($E = \emptyset$), the exploit term defaults to zero, ranking initial queries purely by exploration margin ($1 - \max_{c \in C} s(\mathbf{v}_t, \mathbf{v}_c)$). The multiplicative formulation ensures a candidate is prioritized only when it is both error-proximal and distinct from solved capabilities.

Dynamic Evaluation Loop. Algorithm 3 (Appendix N) details the complete iterative probing procedure. Evaluation begins with a uniform cold-start batch of K queries to initialize C and E. In each round, the top-K scoring candidates from remaining pool $R \subseteq \mathcal{P}$ are evaluated on model $\mathcal{M}$, and empirical outcomes are partitioned into new successes $\mathcal{S}_C$ and errors $\mathcal{S}_E$. To scale scoring across million-scale candidate pools without recomputing against accumulated history, we maintain running maximum similarity vectors $\mathbf{a}_R, \mathbf{b}_R \in [0, 1]^{|R|}$ updated incrementally via matrix-vector products with each observed batch, keeping the per-iteration complexity $\mathcal{O}(|R|K)$ independent of the growing evaluation history (formalized in Eq. 7 and Algorithm 3, Appendix N). This batch evaluation and update cycle repeats until the cumulative testing budget B is reached, delivering the final partition of resolved capabilities C and discovered failures E.

3 EXPERIMENTS

By coupling programmatic task generation (§2.2 – 2.3) with budget-aware dynamic steering (§2.4), ANYPOINT enables active failure-boundary discovery for point-cloud MLLMs under a constrained query budget rather than passive scoring on static benchmarks. To assess whether this dynamic paradigm reliably exposes point-cloud MLLM weaknesses and provides actionable diagnostic value, we evaluate six representative models across four central research questions: **RQ1 (§3.2)** examines whether dynamic evaluation uncovers more failures than static benchmarks and what drives its performance gains; **RQ2 (§3.3)** analyzes how the exploit–explore trade-off governs testing efficiency and search stability; **RQ3 (§3.4)** diagnoses shared bottlenecks and model-specific weaknesses across structural task dimensions; and **RQ4 (§3.5)** investigates whether discovered model failures can effectively guide fine-tuning and how well the gains generalize.

3.1 EXPERIMENTAL SETUP

Candidate Task Pool. We leverage our programmatic generator (§2.2 – §2.3) to synthesize $2{,}000{,}000$ verified tasks, partitioned into two strictly disjoint pools of $1{,}000{,}000$ items each for dynamic probing and training retrieval. The pools are generated from non-overlapping random layout seeds, ensuring tasks across pools never share underlying 3D scene point clouds or scene graphs. To enable standardized assessment across decoding styles, each task is framed as a multiple-choice question with two to four options, with incorrect distractors sampled from other entities situated in the same scene (Appendix F). Crucially, the dynamic selection policy operates in complete blindness to option cardinality: task representations $\mathbf{v}_t$ encode only scene geometry and question semantics without distractor tokens or option counts, ensuring that ANYPOINT and all baselines query identical underlying option-cardinality distributions. Each task couples a question with an $8{,}192$-point colored scene point cloud sampled from $20{,}438$ curated object assets from TexVerse (Zhang et al., 2025c), pre-computed using Qwen3-Embedding-8B (Zhang et al., 2025b) (component ablation in Appendix I).

Evaluated Models and Baselines. We evaluate six point-cloud MLLMs covering diverse visual encoding and multimodal alignment paradigms: PointLLM (Xu et al., 2024), OneLLM (Han et al., 2024), ShapeLLM (Qi et al., 2024a), MiniGPT-3D (Tang et al., 2024), PointAlign (Su et al., 2026), and GreenPLM (Zeng et al., 2022). Model details are provided in Appendix A, and evaluation prompts in Appendix G. To ensure standardized evaluation across architectures, all models are queried under a unified generation configuration with temperature 1.0, top-p 0.95, top-k 50, and a maximum limit of $2{,}048$ tokens. Against ANYPOINT, we compare static uniform sampling (*Uniform Sampling*, under identical query budget) and three competitive adaptive testing strategies re-implemented on our candidate pool: ACD (Lu et al., 2025), AutoBencher (LI et al., 2025), and SEA (Song et al., 2025). Baseline details are provided in Appendix B.

Evaluation Protocol. The dynamic evaluation loop probes point-cloud MLLMs by adaptively drawing queries from the candidate pool under a constrained testing budget of $1{,}000$ queries, evaluated in batches of 10 with default trade-off parameter $\lambda = 0.2$. We investigate the impact of this trade-off through an ablation study across $\lambda \in [0, 1]$ in §3.3. Since point-cloud MLLMs generate free-

Table 1: Failure rate (%) across six point-cloud MLLMs under a constrained query budget ($B = 1{,}000$, $K = 10$). Higher is better, indicating more efficient failure boundary discovery. The best result in each column is shown in **bold**; rows corresponding to ANYPOINT are shaded.

Method	Failure rate within budget (↑)						Avg.
	PointLLM	OneLLM	ShapeLLM	GreenPLM	MiniGPT-3D	PointAlign	
Uniform Sampling (static)	71.80	56.90	66.70	64.70	67.20	67.50	65.80
ACD (Lu et al., 2025)	70.70	63.20	66.00	66.50	68.60	69.40	67.40
AutoBencher (LI et al., 2025)	73.20	68.20	73.50	**86.30**	63.30	65.30	71.63
SEA (Song et al., 2025)	77.30	63.70	74.60	72.40	73.30	74.90	72.70
ANYPOINT	**86.60**	**79.70**	**83.50**	85.70	**84.20**	**82.70**	**83.73**

form textual responses, a language model serves as a parsing judge to extract the predicted option letter, followed by deterministic string matching against certified ground truth (Appendix L). Our primary evaluation metric is the *failure rate within budget*: the percentage of tested queries answered incorrectly, where a higher rate reflects more efficient boundary discovery under query constraints. Metric details are provided in Appendix L.

3.2 RQ1: DYNAMIC EVALUATION DISCOVERS FAILURE BOUNDARIES ACROSS POINT-CLOUD MLLMS

ANYPOINT Consistently Uncovers More Failures Across Models. To evaluate whether dynamic evaluation exposes vulnerabilities overlooked by static testing (**RQ1**), Table 1 compares ANYPOINT against static uniform sampling and three adaptive strategies (ACD (Lu et al., 2025), AutoBencher (LI et al., 2025), SEA (Song et al., 2025)) across six point-cloud MLLMs under budget $B = 1{,}000$. Static evaluation uncovers failures on only 65.80% of queries, squandering testing capacity on already-mastered capabilities. In contrast, ANYPOINT elevates average failure discovery to 83.73%, an advantage of 17.93 percentage points. Dynamic probing exposes severe hidden weaknesses in seemingly robust models: on OneLLM, whose static failure rate is lowest (56.90%), ANYPOINT reaches 79.70% (+22.80 percentage points). Across five of six models, ANYPOINT leads by decisive margins, while prior adaptive strategies trail by 11.0–16.3 points due to unmodeled 3D spatial manifolds (except GreenPLM, where AutoBencher concentrates into a collapsed attribute bucket). An additive gain decomposition confirms that this advantage is overwhelmingly driven by isolating hard boundary instances within task types (16.51 percentage points, $> 92\%$ of total gain) rather than coarse category reallocation (1.43 points; Appendix K). Furthermore, component ablations (Appendix I) confirm this advantage does not rely on privileged answer tokens: omitting answer embeddings ($\mathbf{e}_y$) still achieves 79.30% failure discovery, remaining well ahead of all baselines (67.40%–72.70%).

ANYPOINT's Advantage Widens Monotonically as Query Budget Scales. A key operational question is whether dynamic evaluation maintains its advantage across different testing horizons. Figure 2 tracks cumulative failure discovery across the query budget sweep from $B = 100$ to $B = 1{,}000$. While static uniform sampling remains essentially flat across the entire budget range (reflecting memoryless querying that repeatedly squanders testing capacity), ANYPOINT's advantage emerges immediately at $B = 100$ (11.34 percentage points) and widens monotonically to 17.93 percentage points at $B = 1{,}000$, confirming that accumulated error feedback continuously sharpens query selection.

3.3 RQ2: BALANCING EXPLOITATION AND EXPLORATION MAXIMIZES FAILURE DISCOVERY AND STABILITY

Having established that ANYPOINT uncovers failure boundaries effectively under default $\lambda = 0.2$ (**RQ1**), we examine how the exploit–explore trade-off governs testing efficiency and stability (**RQ2**). Figure 3 illustrates both dimensions across exploration weights $\lambda \in [0, 1]$: the blue curve plots failure discovery rate against λ (left axis), while lavender bars quantify search trajectory instability across random initializations (right axis).

Balanced Exploration Maximizes Failure Discovery Across Models. As shown in Figure 3 (blue curve), both extremes of the exploration spectrum forfeit the adaptive advantage of dynamic evaluation. Pure exploitation ($\lambda = 0$) reaches only 70.78%, as the absence of exploratory repulsion causes the

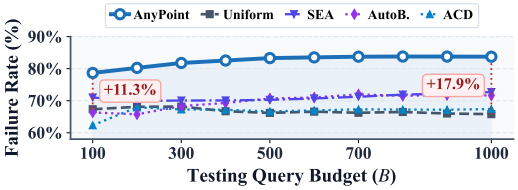

Figure 2: Cumulative failure rate as query budget sweeps from $B = 100$ to $B = 1{,}000$ across six point-cloud MLLMs (RQ1). ANYPOINT maintains a monotonic lead of 11.34 to 17.93 percentage points over uniform sampling.

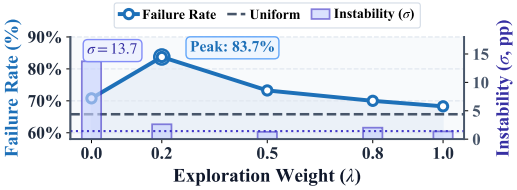

Figure 3: Failure discovery rate (left axis, blue) and search instability (right axis, lavender bars) across exploration weights $\lambda \in [0, 1]$ (RQ2). Peak discovery is attained at balanced $\lambda = 0.2$ (83.73%), where exploratory repulsion suppresses cold-start instability.

search to commit prematurely to narrow error clusters and repeatedly issue semantically redundant questions. Conversely, pure exploration ($\lambda = 1$) achieves 68.28%, barely exceeding uniform sampling (65.80%), as it actively pursues novelty without utilizing the semantic representations of uncovered failures. Peak failure discovery emerges consistently at balanced $\lambda = 0.2$ (83.73%), after which performance degrades smoothly (73.25% at $\lambda = 0.5$ and 70.00% at $\lambda = 0.8$). This confirms that moderate exploratory repulsion is essential to steer error-guided search into fresh, unmapped failure territories without diffusing testing effort into unguided exploration.

Exploratory Repulsion Eliminates Cold-Start Search Degeneracy. Beyond aggregate failure discovery, the trade-off parameter fundamentally governs search trajectory stability. As documented in our random-seed sensitivity study on POINTLLM (Appendix J) and depicted by the lavender bars in Figure 3, pure exploitation ($\lambda = 0$) exhibits severe instability across random initializations ($\sigma = 13.71$ percentage points, ranging from 66.70% to 92.20%), as trajectories without exploration penalties deterministically overfit to stochastic cold-start errors. In contrast, incorporating exploratory repulsion ($\lambda \in [0.2, 1.0]$) eliminates this path dependence, suppressing seed-to-seed variance to between 1.29 and 2.62 percentage points, on par with baseline sampling noise ($\sigma = 1.42$). The exploration term is therefore indispensable not merely for sample efficiency, but for ensuring reproducible, robust capability profiling.

3.4 RQ3: DYNAMIC EVALUATION DIAGNOSES SHARED BOTTLENECKS AND MODEL-SPECIFIC WEAKNESSES

Beyond discovering aggregate failure boundaries, dynamic evaluation provides actionable diagnostic value: by balancing localized failure exploitation with exploratory repulsion ($\lambda = 0.2$), ANYPOINT actively traces diverse vulnerabilities without collapsing into duplicate clusters, transforming the final distribution of queries across task dimensions into an empirical stress map of each model's capability limits (**RQ3**). Figure 4 visualizes these diagnostic query allocations across the three canonical task axes established in Section 2.3 (localization cues, query types, and reasoning depth). Rather than exhibiting uniform deficits, the evaluated models reveal a stark architectural dichotomy: models with paired 3D-text pre-training share a pervasive bottleneck in multi-object spatial counting, whereas models relying on unpaired text-space transfer fail fundamentally on localized attribute binding.

Multi-Object Counting Pervasively Limits Point-Cloud MLLMs Across Architectures. For five of the six models (PointLLM, OneLLM, ShapeLLM, MiniGPT-3D, PointAlign), dynamic probing converges overwhelmingly onto quantity-related tasks (Figure 4b). Object counting alone absorbs between 51.9% and 76.2% of the total 1,000-query testing budget, with all counting queries combined (object counting, region counting, and attribute cardinality) accounting for 71.1% to 91.5% of allocated queries, mirroring severe failure rates (81.2%–87.3%). This exposes a fundamental architectural bottleneck: models project continuous 3D scenes into fixed visual tokens without explicit 3D bounding-box isolation or instance segmentation priors. In multi-object point clouds, we hypothesize that cross-attention suffers from spatial attention diffusion across dense tokens, recognizing object presence while failing to resolve discrete instance cardinalities.

Unpaired Modality Transfer Decouples Geometric Localization from Visual Attributes. In contrast to paired 3D-text models, GreenPLM relies on text-space alignment, producing a sharply divergent diagnostic profile: counting consumes merely 1.9% of its testing budget, while attribute

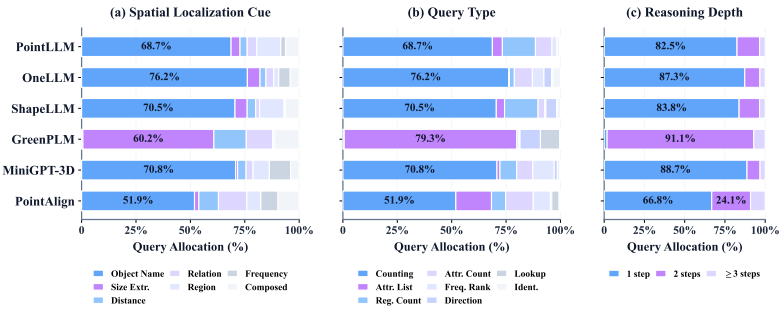

Figure 4: Diagnostic query allocation of ANYPOINT ($\lambda = 0.2$) across the three structural task axes (100% stacked bars per model). Five models heavily concentrate testing budget on object naming, counting, and single-step reasoning, whereas GreenPLM diverges sharply, drawing budget toward size extrema, attribute listing, and two-step reasoning.

listing absorbs 793 queries (79.3%, Figure 4b, 91.9% failure rate), with localization concentrating on size-based extrema (60.2%, Figure 4a) rather than named entities (0.6%). This reveals an intrinsic trade-off of unpaired transfer: while retaining textual relational priors, lacking paired 3D coordinate supervision prevents anchoring continuous geometric features (such as volume extrema) to localized visual attributes, causing performance collapse on fine-grained spatial attribute binding.

Compositional Reasoning Flattens into an Early Perceptual Floor Effect. Analyzing query allocations along the reasoning-step axis (Figure 4c) mirrors these underlying failure modes: counting-limited models concentrate queries on 1-step depth (66.8%–88.7%), while GreenPLM concentrates 91.1% on 2-step depth (attribute listing over extrema). Despite these sharp differences, empirical failure rates across reasoning depths remain remarkably uniform at 64.07% (1-step), 67.15% (2-step), and 66.01% ($\geq$3-step), near the chance floor ($\sim$ 34%). This reflects a perceptual floor effect: in multi-object scenes, current point-cloud MLLMs encounter perceptual bottlenecks even on basic 1-step localization, leaving minimal headroom for multi-hop composition to further degrade performance. These structured failure distributions confirm that ANYPOINT uncovers genuine architectural bottlenecks rather than arbitrary testing noise.

3.5 RQ4: FAILURE-GUIDED TRAINING IMPROVES PERFORMANCE BUT DEMANDS TASK DIVERSITY

To evaluate whether discovered model failures serve as an actionable learning signal to improve point-cloud MLLMs rather than merely ephemeral testing artifacts (**RQ4**), we fine-tune POINTLLM through three sequential rounds of *diagnose – retrieve – train – evaluate*. Each round uses 1,000 diagnostic queries ($\lambda = 0.20$) on the evaluation pool to retrieve 10,000 samples from a disjoint 1,000,000-task training pool via task embedding similarity. A mixture coefficient $\alpha \in [0, 1]$ balances error-guided items against uniform sampling. Checkpoints are evaluated on 2,000 held-out synthetic tasks alongside the real-world scanned benchmark GLUE3D MCQ (256 questions, 25% chance floor; protocol in Appendix M).

Balanced Failure Guidance Improves Performance While Pure Exploitation Induces Collapse. As reported in Table 2 (and visualized in Figure 6, Appendix M), fine-tuning POINTLLM on a balanced mixture ($\alpha = 0.50$) lifts accuracy from 26.65% (Round 0) to a peak of 60.35% at Round 3, outperforming purely uniform retrieval ($\alpha = 0.00$, 58.65%). Conversely, pursuing pure error retrieval ($\alpha = 1.00$) causes severe performance degradation, collapsing to 44.55%. Disaggregating accuracy across individual query types reveals the underlying mechanism: pure error training successfully achieves targeted gains on the model's dominant weakness (object counting accuracy rises from 29.3% to 56.1%), but catastrophically degrades unrepresented capabilities (attribute lookup drops from 90.7% to 41.6%, and attribute listing drops from 94.2% to 50.2%). As diagnostic errors concentrate heavily on multi-object counting, retrieving exclusively on error similarity starves the training distribution of broader diversity. Symmetrically, pure exploitation fails in dynamic testing ($\lambda = 0.0$) and training data curation ($\alpha = 1.00$), establishing that maintaining task diversity is essential to prevent catastrophic forgetting (formalized in Table 9, Appendix M).

Table 2: Dynamic fine-tuning and real-world transfer on POINTLLM across sequential training rounds. Top: In-distribution accuracy (%) on 2,000 unseen synthetic tasks across mixture ratios α. Bottom: Out-of-distribution transfer to real-world scanned scenes on GLUE3D MCQ (256 questions, 25% chance floor). The best result in each column is **bold**; balanced training ($\alpha = 0.50$) is shaded.

Configuration / Category	Initial	Round 1	Round 2	Round 3	Gain (Δ_{R1})
In-Distribution Synthetic Held-out (2,000 tasks)					
$\alpha = 0.00$ (uniform retrieval)	26.65	59.50	58.40	58.65	+32.85
$\alpha = 0.25$	26.65	**60.35**	**59.85**	59.70	+33.70
$\alpha = 0.50$ (balanced mixture)	26.65	59.95	58.85	**60.35**	**+33.30**
$\alpha = 0.75$	26.65	57.10	58.90	57.95	+30.45
$\alpha = 1.00$ (pure error-only)	26.65	50.90	45.90	44.55	+24.25
Out-of-Distribution Real Scans: GLUE3D *(256 questions, $\alpha = 0.50$)*					
Transport (64 queries)	23.44	**34.38**	31.25	34.38	**+10.94**
Object (64 queries)	34.38	**37.50**	29.69	31.25	+3.12
Architecture (64 queries)	32.81	29.69	20.31	20.31	-3.12
Creature (64 queries)	**40.63**	37.50	32.81	34.38	-3.13
Overall (256 queries)	**32.81**	**34.77**	28.52	30.08	**+1.96**

Real-World Transfer Succeeds in Single-Round Training but Decays Under Repeated Iteration. On real-world scans from GLUE3D (256 questions; Table 2, bottom), initial synthetic fine-tuning yields consistent positive transfer, lifting overall accuracy from 32.81% to 34.77% ($\alpha = 0.50$). This gain concentrates sharply on domains represented in our synthetic catalog: accuracy surges by $+10.94$ percentage points on Transport ($23.44\% \rightarrow 34.38\%$) and $+3.12$ on Object, whereas unrepresented categories (Architecture, Creature) remain near baseline noise. However, subsequent rounds (R2–R3) exhibit marked performance decay on real scans (28.52%–30.47%). While held-out synthetic accuracy improves, repeated training over synthetic point clouds induces domain specialization that amplifies sim-to-real discrepancies. This establishes that while failure-guided synthetic data delivers immediate single-round gains, iterative self-improvement requires real-world grounding to avert simulation overfitting.

4 RELATED WORK

Point-Cloud Multimodal Language Models. Point-cloud MLLMs span point-text alignment (Guo et al., 2023; Xu et al., 2024; Qi et al., 2024a; Xue et al., 2023; 2024), multimodal unification (Han et al., 2024), and data-efficient transfer (Tang et al., 2024; 2025). Recent advances integrate point-language generation (Qi et al., 2024b), part-aware grounding (Ahmed et al., 2025), chain-of-thought reasoning (Chen et al., 2026), geometric alignment (Su et al., 2026), and scene understanding (Xiong et al., 2025; Huang et al., 2024; Mao et al., 2026; Huang et al., 2025b;c). Static suites (ScanQA (Azuma et al., 2022), SQA3D (Ma et al., 2022), ScanReQA (Zhang et al., 2025a), GLUE3D (Mariani et al., 2025)) assess these capabilities with fixed questions; ANYPOINT instead synthesizes dynamic tasks with adaptive failure discovery.

Dynamic and Adaptive Evaluation. Dynamic evaluation refreshes test content or adapts query selection to model feedback (Chen et al., 2023). Text-based methods encompass adversarial collection (Kiela et al., 2021), task generation (Zhu et al., 2024; Wang et al., 2025; Huang et al., 2025d), difficulty optimization (LI et al., 2025), error search (Song et al., 2025), skill profiling (Zeng et al., 2025; Ribeiro et al., 2020), psychometrics (Li et al., 2025), and capability discovery (Lu et al., 2025). While vision-language bootstrapping (Yang et al., 2025) and safety suites (Wang et al., 2026) explore multimodal transformations, they remain confined to 2D inputs; ANYPOINT brings budget-constrained dynamic failure discovery to 3D point-cloud MLLMs with program-computed answers.

5 CONCLUSION

We presented ANYPOINT, an adaptive evaluation framework coupling semantic layout generation with deterministic constraint solving over 3D scene graphs to synthesize verified tasks at scale. Under fixed budgets, our dynamic policy elevates failure discovery across six point-cloud MLLMs from 65.80% to 83.73%, with gains driven by isolating hard boundary instances within task types (16.51 percentage points), while dynamic training confirms that preserving task diversity is essential to prevent catastrophic forgetting, extending to open-world real scenes as an exciting future frontier.

AI USE STATEMENT

We used an AI assistant for writing support, including drafting and revising prose, tightening exposition, and copy-editing, as well as for implementation and analysis support. All claims, experimental choices, and interpretations originate with the authors, who reviewed all AI-assisted work and verified the reported numbers against the underlying experimental records. The authors take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work investigates the evaluation and capability diagnosis of point-cloud multimodal language models. All 3D object assets are sourced from the publicly released TexVerse catalog under its academic research license (detailed in Appendix C). Our study does not involve human subjects, personal data, or high-risk decision scenarios, presenting no foreseeable negative societal impacts or ethical concerns.

REPRODUCIBILITY STATEMENT

To facilitate complete reproducibility, comprehensive implementation details, model architectures, and evaluation prompts are documented in Appendices A, B, and G. The deterministic geometric constraint solver and programmatic task generation procedures are fully formalized in Algorithms 1 and 2 (Appendices D and F), with automated physical validity and answerability audits detailed in Appendix D. Furthermore, the budget-aware dynamic evaluation algorithm and incremental update procedures are specified in Algorithm 3 (Appendix N).

REFERENCES

Mahmoud Ahmed, Junjie Fei, Jian Ding, Eslam Mohamed Bakr, and Mohamed Elhoseiny. Kestrel: 3d multimodal llm for part-aware grounded description. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8973–8983. IEEE, 2025.

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.

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.

Huili Chen, Xinqiao Zhang, Ke Huang, and Farinaz Koushanfar. Adatest: Reinforcement learning and adaptive sampling for on-chip hardware trojan detection. *ACM Transactions on Embedded Computing Systems*, 22(2):1–23, 2023.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26418–26428. IEEE, 2024.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in neural information processing systems*, 35:5982–5994, 2022.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in neural information processing systems*, 36:18225–18250, 2023.

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.

Jiaming Han, Kaixiong Gong, Yiyuan Zhang, Jiaqi Wang, Kaipeng Zhang, Dahua Lin, Yu Qiao, Peng Gao, and Xiangyu Yue. Onellm: One framework to align all modalities with language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26584–26595, 2024.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

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

Ian Huang, Yanan Bao, Karen Truong, Howard Zhou, Cordelia Schmid, Leonidas Guibas, and Alireza Fathi. Fireplace: Geometric refinements of llm common sense reasoning for 3d object placement. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13466–13476. IEEE, 2025a.

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

Wencan Huang, Daizong Liu, and Wei Hu. Fast3d: Accelerating 3d multi-modal large language models for efficient 3d scene understanding. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10935–10944, 2025c.

Yue Huang, Siyuan Wu, Chujie Gao, Dongping Chen, Qihui Zhang, Yao Wan, Tianyi Zhou, Chaowei Xiao, Jianfeng Gao, Lichao Sun, et al. Datagen: Unified synthetic dataset generation via large language models. In *International Conference on Learning Representations*, volume 2025, pp. 63739–63773, 2025d.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. Dynabench: Rethinking benchmarking in nlp. In *Proceedings of the 2021 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pp. 4110–4124, 2021.

Peiyu Li, Xiuxiu Tang, Si Chen, Ying Cheng, Ronald Metoyer, Ting Hua, and Nitesh V Chawla. Adaptive testing for llm evaluation: A psychometric alternative to static benchmarks. *arXiv preprint arXiv:2511.04689*, 2025.

XIANG LI, Farzaan Kaiyom, Evan Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. Autobencher: Towards declarative benchmark construction. In *International Conference on Learning Representations*, volume 2025, pp. 94357–94386, 2025.

Cong Lu, Shengran Hu, and Jeff Clune. Automated capability discovery via model self-exploration. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*, 2022.

Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. Spatiallm: Training large language models for structured indoor modeling. *Advances in Neural Information Processing Systems*, 38:45165–45195, 2026.

Giorgio Mariani, Alessandro Raganato, Simone Melzi, and Gabriella Pasi. Glue3d: General language understanding evaluation for 3d point clouds. *Information Fusion*, pp. 104007, 2025.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26407–26417. IEEE, 2024b.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4902–4912, 2020.

Linxin Song, Xuwei Ding, Jieyu Zhang, Taiwei Shi, Ryotaro Shimizu, Rahul Gupta, Yang Liu, Jian Kang, and Jieyu Zhao. Discovering knowledge deficiencies of language models on massive knowledge base. *arXiv preprint arXiv:2503.23361*, 2025.

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.

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 *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 6617–6626, 2024.

Yuan Tang, Xu Han, Xianzhi Li, Qiao Yu, Jinfeng Xu, Yixue Hao, Long Hu, and Min Chen. More text, less point: Towards 3d data-efficient point-language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7284–7292, 2025.

Hanqing Wang, Yuan Tian, Mingyu Liu, Zhenhao Zhang, and Xiangyang Zhu. Sdeval: Safety dynamic evaluation for multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33449–33457, 2026.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuan-Jing Huang, and Zhongyu Wei. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation. In *Proceedings of the 31st international conference on computational linguistics*, pp. 3310–3328, 2025.

Haomiao Xiong, Yunzhi Zhuge, Jiawen Zhu, Lu Zhang, and Huchuan Lu. 3ur-llm: An end-to-end multimodal large language model for 3d scene understanding. *IEEE Transactions on Multimedia*, 27:2899–2911, 2025.

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.

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.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27081–27091. IEEE, 2024.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16277–16287. IEEE, 2024.

Yue Yang, Shuibo Zhang, Kaipeng Zhang, Yi Bin, Yu Wang, Ping Luo, and Wenqi Shao. Dynamic multimodal evaluation with flexible complexity by vision-language bootstrapping. In *International Conference on Learning Representations*, volume 2025, pp. 21399–21420, 2025.

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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19291–19300. IEEE, 2022.

Qingcheng Zeng, Lucas Garay, Peilin Zhou, Dading Chong, Yining Hua, Jiageng Wu, Yikang Pan, Han Zhou, Rob Voigt, and Jie Yang. Greenplm: cross-lingual transfer of monolingual pre-trained language models at almost no cost. *arXiv preprint arXiv:2211.06993*, 2022.

Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. Evaltree: Profiling language model weaknesses via hierarchical capability trees. *arXiv preprint arXiv:2503.08893*, 2025.

Weichen Zhang, Ruiying Peng, Chen Gao, Jianjie Fang, Xin Zeng, Kaiyuan Li, Ziyou Wang, Jinqiang Cui, Xin Wang, Xinlei Chen, et al. The point, the vision and the text: Does point cloud boost spatial reasoning of large language models? *arXiv preprint arXiv:2504.04540*, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025c.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *International Conference on Learning Representations*, volume 2024, pp. 18091–18128, 2024.

A MODELS UNDER EVALUATION

This appendix details the architectural configurations, visual encoders, language backbones, and checkpoint specifications for the six evaluated point-cloud MLLMs. All models are evaluated under the unified generation configuration detailed in Section 3.1 using the evaluation prompts documented in Appendix G.

PointLLM (Xu et al., 2024). PointLLM integrates a Point-BERT (Yu et al., 2022) point-cloud encoder with a frozen LLaMA-7B language model via a linear projection layer. It tokenizes $8{,}192$ input points into 512 point patch tokens with learned positional encodings, trained in two stages covering point-text feature alignment and instruction fine-tuning.

OneLLM (Han et al., 2024). OneLLM establishes a unified multimodal framework aligning eight modalities to a shared LLaMA-7B backbone. For 3D point clouds, it leverages a modality-specific tokenizer and a Universal Multimodal Interleaved (UMI) projection module to project point features into the language embedding space.

ShapeLLM (Qi et al., 2024a). ShapeLLM targets 3D object understanding and embodied interaction. It couples a Point-BERT encoder with a LLaVA-style LLaMA-7B backbone, pre-trained on large-scale point-language datasets to support spatial reasoning, attribute grounding, and 3D shape manipulation.

MiniGPT-3D (Tang et al., 2024). MiniGPT-3D is an efficient, lightweight point-cloud MLLM that aligns 3D representations to a compact Phi-2 (2.7B) language backbone. It leverages pre-trained 2D vision priors and a cross-attention projection mechanism to achieve spatial understanding with low computational overhead.

PointAlign (Su et al., 2026). PointAlign employs a 2.7B architecture built upon Phi-2 and a BLIP-2 style Q-Former. It introduces explicit feature-level geometric alignment regularization during pre-training to enforce fine-grained correspondence between 3D geometric tokens and linguistic concepts.

GreenPLM (Zeng et al., 2022). GreenPLM is an efficiency-oriented 3.8B model based on Phi-3-mini-4k-instruct. Designed for low training cost, it is pre-trained via cross-lingual text-space transfer without paired point-cloud and text supervision, retaining strong textual relational priors but exhibiting distinctive visual grounding characteristics.

B BASELINE EVALUATION STRATEGIES

To isolate the query selection policy as the sole experimental variable, every baseline is re-implemented on our shared $1{,}000{,}000$-task evaluation candidate pool under identical testing conditions: a total query budget of $B = 1{,}000$, a batch size of $K = 10$, seed 42, and the same deterministic parsing judge (Appendix L).

Static Uniform Sampling (*Uniform Sampling*). The static benchmark baseline samples testing queries uniformly at random from the candidate pool without utilizing model interaction feedback. Because conventional static benchmarks are themselves samples drawn independently of evaluated models, *Uniform Sampling* establishes what a fixed test suite achieves under an identical query budget.

Automated Capability Discovery (ACD) (Lu et al., 2025). ACD adaptively steers evaluation by adjusting task difficulty based on the model's historical performance trajectory. In our re-implementation on the candidate pool, ACD clusters tasks by complexity and updates category sampling probabilities via heuristic transition rules based on empirical success rates.

AutoBencher (LI et al., 2025). AutoBencher optimizes test item allocation across predefined categorical bins to maximize difficulty and novelty. On our candidate pool, AutoBencher models multi-armed bandit distributions over the structural task axes (localization cues and query types) to concentrate queries on bins with high empirical failure rates.

Stochastic Error Ascent (SEA) (Song et al., 2025). SEA discovers model weaknesses by traversing a task dependency graph to ascend error gradients. In our implementation, SEA is provided with the identical multi-component TASKEMBEDDER as ANYPOINT, using hierarchical layout-level

candidate pruning followed by full-task embedding similarity ranking to ascend empirical error clusters according to Algorithm 1 of Song et al. (2025).

Representation Fairness and Controlled Parity. To resolve whether ANYPOINT's advantage stems from its exploit–explore utility or from representational metadata, we evaluate strict representation-matched controls. While SEA operates over the identical joint task embedding, AutoBencher (LI et al., 2025) and ACD (Lu et al., 2025) follow their canonical formulations operating over categorical structural bins and difficulty tiers. Crucially, as documented in our component ablation (Table 5, Appendix I), even when ANYPOINT is stripped of all privileged answer and scene-graph metadata and restricted strictly to question sentence embeddings alone ($\mathbf{e}_Q$, identical to the sentence representations used in standard text-only baselines), ANYPOINT still discovers 79.30% failures on POINTLLM. This performance decisively outperforms SEA (77.30%), AutoBencher (71.63%), and ACD (67.40%), isolating the algorithmic superiority of our exploit–explore formulation from feature-representation differences.

Ablation Baselines. We also examine the two endpoint ablations of our own utility function (Eq. 5) in Section 3.3: *Pure Exploitation* ($\lambda = 0$), which greedily chases errors without novelty exploration, and *Pure Exploration* ($\lambda = 1$), which steers exclusively toward unexplored regions without leveraging error feedback.

C OBJECT CATALOG AND POINT CLOUDS

The object catalog is constructed from $20{,}438$ 3D object assets filtered from TexVerse (Zhang et al., 2025c). Each object is semantically decomposed into named functional components (e.g., the seat, legs, and backrest of a chair). Questions are posed over two grounded component attributes, *color* and *shape*, which are directly verifiable from the $xyzrgb$ point-cloud representation (see Appendix E for the full vocabulary). Object point clouds are sampled to $8{,}192$ points with normalized RGB color channels. During scene assembly, each object point cloud is scaled to its instantiated 3D bounding box, rotated by its assigned orientation, and translated to its solved spatial coordinates before all objects are merged into a unified scene point cloud. Assets were rigorously filtered to eliminate non-manifold geometries, degenerate bounding boxes, and ambiguous multi-label colorings. All source assets from TexVerse are utilized in full compliance with their academic research license.

D GEOMETRIC CONSTRAINT SOLVING ALGORITHM

This appendix formalizes the deterministic geometric constraint solving procedure described in Section 2.2. Algorithm 1 details the complete three-step workflow: mapping discrete size tiers into canonical physical scale priors, topologically ordering spatial dependencies while pruning contradictory cycles, and decoupling 3D positioning into vertical contact and horizontal constraint-cone sampling with collision-free verification.

Canonical Physical Scale Priors. Table 3 lists the canonical physical bounding-box volume intervals and representative real-world object categories corresponding to each of the five discrete scale tiers in $\mathcal{T}$. To generate physically realistic scenes, 3D bounding-box dimensions are sampled uniformly within realistic volume bounds matched to everyday indoor objects, preventing uncurated 3D assets from introducing unnatural size proportions.

Table 3: Canonical physical scale priors $\mathcal{T}$ for the five discrete size tiers.

Scale Tier	Volume Interval (m^3)	Representative Real-World Categories
smallest	$[0.001, 0.015]$	Book, cup, phone, mouse, apple, pen
small	$[0.015, 0.080]$	Desk lamp, kettle, laptop, potted plant, vase
medium	$[0.080, 0.350]$	Dining chair, stool, wastebasket, monitor, backpack
large	$[0.350, 1.200]$	Office desk, armchair, cabinet, bookshelf, dresser
largest	$[1.200, 3.500]$	Dining table, bed, three-seater sofa, wardrobe

Algorithm 1 Deterministic Geometric Constraint Solving

Require: Semantic scene graph $G_{\text{sem}} = (V, \mathcal{E}_{\text{sem}})$, size-to-dimension prior mapping $\mathcal{T}$, max attempts $M_{\text{att}} = 50$
Ensure: Continuous bounding boxes $\mathbf{B}$ and positions $\mathbf{p}$ for all $v \in V$, or $\emptyset$ if rejected

 # Stage 1: Size Instantiation and Domain Mapping
for each object $v \in V$ **do**
  Sample 3D bounding-box dimensions $\mathbf{d}(v) \sim \text{Uniform}(\mathcal{T}(s(v)))$ ▷ Canonical physical scale priors
end for

 # Stage 2: Topological Dependency Ordering
Construct directed dependency graph $\mathcal{D} = (V, \mathcal{E}_{\text{dep}})$ from spatial triplets
Detect and discard contradictory cycles in $\mathcal{D}$ to ensure an acyclic structure
$\mathcal{V}_{\text{ordered}} \leftarrow \text{TOPOLOGICALSORT}(\mathcal{D})$

 # Stage 3: Decoupled Placement and Collision Solving
Placed objects $\mathcal{U} \leftarrow \emptyset$
for each node $u \in \mathcal{V}_{\text{ordered}}$ **do**
  success $\leftarrow$ false
  Let v be the parent reference object of u in $\mathcal{D}$ (or ground plane if u is a root)
  for attempt $= 1 \rightarrow M_{\text{att}}$ **do**
    Compute vertical elevation y_u via ground support or contact rules with reference v
    Sample horizontal coordinates (x_u, z_u) within directional constraint cone of reference v
    $\mathbf{p}(u) \leftarrow (x_u, y_u, z_u)$; $\mathbf{B}(u) \leftarrow \text{CONSTRUCTBOX}(\mathbf{p}(u), \mathbf{d}(u))$
    if $\mathbf{B}(u) \cap \mathbf{B}(w) = \emptyset$ for all $w \in \mathcal{U}$ **and** $\text{CHECKSUPPORT}(\mathbf{B}(u))$ **then**
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{u\}$; success $\leftarrow$ true; **break**
    end if
  end for
  if not success **then return** $\emptyset$ ▷ Reject if spatial constraints cannot be satisfied
  end if
end for
return Placed bounding boxes $\mathbf{B}$ and positions $\mathbf{p}$

Quantitative Physical Validity and Answerability Audits. To verify physical realism and geometric solvability across the candidate pool, we conducted two automated audits: (1) **Collision and Support Clearance**: The constraint solver enforces a minimum safety clearance threshold ($\delta = 0.05$ m) across oriented bounding boxes. Automated intersection tests across all synthesized scenes confirm a 100% collision-free and support-valid placement rate among accepted configurations. (2) **Independent**

Symbolic Answerability: A separate deterministic interpreter re-evaluated $10{,}000$ randomly sampled tasks directly over the grounded scene geometries, verifying 100% answerability and ground-truth label consistency ($10{,}000/10{,}000$), ensuring that all retained questions admit unambiguous, well-posed geometric solutions.

E Relation and Attribute Vocabularies

Section 2.3 admits only relations decidable from geometry and only attributes a point cloud encodes; this appendix lists both vocabularies.

Relations. Questions are posed over the six directional relations *to the left of*, *to the right of*, *above*, *below*, *in front of*, and *behind*. Each is decided by the dominant axis of the displacement between the two objects, so the corresponding edge asserts a fact that can be checked against the rendered geometry; a pair whose displacement has no strictly dominant axis yields no edge and no question. The layout solver may produce scenes containing additional relations (*near*, *far*, *beside*, *surrounding*), but questions are derived only over the six directional relations above because those admit unique answers: multiple objects may satisfy *near* or *beside*, which would violate the well-posedness constraint of Section 2.3.

Attributes. Questions are posed over two component attributes, *color* and *shape*. Both are grounded in an $xyzrgb$ point cloud: color is carried by the RGB channels and shape by the geometry. Material and texture are not recoverable from coordinates and colors alone, so a question about them would have an answer not determined by the model's input; we therefore exclude them.

F Localization Cues and Queries

This appendix defines each cue of Eq. 3 and each query by how it is evaluated on a scene graph G. Table 4 summarizes the values taken across the three structural task axes. A cue maps the objects still in play to a smaller collection; a chain of cues therefore narrows from the whole scene to the question's target, and terminates at a uniquely named object supplying the reference the chain is measured against.

Table 4: The values taken across the three structural axes of our programmatic grammar, along which Section 3.4 reads each model's failure distribution. Every sampled derivation is described by one value on each axis.

Axis	Values
Localization	by object name, by frequency, by size, by distance, by spatial relation, by spatial region, composed query
Query type	object identification, object counting, attribute lookup, attribute listing, attribute counting, region counting, frequency ranking, spatial direction
Reasoning steps	$1,\ 2,\ \geq 3$

Localization cues. Size takes the object of largest or smallest bounding-box volume. Distance takes the object nearest to or farthest from a reference. Relation takes the object standing in a given spatial relation to a reference. Region keeps the objects lying in a given direction from a reference. Attribute keeps the objects whose named attribute takes a given value. A chain whose cues jointly resolve to the intended target entities combines spatial constraints into an unambiguous selection.

Queries. Given what the localization leaves, a query asks for one of: the name of the object; the value of one of its attributes; the enumeration of all values of an attribute; the number of distinct values an attribute takes; the number of objects; or the direction between the object and a reference. Two further queries range over the instances of an object type in scenes where a type recurs, asking how many instances are present or which type is the most or least frequent.

Chained cues. Because each cue takes a localization as its argument, a localization of n cues applies them in sequence, and $|\ell|$ counts the number of reasoning steps (Table 4). For instance, asking for the colors of the object closest to a named object chains a single Distance cue and applies an

attribute-enumeration query; adding an ATTRIBUTE cue beforehand would restrict that comparison to objects of a given color, giving a chain of two.

Ground-Truth Certification and Well-Posedness Criteria. To prevent vacuous or ambiguous questions during automated generation, candidate program trees executed over scene graph G must satisfy three deterministic criteria before being retained:

1. **Non-Empty Entity Resolution**: The target set resolved by the localization selector must not be empty ($S = [\![\ell]\!]_G \neq \emptyset$), eliminating questions regarding non-existent entities.
2. **Absence of Metric Ties**: For continuous metric operators (such as selecting the nearest, farthest, largest, or smallest object), programs are rejected if two or more candidate entities yield identical metric values, avoiding arbitrary tie-breaking.
3. **Strict Answer Uniqueness**: For scalar queries (such as querying an object's color, shape, or category), the target set must contain exactly one unambiguous entity ($|S| = 1$), ensuring that the question admits precisely one correct answer on scene graph G.

Only candidate programs satisfying all three criteria are passed to the natural-language template engine and paired with in-scene distractors, as formalized in Algorithm 2.

Algorithm 2 Programmatic Task Synthesis and Verification on Scene Graph G

Require: Grounded 3D scene graph $G = (V, \mathcal{E})$, max localization depth D, distractor count $m \geq 0$
Ensure: Certified evaluation task $(\mathbf{P}, Q, y^\star)$ with optional option set $\mathcal{O}$, and descriptor $(\text{outer}(\ell), \pi, |\ell|)$

 # Subroutine: Recursive Localization Grammar Sampling
function SAMPLELOCALIZATION(G, d)
  if $d = 0$ **then return** sampled base set from $\{V\} \cup \{V_c \mid c \in \{c(v) \mid v \in V\}\}$
  end if
  $\mathcal{C}_{\text{adm}} \leftarrow$ {selectors in Eq. 3 admissible on G} ▷ Must support unique resolution
  $s \sim \text{Uniform}(\mathcal{C}_{\text{adm}})$; $\ell_{\text{sub}} \leftarrow$ SAMPLELOCALIZATION$(G, d-1)$
  return $s(\ell_{\text{sub}})$
end function

 # Main Synthesis and Verification Loop
repeat
  $d \sim \text{Uniform}(\{1, \ldots, D\})$; $\ell \leftarrow$ SAMPLELOCALIZATION(G, d); $S \leftarrow [\![\ell]\!]_G$
  $\pi \sim \text{Uniform}(\{\text{queries admissible on } S\})$; $y^\star \leftarrow [\![\pi]\!]_G(S)$
until $S \neq \emptyset$ **and** $y^\star$ is unique ▷ Certify well-posed ground truth without ambiguity
$Q \leftarrow$ REALIZENATURALLANGUAGE(π, ℓ)
if $m > 0$ **then** ▷ Optional closed-set option construction
  $\mathcal{D} \leftarrow$ SYNTHESIZEINSCENEDISTRACTORS$(G, \pi, y^\star, m)$; $\mathcal{O} \leftarrow$ SHUFFLE$(\{y^\star\} \cup \mathcal{D})$
else $\mathcal{O} \leftarrow \emptyset$
end if
return Task $(\mathbf{P}, Q, y^\star)$, option set $\mathcal{O}$, and descriptor $\big(\text{outer}(\ell),\ \pi,\ |\ell|\big)$

G PROMPTS

This appendix provides the exact system and task prompts used across the ANYPOINT pipeline.

G.1 SYMBOLIC 3D SCENE GRAPH GENERATION PROMPT

Given a sampled category subset from the asset catalog, the language model (DeepSeek-V4-Pro) proposes a layout structured as a semantic scene graph. Box G.1 documents the system prompt (condensed for formatting; in-context demonstrations are omitted for brevity).

System Prompt for Symbolic 3D Scene Graph Generation

Role: You are an expert at generating realistic spatial scene layouts for 3D object placement. Given a list of objects, generate a JSON scene graph describing their arrangement in a physically plausible, real-world indoor scene.

Task Requirements:

- **Scale coverage:** Include both `largest` and `smallest` scale tiers ($\geq$ 1 object each).
- **Strict entity naming:** Every relation reference must be an actual object name from the input list.
- **No ungrounded descriptors:** State spatial relationships only; strictly avoid visual adjectives or stylistic embellishments.
- **Physical plausibility:** Arrange objects adhering to physical stability, support logic, functional grouping, and reachable orientations.

Target JSON Schema:

```
{
  "description": "<concise scene arrangement summary>",
  "objects": [
    {"name": "<category>", "size": "<scale_tier>", "rotation":
<degrees>}
  ],
  "relations": [
    {"subject": "<category>", "relation": "<rel>", "reference":
"<category>"}
  ]
}
```

Spatial Relation and Scale Vocabulary:

- **Scale tiers:** `largest` (required), `large`, `medium`, `small`, `smallest` (required).
- **Horizontal relations:** `in front of`, `behind`, `to the left of`, `to the right of`, `next to`, `near`, `far from`.
- **Vertical relations:** `on`, `above`, `below`, `under`.
- **Physical constraints:** Gravitational support (smaller objects on larger surfaces), collision clearance, functional clustering, realistic 3D yaw rotations.

G.2 Closed-Set Evaluation Prompt

Each evaluated point-cloud MLLM receives the rendered 3D scene point cloud $\mathbf{P} \in \mathbb{R}^{N\times 6}$ together with the multiple-choice prompt detailed in Box G.2.

Prompt Template for Multiple-Choice Evaluation Probing

Context: You are given a 3D point cloud of an indoor scene and a spatial understanding question.

Input Query Template:

```
Question: [Synthesized Question Text Q]
Options:
A: [Option 1]
B: [Option 2]
C: [Option 3]
[Optional D: Option 4]
Please answer directly with the letter of the correct option.
```

Evaluation Protocol: The model's free-form decoded output is processed by a deterministic parsing judge (Appendix L) that maps the response to discrete option letters $\{A, B, C, D\}$ to compare against certified ground truth $y^\star$.

Asset filtering and component annotation prompts. Detailed prompts used during 3D asset curation, geometric quality filtering, and functional component decomposition from TexVerse are documented here.

H Qualitative Task Generation Gallery

This appendix provides concrete qualitative examples of evaluation tasks synthesized by AnyPoint, illustrating the coupling between physically grounded 3D scene point clouds, programmatic question derivation, and geometrically certified ground-truth answers.

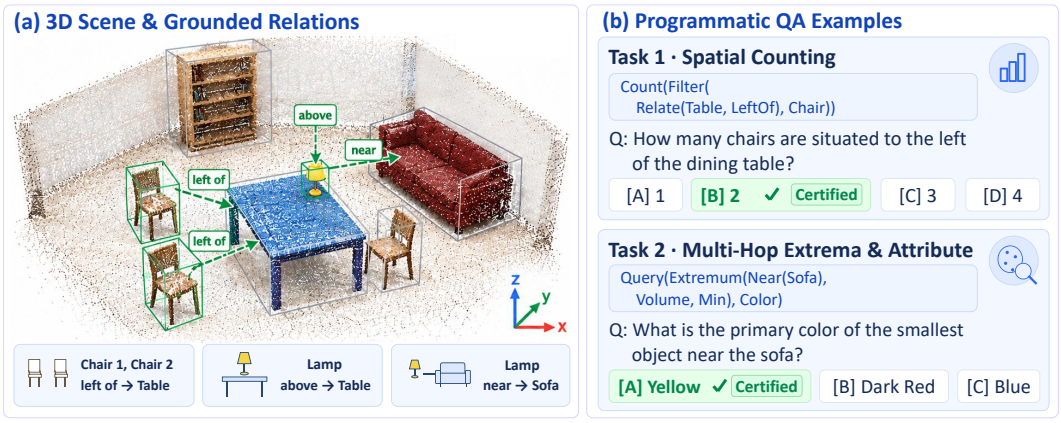

Figure 5: Qualitative evaluation task generation in AnyPoint. (a) **3D Scene & Grounded Relations**: A collision-free 3D scene point cloud assembled from TexVerse object assets, grounded with continuous 3D bounding boxes and semantic spatial relations (*left of*, *above*, *near*). (b) **Programmatic QA Examples**: Executable program trees derived over the grounded scene graph synthesize spatial counting questions (Task 1) and multi-hop extrema with attribute queries (Task 2), paired with certified ground-truth answers computed directly from geometry and multiple-choice distractors sampled from co-occurring in-scene entities.

I TASK-VECTOR COMPONENTS

Selection depends on the task vector only through cosine similarity, so what the vector represents determines what the policy can distinguish. Each of the three components in Eq. 4 ($\mathbf{e}_G, \mathbf{e}_Q, \mathbf{e}_y$) is encoded using Qwen3-Embedding-8B (Zhang et al., 2025b) to unit length (measured at 1.0000 ± 0.0003), so scaling the concatenation by $1/\sqrt{3}$ makes the inner product between two task vectors mathematically equal to the unweighted mean of their three component cosine similarities, enabling fair comparison when ablating components.

Table 5 replaces the full joint vector by each individual component, holding λ=0.2 and the rest of the protocol fixed. No single component suffices. The question ($\mathbf{e}_Q$) and answer ($\mathbf{e}_y$) each carry significant diagnostic signal (losing 7.30 and 8.60 points when used in isolation), while the scene graph alone ($\mathbf{e}_G$) drops performance by 18.50 points: scene structure separates tasks by spatial topology and object entities but cannot discern what is queried, whereas tasks sharing a scene are precisely those the policy must distinguish.

Table 5: Failure rate (%) on POINTLLM when the task vector uses one component in place of all three (B=1000, λ=0.2).

Task vector	Dim.	Failure rate	Δ
Full ($\mathbf{e}_G + \mathbf{e}_Q + \mathbf{e}_y$)	12288	**86.60**	—
Question only ($\mathbf{e}_Q$)	4096	79.30	−7.30
Answer only ($\mathbf{e}_y$)	4096	78.00	−8.60
Scene graph only ($\mathbf{e}_G$)	4096	68.10	−18.50

J SENSITIVITY TO THE COLD START

The seed governs the cold-start batch, the draws of the *Random* arm, and per-question sampling, so varying it perturbs the whole pipeline. Its effect on an adaptive run is concentrated in the cold start: a different first batch surfaces different early errors, and every subsequent selection follows from those. Table 6 repeats the sweep on POINTLLM over three seeds under an otherwise identical configuration.

Table 6: Failure rate (%) on POINTLLM across three seeds, B=1000. *SD* is the standard deviation over seeds; the *Random* arm's 1.42 is the scale of ordinary sampling noise. Every setting is near that scale except λ=0, whose exploit-only utility leaves the run determined by whichever errors the cold start happens to surface.

Policy	42	43	44	Mean	SD
Static (uniform)	71.80	73.90	74.50	73.40	1.42
λ=0 (errors only)	66.70	92.20	70.70	76.53	**13.71**
λ=0.2 (ours)	86.60	88.10	91.70	**88.80**	2.62
λ=0.5	78.90	81.30	80.90	80.37	1.29
λ=0.8	72.70	75.80	72.00	73.50	2.02
λ=1 (novelty only)	72.00	71.70	74.20	72.63	1.37

Three readings follow. The setting we adopt is best on average, at 88.80% against 80.37% for the next best, and its margin over static evaluation is stable when the two are compared within a seed, at +14.8, +14.2, and +17.2 points. Its weakest run, 86.60%, exceeds the strongest run of static evaluation and of every other mixture setting, $\lambda \in \{0.5, 0.8, 1\}$.

The exception is λ=0, which is not weaker on average but is erratic: its three runs give 66.70, 92.20, and 70.70, a standard deviation of 13.71 against 1.42 for uniform sampling. This is what Section 3.3 predicts. At λ=0 the explore factor carries exponent zero and drops out, leaving $U(t) = \mathrm{error}(t)$, so selection depends only on similarity to errors already recorded and the entire trajectory is fixed by whichever errors the cold start happens to reveal. On one seed that yields the highest rate in the table; on another it yields less than uniform sampling, and nothing in the run indicates which case obtains.

K Detailed Failure Gain Decomposition

This appendix provides the mathematical formalization, empirical methodology, and complete multi-axis breakdown for the gain decomposition summarized in Section 3.2.

Mathematical Formalization. Let $\mathcal{C}$ denote the set of task categories along a given structural task axis. Let w_c^{rand} and w_c^{dyn} denote the query proportion allocated to category $c \in \mathcal{C}$ under uniform sampling and dynamic evaluation, respectively. Let $\bar{r}_c^{\text{rand}}$ denote the baseline error rate in category c, and r_c^{dyn} denote the empirical error rate achieved by dynamic evaluation within category c. The total advantage $\Delta_{\text{total}} = \sum_c w_c^{\text{dyn}} r_c^{\text{dyn}} - \sum_c w_c^{\text{rand}} \bar{r}_c^{\text{rand}}$ decomposes additively into:

$$\Delta_{\text{type}} = \sum_{c \in \mathcal{C}} (w_c^{\text{dyn}} - w_c^{\text{rand}}) \bar{r}_c^{\text{rand}}, \qquad \Delta_{\text{inst}} = \sum_{c \in \mathcal{C}} w_c^{\text{dyn}} (r_c^{\text{dyn}} - \bar{r}_c^{\text{rand}}), \tag{6}$$

where Δ_{type} captures the contribution of shifting sample weights across task categories under average baseline difficulty, and Δ_{inst} measures the residual gain from isolating harder instances within categories.

Multi-Axis and Model-by-Model Breakdown. Table 7 presents the full model-by-model breakdown along the 3D spatial localization axis, which represents the core multimodal capability distinguishing point-cloud understanding from 2D vision. The bottom summary rows verify consistency across all three structural task axes (§2.3).

Table 7: Decomposition of AnyPoint's failure discovery advantage over uniform sampling. The total advantage decomposes additively into gains from shifting query proportions across task types (Δ_{type}) and gains from discovering hard instances within types (Δ_{inst}). Individual models are evaluated along the 3D spatial localization axis; summary rows verify consistency across all three structural task axes (§2.3). Over 92% of the total advantage stems from instance hardness.

Model / Structural Axis	Uniform Sampling	AnyPoint	Total Advantage	Type Shift (Δ_{type})	Instance Hardness (Δ_{inst})
PointLLM	71.80	**86.60**	+14.80	-1.63	+16.43
OneLLM	56.90	**79.70**	+22.80	+7.65	+15.15
ShapeLLM	66.70	**83.50**	+16.80	-2.17	+18.97
GreenPLM	64.70	**85.70**	+21.00	+1.92	+19.08
MiniGPT-3D	67.20	**84.20**	+17.00	+3.73	+13.27
PointAlign	67.50	**82.70**	+15.20	-0.93	+16.13
Average (Spatial Localization, 7 types)	**65.80**	**83.73**	**+17.93**	**+1.43** (8.0%)	**+16.51** (92.0%)
Average (Query Type, 8 types)	**65.80**	**83.73**	**+17.93**	**+3.37** (18.8%)	**+14.56** (81.2%)
Average (Reasoning Depth, 3 tiers)	**65.80**	**83.73**	**+17.93**	**+0.03** (0.2%)	**+17.91** (99.8%)

L Evaluation Metrics and Answer Grading

Models answer in free form, so correctness is assigned in two stages: a deterministic parsing judge extracts which discrete option $\{A, B, C, D\}$ the response selects, after which the comparison against the program-computed answer is a deterministic string match. The judge therefore performs structural extraction rather than subjective assessment, ensuring objective verification. The identical parsing judge is applied to every evaluated arm, so grading applies uniformly and cannot favor one selection policy over another.

Formal Definitions of Evaluation Metrics. We formally define the evaluation metrics reported in Section 3:

- **Failure Rate within Budget ($|E|/B$):** Given a testing budget of B queries and the resulting error set $E \subseteq \mathcal{S}$ ($|\mathcal{S}| = B$), the failure rate is defined as $\mathrm{FR} = \frac{|E|}{B} \times 100\%$. Under a fixed query constraint, a higher failure rate indicates more efficient allocation toward unresolved failure boundaries.
- **Additive Gain Decomposition ($\Delta_{\text{type}}, \Delta_{\text{inst}}$):** Full formalization, mathematical derivations, and multi-axis empirical breakdowns are detailed in Appendix K.
- **Diagnostic Allocation Fractions:** For each task axis (localization cues, query types, reasoning steps), the diagnostic profile reports the percentage of the budget allocated to category c: $p_c = \frac{B_c}{B} \times 100\%$, where B_c is the number of queries evaluated in category c.

M Dynamic-Training Configuration

Section 3.5 trains PointLLM (LLaMA-7B backbone) for three rounds of *diagnose – retrieve – train – evaluate*. The LLM is fully fine-tuned with the point-cloud backbone frozen and the projection layer trainable.

Loop. Each round spends a diagnostic budget of 1000 queries on the one-million-task evaluation pool under the selection policy of Eq. 5 with λ=0.2 and batch size 10, then retrieves 10,000 items from a disjoint one-million-task training pool by maximum cosine similarity to the error set, under the task embedding of Eq. 4. A round trains only on its own retrieved data; earlier rounds are not replayed. The mixing coefficient α sets how many of the 10,000 items come from similarity-ranked retrieval, the remainder being drawn uniformly from the pool, so α=0 performs no diagnosis at all. Exclusion sets are maintained independently per α, and the seed is fixed to 42 throughout.

Optimization. We use the official PointLLM stage-two recipe: three epochs per round, per-device batch size 4 across eight GPUs, learning rate 2×10^{-5} with cosine schedule and warmup ratio 0.03, no weight decay, maximum sequence length 2048, and `bf16` precision with gradient checkpointing. Each round is therefore roughly 939 optimizer steps. Re-running $\alpha \in \{0, 0.25, 0.5\}$ at one epoch per round reproduces the same per-round shape, with the first round gaining 30–32 points and subsequent rounds moving within ± 1.5 points, indicating the conclusions are not an artifact of this setting.

Evaluation. The synthetic held-out set contains 2,000 tasks disjoint from both the diagnostic and training pools, evaluated after every round with single-task inference and the per-question seed scheme of Appendix A. GLUE3D MCQ contains 256 four-way questions over real scanned objects, giving a 25% chance baseline; it is evaluated for the untrained model and for all three rounds at $\alpha \in \{0, 0.5\}$. Because the retrieval signal is computed on the synthetic pool, α is not a meaningful ablation axis out of distribution, and the two trajectories serve as controls for one another.

Category-Level Breakdown on GLUE3D. Table 8 provides the fine-grained category-level breakdown of out-of-distribution transfer on the 256 questions of GLUE3D across its four official object categories (64 questions each). As shown, initial positive transfer in Round 1 is concentrated heavily on object domains represented in the synthetic catalog: accuracy surges by +10.94 percentage points on TRANSPORT (23.44% → 34.38%) and +3.12 on OBJECT (34.38% → 37.50%). Conversely, categories outside the synthetic object vocabulary (CREATURE and ARCHITECTURE) exhibit negligible movement, remaining near baseline levels. Subsequent training rounds (R2 and R3) exhibit marked decay across categories, confirming simulator overfitting under repeated iterations.

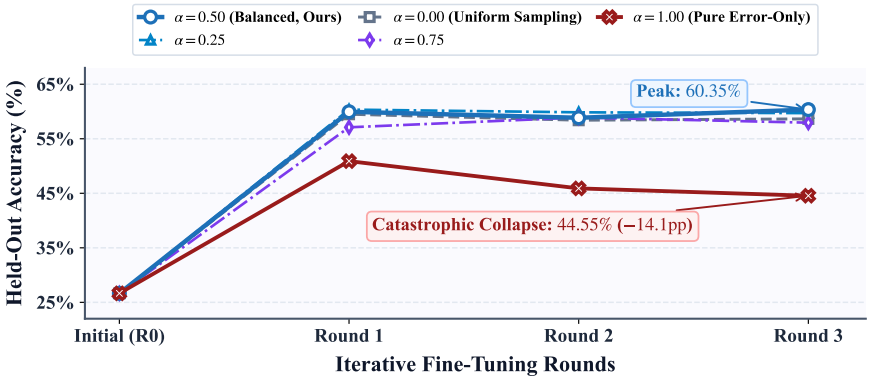

Figure 6: Iterative fine-tuning trajectories on POINTLLM across sequential rounds on 2,000 unseen synthetic tasks. Balanced failure guidance ($\alpha = 0.50$, blue) climbs steadily to peak 60.35%, whereas pure error retrieval ($\alpha = 1.00$, wine crimson) triggers catastrophic forgetting, collapsing to 44.55%.

Table 8: Detailed category-level accuracy (%) on GLUE3D MCQ (256 questions, 64 questions per category) across fine-tuning rounds on POINTLLM.

Category	**Count**	**R0 (Base)**	**R1** (α=0)	**R1** (α=0.5)	$\Delta_{\mathbf{R1}}$
Transport	64	23.44	**34.38**	**34.38**	+10.94
Object	64	34.38	28.12	**37.50**	+3.12
Architecture	64	32.81	**39.06**	29.69	-3.12
Creature	64	**40.63**	39.06	37.50	-3.13
Overall	**256**	**32.81**	**35.16**	**34.77**	**+1.96**

Symmetric Collapse Across Evaluation and Training. Table 9 formalizes the conceptual symmetry between dynamic evaluation and dynamic training. Pursuing pure exploitation without exploration ($\lambda = 0$ in evaluation or $\alpha = 1.0$ in training data curation) causes performance collapse on both testing and training sides of the pipeline, confirming that exploration during evaluation and task diversity during training are mutually essential to prevent capability collapse.

Table 9: Symmetric collapse under pure exploitation across dynamic evaluation and training.

	Pure Exploitation	**Balanced Mixture**	**Uniform Sampling**
Evaluation: Failure Discovery Rate (%)	70.78 (λ=0)	**83.73** (λ=0.2)	65.80
Training: Held-out Accuracy (%)	44.55 (α=1)	**60.35** (α=0.5)	58.65

N Dynamic Evaluation Algorithm and Implementation Details

Algorithm 3 provides the complete procedural specification of the budget-aware dynamic evaluation loop, detailing the initialization, uniform cold start, batch evaluation, and incremental matrix-vector updates. The matrix-vector update and scoring conventions are formalized below.

Incremental Matrix-Vector Similarity Updates. To scale utility computation across million-scale candidate pools without recomputing against accumulated history, we maintain running similarity vectors $\mathbf{a}_R, \mathbf{b}_R \in [0,1]^{|R|}$ updated incrementally with newly observed batches $\mathcal{S}_E$ and $\mathcal{S}_C$:

$$\mathbf{a}_R \leftarrow \max\left(\mathbf{a}_R, \max_{e\in\mathcal{S}_E} \mathbf{V}_R \mathbf{v}_e^\top\right), \qquad \mathbf{b}_R \leftarrow \max\left(\mathbf{b}_R, \max_{c\in\mathcal{S}_C} \mathbf{V}_R \mathbf{v}_c^\top\right), \tag{7}$$

where $\mathbf{V}_R \in \mathbb{R}^{|R|\times 3d}$ contains pre-computed embeddings of remaining candidates R, keeping the per-iteration complexity $\mathcal{O}(|R|K)$ independent of the growing evaluation history.

Algorithm 3 Budget-Aware Dynamic Evaluation Loop

Require: Candidate pool $\mathcal{P}$, evaluated MLLM $\mathcal{M}$, query budget B, batch size K, trade-off λ
Ensure: Correct set C, error set E

 # Stage 1: Initialization and Uniform Cold Start

$R \leftarrow \mathcal{P}$; $C \leftarrow \emptyset$; $E \leftarrow \emptyset$; Initialize similarity vectors $\mathbf{a}_R \leftarrow \mathbf{0}$, $\mathbf{b}_R \leftarrow \mathbf{0}$
$\mathcal{S} \leftarrow$ UniformSample(R, K); $R \leftarrow R \setminus \mathcal{S}$ ▷ Cold-start seed batch
Evaluate $\mathcal{M}$ on $\mathcal{S}$; split into successes $\mathcal{S}_C$ and errors $\mathcal{S}_E$; update $C \leftarrow C \cup \mathcal{S}_C$, $E \leftarrow E \cup \mathcal{S}_E$
Update $\mathbf{a}_R, \mathbf{b}_R$ via incremental matrix-vector updates with $\mathcal{S}_E, \mathcal{S}_C$

 # Stage 2: Adaptive Failure-Boundary Steering Loop

while $|C| + |E| < B$ **do**
  Compute $U(t) = (\mathbf{a}_R(t))^{1-\lambda} \cdot (1 - \mathbf{b}_R(t))^{\lambda}$ for all $t \in R$ ▷ Exploit–explore utility scoring
  $\mathcal{S} \leftarrow$ top-$\min(K,\ B - |C| - |E|)$ items in R by $U(t)$; $R \leftarrow R \setminus \mathcal{S}$
  Evaluate $\mathcal{M}$ on $\mathcal{S}$; split into $\mathcal{S}_C$ and $\mathcal{S}_E$; update $C \leftarrow C \cup \mathcal{S}_C$, $E \leftarrow E \cup \mathcal{S}_E$
  Update running similarities $\mathbf{a}_R, \mathbf{b}_R$ via incremental matrix-vector updates with $\mathcal{S}_E, \mathcal{S}_C$ ▷ Incremental update
end while
return C, E

Setup and notation. Fix a candidate task t with embedding $\mathbf{v}_t$, an exploration weight $\lambda \in [0,1]$, a correct set C, and an error set E. Write

$$a := \max_{e\in E} s(\mathbf{v}_t, \mathbf{v}_e), \qquad b := \max_{c\in C} s(\mathbf{v}_t, \mathbf{v}_c), \qquad x := 1 - b,$$

where $s(\mathbf{u}, \mathbf{v}) = \max(0, \langle \mathbf{u}, \mathbf{v}\rangle) \in [0,1]$ denotes the non-negative cosine similarity, ensuring $a, b, x \in [0,1]$. With this notation Eq. 5 reads $U(t) = a^{\,1-\lambda}\, x^{\lambda}$, and we call a the *exploit similarity* and x the *explore margin*. An empty set contributes a similarity of 0, so $a = 0$ before the first failure is recorded and $x = 1$ before the first correct answer. In the implementation, both bases are floored at $\varepsilon = 10^{-8}$ before exponentiation to keep U finite; at the cold-start boundary ($E = \emptyset$) the floored exploit factor is identical across all candidates, so U ranks by exploration margin x alone as intended.