SSR3D-LLM: Structured Spatial Reasoning via Latent Steps for Fine-Grained Grounding in Unified 3D-LLMs

Jiawei Li[1] **Ziyi Liu**[2,†] **Weijie Shi**[1]
Long Chen[1] **Jiajie Xu**[2] **Xiaofang Zhou**[1]
[1]The Hong Kong University of Science and Technology
[2]Soochow University
jlijz@cse.ust.hk liuziyi@suda.edu.cn

Code Dataset

Abstract

3D object grounding localizes referred objects in a 3D scene from natural language. Unified instance-centric 3D-LLMs aim to solve grounding together with dialog, QA, and captioning, yet many rely on a single pointer-style grounding decision that compresses a relational instruction into one selection. This is brittle for fine-grained queries where multiple same-class candidates must be ruled out by context objects and spatial relations. We propose *Structured Spatial Reasoning 3D-LLM* (SSR3D-LLM), a structured grounding interface for unified 3D-LLMs. Given fixed Mask3D object proposals, the LLM writes a sequence of latent spatial reasoning steps and memory tokens from the query, and a geometry-aware scorer reads these latent steps in order to refine candidate rankings step by step with step-length masking. The latent steps are learned from standard benchmark target supervision with auxiliary referential-cue supervision during training, while inference uses only the input query and Mask3D proposals. Across ReferIt3D, ScanRefer, and Multi3DRef, SSR3D-LLM achieves highly competitive results among unified 3D-LLMs with a 1B backbone, improving over the evaluated Grounded 3D-LLM and Chat-Scene baselines, while its default language route remains functional with task-dependent changes.

1 Introduction

When we instruct a robot to find an object in a 3D scene, many real-world instructions are fine-grained and compositional, specifying not only object category and attributes, but also spatial position and relations to other context objects, e.g., "find the white chair to the right of the brown sofa in the center of the room". Here, the target object is the referent to be localized, while context objects are the other mentioned objects that help disambiguate it, such as "the brown sofa". The robot must parse the linguistic structure and combine it with geometric cues from the 3D scene to precisely localize the target instance. This is the 3D object grounding task, a core capability for embodied AI and spatial intelligence that enables downstream manipulation, navigation, and situated question answering [10, 24].

Early 3D object grounding methods mainly learned supervised cross-modal matching on human-annotated datasets such as ScanRefer and ReferIt3D [6, 3, 9]. They can work well on benchmark expressions, but their language side is usually task-specific: it is less suited to open-form instructions

† Corresponding author

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

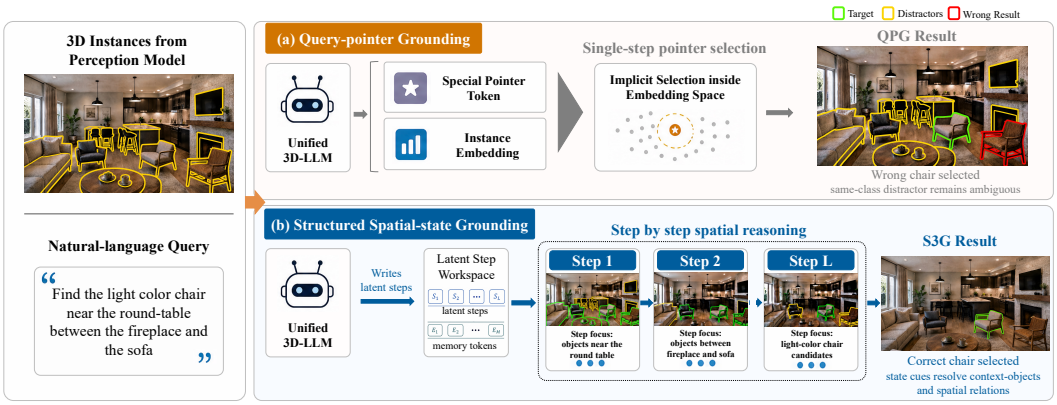


Figure 1: **QPG vs. S3G.** Fine-grained 3D grounding often requires ruling out candidates through context objects and spatial relations. QPG makes one pointer-style selection, while S3G writes latent spatial reasoning steps and refines candidate rankings step by step.

and does not naturally share a model with dialog, question answering, or captioning. Recent foundation models that jointly process text and 3D scenes address this language-interface limitation by building *3D-LLMs* that support language generation and 3D object grounding in a single model [8, 17, 37, 4, 20, 36, 31]. Representative grounding-capable unified 3D-LLMs, such as Grounded 3D-LLM [8] and Chat-Scene [17], extend instruction-tuned LLMs with 3D object representations so the same model can both generate language and select referred objects. Because grounding must share the model with dialog, question answering, and captioning, it is often exposed through a compact output interface rather than a task-specific grounding architecture. In Grounded 3D-LLM, this appears as *Query-Pointer Grounding* (*QPG*) [12]: a single query/pointer representation is used to select one candidate object representation. Related unified systems use similarly compact object-id or pointer-style readouts, forming a broader family of single-readout grounding interfaces. As illustrated in Figure 1 (a), this makes grounding compatible with the LLM's token-level training interface, but it also gives fine-grained spatial grounding a narrow readout.

However, this design is not sufficient to solve fine-grained compositional grounding due to three challenges. (i) Grounding becomes a single selection where target category, attributes, context objects, and spatial relations all have to be reflected in one pointer-style decision. This leaves no intermediate workspace for progressively checking context objects or ruling out same-class distractors, which is exactly what fine-grained spatial queries often require. (ii) Standard final-target supervision only says which object is correct. It does not directly supervise how the model should use context objects and relations when comparing several plausible candidates, so a single-step selector can rely on shortcuts that work on easy expressions but fail on queries requiring ordered spatial disambiguation. (iii) When fine-grained grounding fails, the error may come from parsing the target, choosing the wrong context object, or applying the wrong spatial relation, yet a single pointer output gives little insight into which part failed.

Our method. To address these challenges, we propose *Structured Spatial-State Grounding (S3G)*, shown in Figure 1 (b), and instantiate it as *Structured Spatial Reasoning 3D-LLM (SSR3D-LLM)*.

First, to better handle fine-grained queries with multiple context objects and spatial relations, S3G adds a sequence of latent spatial reasoning steps inside the 3D-LLM, giving the grounding branch a lightweight structured interface instead of squeezing all query cues into a single pointer token. Each step is a latent state read from a reserved step marker, and memory tokens provide shared query context across steps. The steps are not generated text, so the model keeps a compact latent grounding interface without exposing an explicit chain-of-thought output.

Second, to ensure the model actually uses these latent steps, we supervise the reserved step markers during training with auxiliary referential cues, on top of the same standard query-target supervision used for grounding. When intermediate cues are not provided by a benchmark, we derive them offline from the training queries and annotations, so the S3G pathway remains supervised without changing the test-time input. At inference, the model receives only the input query and fixed Mask3D proposals.

Third, to make the procedure stable and easy to inspect, a geometry-aware scorer refines candidate rankings step by step over the fixed Mask3D proposal set. Step-length masking freezes updates after the effective steps are done, and the resulting per-step rankings form a compact non-textual trace that helps localize failures. SSR3D-LLM keeps the upstream point-cloud-derived proposal pipeline and the LLM backbone family in place, and changes how grounding information is represented and consumed. Finally, because a 3D-LLM must remain a capable language model, we confine grounding-specific parameter updates to lightweight adapters and grounding modules to minimize interference with default language behavior.

Our main contributions are:

- We introduce SSR3D-LLM, a unified 3D-LLM framework that keeps the default dialog/QA/captioning pathway while adding a structured grounding route for fine-grained object localization.
- We propose a structured grounding method S3G, which replaces single-pointer QPG with latent spatial reasoning steps learned from training-time referential cues, a geometry-aware candidate scorer, step-length masking, and an inspectable score trace.
- We evaluate on ReferIt3D, ScanRefer, and Multi3DRef: our SSR3D-LLM outperforms Grounded 3D-LLM and Chat-Scene across fine-grained candidate selection, box localization, and set-level grounding, while also achieving lower grounding-readout latency and higher throughput after proposal extraction.

2 Related Work

Early and specialized 3D grounding systems treat grounding as a dedicated cross-modal matching problem over 3D scenes and referring expressions [6, 3]. Many later models add explicit geometric priors, such as distances, directions, object relations, and candidate comparison, to distinguish similar objects in cluttered scenes [30, 41, 27, 15, 40]. Representative systems such as MiKASA [5], ViL3DRel [7], and MVT [19] show that geometry-aware interaction is important for fine-grained 3D reference resolution. These designs motivate our geometry-aware candidate scorer, but they are usually task-specific rather than unified language models.

Large-model and agentic 3D grounding methods use stronger language and visual priors through prompting, rendered views, external 2D detectors, or verification loops. Zero-shot or training-free systems such as ZSVG3D [38], SeeGround [23], LaSP [28], and SORT3D [39] treat grounding as a dedicated inference pipeline with rendered or multi-view evidence. Reasoning-centric systems such as LLM-Grounder [35], VLM-Grounder [34], ViewRefer [14], ViewInfer3D [13], Transcrib3D [11], SPAZER [21], and GPT4Scene [29] use external planning, 2D/multi-view foundation-model prompting, or agentic candidate verification to improve grounding. These methods provide useful broader context and often strong performance, but they operate under different system assumptions from our instance-centric unified 3D-LLM setting.

Unified 3D-LLMs for grounding extend instruction-tuned LLMs with instance-centric 3D representations so a single model can both generate language and ground a referred target object in a 3D scene [8, 17, 16, 20, 36, 31]. Grounded 3D-LLM [8] and Chat-Scene [17], among others, realize grounding by having the model output a discrete referent token or object identifier that selects one object representation. This pointer-style interface (*QPG*) [12] keeps grounding compatible with token prediction, but it gives fine-grained spatial reasoning only a single-step readout. SSR3D-LLM keeps the same unified-model premise and point-cloud-derived candidate setting, but replaces the single pointer route with compact latent spatial reasoning steps for step-wise candidate ranking. Recent Chat-Scene++ [18] keeps object-identifier grounding but uses context-rich object features, a larger backbone, and grounded chain-of-thought training data; Appendix A.11 discusses its reported results.

Intermediate supervision and capability preservation such as referential order or context-object chains can make fine-grained grounding easier to guide and inspect [1, 33, 2]. Our *S3G* uses such cues to shape an ordered internal state, while inference still uses only the input query and 3D candidates. At the same time, grounding-specific supervision can interfere with the language-generation behavior of multimodal LLMs. LISA [22] ties a language token embedding to a segmentation control vector, while STAMP [25] confines dense supervision to the visual side to better preserve conversational capability. Following this concern, we keep the default language-generation pathway intact and confine grounding-specific training signals to lightweight adaptation and grounding modules. Surveys

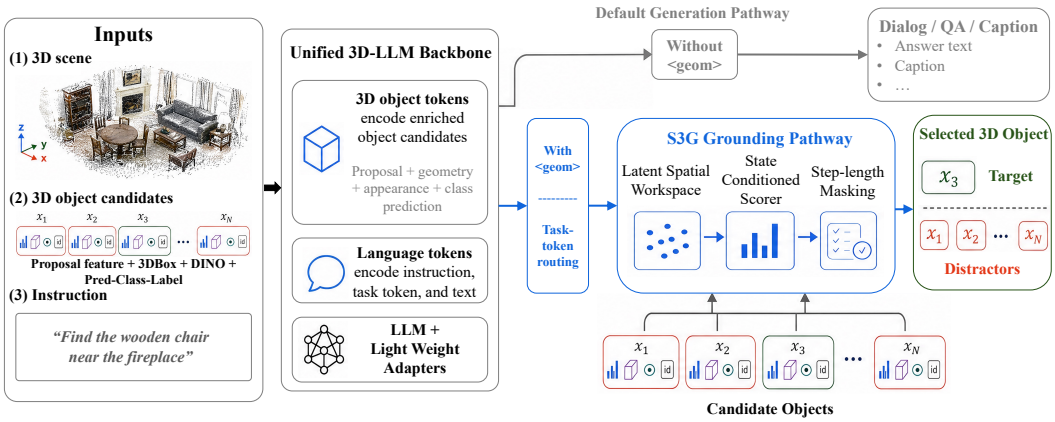


Figure 2: **SSR3D-LLM overview.** One 3D-LLM backbone handles language tasks and grounding. Without `<geom>`, it uses the default dialog/QA/caption route; with `<geom>`, it routes the instruction and Mask3D proposal representations to S3G for target selection and step-wise score traces.

on spatial reasoning in multimodal LLMs [26] similarly highlight the value of separating language generation from geometric decision-making.

3 Preliminaries

Data and Inputs. We use an instance-centric 3D scene S, a referring query u, and N candidate objects or proposals indexed by i, from official ReferIt3D sets or Mask3D proposals after benchmark matching. For proposal i, $\mathbf{f}_i, \mathbf{b}_i, \mathbf{a}_i, \mathbf{c}_i$ denote its pooled Mask3D feature, rotated-box geometry, optional DINOv2 multi-view appearance, and proposal-side predicted-class label embedding.

Candidate-Set Formulation. Grounding ranks $\{1, \ldots, N\}$ with logits $\boldsymbol{\ell} \in \mathbb{R}^N$. The supervised answer is a target index $i^\star$ for single-target localization (Nr3D and ScanRefer-style tasks) or a target set $Y^\star \subseteq \{1, \ldots, N\}$ for set-level grounding such as Multi3DRef.

Learning Setting. We study supervised instance-level 3D grounding in a 3D-LLM that also supports general language generation. In S3G, D is the scorer dimension, K the latent-step budget, and M the memory-token count; `<geom>` activates grounding, with reserved step and memory tokens τ_k^{step} and τ_j^{mem}. Auxiliary referential cues supervise latent steps during training only and are not part of the inference input.

Problem Definition. In grounding mode, SSR3D-LLM predicts $\hat{i} = \arg\max_i \boldsymbol{\ell}_i$ for single-target evaluation, or thresholds/ranks logits under the benchmark protocol to obtain $\hat{Y} \subseteq \{1, \ldots, N\}$ for set-level evaluation. All metrics are computed after the candidate matching or restriction required by each benchmark.

4 Method

SSR3D-LLM keeps the unified 3D-LLM backbone and the Mask3D-based candidate pipeline, but replaces the single *QPG* pointer with a structured grounding route. When `<geom>` is present, proposal, geometry, optional appearance, and proposal-side predicted-class evidence form object tokens; the LLM writes latent spatial reasoning steps and memory tokens; and a geometry-aware scorer refines candidate rankings step by step. Without this task token, the model follows the default dialog/QA/caption path. S3G is an interface-level design trained around Mask3D proposal tokens, latent-step supervision, and a candidate-level scoring head. Figure 2 gives the route, while Figure 3 details the latent-step workspace, scorer, and mask.

4.1 3D Candidate Space

SSR3D-LLM grounds over point-cloud-derived Mask3D object proposals [32]; the *QPG* comparison uses the same proposal rows, proposal-level representation protocol, and evaluation mapping. For each proposal i, we combine its pooled proposal feature $\mathbf{f}_i$, rotated-box geometry $\mathbf{b}_i$, optional

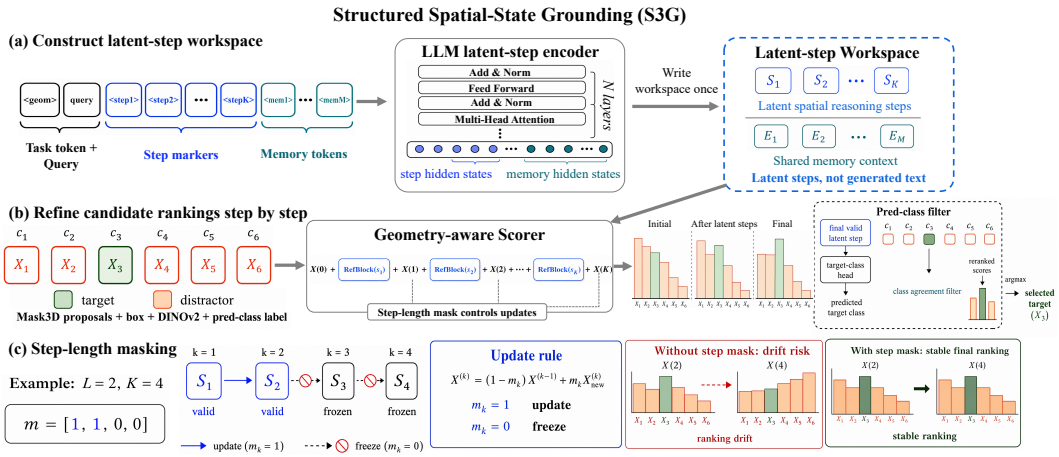


Figure 3: **S3G mechanism.** (a) The LLM writes a latent-step workspace in one forward pass by reading hidden states at reserved step markers and memory tokens. (b) A geometry-aware scorer reads these latent steps to refine Mask3D proposal rankings step by step, then a pred-class filter uses the predicted target class for final reranking. (c) Step-length masking determines which steps update candidate states. One forward pass still writes all K slots, so freezing later states does not remove the cost of producing those slots.

DINOv2 multi-view appearance $\mathbf{a}_i$, and proposal-side predicted-class label $\mathbf{c}_i$:

$$\mathbf{x}_i^{(0)} = W_f\mathbf{f}_i + W_b\mathbf{b}_i + \alpha W_a\mathbf{a}_i + W_c\mathbf{c}_i, \quad \mathbf{X}^{(0)} = [\mathbf{x}_1^{(0)}; \ldots; \mathbf{x}_N^{(0)}]. \tag{1}$$

Here $\mathbf{x}_i^{(0)} \in \mathbb{R}^D$ is the initial object candidate embedding for the i-th Mask3D proposal, $\mathbf{X}^{(0)} \in \mathbb{R}^{N\times D}$ stacks the N candidate embeddings, W_f, W_b, W_a, W_c are learned projections, and α=0 disables DINOv2.

4.2 Constructing Latent Spatial Reasoning Steps

The latent-step workspace uses K reserved *step-marker* tokens and M learnable *memory* tokens. Step-marker hidden states serve as ordered latent spatial reasoning steps, while memory tokens carry shared query context. For query u, the inference-time grounding prompt is the token sequence $p_{\text{pred}} = [\texttt{<geom>}, u, \tau_{1:K}^{\text{step}}, \tau_{1:M}^{\text{mem}}]$. This layout lets the steps form representations after reading the query, followed by memory tokens that summarize them; the scorer reads both. In a causal decoder, token positions determine the preceding context available to each token, so this is an information-flow design rather than a uniquely optimal ordering. Let T_p be the number of prompt tokens before the memory tokens. One LLM forward gives hidden states $\mathbf{H} \in \mathbb{R}^{(T_p+M)\times d_{\text{llm}}}$, which are projected into scorer space:

$$\begin{aligned} \mathbf{s}_k &= \text{LN}(W_s\,\mathbf{H}_{\text{pos}(\tau_k^{\text{step}})}) \in \mathbb{R}^D, \quad k = 1, \ldots, K, \\ \mathbf{e}_j &= \text{LN}(W_m\,\mathbf{H}_{T_p+j}) \in \mathbb{R}^D, \quad j = 1, \ldots, M, \end{aligned} \tag{2}$$

Here LN is layer normalization, W_s, W_m are learned projections, and $\text{pos}(\cdot)$ returns a token position in p_{pred}. This yields latent step representations $\mathbf{S} \in \mathbb{R}^{K\times D}$ and memory representations $\mathbf{E} \in \mathbb{R}^{M\times D}$. K is the reserved slot budget, whereas L is the effective length used for candidate-state updates. We use $K = 4$: among 7,485 Nr3D queries, approximately 98.8% have clipped lengths of at most three, and at most about 1.2% may be truncated from lengths above four (Appendix Table 20). This distribution supports the coverage of the chosen budget without establishing a universally optimal K. The default inputs are the scene and the original query: the fixed frontend constructs candidate representations from the scene, while the query and reserved markers construct the latent steps. Human-written cues, annotated lengths, and test ground-truth target classes are not inputs.

4.3 Step-Wise Candidate Refinement

Starting from $\mathbf{X}^{(0)}$, the scorer maintains a candidate-state matrix $\mathbf{X}^{(k)} = [\mathbf{x}_1^{(k)}; \ldots; \mathbf{x}_N^{(k)}] \in \mathbb{R}^{N\times D}$ after latent step k. At step k, it reads latent step representation $\mathbf{s}_k$ and memory $\mathbf{E}$, then applies a

masked update with step mask $m_k \in \{0, 1\}$:

$$\mathbf{X}^{(k)}_{\text{new}} = \text{ReferBlock}_k\left(\mathbf{X}^{(k-1)}, \mathbf{E}, \mathbf{s}_k\right), \quad \mathbf{X}^{(k)} = (1 - m_k)\mathbf{X}^{(k-1)} + m_k\mathbf{X}^{(k)}_{\text{new}}. \tag{3}$$

ReferBlock lets candidates attend to memory under the current step cue, mix with each other, and undergo a small feed-forward refinement (Appendix Algorithm 1). Because the initial candidate embeddings already combine Mask3D proposal features, rotated-box geometry, optional DINOv2 appearance, and predicted-class labels, the updates remain tied to the fixed proposal set. Candidate logits are produced by a shared linear readout, $\ell_i^{(k)} = \mathbf{w}_o^\top \mathbf{x}_i^{(k)} + b_o$, with final logits $\boldsymbol{\ell} = \boldsymbol{\ell}^{(K)}$; intermediate logits form the trace.

4.4 Handling Variable-Length Queries

Queries require different numbers of useful steps, so candidate states remain unchanged after the effective length. For $L \in \{1, \ldots, K\}$, we set $m_k = 1$ when $k \le L$ and $m_k = 0$ otherwise. During training, cue annotations provide effective-length supervision. Default inference uses model-predicted $\hat{L}$; only the oracle-L diagnostic uses annotated lengths. The length predictor outputs $q_k = \sigma(g_\psi(\mathbf{s}_k)) \in (0, 1)$, where σ is the sigmoid and g_ψ is a learned scalar head; q_k is the probability that step k lies beyond the useful boundary. With threshold $\eta = 0.5$, we stop before the first boundary hit and retain at least one step: $\hat{L} = \text{clip}_{1:K}(\min(\{k - 1 : q_k \ge \eta\} \cup \{K\}))$, where $\text{clip}_{1:K}$ clips the length to $\{1, \ldots, K\}$. The default mask is $m_k = \mathbf{1}\{k \le \hat{L}\}$. For example, $q_1 = 0.05, q_2 = 0.12, q_3 = 0.83$ gives $\hat{L} = 2$, so candidate states stop updating after step 2. At inference, one LLM forward first produces all of $(\mathbf{S}, \mathbf{E})$; the model then predicts the length mask and updates candidates using Eq. (3). Masking does not remove the cost already incurred in producing all K slots. The *pred-class filter* predicts a target class from the final valid step $\mathbf{s}_{\hat{L}}$ and reranks candidates with matching proposal-side classes at final selection. The query-side class is model-predicted, whereas proposal-side class representations already contribute to $\mathbf{X}^{(0)}$. Both differ from the length mask m_k, which controls state updates; default filtering does not use test ground-truth classes. Step logits $\boldsymbol{\ell}^{(k)}$ and $m_{1:K}$ form an inspectable non-textual trace.

4.5 Training Objectives and Protocol

Training uses answer supervision and auxiliary cue supervision. The answer is the standard benchmark label $i^\star$ or $Y^\star$: the target candidate in ReferIt3D, and the target box or target set matched to Mask3D proposal rows for ScanRefer and Multi3DRef. Cues and their effective lengths are training supervision; the oracle-L diagnostic additionally uses annotated lengths, which are not default test inputs. For Sr3D, we deterministically convert the annotated target class, relation, and context-object fields into ordered cue phrases. For Nr3D, which provides open-form utterances, offline Qwen3-32B served through vLLM extracts target, context-object, and relation cues from the query; object names in the resulting training labels are then normalized to the training vocabulary when needed. For ScanRefer and Multi3DRef, the same offline annotator uses the expression and, when available in the training annotation, the target class/name field to produce referential cue phrases in the same target/context-object/relation format. The annotation prompt requests JSON with helpful anchors before the target and the target last; Figure 10 gives the template. Appendix A.10 reports a human audit of training cues; this label processing adds no default test inputs. For a training example with L cues, the first L latent steps match their text representations and the remaining steps are masked by m_k. Phase C and default inference use p_{pred}: only reserved step and memory markers follow the original query, with no cue phrases in the prompt. Training may still use cue and length labels; inference generates latent steps and predicts their length from the query.

The training workflow comprises scorer warm-start, latent-step interface training, and no-cue adaptation. The scorer is first warm-started with the candidate-ranking objective using frozen LLM text features as language context. The LLM-side workspace is then trained so that $\mathbf{S}$ and $\mathbf{E}$ replace the fixed text features; a frozen BERT encoder supplies step targets $\{\mathbf{t}_k\}$ and the full-query target $\mathbf{t}_g$. Within interface training, Phase A fixes the scorer and uses cue distillation; Phase B continues training the scorer with auxiliary language-class supervision. Phase C follows B with the no-cue inference prompt, adapting the model to form latent steps from the query itself. Base LLM weights remain frozen, and the LLM side uses lightweight adapters and projections; Phase B also trains the scorer. Appendix Table 15 gives the A/B/C settings; scorer warm-start is separate from Phase A. For single-target examples, grounding logits use target cross-entropy: $\mathcal{L}_{\text{ref}} = \text{CE}(\text{softmax}(\boldsymbol{\ell}), i^\star)$. For

target-set examples, $\mathcal{L}_{\text{ref}}$ is the corresponding multi-positive objective over matched candidates in $Y^\star$, consistent with the set prediction metrics in Section 3. Latent-step supervision uses cosine losses:

$$\mathcal{L}_{\text{step}} = \frac{1}{\sum_k m_k} \sum_{k=1}^{K} m_k \big(1 - \cos(\mathbf{s}_k, \mathbf{t}_k)\big), \quad \mathcal{L}_{\text{global}} = 1 - \cos(\mathbf{g}, \mathbf{t}_g). \tag{4}$$

where $\mathbf{g} = \mathbf{e}_1$ is the global memory vector. For ScanRefer-style identification, a target-class head on the last valid latent step adds $\mathcal{L}_{\text{cls}}$ and supplies the query-side predicted class at inference. Here CE is cross-entropy, cos is cosine similarity, and the λ's are loss weights. The grounding and latent-workspace training objective is

$$\mathcal{L} = \mathcal{L}_{\text{ref}} + \lambda_{\text{step}}\mathcal{L}_{\text{step}} + \lambda_{\text{global}}\mathcal{L}_{\text{global}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}. \tag{5}$$

Cue lengths additionally provide supervision for the length predictor; at inference, g_ψ predicts the mask from model-generated steps without reading annotated lengths.

5 Experiments

5.1 Setup and Protocol

We evaluate on three main grounding benchmarks. Nr3D/ReferIt3D (Table 1) uses official candidate-set Top-1 accuracy and tests fine-grained object selection; ScanRefer and Multi3DRef (Table 2) test box localization under proposal noise and set-level grounding. Additional comparisons, language examples, and trace diagnostics are in the appendix.

For ReferIt3D, we use the official supervised splits: 28,716 train / 7,485 test examples for Nr3D and 65,844 train / 17,726 test examples for Sr3D. Across benchmarks, the supervised answer label is the dataset target object, box, or target set after proposal matching; auxiliary referential cues supervise only latent steps during training. Unless stated otherwise, SSR3D-LLM uses $K{=}4$ latent spatial reasoning steps and $M{=}16$ memory tokens, with the Mask3D proposal representation from Section 4.1. Default grounding uses predicted-L: steps, length, and query-side class are generated from the expression and Mask3D proposals without human cues or annotated length. Training cues supervise steps and length; oracle-L is diagnostic only (Appendix A.7). Language-capability experiments run the same checkpoint on ScanQA, Scan2Cap, object description, and embodied dialog through the default pathway without `<geom>`.

We organize baselines by system paradigm. Specialized and task-specific grounding systems provide task-level references on the same benchmarks, showing how methods optimized directly for 3D grounding perform. Reasoning-centric and agentic vision-LLM pipelines show what recent large-model grounding stacks can reach when they use rendered views, external prompting, or verification. Our main interface comparison is the Unified 3D-LLMs block, where methods share the goal of preserving a general 3D-LLM while adding grounding. Appendix Tables 7–8 summarize the fuller interface context. For unified baselines with internal object indices, the fixed mapping in Appendix A.4 converts predictions to the same ReferIt3D candidate protocol.

5.2 Main Grounding Benchmark Results

Table 1 reports official Nr3D candidate-set Top-1 accuracy. The Grounded 3D-LLM and Chat-Scene rows are ReferIt3D adaptations: both are finetuned on the official training split, and their internal pointer or object-ID outputs are converted to candidate predictions using the fixed mapping in Appendix A.4. SSR3D-LLM reaches 47.2% with predicted-L, exceeding Grounded 3D-LLM's 32.8% by 14.4 percentage points and Chat-Scene's 25.9% by 21.3 points. Grounded 3D-LLM provides a more direct interface comparison through its *QPG* readout under the same proposal-level representation protocol. Chat-Scene provides a supplementary reference, with differences in backbone, object-token construction, output interface, and training adaptation; the ReferIt3D adaptation may also underestimate these baselines.

Table 2 extends the comparison to ScanRefer and Multi3DRef. On ScanRefer, SSR with predicted-L reaches 55.8 / 51.5, exceeding Chat-Scene's 55.5 / 50.2 by 0.3 / 1.3 percentage points and Grounded 3D-LLM's 48.6 / 44.0 by 7.2 / 7.5 points. On Multi3DRef, SSR reaches 62.6 / 57.5, exceeding Chat-Scene's 57.1 / 52.4 by 5.5 / 5.1 points and Grounded 3D-LLM's 44.7 / 40.8 by 17.9 / 16.7 points. These two benchmarks also avoid the limitations introduced by ReferIt3D adaptation. The results

Table 1: **Nr3D grounding results.** We report candidate-set Top-1 accuracy (%) on the 7,485-query official test split. SSR's Overall and three-seed 95% CI follow the reported predicted-L setting with predicted-class filtering (Appendix Table 11). Subgroup scores are from a separate single-checkpoint predicted-L evaluation (3,536/7,485 correct overall); its class-filter configuration is unverified. Grounded 3D-LLM and Chat-Scene are adapted on the official training split and use a fixed instance-to-candidate mapping. Grounded uses the *QPG* interface under the same proposal-level representation protocol; Chat-Scene differs in backbone, object tokens, and training adaptation. FT denotes grounding-specific training or finetuning (✓/×); "–" denotes an unreported split; bold marks the best Overall result among the unified 3D-LLMs listed here.

Method	Backbone	FT	Overall	Easy	Hard	V-Dep.	V-Ind.
Specialized / task-specific grounding systems							
ReferIt3DNet (ECCV'20)	3D listener	✓	35.6	43.6	27.9	32.5	37.1
WS-3DVG (ICCV'23)	3DVG net	✓	22.5	27.3	18.0	21.9	22.9
M3DRef-CLIP (ICCV'23)	CLIP	✓	49.4	55.6	43.4	42.3	52.9
ZSVG3D (CVPR'24)	GPT-4 + CLIP	×	39.0	46.5	31.7	36.8	40.0
MiKASA (CVPR'24)	Transformer	✓	64.4	69.7	59.4	65.4	64.0
SeeGround (CVPR'25)	Qwen2-VL-72B	×	46.1	54.5	38.3	42.3	48.2
LaSP (EMNLP'25)	LLM program	×	52.9	60.7	45.3	49.2	54.7
ViGOR (WACV'25)	3D listener	✓	59.7	66.6	53.1	59.2	59.9
SORT3D (arXiv'25)	LLM toolbox	×	62.0	–	–	56.6	64.3
Reasoning-centric / agentic large-model pipelines							
VLM-Grounder (CVPR'24)	GPT-4V	×	48.0	55.2	39.5	45.8	49.4
SPAZER (arXiv'25)	GPT-4o	×	63.8	68.0	58.8	59.9	66.2
Unified 3D-LLMs							
Chat-Scene (NeurIPS'24)	Vicuna-7B	✓	25.9	31.0	21.4	21.6	28.3
Grounded 3D-LLM (arXiv'24)	Tiny-Vicuna-1B	✓	32.8	40.3	25.5	29.7	34.3
SSR3D-LLM (predicted-L)	Tiny-Vicuna-1B	✓	**47.2**	57.3	37.6	42.0	49.9

support grounding gains over the two listed unified baselines; the higher GPT4Scene-HDM scores indicate that stronger rendered-view reasoning stacks remain complementary to this interface design.

Appendix A.11 relates these results to reported values of the concurrent Chat-Scene++ [18].

5.3 Interface Evidence and Language Trade-offs

Contribution of the structured interface. Spatial-Cue-Chain-of-Thought (SCCT) uses the same cues and samples as textual targets but does not reproduce SSR's gains (Table 3). Removing query-side filtering lowers SSR to 42.8, still above Grounded; proposal-side class representations remain. These controls support an interface contribution without excluding all training differences (Appendix A.7).

Mechanism ablations. Table 4 reports length/mask comparisons and fixed-model step perturbations under their respective reference settings. The Nr3D oracle-to-predicted-L drop is significant at 3.1 points; removing the mask changes 47.2 to 46.9 ($p = 0.5$), without a significant accuracy benefit. Shuffling or reversing valid steps degrades the separate mechanism runs, showing use of order, not unique optimality. These perturbations differ from cue-order retraining (Appendix A.7).

Language functions and their cost. The same checkpoint retains its language route without `<geom>`, but performance changes: Scan2Cap improves and the other tasks decline, including Dialog CIDEr by 12.6% (Table 5; protocols/examples: Appendix A.8).

Supporting cue, budget/order, Phase C, and 7B diagnostics appear in Appendices A.10–A.11.

Figure 4 illustrates the same mechanism on a concrete case. The single-pointer baseline can lock onto a visually plausible but relationally wrong object, while S3G first stabilizes context-object evidence, then uses the later spatial cue to separate competing candidates. The visualization is meant to show

Table 2: **ScanRefer and Multi3DRef grounding results.** ScanRefer reports Acc@0.25IoU / Acc@0.50IoU; Multi3DRef reports F1@0.25IoU / F1@0.50IoU (all in %). SSR uses predicted-L and predicted-class filtering without human-provided cues or oracle length. SSR's ScanRefer values are means over three training seeds, with 95% CIs in Appendix Table 11; Multi3DRef values are presented as point estimates. FT denotes grounding-specific training or finetuning (✓/×). "–" denotes an unreported result or an unevaluated output type; bold marks the best result among Grounded 3D-LLM, Chat-Scene, and SSR within this table. Native-protocol Chat-Scene++ references are listed separately in Appendix Table 24.

Method	Backbone	FT	ScanRefer		Multi3DRef	
			A@0.25	A@0.50	F1@0.25	F1@0.50
Specialized / task-specific grounding systems						
WS-3DVG (ICCV'23)	3DVG net	✓	27.37	21.96	–	–
M3DRef-CLIP (ICCV'23)	CLIP	✓	–	–	42.8	38.4
ZSVG3D (CVPR'24)	GPT-4 + CLIP	×	36.4	32.7	–	–
SeeGround (CVPR'25)	Qwen2-VL-72B	×	44.1	39.4	–	–
Reasoning-centric / agentic large-model pipelines						
VLM-Grounder (CVPR'24)	GPT-4V	×	51.6	32.8	–	–
SPAZER (arXiv'25)	GPT-4o	×	57.2	48.8	–	–
GPT4Scene (arXiv'25)	GPT-4o	×	40.5	36.7	45.4	42.1
GPT4Scene-HD (arXiv'25)	GPT-4o	×	50.9	46.4	53.7	50.0
GPT4Scene-HDM (arXiv'25)	GPT-4o	×	62.6	57.0	64.5	59.8
Unified 3D-LLMs						
Chat-Scene (NeurIPS'24)	Vicuna-7B	✓	55.5	50.2	57.1	52.4
Grounded 3D-LLM (arXiv'24)	Tiny-Vicuna-1B	✓	48.6	44.0	44.7	40.8
SSR3D-LLM	Tiny-Vicuna-1B	✓	**55.8**	**51.5**	**62.6**	**57.5**

Table 3: **Cue and class-filter controls (%).** Metrics follow Tables 1–2. Both SSR rows use predicted-L; only query-side filtering is removed. Default SSR Nr3D/ScanRefer values are three-seed means; other cells are point estimates.

Method / setting	Nr3D	ScanRefer	Multi3DRef
Grounded 3D-LLM	32.8	48.6 / 44.0	44.7 / 40.8
Grounded 3D-LLM + SCCT	30.3	47.2 / 42.7	44.0 / 40.4
SSR3D-LLM (predicted-L)	47.2	55.8 / 51.5	62.6 / 57.5
w/o query-side class filter	42.8	52.8 / 48.8	62.4 / 57.3

Table 4: **Mechanism ablations (accuracy, %).** Shuffle/reverse perturb valid steps of a fixed model under oracle-L. Mask removal uses predicted-L for Nr3D and oracle-L for Sr3D. Nr3D length/mask statistics are in Appendix Table 12; other mechanism results are separate point estimates. Sr3D uses a separate setting from the default 69.5.

Variant	Nr3D	Sr3D
Oracle-L	50.3	67.5
+ shuffle valid steps	46.2	56.8
+ reverse valid steps	41.7	46.3
Predicted-L	47.2	67.0
w/o step mask (fixed K)	46.9	59.1

what the score trace adds beyond the final prediction: each step exposes which candidates remain plausible and when the target becomes separated.

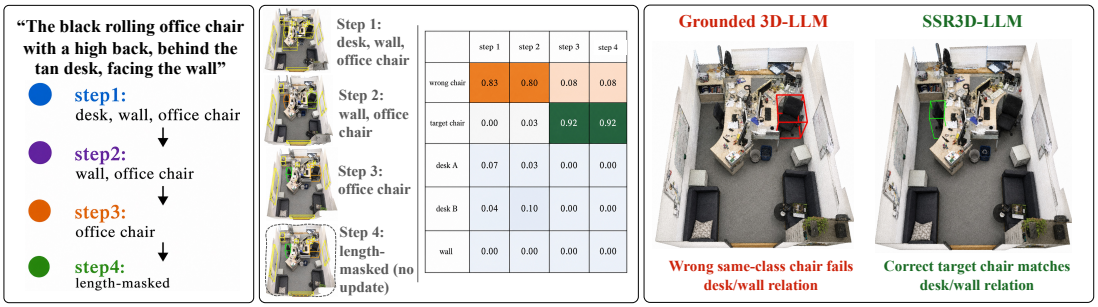
Readout and end-to-end cost. Readout speedups are 17.74×/22.91× over Grounded/Chat-Scene (Table 6). E2E is faster on Nr3D but slower than Grounded on ScanRefer/Multi3DRef, so speedups

Table 5: **Default language-path results.** Base: Grounded 3D-LLM QPG. Both use Tiny-Vicuna-1B, matched settings, and one evaluation run. Δ: relative change, except ScanQA in percentage points (pp); CIDEr retains its reported scale.

Task	Metric	Base	SSR	Δ
Dialog	CIDEr	1.247	1.090	-12.6%
ScanQA	EM (%)	20.11	19.83	-0.28 pp
Scan2Cap	CIDEr@0.5	0.626	0.658	$+5.1\%$
Object description	CIDEr	1.052	1.027	-2.4%

Table 6: **Mean latency (s/query).** Same GPU worker, batch 1. Readout excludes front-end costs; E2E includes per-query visual features and proposals, excluding only one-time loading. Matched conditions and full statistics: Appendix A.9. Readout is separate, not per-benchmark.

Method	Readout	End-to-end (E2E)		
		Nr3D	ScanRefer	Multi3DRef
Grounded 3D-LLM	1.0662	3.449	2.957	2.745
Chat-Scene	1.3772	57.581	44.083	44.041
SSR3D-LLM	0.0601	0.827	8.120	8.325



(a) Input and latent step trace (b) Step-wise grounding scores (c) Final prediction comparison

Figure 4: **Qualitative grounding case.** A real ScanRefer example shows the query, step-wise candidate scores, proposal boxes, and baseline vs. SSR3D-LLM predictions. The masked fourth slot preserves the step-3 ranking; cue labels are visualization-only annotations for the figure.

are not universal. All latent steps use one forward pass without autoregressive trace decoding; timing scopes and full statistics are in Appendix A.9.

Full efficiency results are in Appendix Tables 16–17; GT-rank flow, score heatmaps, and reliability plots are in Appendix Figures 13–17.

Remark. The experiments isolate S3G over fixed Mask3D candidates and proposal-level representations; stronger proposal sources, rendered views, and richer visual inputs remain complementary.

6 Conclusion

We propose SSR3D-LLM, which reframes fine-grained grounding in unified 3D-LLMs as an interface problem. S3G replaces single-pointer *QPG* with latent spatial reasoning steps: the LLM writes a workspace from the query, and a geometry-aware scorer ranks fixed Mask3D candidates step by step. Auxiliary referential cues supervise this workspace during training, but inference uses only the original query and object proposals. Across ReferIt3D, ScanRefer, and Multi3DRef, this route improves over *QPG* and the evaluated unified 3D-LLM baselines while keeping the language-task pathway functional, with task-dependent changes. The current system still depends on upstream proposal quality and uses a fixed latent-step budget; extending S3G to stronger proposal sources, richer visual inputs, adaptive or longer latent steps, and broader 3D-LLM backbones are natural next steps.

Acknowledgments and Disclosure of Funding

The research work described in this paper was supported by the HKUST – HKUST(GZ) Cross-campus Research Collaboration "1+1+1" Joint Funding Program (grant#C_2025_028). It was partially conducted in JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

This work was also supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62602454 and the Suzhou Young Elite Scientists Development Program under Grant No. SZTJ-2026-A031.

We thank the PhD students who assisted with the quality assessment of the referential cues.

Competing interests. The authors declare no competing interests.

References

[1] Eslam Abdelrahman, Mohamed Ayman, Mahmoud Ahmed, Habib Slim, and Mohamed Elhoseiny. Cot3dref: Chain-of-thoughts data-efficient 3d visual grounding. *arXiv preprint arXiv:2310.06214*, 2023.

[2] Ahmed Abdelreheem, Kyle Olszewski, Hsin-Ying Lee, Peter Wonka, and Panos Achlioptas. Scanents3d: Exploiting phrase-to-3d-object correspondences for improved visio-linguistic models in 3d scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3524–3534, 2024.

[3] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. *16th European Conference on Computer Vision (ECCV)*, 2020.

[4] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19129–19139, 2022.

[5] Chun-Peng Chang, Shaoxiang Wang, Alain Pagani, and Didier Stricker. Mikasa: Multi-key-anchor & scene-aware transformer for 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14131–14140, 2024.

[6] Dave Chen, Angel Chang, and Matthias NieSSner. *ScanRefer: 3D Object Localization in RGB-D Scans Using Natural Language*, pages 202–221. 11 2020. ISBN 978-3-030-58564-8. doi: 10.1007/978-3-030-58565-5_13.

[7] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Language conditioned spatial relation reasoning for 3d object grounding. *Advances in neural information processing systems*, 35:20522–20535, 2022.

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

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[10] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[11] Jiading Fang, Xiangshan Tan, Shengjie Lin, Igor Vasiljevic, Vitor Guizilini, Hongyuan Mei, Rares Ambrus, Gregory Shakhnarovich, and Matthew R Walter. Transcrib3d: 3d referring expression resolution through large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9737–9744. IEEE, 2024.

[12] Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-llm: Extending language model for 3d visual reasoning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2195–2206. IEEE, 2025.

[13] Liang Geng and Jianqin Yin. Viewinfer3d: 3d visual grounding based on embodied viewpoint inference. *IEEE Robotics and Automation Letters*, 9:7469–7476, 2024. doi: 10.1109/LRA.2024.3426286.

[14] Ziyu Guo, Yiwen Tang, Renrui Zhang, Dong Wang, Zhigang Wang, Bin Zhao, and Xuelong Li. Viewrefer: Grasp the multi-view knowledge for 3d visual grounding with gpt and prototype guidance. *arXiv preprint arXiv:2303.16894*, 2023.

[15] Dailan He, Yusheng Zhao, Junyu Luo, Tianrui Hui, Shaofei Huang, Aixi Zhang, and Si Liu. TransRefer3D: Entity-and-relation aware transformer for fine-grained 3d visual grounding. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2344–2352, 2021. doi: 10.1145/3474085.3475397. URL `https://arxiv.org/abs/2108.02388`.

[16] 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.

[17] 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.

[18] Haifeng Huang, Yilun Chen, Zehan Wang, Jiangmiao Pang, and Zhou Zhao. Chat-Scene++: Exploiting context-rich object identification for 3D LLM. *arXiv preprint arXiv:2603.27507*, 2026.

[19] Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang. Multi-view transformer for 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15524–15533, 2022.

[20] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. In *European Conference on Computer Vision (ECCV)*, 2024.

[21] Zhao Jin, Rong-Cheng Tu, Jingyi Liao, Wenhao Sun, Xiao Luo, Shunyu Liu, and Dacheng Tao. Spazer: Spatial-semantic progressive reasoning agent for zero-shot 3d visual grounding, 2025. URL `https://arxiv.org/abs/2506.21924`.

[22] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9579–9589, 2024.

[23] Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[24] Daizong Liu, Yang Liu, Wencan Huang, and Wei Hu. A survey on text-guided 3d visual grounding: Elements, recent advances, and future directions. *arXiv preprint arXiv:2406.05785*, 2024.

[25] Jiazhen Liu, Mingkuan Feng, and Long Chen. Better, stronger, faster: Tackling the trilemma in mllm-based segmentation with simultaneous textual mask prediction, 2025. URL `https://arxiv.org/abs/2512.00395`.

[26] Weichen Liu, Qiyao Xue, Haoming Wang, Xiangyu Yin, Boyuan Yang, and Wei Gao. Spatial reasoning in multimodal large language models: A survey of tasks, benchmarks and methods. *arXiv preprint arXiv:2511.15722*, 2025.

[27] Junyu Luo, Jiahui Fu, Xianghao Kong, Chen Gao, Haibing Ren, Hao Shen, Huaxia Xia, and Si Liu. 3d-sps: Single-stage 3d visual grounding via referred point progressive selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18212–18221, 2022.

[28] Boyu Mi, Hanqing Wang, Tai Wang, Yilun Chen, and Jiangmiao Pang. Language-to-space programming for training-free 3d visual grounding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3864, 2025.

[29] Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models, 2025. URL `https://arxiv.org/abs/2501.01428`.

[30] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. Languagerefer: Spatial-language model for 3d visual grounding. *arXiv preprint arXiv:2107.03438*, 2021.

[31] Li Ruihuang, Zhang Zhengqiang, He Chenhang, Ma Zhiyuan, Patel Vishal M., and Zhang Lei. Dense multimodal alignment for open-vocabulary 3d scene understanding. In *ECCV*, 2024.

[32] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. 2023.

[33] Tung-Yu Wu, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Data-efficient 3d visual grounding via order-aware referring. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3107–3117. IEEE, 2025.

[34] Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. In *CoRL*, 2024.

[35] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. *arXiv preprint arXiv:2309.12311*, 2023.

[36] Jianing Yang, Xuweiyi Chen, Nikhil Madaan, Madhavan Iyengar, Shengyi Qian, David F Fouhey, and Joyce Chai. 3d-grand: A million-scale dataset for 3d-llms with better grounding and less hallucination. *arXiv preprint arXiv:2406.05132*, 2024.

[37] Ting Yu, Xiaojun Lin, Shuhui Wang, Weiguo Sheng, Qingming Huang, and Jun Yu. A comprehensive survey of 3d dense captioning: Localizing and describing objects in 3d scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(3):1322–1338, 2023.

[38] Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20623–20633, 2023. doi: 10.1109/CVPR52733.2024.01949.

[39] Nader Zantout, Haochen Zhang, Pujith Kachana, Jinkai Qiu, Guofei Chen, Ji Zhang, and Wenshan Wang. Sort3d: Spatial object-centric reasoning toolbox for zero-shot 3d grounding using large language models, 2025. URL `https://arxiv.org/abs/2504.18684`.

[40] Yiming Zhang, ZeMing Gong, and Angel X Chang. Multi3drefer: Grounding text description to multiple 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15225–15236, October 2023.

[41] Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3dvg-transformer: Relation modeling for visual grounding on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2928–2937, October 2021.

A Additional Results and Ablations

A.1 Discussion, Limitations, and Future Work

Scope of the claim. SSR3D-LLM targets the grounding interface inside unified, instance-centric 3D-LLMs. The main comparison therefore emphasizes unified 3D-LLM baselines that expose grounding through compact object-id or pointer-style readouts. Specialized grounding systems, agentic pipelines, and stronger vision-LLM stacks remain important broader references, but they often use different visual front ends, rendered views, external verification loops, or supervision protocols.

Limitations. Our experiments use fixed Mask3D proposal sets and ScanNet-style indoor benchmarks, so grounding quality is still bounded by upstream proposal recall, instance segmentation quality, and the domain coverage of the underlying datasets. Rotated-box geometry, DINOv2 appearance, and proposal-side predicted classes improve the proposal representation, but they do not replace the proposal generator. The latent steps are also shaped by training-time referential cues, whose coverage and noise can affect how cleanly the step-wise trace aligns with human-interpretable context objects and relations. We use a fixed budget $K = 4$ with step-length masking. The Nr3D length distribution bounds potential truncation at about 1.2% (Appendix Table 20), but applicability to longer or nested relational chains and universal optimality of this budget remain unestablished.

Future work. Promising directions include combining S3G with stronger open-vocabulary 3D perception, covering temporal, causal, and functional relations beyond spatial ones, extending evaluation beyond static indoor ScanNet-style scenes, and studying closed-loop use in embodied agents where grounding errors can affect downstream actions. Another direction is to reduce reliance on offline referential-cue construction by learning latent-step structure from weaker or self-supervised signals while preserving the default language pathway of unified 3D-LLMs.

A.2 Additional Qualitative Visualizations

Figures 5–8 provide additional qualitative grounding cases. They complement the main qualitative example by showing how candidate rankings evolve across latent steps and how the final prediction is selected from the fixed Mask3D proposal set. In all four examples, the final S3G selection matches the ground-truth target object. These cases illustrate ranking behavior only; they are neither an accuracy estimate nor evidence for the default predicted-L setting, whose results are reported in the main tables. Across these visualizations, yellow 3D boxes denote the Mask3D proposal candidates considered by the grounding model. Cyan and orange boxes highlight intermediate proposals that receive high scores or remain competitive at a latent step, green boxes mark the final S3G selection, and red boxes mark the baseline prediction when it selects a different object. The colors are visualization overlays for readability; the model operates on the same fixed proposal representations described in the 3D candidate-space subsection.

A.3 ReferBlock Details

What it does. At each step k, $\mathrm{ReferBlock}_k(\cdot)$ takes the current candidate tokens $\mathbf{X}^{(k-1)}$ and updates them using (i) the step cue $\mathbf{s}_k$ and (ii) the shared memory $\mathbf{E}$. Intuitively, candidates first "read" the global instruction from $\mathbf{E}$, then mix information across candidates, and finally apply a small feed-forward refinement. In Algorithm 1, LN denotes layer normalization, $\mathrm{CrossAttn}$ and $\mathrm{SelfAttn}$ are standard multi-head attention blocks, and MLP is a feed-forward block. The algorithm returns $\mathbf{X}^{(k)}_{\text{new}}$ before length masking; Eq. (3) then either applies this update or retains $\mathbf{X}^{(k-1)}$. An unchanged state does not imply that the LLM skipped the slot.

A.4 Unified Baseline Adaptation and Index-to-Candidate Evaluation

Grounded 3D-LLM and Chat-Scene use internal object-index outputs, so Table 1 evaluates ReferIt3D adaptations under the official candidate-set protocol. Both adapted unified baselines are finetuned on the official ReferIt3D training splits and evaluated on the official test split. Their grounding heads produce internal identifiers–a *QPG* backbone index for Grounded 3D-LLM, and an object-ID style output for Chat-Scene–rather than a native score for every ReferIt3D candidate. To evaluate

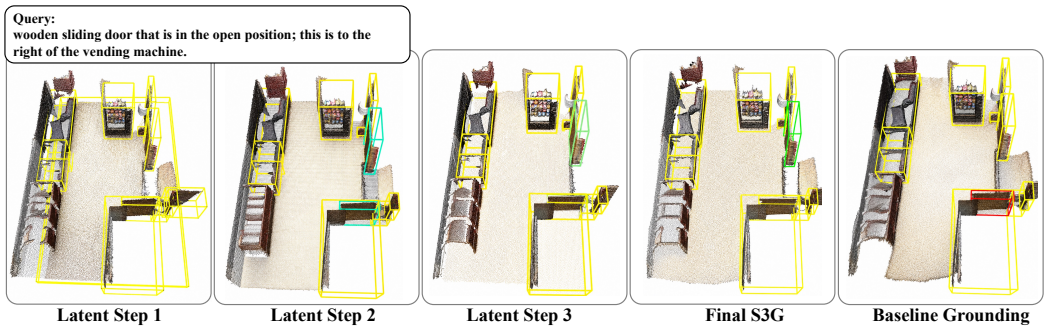


Figure 5: **Additional qualitative visualization I.**

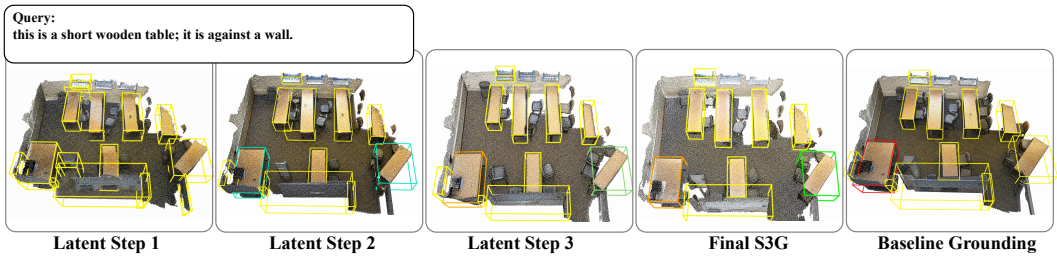


Figure 6: **Additional qualitative visualization II.**

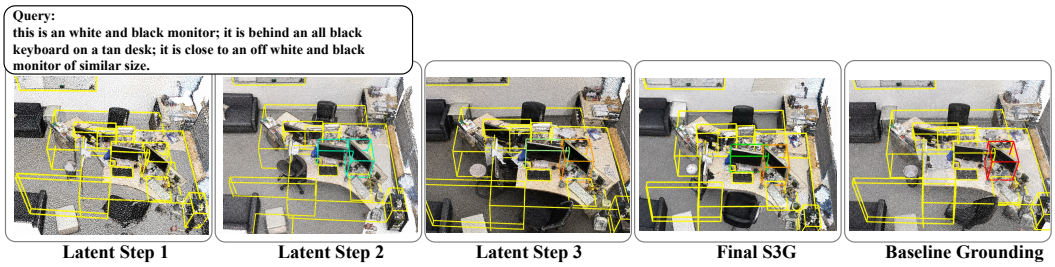


Figure 7: **Additional qualitative visualization III.**

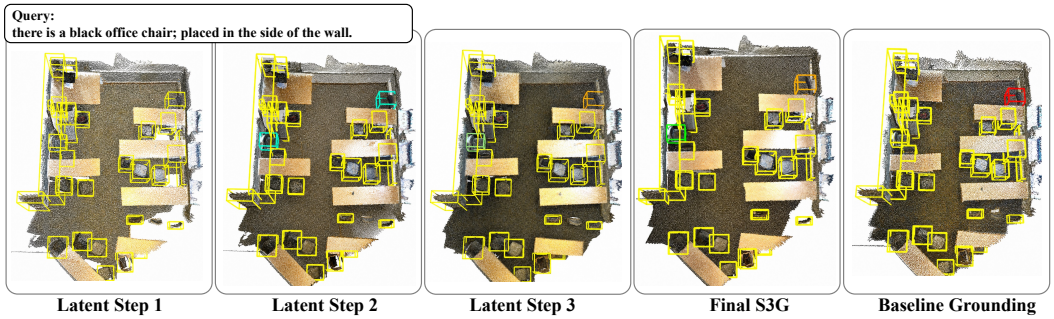


Figure 8: **Additional qualitative visualization IV.**

Algorithm 1 ReferBlock computation at step k.

Require: Candidate tokens $\mathbf{X}^{(k-1)} \in \mathbb{R}^{N\times D}$, memory $\mathbf{E} \in \mathbb{R}^{M\times D}$, step cue $\mathbf{s}_k \in \mathbb{R}^D$
Ensure: Updated tokens $\mathbf{X}^{(k)}_{\text{new}} \in \mathbb{R}^{N\times D}$
$\mathbf{Q} \leftarrow \mathbf{X}^{(k-1)}$
$\mathbf{Q} \leftarrow \mathbf{Q} + \mathrm{CrossAttn}(\mathrm{LN}(\mathbf{Q}), \mathrm{LN}(\mathbf{E}), \mathbf{s}_k)$ {read instruction with step cue}
$\mathbf{Q} \leftarrow \mathbf{Q} + \mathrm{SelfAttn}(\mathrm{LN}(\mathbf{Q}))$ {mix candidates}
$\mathbf{Q} \leftarrow \mathbf{Q} + \mathrm{MLP}(\mathrm{LN}(\mathbf{Q}))$
$\mathbf{X}^{(k)}_{\text{new}} \leftarrow \mathbf{Q}$

these outputs under the official candidate-set protocol, we map the predicted internal identifier to the corresponding candidate object using a fixed mapping derived from the same Mask3D perception outputs.

Definition. We define a *reference index* $j^\star$ by mapping the ReferIt3D target instance into the backbone index space using a fixed instance-to-index mapping built from the same Mask3D perception outputs. Given the model-predicted top-1 index $\hat{j}$, the prediction is correct iff $\hat{j} = j^\star$; invalid indices count as misses.

Evaluation protocol. All evaluations use (i) the official ReferIt3D splits and the same context construction/sampling procedure, (ii) the same fixed mapping (derived from the same Mask3D perception outputs) to define $j^\star$, and (iii) the same evaluation pipeline (mapping, filtering, and hit criterion), where we only swap the evaluated model. For the Grounded 3D-LLM *QPG* comparison, the candidate representation also uses the same CLASP-pretrained Mask3D features, rotated-box geometry, and DINOv2 appearance used by SSR3D-LLM; only the grounding readout changes. The Chat-Scene row uses related proposal-level visual evidence but keeps the corresponding Chat-Scene object-token pipeline, backbone, output interface, and training adaptation, so it is a supplementary unified-model reference rather than the matched interface baseline. The shared Mask3D proposal-generation and instance-to-candidate mapping pipeline controls transferred visual candidates, but ReferIt3D adaptation may still underestimate Chat-Scene or QPG-style baselines. The paired-subset readout replacement in Table 10 and the main ScanRefer/Multi3DRef comparisons provide complementary evidence with distinct scopes.

A.5 Auxiliary Representation Visualization

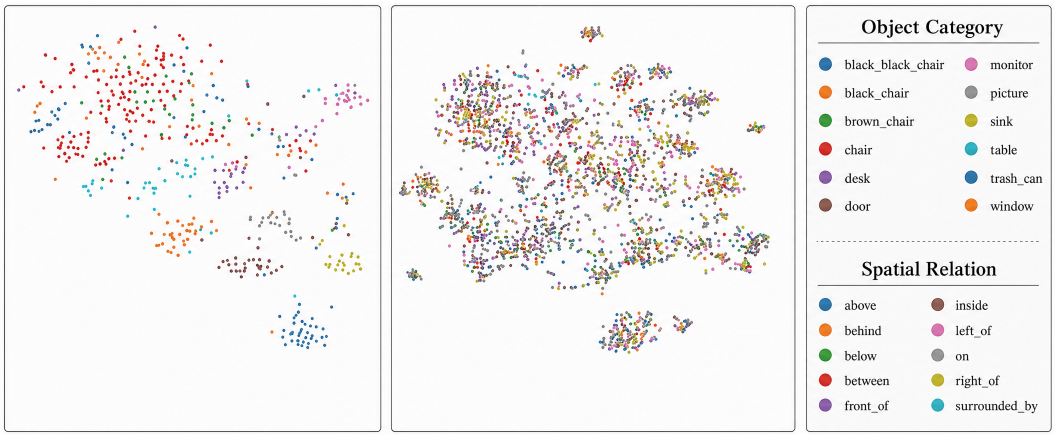
The main evidence comes from behavioral comparisons under the same candidate pipeline, proposal-level representation protocol, and backbone. The following visualization provides an auxiliary qualitative view of the learned pointer representation.

A.6 Additional Comparisons and Transfer Evidence

Interface positioning. This subsection adds interface positioning, predicted-L results on Sr3D, paired-subset transfer on Chat-Scene, and the cost of latent-step traces; length diagnostics and statistical analysis appear in Appendix A.7. Tables 7 and 8 separate task-specific grounding systems, reasoning-centric pipelines, and unified 3D-LLMs; rows outside the unified block provide literature context under different visual and inference protocols. SSR3D-LLM keeps the unified dialog/QA/captioning setting, but replaces compact single-readout grounding with latent-step candidate scoring over fixed Mask3D proposals.

Sr3D default results and filtering control. Table 9 reports overall accuracy on the full official Sr3D test split. SSR obtains 69.5% with predicted-L and 67.3% after removing only query-side predicted-class filtering, a drop of 2.2 percentage points. Both rows generate their own steps and lengths from the query and candidates, retaining the proposal-side representations.

Cross-backbone transfer. Table 10 is a paired-subset transfer diagnostic on Chat-Scene, using examples where annotations, proposals, and object-ID mappings can be aligned sample by sample. With the same Chat-Scene backbone and paired evaluation subset, replacing only its readout raises



(a) Object Category Semantics **(b) Spatial Relation Semantics**

Figure 9: **Auxiliary representation visualization for *QPG*.** The *QPG* representation separates object-category semantics more clearly than spatial-relation semantics. We use this as a qualitative view of the single-step bottleneck; Section 5.2 evaluates its effect on object selection.

Table 7: **Grounding interfaces across method families.** Comparison of task coverage, whether grounding uses an intermediate reasoning workspace, and what representation the final grounding readout uses.

Method	**3DVG**	**QA**	**Cap.**	**Dia-log**	**Reasoning workspace**	**Grounding readout**
Specialized / task-specific grounding systems						
ZSVG3D	✓	×	×	×	None	Feature / text matching
SeeGround	✓	×	×	×	None	VLM candidate matching
LaSP	✓	×	×	×	None	Spatial feature matching
SORT3D	✓	×	×	×	None	Object-relation matching
Reasoning-centric / agentic large-model pipelines						
VLM-Grounder	✓	×	×	×	External prompts	2D-to-3D verified matching
SPAZER	✓	×	×	×	External agent	Agentic candidate verification
GPT4Scene	✓	×	×	×	External prompts	Scene-level verification
Unified 3D-LLMs						
Grounded 3D-LLM	✓	✓	✓	✓	None	Single-token *QPG* pointer
Chat-Scene	✓	✓	✓	✓	None	Single-step object-ID token
SSR3D-LLM (ours)	✓	✓	✓	✓	**Latent steps**	**Reasoning-state scoring**

Nr3D from 25.9 to 30.3 and Sr3D from 31.5 to 40.5, showing that the S3G grounding route can improve another unified 3D-LLM backbone when the candidate mapping is available. These values apply only to that paired subset and are separate from official full-set results.

One-pass latent traces. All step representations are read from reserved markers in one LLM forward pass; the inspectable trace comes from step-wise candidate scores rather than generated text. This design avoids autoregressive trace decoding. Measured readout and end-to-end costs are reported separately in Tables 16 and 17.

A.7 Length Diagnostics, Control Protocols, and Statistics

Default inference and oracle diagnostics. Main results use predicted-L: inference takes the original expression and Mask3D candidates, and the model generates the steps, useful length, and query-side target class. Oracle-L instead supplies the useful length derived from referential-cue annotations to diagnose the cost of length prediction; it is not a standard inference input. Table 11

Table 8: **Integrated comparison with the unified 3D-LLM block highlighted.** This table clarifies which rows provide broader literature context and which rows share the instance-centric unified 3D-LLM setting.

Method	Input regime	Grounding interface
Specialized / task-specific grounding systems		
SeeGround	Mask3D proposals with rendered multi-view images and spatial text prompts	VLM scores candidate IDs over precomputed 3D detections using rendered-view evidence
Reasoning-centric / agentic large-model pipelines		
VLM-Grounder	Grounding DINO detections, SAM masks, and multi-view RGB-D projection cues	Grounds in 2D views, lifts SAM masks to 3D, then merges projected evidence
SPAZER	ScanNet detections with rendered 2D/3D views and large-model spatial verification prompts	Uses an agentic screening loop to verify candidate objects through 2D/3D observations
Unified 3D-LLMs		
Chat-Scene	Mask3D proposals with Uni3D object features and DINOv2 multi-view appearance	Single-step object-ID route selects a candidate token inside a unified 3D-LLM
Grounded 3D-LLM	Mask3D proposals encoded by the unified instance-centric 3D-LLM input pipeline	Single pointer-style *QPG* route selects a query-index token for the target candidate
SSR3D-LLM (ours)	**Mask3D proposals with rotated-box geometry and DINOv2 multi-view appearance as proposal-level feature enhancement**	**LLM writes latent spatial reasoning steps; a geometry-aware scorer ranks candidates step by step**

Table 9: **Sr3D results with predicted-L.** Candidate-set Top-1 accuracy (%) on the 17,726-query official test split. Both rows use no human-provided cues or annotation-derived length and are presented as point estimates; the no-filter row changes only query-side filtering at final candidate selection.

Setting	Overall
SSR3D-LLM (predicted-L)	69.5
w/o query-side class filter	67.3

compares these settings with predicted-class filtering enabled in both. Nr3D and ScanRefer values are means and 95% CIs over three training seeds, describing training variability; both Multi3DRef settings are presented only as point estimates.

Scope of SCCT and class-filter controls. SCCT gives Grounded 3D-LLM the same step-wise cue content and training examples as SSR, expressing the cues as causal textual targets, for example by prepending the original answer with `[SCCT begin] [step1] Door [step2] Table [step3] Chair [step4] NA [SCCT end]`. SSR uses these cues to supervise structured spatial states, whereas SCCT learns them through textual targets. The comparison tests whether textual supervision with matched cue content reproduces the gains; it does not exclude every optimization or training-procedure difference. The class-filter control removes final candidate filtering/reranking driven by the query-side predicted class under predicted-L, retaining proposal-side visual and predicted-class representations; it does not test removal of all class information. This class filter is distinct from the step mask that controls recursive updates in the following analysis.

Mechanism diagnostics under separate settings. Table 4 examines step order, length prediction, and the step mask on fixed proposal sets. Under oracle-L, shuffling or reversing valid steps at inference, without retraining, changes Nr3D from 50.3 to 46.2/41.7 and Sr3D from 67.5 to 56.8/46.3; these fixed-model perturbations test the learned alignment, whereas cue-order retraining in Table 21 relearns it. Predicted length gives 47.2 on Nr3D and 67.0 on Sr3D (-0.5 points from the 67.5 oracle-L reference). Step-mask removal is compared with predicted-L on Nr3D, from 47.2 to 46.9, and with oracle-L on Sr3D, where fixed-K updates give 59.1 (-8.4 points from 67.5). The Sr3D runs in this table use a separate setting from the default predicted-L result of 69.5 in Table 9. The

Table 10: **Cross-model transfer evidence on Chat-Scene.** This subset uses samples for which annotations, proposal outputs, and object-ID mappings can be paired, and complements the official full-set results reported in the main tables. It measures cross-backbone adaptation on the paired evaluation subset.

Method	Nr3D Overall	Sr3D Overall
Chat-Scene	25.9	31.5
Chat-Scene + SSR grounding	30.3	40.5

Table 11: **Separate oracle-L and predicted-L diagnostics.** All metrics are percentages. Nr3D and ScanRefer cells contain three-seed means and cross-seed 95% CIs; Multi3DRef cells contain point estimates. Oracle-L uses annotation-derived length and predicted-L uses model-predicted length; both retain predicted-class filtering.

Benchmark	Metric	Oracle-L	Predicted-L
Nr3D	Accuracy	50.3 [48.1, 52.4]	47.2 [45.2, 49.2]
ScanRefer	Acc@0.25	58.7 [54.9, 62.5]	55.8 [52.4, 59.2]
ScanRefer	Acc@0.50	53.9 [50.2, 57.6]	51.5 [48.3, 54.7]
Multi3DRef	F1@0.25	63.1	62.6
Multi3DRef	F1@0.50	57.9	57.5

Nr3D length and mask comparisons are three-seed means with the paired statistics below; the other mechanism results are point estimates.

Scene-clustered paired tests. Table 12 uses paired outcomes on the same Nr3D queries: 7,485 queries $\times$ 3 seeds, giving 22,455 paired query-seed observations, with scene-clustered bootstrap and two-sided paired tests over 312 scenes. Differences are setting B minus setting A in percentage points; their CIs and p-values quantify scene/test-query sampling uncertainty, not cross-seed uncertainty, and the 22,455 observations are not independent training repetitions. The -3.1-point oracle-L to predicted-L gap is significant, whereas the -0.3-point gap from removing the step mask relative to predicted-L is not. A nonsignificant difference does not establish equivalence between the settings.

A.8 Language Evaluation and Examples

Evaluation scope and settings. Table 13 lists the splits, actual sample counts, and inputs for Table 5. Base (Grounded 3D-LLM *QPG*) and SSR both use Tiny-Vicuna-1B and matched configurations, generating through the default language pathway without `<geom>` or S3G's spatial states. All four tasks use a single evaluation run: seed 1, batch size 1, three beams, no sampling, a maximum input length of 512 tokens, and at most 150 new tokens. The scoring-tool configuration is `pycocoevalcap` 1.2 with PTBTokenizer; the task-specific metrics are CIDEr, EM, or CIDEr@0.5. The evaluation scope is limited to the sets listed in the table.

Role of the examples. Table 14 retains randomly sampled inputs, scenes, and the two models' original answers, using a fixed sampling seed and the checkpoints and default language pathway described in Section 5.3. These examples illustrate output forms; they neither replace the four-task quantitative results nor establish unchanged task performance.

A.9 Reproducibility, Assets, and Responsible Use

Reproducibility. The main experiments use ReferIt3D/Nr3D/Sr3D, ScanRefer, and Multi3DRef with the official splits and metrics described in Section 5.1. Main-table grounding results use Mask3D proposal-based candidates, proposal matching or instance-to-candidate mapping, and fixed evaluation scripts so that methods are evaluated under the same candidate hit criterion within each benchmark protocol. The release package includes the training, evaluation, referential-cue preprocessing, mapping, and table-generation scripts for the main results and key controlled comparisons, together with configuration files and checkpoint files or checkpoint-reconstruction instructions. For external

Table 12: **Scene-clustered paired statistics on Nr3D.** The 22,455 paired query-seed observations cover 7,485 queries and 3 seeds, with 312 scenes as the bootstrap clusters. Δ denotes B$-$A in percentage points and tests are two-sided; both comparisons retain predicted-class filtering.

Setting A $\rightarrow$ B	**Accuracy**	**Δ**	**95% CI**	**p**
Oracle-L $\rightarrow$ predicted-L	$50.3 \rightarrow 47.2$	-3.1	$[-3.8, -2.3]$	$< 10^{-4}$
Predicted-L $\rightarrow$ w/o step mask	$47.2 \rightarrow 46.9$	-0.3	$[-1.8, +0.9]$	0.5

Table 13: **Four-task language evaluation scope.** Inputs omit <geom>; both models use the same splits, samples, and matched configurations. Results are in Table 5.

Task	**N**	**Split**	**Input**
Dialog	713	Test	Dialogue history
ScanQA	696	Val	Question
Scan2Cap	1,518	Test	Object/spatial prompt
Object description	3,032	Test	Object prompt

benchmarks and pretrained models, users should obtain the original assets from their official sources and follow the corresponding terms of use.

Training and experimental details. SSR3D-LLM uses a Tiny-Vicuna-1B unified 3D-LLM backbone, Mask3D proposals, $K{=}4$ latent spatial reasoning steps, $M{=}16$ memory tokens, rotated-box geometry, optional DINOv2 proposal appearance, and the target-class head/pred-class filter where reported. The scorer is first warm-started using frozen LLM text features and the candidate-ranking objective. Table 15 lists the subsequent A/B/C settings for the ReferIt3D latent-step model. Phase A keeps the scorer fixed and uses cue distillation; Phase B trains the scorer with a language-class auxiliary weight of 0.05. Phase C removes step phrases from the prompt and trains the reserved step markers from the original query, adapting the Phase B model to no-cue inference.

The LLM-side adapter applies LoRA to the last four Transformer layers' q- and v-projection matrices with rank 8, alpha 16, and dropout 0.0; base Tiny-Vicuna weights remain frozen in these stages. To retain the default language pathway, we keep the same multi-task SFT recipe as Grounded 3D-LLM over dialog, ScanQA, Scan2Cap, and object-description language data; inputs without <geom> are trained and evaluated through the default generation pathway. For ScanRefer and Multi3DRef, we continue from the Phase C checkpoint with trainable scorer and LLM-side adapter/projection parameters, initial learning rate 1×10^{-5}, scorer learning rate 1×10^{-5}, LLM-side learning rate 2×10^{-5}, and evaluation after every epoch. The selected ScanRefer run uses batch size 48, DINOv2 appearance weight $\alpha = 1.0$, and target-class loss weight 0.2; the selected Multi3DRef run uses batch size 40 and DINOv2 appearance weight $\alpha = 2.0$. The released configuration files specify the remaining command-level settings, optimizer choices, schedules, seeds, checkpoint paths, and run-specific overrides for the main results.

Training supervision and inference inputs. Figure 10 gives the offline annotation prompt for training cues: the annotator returns anchors and a target-last referential order for step supervision. A frozen BERT encoder supplies text targets for cues and the full query, corresponding to the step/global alignment losses in Eq. (4); target-class supervision supplies the classification term in Eq. (5). The effective lengths of annotated cues provide length supervision. Default inference uses the original query and Mask3D candidates, predicts $\hat{L} \in \{1, \ldots, K\}$ from steps produced in one forward pass, constructs m_k with threshold 0.5, and predicts the query-side class from the last valid step. The length mask controls candidate-state updates, whereas predicted-class filtering acts at final candidate selection. At test time, annotated lengths are used only for oracle-L diagnostics, and ground-truth target classes are not default filtering conditions.

Training compute resources. Training used NVIDIA A100 80GB GPU workers, with single-GPU execution for the final grounding adaptations.

Table 14: **Qualitative examples from the default language pathway (random samples).** Same checkpoints and the default route without <geom>; examples do not establish overall performance preservation. Quantitative comparisons are in Table 5.

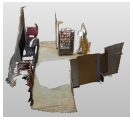
Scene	Task	Input	*QPG* baseline output	SSR3D-LLM output
 scene0019_01	Dialog	Can you describe the general layout of this setting?	Armchairs of curved backrest chairs have various features like leather and metallic legs.	Certainly! Of curved backrest chairs have black upholstery and metallic legs.
 scene0441_00	ScanQA	To the left of the urinal is what? Please answer with a single word or phrase.	toilet paper dispenser	toilet paper dispenser
 scene0389_00	Scan2Cap	Describe the aesthetics of the object and its orientation within its immediate vicinity.	there is a rectangular picture; it is on the wall above the bed.	there is a rectangular picture; it is on the wall above the bed.
 scene0568_01	ObjDesc	Could you give us a quick overview of the object?	A wooden chair with a vertical slatted back design, upholstered in light-colored fabric.	A black office chair with a mesh backrest.

Table 15: **ReferIt3D interface-training settings.** Scorer warm-start precedes these stages. Phase A/B use training cues, whereas Phase C removes cue phrases from the prompt; a no-cue prompt still permits training supervision. LR denotes learning rate.

Stage	LLM prompt	Main training role	Epochs / Batch size / LR
Phase A	Cue-conditioned	Fix scorer; train latent-step interface with cue distillation	20 / 64 / 1×10^{-4}
Phase B	Cue-conditioned	Train scorer with language-class auxiliary (weight 0.05)	50 / 32 / 5×10^{-5}
Phase C	No cue	Train reserved step markers from query for no-cue inference	50 / 64 / 1×10^{-4}

Efficiency measurement scopes. Tables 16 and 17 use the same GPU worker and batch size 1, with matched warm-up, measured queries, and timing boundaries. Readout maps prepared object/query representations to the final prediction, excluding front-end costs. E2E includes per-query visual-feature and proposal construction, excluding only one-time model loading. The three readout rows are not benchmark-specific results. Both tables report mean latency $\pm$ standard deviation and throughput; the readout multipliers are the reported latency ratios relative to SSR. Peak-memory units retain their reported scales: GB denotes 10^9 bytes and GiB denotes 2^{30} bytes, so the units are not interchangeable. Readout memory values are not reported.

Open assets. The public GitHub and Hugging Face repositories linked on the first page provide SSR3D-LLM training/evaluation code, S3G modules and configurations, referential-cue annotations or regeneration scripts, evaluation mappings, and trace/qualitative visualization scripts. We release derived annotations, code, and configs as allowed, while raw datasets, pretrained backbones, Mask3D assets, DINOv2 models, and LLM weights remain governed by their original licenses and access terms. Release documentation lists required external assets and where to obtain them.

User Prompt Template

I have a sentence describing a specific target object in a 3D indoor scene.
Your task is to provide the mentioned anchor objects and a correct referential order.

Known target object name: {target_phrase}
Sentence: {query}

Rules:

1. Output JSON only. Do not add explanations.
2. The final referential_order element must be exactly the known target object name.
3. Use 0 to 3 anchor objects before the target.
4. Use object category names only: remove color, material, size, and instance adjectives.
5. Do not use rooms/areas/spatial words as anchors, e.g. "room", "kitchen", "entrance", "left", "right", "corner", or "front".
6. If the sentence has no useful anchor object, use only the target object.

System Prompt and Expected Output

System prompt:
You are an expert at 3D visual grounding. You convert a natural-language object reference into a short Vigor-style referential chain: helpful anchor objects first, target object last.

Expected JSON output:

```
{
  "target_phrase": "backpack",
  "anchors": ["desk", "pencil", "desk lamp"],
  "referential_order": ["desk", "pencil", "desk lamp", "backpack"]
}
```

Training conversion:

```
desk <step1> pencil <step2> desk lamp <step3> backpack <step4>
```

Figure 10: **Offline referential-cue annotation prompt.** The Qwen3-32B/vLLM annotator receives the training query and target phrase and returns JSON anchors and a target-last order. Cue phrases precede their step markers in Phase A/B and provide step supervision. Phase C removes cues from the prompt while retaining training supervision. Default inference uses only the query and Mask3D candidate representations, without generated or human-written cue chains.

Table 16: **Full grounding-readout efficiency results.** Latency is in s/query (mean $\pm$ standard deviation), throughput in query/s, and multipliers are latency ratios relative to SSR.

Method	Latency	Throughput	Relative to SSR
Grounded 3D-LLM	1.0662 ± 0.0143	0.938	17.74$\times$
Chat-Scene	1.3772 ± 0.2626	0.726	22.91$\times$
SSR3D-LLM	0.0601 ± 0.0010	16.638	1.00$\times$

Asset sources and licenses. Table 18 summarizes the main assets and their published licenses or access terms. Original SSR3D-LLM code is released under MIT; upstream assets retain their applicable terms and attribution. The repository provides third-party notices, asset documentation, and preserved license texts.

Broader impacts. SSR3D-LLM may benefit embodied AI, navigation, assistive robotics, and spatial scene understanding by improving fine-grained grounding of objects and spatial relations in 3D scenes. The same capability can inherit biases and errors from indoor datasets, upstream proposal detectors, and pretrained visual/language models. Incorrect grounding in embodied systems could lead to unsafe or disruptive actions if deployed without validation. This paper evaluates static ScanNet-style research benchmarks and does not deploy a robot or safety-critical system; deployment with richer visual front ends should include task-specific validation, monitoring, and human oversight.

Table 17: **End-to-end efficiency by benchmark.** Latency is in s/query (mean $\pm$ standard deviation), throughput in query/s, and memory retains each row's GB or GiB unit. Timing excludes one-time model loading and includes per-query visual-feature and proposal construction.

Dataset	Method	Latency	Throughput	Peak memory
Nr3D	Grounded 3D-LLM	3.449 ± 0.427	0.290	4.77 GB
	Chat-Scene	57.581 ± 17.821	0.017	21.98 GiB
	SSR3D-LLM	0.827 ± 0.233	1.209	6.88 GB
ScanRefer	Grounded 3D-LLM	2.957 ± 0.036	0.338	4.08 GB
	Chat-Scene	44.083 ± 4.496	0.023	22.10 GiB
	SSR3D-LLM	8.120 ± 0.136	0.123	7.44 GB
Multi3DRef	Grounded 3D-LLM	2.745 ± 0.371	0.364	4.55 GB
	Chat-Scene	44.041 ± 5.648	0.023	22.05 GiB
	SSR3D-LLM	8.325 ± 3.148	0.120	7.92 GB

Table 18: **Licenses and sources of the main assets.** Asset names link to official releases. Dataset access terms and model-specific notices apply alongside the listed repository licenses.

Asset	License / applicable terms
ScanNet v2 / ScanNet200	ScanNet Terms of Use: approved non-commercial research and education.
ReferIt3D (Nr3D / Sr3D)	MIT (code); annotation license not separately stated. ScanNet terms apply to scenes.
ScanRefer / Scan2Cap	CC BY-NC-SA 3.0; ScanRefer dataset access terms. Scan2Cap uses ScanRefer annotations.
Multi3DRefer	MIT (official dataset card); ScanNet terms apply to scenes.
ScanQA	CC BY-NC-SA 3.0; ScanNet terms apply to scenes.
Stanford Alpaca	CC BY-NC 4.0; upstream research-use notice.
3D-LLM	MIT (repository); source-dataset terms apply to language data.
Grounded 3D-LLM	Component-specific notices; no project-wide license stated.
Tiny-Vicuna-1B	Apache-2.0 (model card); TinyLlama/WizardVicuna notice retained.
BERT-base-uncased, Qwen3-32B, DINOv2	Apache License 2.0.
Vicuna-7B	Version-specific LLaMA / Llama 2 terms, as listed by LMSYS.
Mask3D, Vigor, Chat-Scene	MIT (repositories); inherited component and model notices retained.

A.10 Cue Quality, Step Budget, and Ordering

Cue construction and structural constraints. Sr3D cues are deterministically converted from annotated target classes, relations, and context objects. For Nr3D, Qwen3-32B/vLLM extracts cues from the query, after which object names in the training labels are normalized to the training vocabulary when needed. For ScanRefer and Multi3DRef, the annotator can additionally use target class/name fields available in the training annotations. The template in Figure 10 requests JSON only, with zero to three anchors before the target and the target name as the final item; if no useful anchor exists, it uses the target alone. Anchors use object-category names without color, material, size, or instance modifiers, and exclude rooms, areas, and spatial words. These are prompt constraints; the same figure shows how training phrases are converted into step-marker sequences.

Human audit. We sampled 100 training cues from each of four benchmarks, 400 in total, using fixed proportional strata without replacement and random seed `20260727`. Five PhD students checked each cue for target-last ordering and anchor validity, with majority vote determining passage of the full rubric; Table 19 reports Wilson 95% intervals. Pairwise rater agreement was 98.70%, Fleiss' $\kappa = 0.941$, and 387/400 samples received unanimous judgments. All 400 satisfied the one-to-four-step, target-last structure, while 349 passed the full rubric. The 51 failures comprised 25 unsupported-anchor cases, 25 target-reuse cases, and one repeated anchor, showing that semantic errors persist despite structural compliance. The equal-weight total gives each benchmark the same

Table 19: **Human quality audit of training cues.** 100 cues per benchmark, judged by five raters through majority vote. Passing requires both target-last structure and anchor validity. Intervals are Wilson 95% CIs for pass rates (%).

Benchmark	Majority pass / N	Pass rate (%)	Wilson 95% CI
Nr3D	90/100	90.0	[82.56, 94.48]
Sr3D	100/100	100.0	[96.30, 100.00]
ScanRefer	79/100	79.0	[70.02, 85.83]
Multi3DRef	80/100	80.0	[71.12, 86.66]
Equal-weight total	349/400	87.25	[83.62, 90.17]

Table 20: **Clipped length distribution on Nr3D.** 7,485 queries in total. The $L = 4$ bin includes original lengths equal to and above four, so not every query in this bin was truncated.

Clipped length L	1	2	3	4	Total
Queries	1,240	5,034	1,120	91	7,485

weight rather than weighting by training-set size. This audit assesses label quality; it does not establish the grounding effect of replacing the annotator with a stronger model.

Coverage of the four-step budget. Table 20 gives effective lengths after clipping at four for 7,485 Nr3D queries. The first three bins contain 7,394 queries, approximately 98.8%; the final 91 include queries originally of length four as well as longer queries. Thus, at most 91 queries, or about 1.2%, can have been truncated from lengths above four. This upper bound describes the coverage of the chosen budget. A K ablation and a dedicated comparison on original lengths above four have not been performed, leaving the benefit of a larger budget untested.

Token layout, inference perturbations, and retraining. In p_{pred}, the query precedes the steps and memory follows them: steps read the query, subsequent memory summarizes the steps, and the scorer reads both. This explains causal information flow without ruling out other layouts. Table 21 compares matched retraining with two consistently ordered cue sequences, evaluated with predicted-L. Forward and reversed orders obtain 47.7% and 48.3% Nr3D accuracy. The paired difference's 95% interval includes zero, establishing neither a directional advantage nor statistical equivalence. Inference-time shuffle/reverse perturbations reorder latent steps of a fixed model without relearning slot alignment, testing its dependence on positions; retraining instead permits slots to realign, so the two controls are not interchangeable. This comparison is also separate from the main-table cross-seed means.

A.11 Training Stage, Backbone Scale, and Native-Protocol References

Phase C checkpoint comparison. Table 22 evaluates Phase A+B and Phase C on the same full Nr3D split, with identical no-cue prompts, candidates, oracle-L, and evaluator; only the checkpoint changes. Each row evaluates a single checkpoint and reports a query-level 95% CI. The paired change from 48.5 to 50.8 is $+2.3$ percentage points, consistent with Phase C adapting B to no-cue inputs; it does not establish a training-stage benefit across seeds. The 50.8 result is from one Phase C checkpoint, whereas 50.3 in Table 11 is the three-seed oracle-L mean; they are not interchangeable.

Nr3D grounding with a 7B backbone. Table 23 separately reports the two length settings for 1B and Vicuna-7B. With oracle-L and predicted-L, 7B reaches 52.0 and 49.0, respectively, exceeding the corresponding 1B values by 1.7 and 1.8 percentage points. This evidence is limited to Nr3D grounding: it does not establish a full multi-task benefit or an equal-budget comparison between backbones.

Comparison with concurrent work Chat-Scene++. Chat-Scene++ [18] is a concurrent successor of Chat-Scene. It keeps object-identifier grounding but extracts context-rich object features with pre-trained 3D scene-level and 2D image-level encoders, uses a Vicuna-7B backbone for its main results, and adds grounded chain-of-thought (CoT) training data; its own ablations identify visual-

Table 21: **Matched retraining with different cue orders.** Nr3D, predicted-L. Accuracy is in %; the difference is reversed minus forward in percentage points, with a paired 95% CI.

Training cue order	Nr3D accuracy	Δ (paired 95% CI)
Forward ordered	47.7	—
Reversed ordered	48.3	$+0.6$ $[-0.6, +1.7]$

Table 22: **Separate checkpoint comparison for Phase C.** Full Nr3D, no-cue prompt, identical candidates, oracle-L, and evaluator; only the checkpoint differs. Accuracy and query-level 95% CIs are in %; neither row is a cross-seed mean.

Checkpoint	Nr3D accuracy	Query-level 95% CI
Phase A+B	48.5	[47.4, 49.6]
Phase C	50.8	[49.6, 51.9]

feature enrichment as the larger contributor. Table 24 lists its reported ScanRefer/Multi3DRef values together with Chat-Scene and SSR; it reports no Nr3D/Sr3D candidate-set results, and its 1B backbone ablation reports only @0.50. At the same 1B scale, Chat-Scene++ and SSR are comparable (ScanRefer Acc@0.50: 52.6 vs. 51.5; Multi3DRef F1@0.50: 56.1 vs. 57.5). Its grounded CoT option generates reasoning text before answering, a complementary route to SSR's latent steps read in one forward pass.

A.12 Trace Diagnostics and Error Analysis

The following figures analyze saved per-step scores, masks, and confidence values from the checkpoints that produced these traces. They provide descriptive views of ranking dynamics, step-state freezing, where high-confidence errors occur, and how the *QPG* baseline behaves under the same candidate-set analysis. Their correspondence to the default predicted-L setting, including the inference-time length source and class-filter status, is not established here; they are therefore not evidence for default-setting accuracy or accuracy gains, which are reported in Tables 1, 4, and 12. In the example traces, mask analysis, and rank flow, L and L_{used} denote the valid steps in each saved trace; in the macro trends, L is the annotation-derived length used only to bin queries, which does not by itself indicate the length used at inference.

Example traces. Figure 11 visualizes individual step-wise score trajectories. These examples complement the main qualitative case by showing the same information in a compact heatmap form: which candidates become competitive, when the ground-truth object rises, and how much probability mass remains outside the tracked candidates.

State behavior of the step mask. Figure 12 illustrates the recursive scorer's deterministic behavior: valid steps $k \leq L$ update the states, whereas states remain unchanged for $k > L$. This allows a fixed maximum step budget to accommodate different useful lengths without padded steps changing the final ranking. The figure does not test whether overall accuracy improves significantly; that question is addressed by the paired statistics in Table 12.

GT rank flow. Figure 13 aggregates the rank of the ground-truth object over normalized latent-step progress. It provides a complementary view to the per-example heatmaps: instead of showing one query, it summarizes whether the recursive updates tend to move the target object closer to the top of the candidate list across evaluation traces.

Full-evaluation macro trends. We summarize *all* evaluation utterances using the per-candidate probability distributions exported by our evaluation pipeline. Figure 14 shows accuracy and high-confidence conditional error as a function of oracle step length L (clipped to $K = 4$) and candidate-set size. Figure 17 further visualizes the confidence separation and the high-confidence conditional error across L. These plots show how accuracy and high-confidence conditional error vary across

Table 23: **Nr3D accuracy across backbones (%).** The 1B reference values are three-seed means; 7B values are point estimates. Parentheses give percentage-point differences from 1B.

SSR backbone	**Oracle-L**	**Predicted-L**
Tiny-Vicuna-1B	50.3	47.2
Vicuna-7B	52.0 (+1.7)	49.0 (+1.8)

Table 24: **Reported ScanRefer and Multi3DRef values of Chat-Scene, Chat-Scene++, and SSR.** ScanRefer reports Acc and Multi3DRef reports F1 (%). Chat-Scene and Chat-Scene++ values are reported under their native protocols; SSR values (predicted-L) are from Table 2. NR means not reported, rather than zero.

Model	**ScanRefer Acc**		**Multi3DRef F1**	
	@0.25	**@0.50**	**@0.25**	**@0.50**
Chat-Scene (7B)	55.5	50.2	57.1	52.4
Chat-Scene++ (7B)	59.8	53.4	62.9	58.7
Chat-Scene++ (1B backbone ablation)	NR	52.6	NR	56.1
SSR3D-LLM (1B, predicted-L)	55.8	51.5	62.6	57.5

cue-length and candidate-set-size bins; the coverage and conditional-error panels separate confident predictions from reliable confident predictions.

***QPG* diagnostic macro trends.** Figures 18–21 repeat the same binning as Fig. 14, but use an exported candidate-context diagnostic for the single-readout *QPG* baseline. This diagnostic is separate from the main benchmark tables and is used to inspect how the *QPG* readout behaves across step-length and candidate-set-size bins. Because its confidence proxy is derived from similarity scores rather than a calibrated listener softmax, we use a fixed-coverage high-confidence definition (top-5% by $p_{\max}$, per seed) and report $\Pr(\text{wrong} \mid \text{high-conf})$.

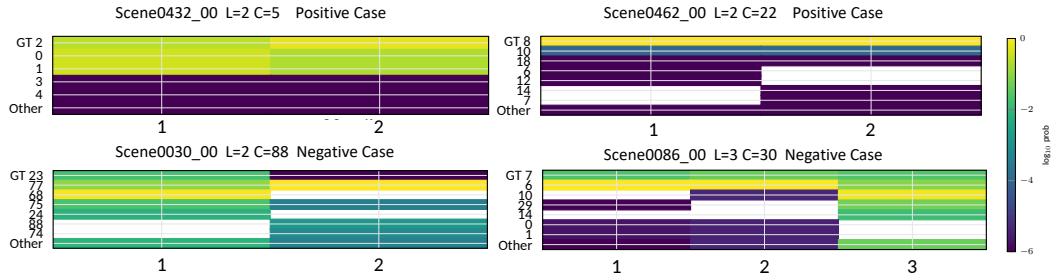


Figure 11: **Step-wise score heatmaps (examples).** For each query, we track the ground-truth (GT) object and the union of candidates that enter the per-step top-k list, and visualize their step-wise probabilities over effective steps ($k \leq L$) as a heatmap (color indicates $\log_{10}$ probability). Rows show candidate instance ids (and labels when available); "Other" aggregates the remaining probability mass outside the tracked candidates. These examples make the intermediate scoring process directly inspectable.

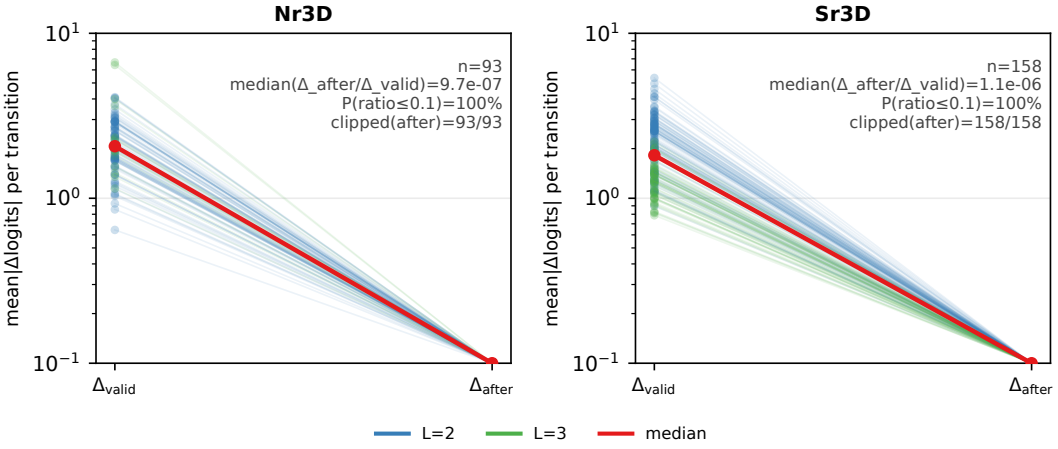


Figure 12: **Step-length masking analysis.** From exported stepwise traces, we directly compare the transition magnitude within valid steps (Δ_{valid}) and after termination (Δ_{after}). The near-zero Δ_{after} confirms step-length masking makes padded steps behaviorally inert.

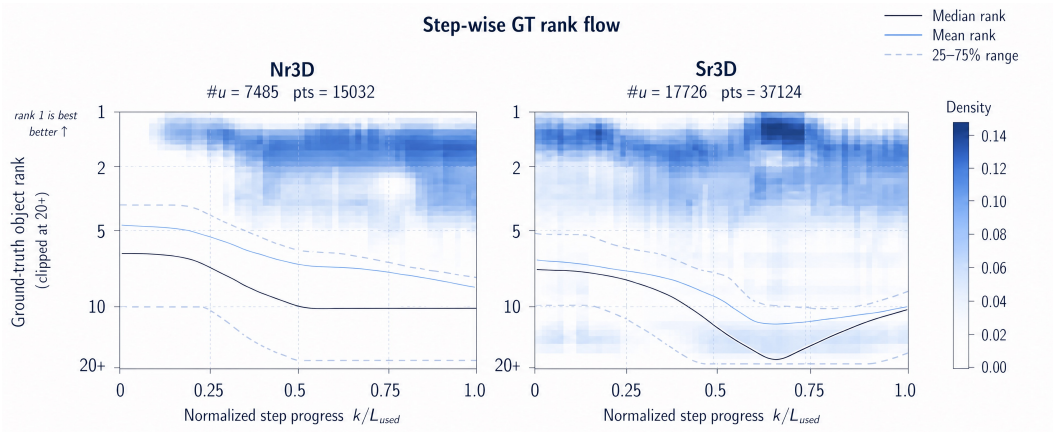


Figure 13: **GT rank flow from step-wise traces.** Smoothed density of GT rank over normalized step progress k/L_{used}. Columns are normalized within each progress step; curves summarize the median, mean, and 25–75% interquartile range. This aggregate diagnostic shows whether recursive updates tend to move the ground-truth object toward the top of the candidate ranking.

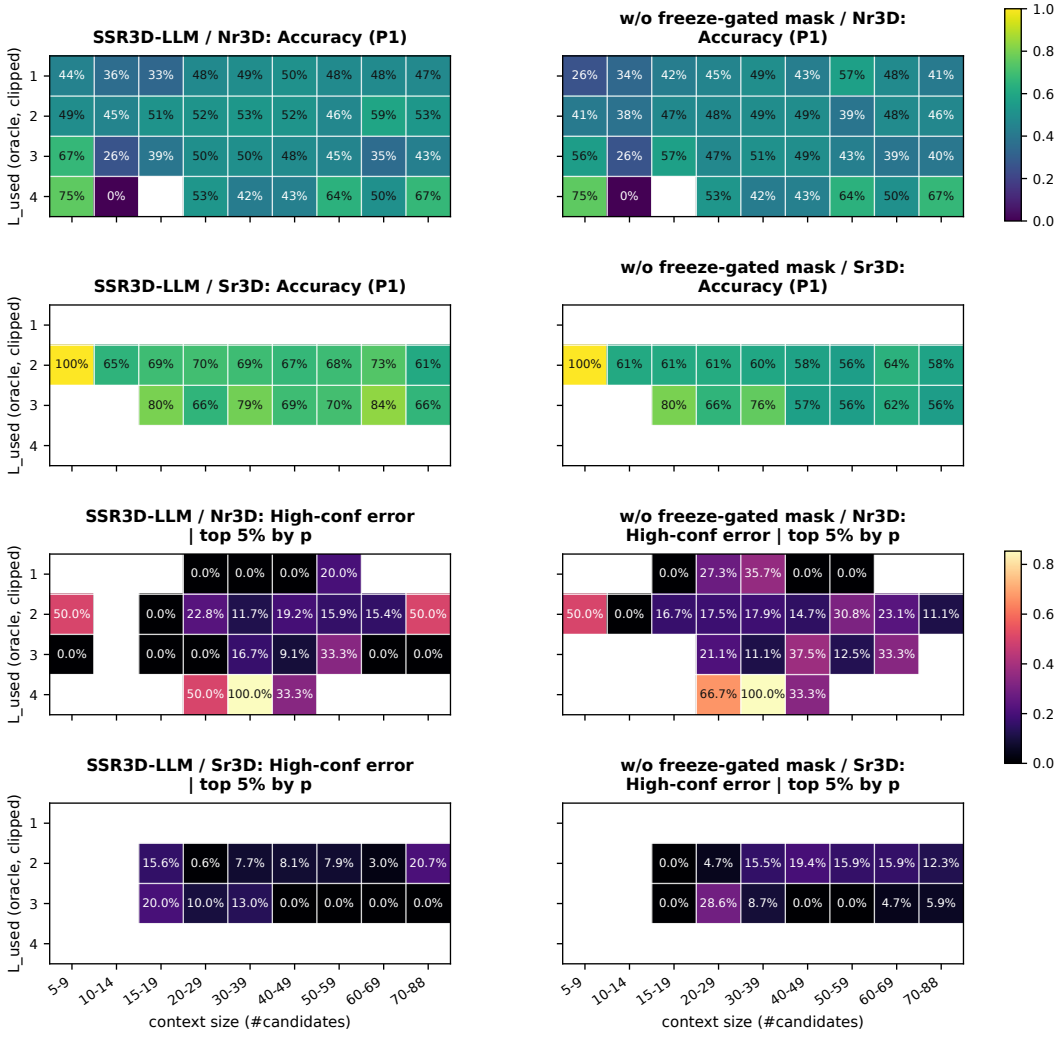


Figure 14: **Macro trends from full evaluation (Nr3D, Sr3D).** We aggregate the full evaluation set using the saved per-candidate probability distributions (no re-inference). Rows correspond to oracle step length L (from referential-cue annotations, clipped to $K = 4$); columns bin the candidate-set size (#objects). We show two methods side-by-side: SSR3D-LLM (ours) and a paired baseline *w/o step-length mask* (fixed-K updates), using the same candidate sets and the same evaluation artifacts. Note Sr3D only contains $L \in \{2, 3\}$; rare Nr3D cases with $L > 4$ are clipped to 4 to match the model. The top row reports candidate-set accuracy; the bottom row reports *high-confidence conditional error* (a.k.a. selective risk): we define *high confidence* as the top-5% predictions by $p_{\max}$ (threshold computed per eval repeat), and report $\Pr(\text{wrong} \mid \text{high-conf})$ within each bin. Figure 15 reports the corresponding coverage $\Pr(\text{high-conf})$ (which can vary across bins), enabling correct interpretation of near-zero cells.

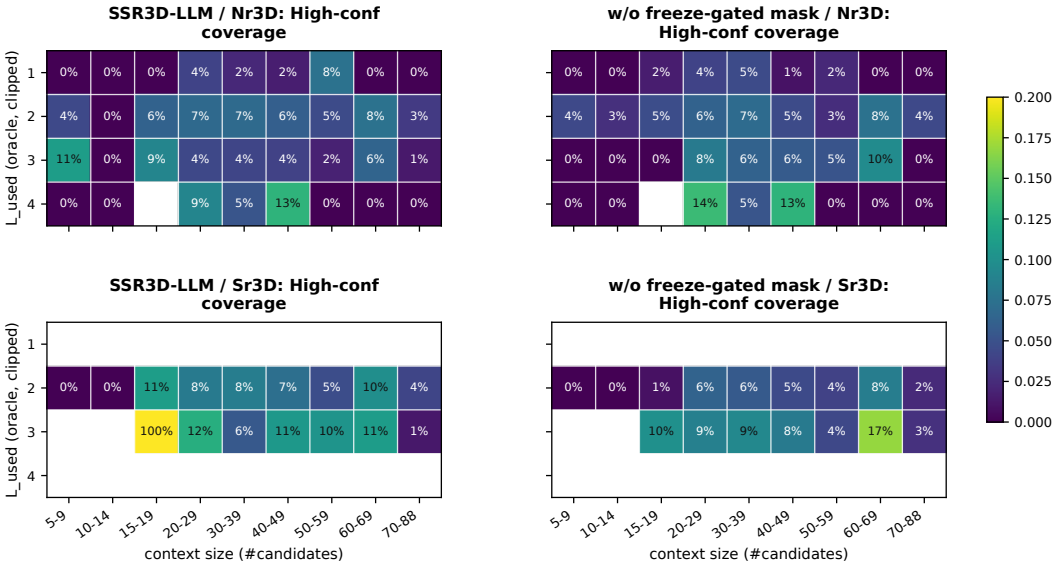


Figure 15: **High-confidence coverage from full evaluation.** Same binning as Fig. 14. We report $\Pr(\text{high-conf})$, where high-conf is defined as top-5% by $p_{\max}$ (per eval repeat). These maps disambiguate whether a low conditional error cell stems from genuinely low risk or simply from low coverage.

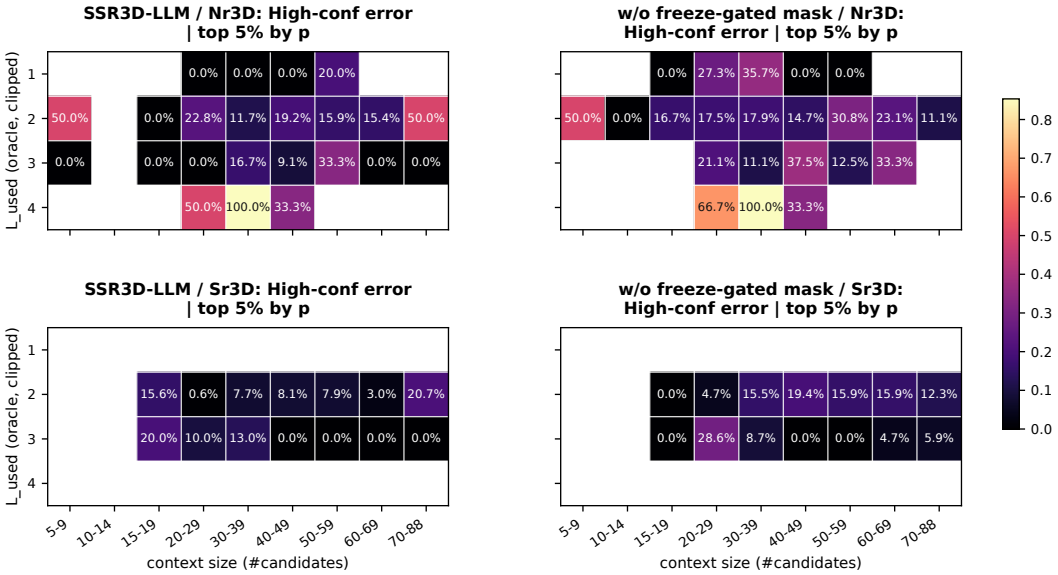


Figure 16: **High-confidence conditional error from full evaluation.** Same data as the bottom row of Fig. 14, shown alone for readability.

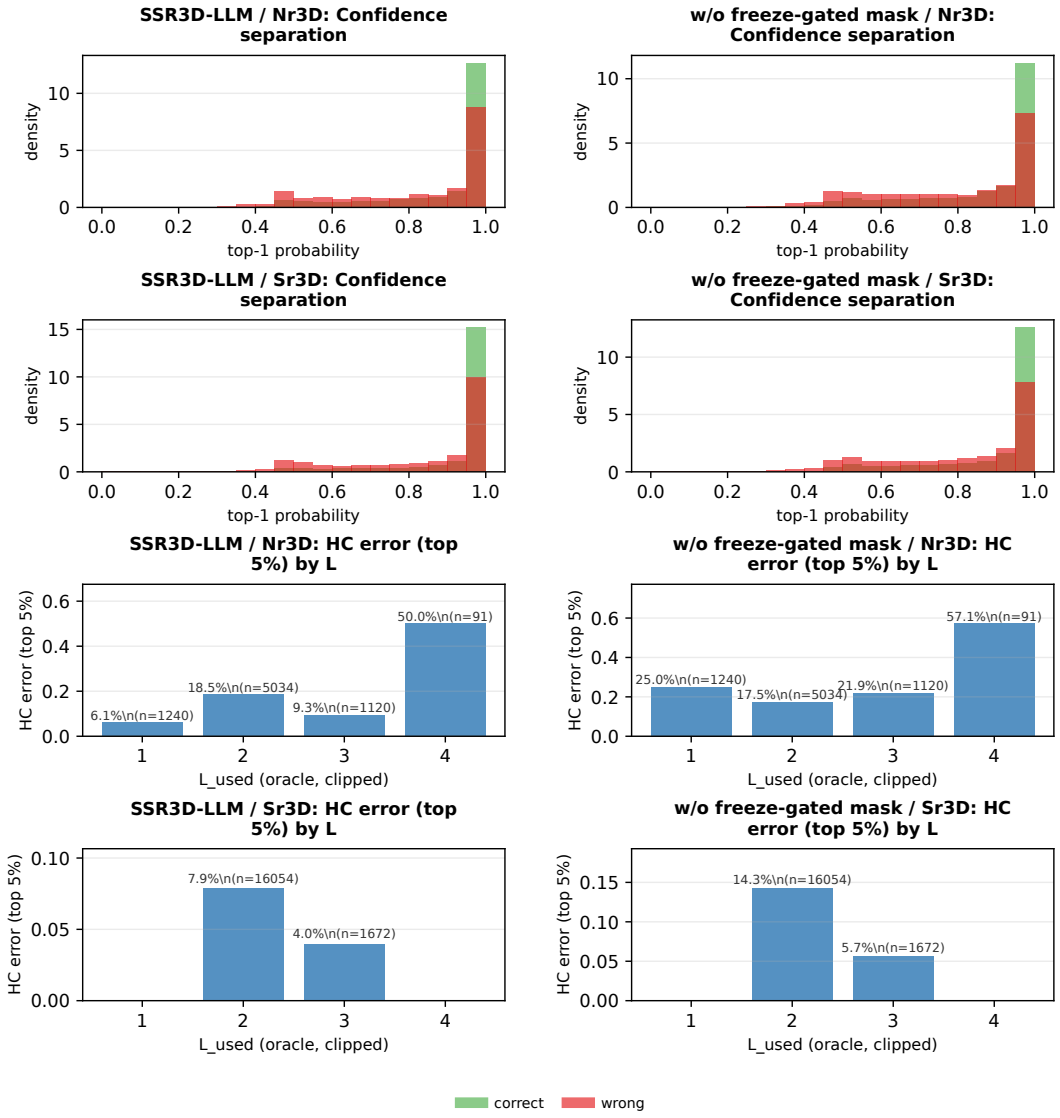


Figure 17: **Error-mode breakdown from full evaluation.** Top: top-1 probability histograms for correct vs wrong predictions. Bottom: high-confidence conditional error by oracle step length L (clipped to $K = 4$) with sample counts.

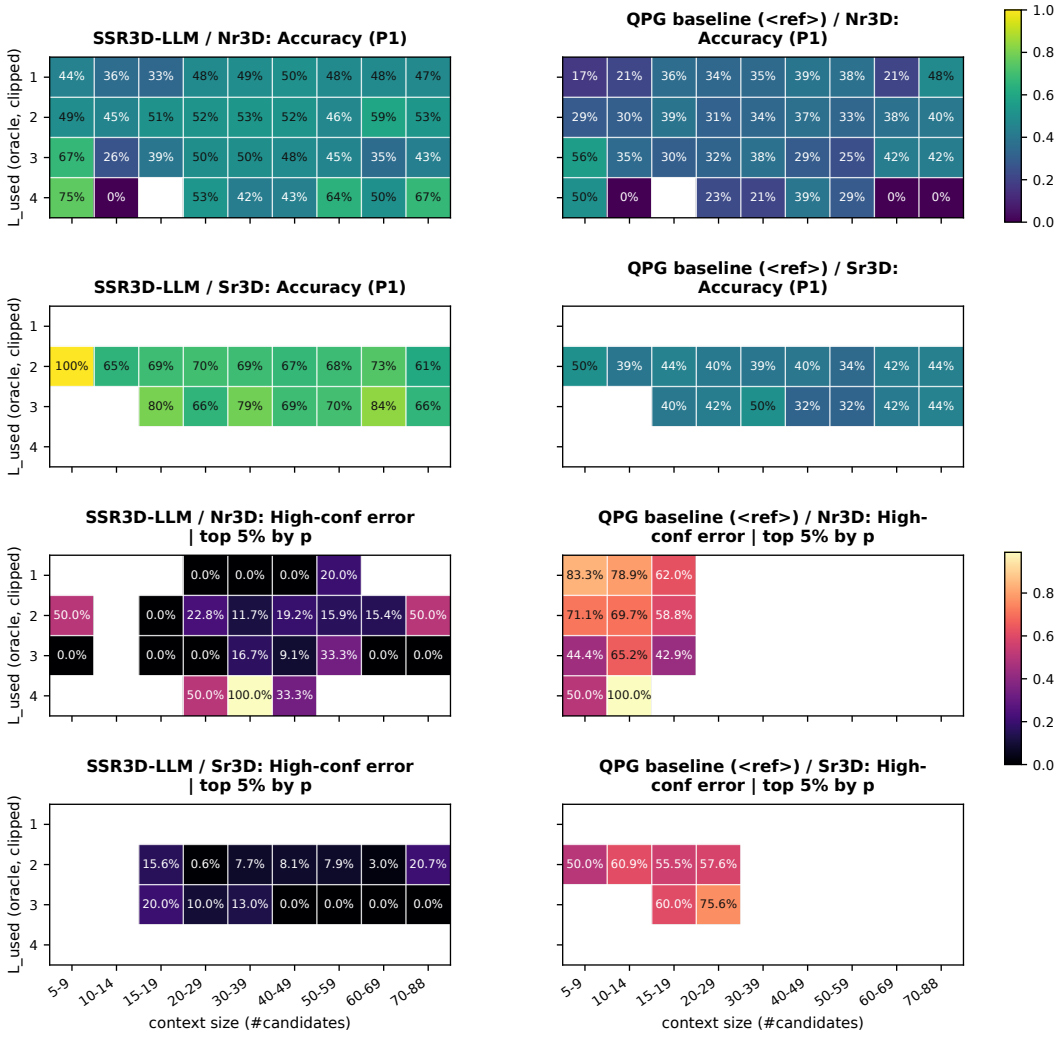


Figure 18: ***QPG* diagnostic macro trends.** Same binning as Fig. 14, with the right column showing a single-readout *QPG* diagnostic on the exported candidate contexts. This figure is an appendix diagnostic rather than the main same-protocol benchmark comparison. The confidence proxy is a softmax-normalized top-1 probability among candidate-context scores (temperature $T = 1$); because it comes from similarity scores rather than a calibrated listener softmax, high confidence is defined by fixed coverage: top-5% by proxy $p_{\max}$, computed per seed. Blank cells indicate bins with no samples or no high-confidence samples under this diagnostic.

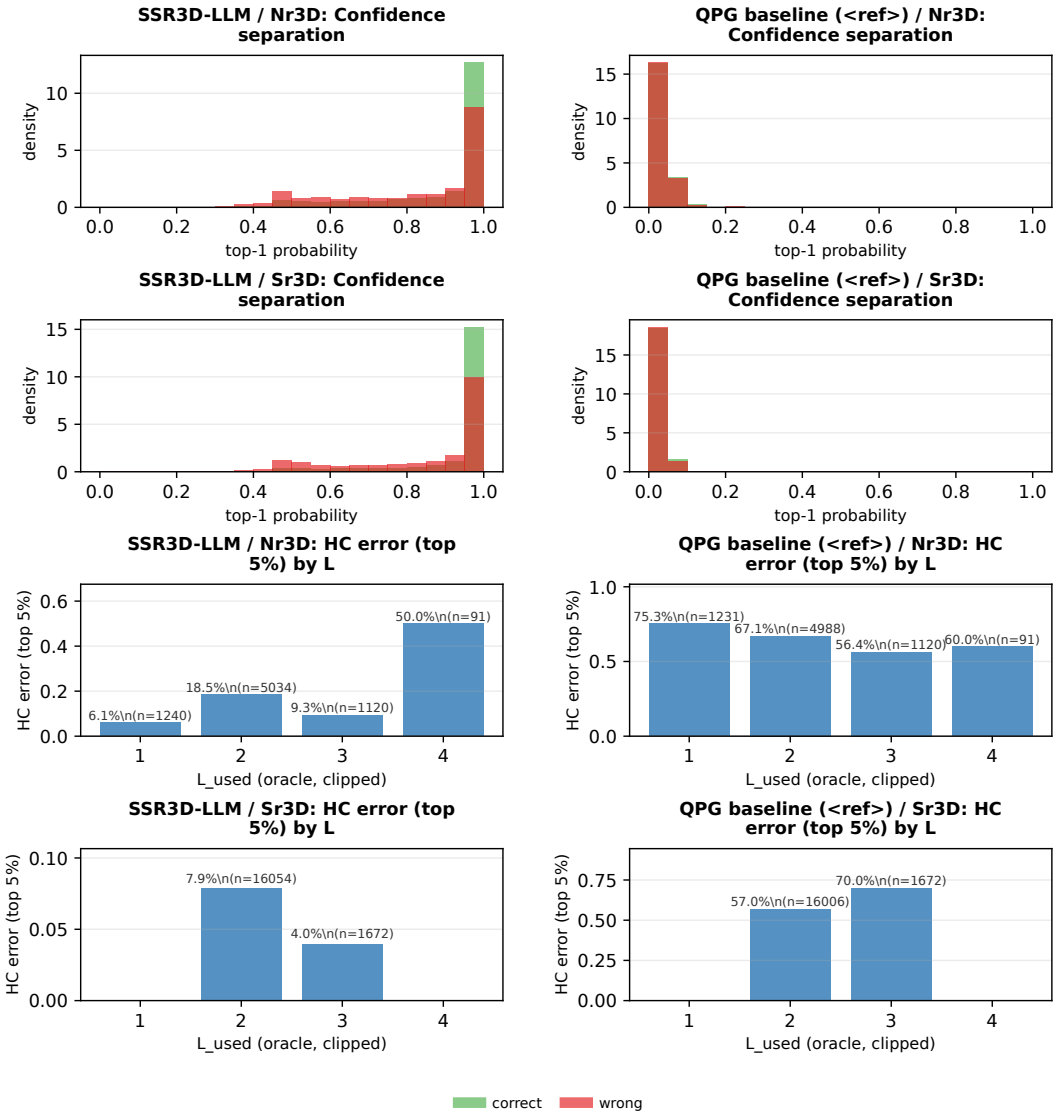


Figure 19: ***QPG* diagnostic error-mode breakdown.** Same layout as Fig. 17, with the right column showing the exported candidate-context *QPG* diagnostic. The high-confidence definition matches Fig. 18: top-5% by proxy $p_{\max}$, computed per seed.

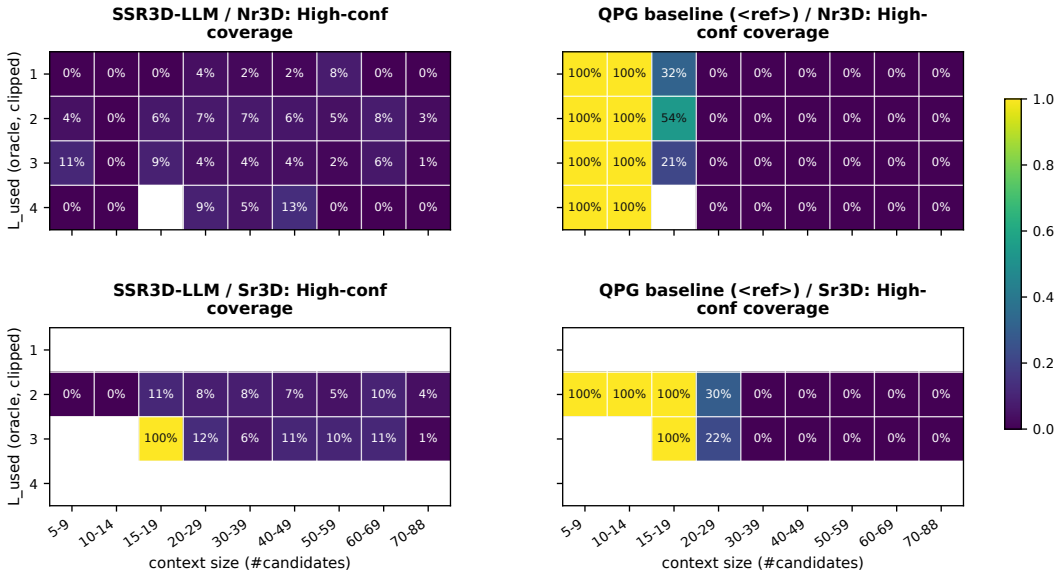


Figure 20: ***QPG* diagnostic high-confidence coverage.** Same as Fig. 15, with the exported candidate-context *QPG* diagnostic column.

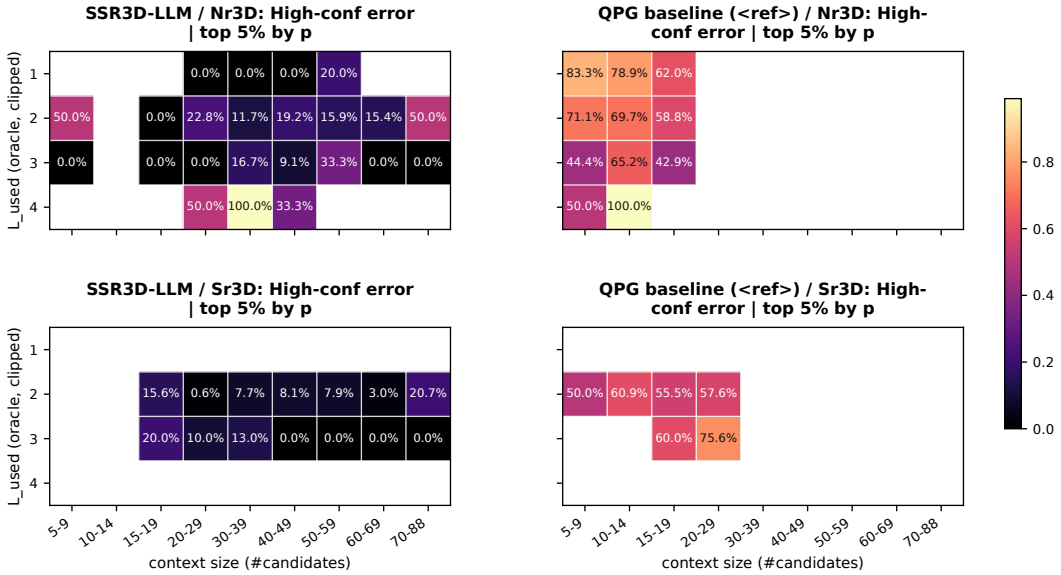


Figure 21: ***QPG* diagnostic high-confidence conditional error.** Same as Fig. 16, with the exported candidate-context *QPG* diagnostic column.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Introduction describe the proposed latent spatial reasoning steps, the step-wise scorer, and the empirical grounding gains that are later supported by the method and experiments.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The Conclusion and Appendix Section A.1 discuss dependence on upstream proposals, the fixed latent-step budget, richer visual inputs, and responsible deployment boundaries.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper is empirical and does not claim formal theoretical theorems or proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Sections 4.5 and 5.1 specify the model, training protocol, official splits, proposal-based candidate protocol, and metrics; Appendix Section A.9 gives the remaining training settings and describes the released scripts, configuration files, mappings, and checkpoint material.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The first page links to the public GitHub code repository and Hugging Face resource repository. They provide training, evaluation, referential-cue preprocessing, mapping, and table-generation scripts, configuration files, and checkpoints or reconstruction instructions for the main results and key controlled comparisons, as described in Appendix Section A.9.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The main paper gives the benchmark splits, evaluation protocol, candidate representation, latent-step budget, memory-token count, and three-stage training protocol; Appendix Section A.9 reports the main batch sizes, learning rates, epoch schedules, adapter settings, and loss weights. Released configurations specify run-specific seeds and checkpoint settings.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Nr3D and ScanRefer main results are means over three training seeds with cross-seed 95% CIs (Appendix Table 11); Appendix Table 12 reports scene-clustered bootstrap CIs and two-sided paired tests, and Appendix Tables 16 and 17 report latency as mean $\pm$ standard deviation. Multi3DRef results and some diagnostic comparisons are point estimates.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix Section A.9 states that training used NVIDIA A100 80GB GPU workers with single-GPU execution for the final grounding adaptations; Table 6 and Appendix Tables 16 and 17 report inference latency on the same GPU worker, throughput, and end-to-end peak memory.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: To the best of our knowledge, the research follows standard academic practice, uses public benchmarks and cited prior assets, and presents results and limitations transparently.

Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix Section A.9 discusses positive uses in embodied AI, navigation, assistive robotics, and spatial understanding, as well as risks from dataset/model bias, detector errors, and unsafe embodied deployment without validation.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The work focuses on 3D grounding on existing research benchmarks and does not present a release framed as a high-risk generative or scraped-data asset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Original creators are credited, and Appendix Section A.9, Table 18, and the repository documentation list official sources, published licenses, and access terms. We retain upstream attribution and license notices; releases without a separate project-wide license are identified as such.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Appendix Section A.9 describes the new assets (SSR3D-LLM training/evaluation code, S3G modules and configuration files, referential-cue annotations or scripts to regenerate them, evaluation mapping utilities, and visualization scripts) and states that release documentation lists the required external assets and where to obtain them.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The reported experiments do not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve human-subject studies that would require IRB-style approval.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: Sections 4.2 and 4.5 describe both uses relevant to the core method: the 3D-LLM writes latent spatial reasoning steps at inference, and an offline Qwen/vLLM annotator generates training-time referential cues for latent-step supervision.

    Guidelines:

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