GEAR: AN EVIDENCE-ADAPTIVE VLM AGENT FOR ZERO-SHOT 3D VISUAL GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot 3D visual grounding aims to localize objects from natural-language descriptions without task-specific 3D training. Existing approaches largely follow two paradigms: proposal-based methods enable efficient and reliable geometric reasoning but cannot recover missing objects, while RGB-D reconstruction provides stronger visual recoverability at substantially higher computational cost and with additional sources of error. To bridge this gap, we introduce **GEAR**, an evidence-adaptive ReAct-style agent that unifies proposal-based grounding, selective verification, and on-demand candidate repair. Starting from proposal hypotheses, GEAR adaptively selects and revisits grounding operations based on the query and evolving evidence, invoking semantic or visual verification for ambiguous hypotheses and bounded RGB-D reconstruction only when candidates are missing or unreliable. A candidate-bound memory maintains consistent evidence across grounding operations, while a role-aware geometric solver handles compound ordinal and local-anchor expressions. GEAR achieves 60.4% Acc@0.25 and 52.4% Acc@0.5 on ScanRefer and 56.0% accuracy on Nr3D, surpassing TAB, the previous state-of-the-art method, while reducing token usage and inference time by approximately 50%.

1 INTRODUCTION

3D visual grounding (3DVG) aims to localize a target object in a 3D scene from a natural-language description. This capability is fundamental to embodied intelligence, where an agent must identify the correct physical instance before it can interact with, manipulate, or navigate toward an object. Grounding can require substantially more than category recognition: an instruction such as "the second chair from the right next to the table" requires jointly resolving object identity, visual attributes, spatial relations, and ordinal structure. While conventional 3DVG methods have achieved strong performance with task-specific supervision, they typically rely on densely annotated 3D bounding boxes and predefined object taxonomies, limiting their scalability to open-vocabulary objects, unseen environments, and diverse language instructions.

Zero-shot 3DVG seeks to overcome these limitations by leveraging pretrained vision-language models (VLMs) and geometric scene representations without training a dedicated grounding model on the target benchmark. Existing systems, however, face an inherent tension between *efficient geometric reasoning* and *visual recoverability*. Proposal-based methods operate over compact 3D object candidates and therefore support efficient and reliable geometric reasoning, but their search space is fundamentally bounded by the recall of the initial proposals. Once the target or a critical anchor is missing, subsequent reasoning cannot recover it. In contrast, methods that reason directly from RGB-D observations can discover objects beyond a fixed proposal set, but doing so requires additional frame selection, segmentation, tracking, depth processing, and multi-view reconstruction. Applying these operations extensively not only increases inference cost but also introduces additional failure modes from occlusion, depth noise, camera poses, and inconsistent visual evidence.

Recent systems exemplify these two paradigms. AgentGrounder (Fig.1(a)) constructs an offline Object Lookup Table (OLT) from 3D instance segmentation and performs online candidate retrieval, deterministic geometric reasoning, and selective visual inspection Huynh et al. (2026). Its proposal-centric representation enables compact and interpretable reasoning, but objects absent from the initial OLT remain inaccessible during inference. TAB (Fig.1(b)) instead operates on RGB-D streams

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

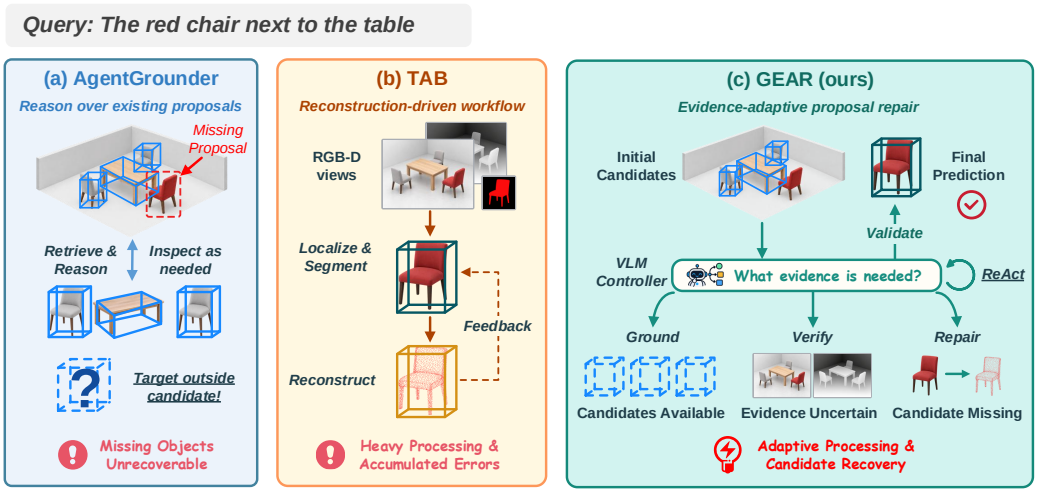

Figure 1: **Comparison of zero-shot 3D visual grounding paradigms.** (a) AgentGrounder reasons over existing proposals, limited by candidate coverage. (b) TAB reconstructs objects from RGB-D observations at additional processing cost. (c) GEAR adaptively combines proposal-based reasoning, verification, and bounded RGB-D repair according to evidence needs.

and combines a ReAct-style agent with semantic-anchored geometric expansion Wang et al. (2026). Its temporal and multi-view expansion mechanisms enable proposal-independent reconstruction, but entail heavy RGB-D processing through a substantially richer visual inference path and remain exposed to errors accumulated during segmentation, temporal association, depth processing, and reconstruction. These complementary strengths suggest that the central question is not simply *which representation should be used*, but rather:

When is the current geometric evidence sufficient to make a grounding decision, and when is additional visual evidence worth acquiring?

We address this question with **GEAR** (**G**rounding with **E**vidence-**A**daptive **Re**Act), an *evidence-adaptive* agent for zero-shot 3D grounding, as shown in Fig.1(c). Rather than committing to either proposal-only reasoning or reconstruction-heavy visual processing, our agent follows a proposal-to-repair strategy. Given a language query, an initial set of object proposals, and an RGB-D video stream, it maintains candidate hypotheses together with their accumulated evidence. Inexpensive proposal retrieval and deterministic geometric reasoning form the default inference path. Additional semantic inspection, view-based verification, and RGB-D lifting are progressively invoked only when the current hypothesis cannot be sufficiently supported—for example, when a target or anchor may be missing, semantic ambiguity remains unresolved, or the available geometric evidence is unreliable. RGB-D reconstruction therefore serves as a bounded recovery mechanism rather than a mandatory processing stage.

A key challenge in such adaptive inference is that evidence obtained at different stages is not necessarily interchangeable. A reconstructed object may overlap an existing proposal without sharing its semantic identity; evidence supporting an anchor should not be inherited by a target candidate; and an updated bounding box may invalidate geometric relations established for an earlier version. We therefore introduce a candidate-bound evidence memory that explicitly records candidate identity, provenance, target/anchor role, bounding-box version, and modality-specific support or contradiction. Newly acquired visual evidence is attached only to the candidate it actually verifies, while reconstructed candidates must satisfy semantic, role, geometric-alignment, and trust constraints before entering the active candidate set. Together with a conservative termination policy, this design prevents cross-candidate evidence contamination, target drift, and premature commitment to insufficiently supported hypotheses.

Beyond evidence acquisition, zero-shot grounding also requires correctly translating compositional language into executable geometric operations. In particular, compound ordinal and local-anchor expressions cannot always be resolved by flat global relation scores. For “the second chair from the right near the table,” for example, the relevant ordering is defined over a query-dependent local group

rather than all chairs in the scene, and ordinal constraints may apply to either target or anchor entities. We introduce a role-aware structured resolver that explicitly separates target-side and anchor-side constraints, constructs query-relevant spatial groups, and performs deterministic ranking within the appropriate scope. This combines the open-vocabulary semantic capability of VLMs with explicit and inspectable geometric reasoning.

Overall, our framework unifies proposal-based reasoning, visual verification, and bounded RGB-D reconstruction within an adaptive grounding loop guided by accumulated evidence. It preserves the efficiency of proposal-based reasoning when the initial scene representation is sufficient, while enabling the recovery of missing or unreliable candidates from RGB-D observations when necessary.

Our contributions are summarized as follows:

- We introduce an evidence-adaptive framework for zero-shot 3D visual grounding that dynamically routes between efficient proposal-based reasoning and progressively more expensive visual evidence acquisition, using RGB-D reconstruction as a bounded recovery mechanism rather than a mandatory inference path.
- We develop a candidate-bound evidence memory with guarded candidate repair and termination, explicitly preserving target/anchor roles, candidate provenance, bounding-box versions, and modality-specific evidence to support reliable evidence accumulation across heterogeneous grounding operations.
- We evaluate GEAR on 250-query subsets of ScanRefer and Nr3D through grounding, cost, and ablation comparisons. GEAR achieves 60.4% Acc@0.25 and 52.4% Acc@0.5 on ScanRefer and 56.0% accuracy on Nr3D, with lower token usage and total inference time than previous SOTA method TAB.

2 Related Work

2.1 Supervised 3D Visual Grounding

Early 3D visual grounding systems learn to associate referring expressions with objects in annotated 3D scenes. ScanRefer introduced language-based localization in RGB-D scans and learned a fused representation of object proposals and sentence features for 3D box prediction Chen et al. (2020). ReferIt3D introduced the Nr3D and Sr3D benchmarks and neural listeners for fine-grained instance identification, emphasizing the importance of object context and spatial relations Achlioptas et al. (2020). Subsequent detection-style models such as BUTD-DETR combine language guidance with objectness cues and can decode boxes beyond a fixed proposal pool Jain et al. (2022). These approaches demonstrate strong language–geometry alignment, but training a dedicated grounder generally requires substantial 3D grounding annotations. We instead study inference-time grounding without training on target-query annotations.

2.2 Zero-Shot Grounding from 3D Proposals and 2D Views

Zero-shot methods use pretrained language and vision models to reduce dependence on task-specific 3D labels. LLM-Grounder decomposes a query, grounds its semantic components with external scene tools, and reasons over their relations Yang et al. (2023). Visual Programming for Zero-shot Open-Vocabulary 3D Visual Grounding expresses the task through view-independent, view-dependent, and functional program modules Yuan et al. (2024). SeeGround combines query-aligned rendered views with spatially enriched scene descriptions for a 2D VLM Li et al. (2025). Agent-Grounder builds an offline object lookup table and uses an online agent for selective candidate retrieval, geometric scoring, and on-demand image rendering Huynh et al. (2026). These structured scene representations make geometric reasoning efficient, while their available candidates still depend on the coverage and quality of the underlying scene representation.

Complementary approaches reason more directly from image sequences. VLM-Grounder dynamically stitches views, uses grounding feedback, and projects multi-view observations into a 3D box without requiring an initial 3D object proposal set Xu et al. (2025). Such visual evidence can recover instances that a proposal table does not contain, but obtaining reliable masks and 3D extents from

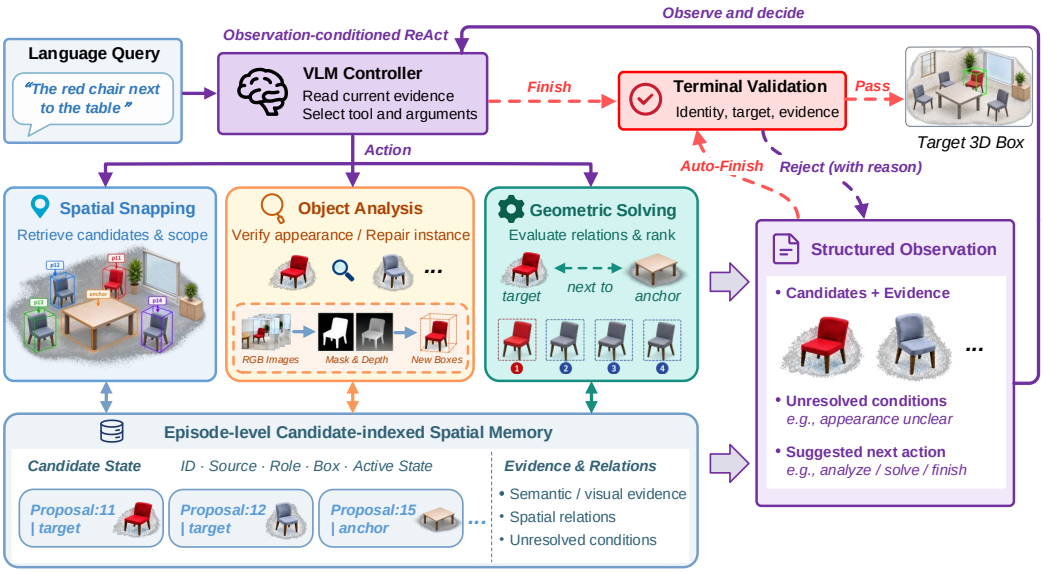

Figure 2: **Overview of GEAR.** GEAR employs a VLM Controller that adaptively selects grounding operations, a candidate-indexed memory that maintains consistent multimodal evidence, and a terminal validation module that verifies final grounding decisions.

multiple RGB-D views introduces additional computation and sensitivity to visual or geometric errors. Our system treats proposals and RGB-D reconstruction as alternative evidence sources that can be revisited within the same grounding episode.

2.3 Agentic Tool Use and Evidence Control

ReAct interleaves reasoning, tool actions, and environmental observations, allowing an agent to revise its plan after each interaction Yao et al. (2023). Agentic behavior is therefore not new to 3D grounding: VLM-Grounder uses iterative grounding feedback, AgentGrounder performs tool-driven retrieval and inspection, and TAB employs a VLM agent that dynamically invokes visual tools on raw RGB-D streams Xu et al. (2025); Huynh et al. (2026); Wang et al. (2026). TAB further introduces semantic-anchored geometric expansion to propagate target evidence across views before 2D-to-3D reconstruction Wang et al. (2026). Our distinction is not the use of ReAct or multi-view reconstruction alone. We study *candidate-centric control across evidence sources*: a high-level VLM controller decides, from the latest observation and episode-level candidate memory, whether to continue with proposal-based reasoning, inspect a candidate, invoke bounded RGB-D repair, or terminate. Candidate identities and target/anchor roles are carried across these decisions, while evidence-aware checks constrain candidate promotion and finalization.

3 Method

3.1 Task Formulation and Overview

Given a natural-language query q, an RGB-D sequence $\mathcal{V} = \{(I_i, D_i, T_i)\}_{i=1}^{N}$ with camera poses T_i, and an available set of 3D object proposals $\mathcal{P}$, zero-shot 3D visual grounding aims to identify the target instance and return its 3D bounding box $b = [c_x, c_y, c_z, d_x, d_y, d_z]$. The proposals provide inexpensive object hypotheses and geometric primitives, but the target may be absent from $\mathcal{P}$ or a proposal may have an unreliable extent. RGB-D observations provide a complementary source from which an instance can be inspected or reconstructed, at higher computational cost.

GEAR addresses this trade-off with a VLM agent that follows an *evidence-adaptive proposal-to-repair strategy*, as shown in Fig.2. The agent uses existing proposals as initial hypotheses and autonomously selects and parameterizes subsequent operations according to the query and accumulated evidence. It can reason over candidate geometry, inspect appearance to distinguish rivals, or acquire RGB-D evidence to repair missing or unreliable candidates. During the episode, it revises

its decisions in response to new evidence and remaining uncertainties. Reconstruction is invoked selectively, and its output returns to the shared candidate state for further reasoning. This makes both evidence acquisition and candidate selection part of the agent's decision process.

Four modules implement this strategy.The *VLM Controller* chooses operations and proposes grounding decisions; *Spatial Memory* preserves candidate identities, roles, geometry, and evidence; *Observation* returns structured evidence and action guidance to the controller; and *Terminal Validation* checks whether a proposed decision is sufficiently supported. The controller acts through three re-entrant tools—spatial snapping, object analysis, and geometric solving—and can revisit them as the candidate state changes. Its autonomous decisions are thus grounded in persistent evidence and constrained by candidate consistency and termination checks.

3.2 Evidence-Adaptive Agent Loop

GEAR interleaves reasoning, tool execution, and observation within a query–scene episode. At step t, M_t denotes Spatial Memory (Sec. 3.4) and H_t denotes the ordered interaction history, initialized as $H_0 = \emptyset$.

VLM Controller. The controller interprets the query together with previous actions and observations to select its next action and arguments:

$$(a_t, \xi_t) = \pi_{\mathrm{VLM}}(q, H_t), \qquad a_t \in \{\mathrm{Snap}, \mathrm{Analyze}, \mathrm{Solve}, \mathrm{Finish}, \mathrm{Abort}\}. \tag{1}$$

The arguments ξ_t specify a tool request, a stored candidate reference for `Finish`, or a reason for `Abort`. The controller chooses the operation and its parameters, while each tool carries out its dependent vision or geometry steps internally. Tool descriptions and execution instructions are fixed in the controller context.

Structured Observation. The Observation module connects tool execution to the next agent decision. It reports the candidates and evidence produced by the current operation, unresolved query conditions, and guidance on what evidence to acquire next. Let r_{t+1} denote the raw tool feedback and h_{t+1} a policy hint derived from the updated memory and available object and geometry diagnostics. A grounding operation produces the following transition:

$$\begin{aligned} (M_{t+1}, r_{t+1}) &= \mathcal{T}_{a_t}(M_t, \xi_t; q, \mathcal{P}, \mathcal{V}), \\ o_{t+1} &= \Phi(r_{t+1}, h_{t+1}), \\ H_{t+1} &= H_t \parallel \big((a_t, \xi_t), o_{t+1}\big), \end{aligned} \tag{2}$$

where $\mathcal{T}_{a_t}$ executes one of the three grounding tools, Φ constructs the controller-visible observation, and $\parallel$ denotes ordered concatenation.

The feedback preserves candidate and anchor identities, bounding boxes, confidence scores, relational or visual evidence, and tool-specific recommendations. These hints translate deterministic evidence diagnostics into actionable suggestions, while the VLM decides and parameterizes the next operation based on the hints and interaction history H_t. The history maintains previous actions and observations, enabling the controller to reason over the evolving state. Spatial Memory provides persistent support for tool execution, policy constraints, and validation. Moreover, tool failures and rejected grounding hypotheses are recorded as new observations, allowing the controller to adaptively revise subsequent actions.

3.3 Three Re-entrant Grounding Operations

Spatial Snapping. The `space_snapper` establishes candidate hypotheses from the predicted 3D object table. It parses the query into a target expression, attributes, and spatial conditions, retrieves target and anchor candidates, and assigns stable memory IDs. Its output identifies the active candidates, search scope, and unresolved conditions such as missing objects or unverified appearance attributes. The agent can revisit this operation to revise the candidate set or spatial scope.

Object Analysis. The `object_analyzer` acquires evidence for deciding whether an existing hypothesis is adequate or a new one is needed. Its modes provide three complementary functions within the same controller-visible tool.

Semantic verification. Candidate descriptions are ranked or compared against the target expression to assess semantic compatibility and narrow the alternatives.

View-based verification. A view probe examines candidate-associated RGB views to check queried appearance attributes and distinguish visually ambiguous rivals. It returns evidence tied to the inspected candidates for subsequent reasoning.

Candidate repair. When objects are missing or unreliable, `instance_repair` or `video_lift` performs bounded RGB-D reconstruction to generate new hypotheses. The process integrates view selection, segmentation, temporal aggregation, and geometric reconstruction into a single operation. Each repaired candidate records its source, role, visual support, and drift diagnostics, while admission guards determine whether it enters the active candidate set.

Geometric Solving. The `geometry_solver` evaluates candidate hypotheses and anchor relations using 3D boxes. It supports directional, proximity, support, and between relations, returning rankings, relation traces, margins, and unresolved constraints for subsequent reasoning.

For compound and ordinal queries, a role-aware resolver determines whether the ordinal applies to the target or anchor. Target-side ordinals are solved within query-relevant groups constructed from geometric context, while anchor-side ordinals are resolved before target grounding. The resolver records the grouping and ordering evidence used for each decision; other queries follow the general relation-ranking process.

3.4 Candidate-Indexed Spatial Memory

Adaptive evidence acquisition requires preserving which object each observation concerns as candidates are inspected or repaired. Spatial Memory provides this persistent state across the three grounding tools:

$$M_t = \big(q_0, \mathcal{C}_t, \mathcal{A}_t, \mathcal{G}_t, \mathcal{E}_t, w_t\big), \tag{3}$$

where q_0 is the original query and its initial parse, $\mathcal{C}_t$ stores candidate hypotheses, $\mathcal{A}_t$ contains anchor IDs, $\mathcal{G}_t$ contains object nodes and scored relation edges, $\mathcal{E}_t$ contains bounded evidence records, and w_t stores the active candidate set and working state.

Each candidate retains its identity, source, target/anchor role, and box. Proposal and reconstructed hypotheses have separate IDs, such as `proposal:11` and `repaired:2`. The original target is preserved across later tool queries, so investigating an anchor does not redefine the grounding objective. Tool events reference candidate IDs when available, while detailed outputs remain in the corresponding tool artifacts.

Visual probes and video lifting evidence is further bound to the candidate identity and box snapshot at acquisition time. Reuse requires matching identity and role, a compatible box snapshot, and, for lifted candidates, valid trust and drift checks. Spatial overlap alone does not transfer verification across candidates, and appearance evidence remains distinct from spatial-relation support. This candidate-scoped memory enables later decisions to verify that each piece of evidence applies to the candidate under consideration.

3.5 Evidence-Guided Routing and Termination

Evidence-guided routing. The agent adapts its next step to what remains unresolved: candidate coverage, semantic or appearance ambiguity, spatial-relation uncertainty, or box reliability. Adequate candidate coverage supports geometric reasoning; uncertain appearance motivates semantic analysis or a view probe; a missing or unreliable instance motivates candidate repair. New evidence is incorporated into memory and can lead the agent to reconsider its candidate choice or revisit earlier operations. Policy hints expose these evidence requirements to the controller, which selects the actual tool and arguments. Execution guards can defer reconstruction while existing candidates need geometric evaluation. The resulting sequence depends on the query and evolving evidence state.

Terminal Validation. The agent terminates through `Finish` by proposing a stored candidate. Terminal Validation verifies compatibility with the query and evidence state. A candidate can be accepted through object/visual verification or geometry reasoning satisfying ranking and relation constraints. Rejected proposals return failure reasons as observations for further reasoning.

Table 1: Comparison of 3D visual grounding methods on ScanRefer. Acc@0.25/Acc@0.5 denote grounding accuracy at 3D IoU thresholds of 0.25/0.50, respectively. An asterisk (*) denotes results reproduced by us under the same evaluation protocol as GEAR.

Method	Agent	Unique		Multiple		Overall	
		Acc@0.25↑	Acc@0.5↑	Acc@0.25↑	Acc@0.5↑	Acc@0.25↑	Acc@0.5↑
Fully-Supervised Methods							
MCLN Qian et al. (2024)	–	86.9	72.7	52.0	40.8	57.2	45.7
ConcreteNet Unal et al. (2024)	–	86.4	82.1	42.4	38.4	50.6	46.5
Chat-Scene Huang et al. (2024)	Vicuna-7B	89.6	82.5	47.8	42.9	55.5	50.2
Video-3DLLM Zheng et al. (2025)	LLaVA-Video7B	88.0	78.3	50.9	45.3	58.1	51.7
GPT4Scene Qi et al. (2026)	Qwen2-VL-7B	90.3	83.7	56.4	50.9	62.6	57.0
Zero-Shot Methods							
LLM-Grounder Yang et al. (2023)	GPT-4turbo	–	–	–	–	17.1	5.3
ZSVG3D Yuan et al. (2024)	GPT-4turbo	63.8	58.4	27.7	24.6	36.4	32.7
VLM-Grounder Xu et al. (2025)	GPT-4V	66.0	29.8	48.3	33.5	51.6	32.8
SeeGround Li et al. (2025)	Qwen2-VL-72B	75.7	68.9	34.0	30.0	44.1	39.4
AgentGrounder* Huynh et al. (2026)	Qwen3-VL-32B	64.1	59.8	34.8	27.9	45.6	39.6
TAB* Wang et al. (2026)	Qwen3-VL-32B	**81.5**	70.7	46.2	39.2	59.2	50.8
GEAR (Ours)	Qwen3-VL-32B	79.4	**73.9**	**49.4**	**40.0**	**60.4**	**52.4**

Object-analysis and geometry tools may also trigger automatic completion `Auto-Finish` when sufficient evidence is available. This path shares the same validator while avoiding an additional controller call. Under weak evidence, drift risk, or unresolved repair requests, completion can be deferred for further evidence acquisition.

For accepted candidates with close rivals, unresolved relations, or unsupported visual-verification signals, the agent performs at most one bounded final review using up to two views. Replacement requires consistent support from both views and must pass Terminal Validation again; otherwise, the original answer is preserved. Successful grounding returns the selected candidate box, while `Abort` or budget exhaustion terminates without a validated result.

4 Experiments

4.1 Experimental Setup

Benchmarks and protocols. We evaluate on ScanRefer (Chen et al., 2020) and Nr3D (Achlioptas et al., 2020). Following the established subset evaluation setting in zero-shot 3D visual grounding (Xu et al., 2025; Lin et al., 2025; Jin et al., 2026), we use the publicly released validation subsets containing *250* and *250* queries, respectively. We reuse the released query lists and retain the original benchmark descriptions. For ScanRefer, we report Acc@0.25 and Acc@0.50, with Unique and Multiple denoting targets without and with same-class distractors. For Nr3D, predictions are matched to instances by the largest positive 3D IoU (ties resolved by instance ID); invalid or zero-overlap predictions are counted as incorrect. We report overall accuracy and results on the official Easy/Hard and View-dependent/View-independent splits.

Implementation Details. The controller and visual reasoning tools use *Qwen3-VL-32B-Instruct* Yang et al. (2025) with temperature 0. We use predicted *Mask3D* Yang et al. (2025) object proposals and *SAM3* Carion et al. (2026) for RGB-D repair. No task-specific grounding training is performed; pretrained segmentation provides the initial 3D proposal prior. Before evaluation, we select the Mask3D proposal that best agrees with the predicted box, using predicted boxes only. The selected proposal is then evaluated against the ground-truth target with the standard mIoU and Acc@0.25/0.50 metrics. The reference configuration allows 20 agent decisions, uses a geometry candidate limit of 8, and sets both base and tie probe caps to 8. The VLM frame scoring cap is 48 for the fast/standard repair routes and does not limit the total number of frames processed during reconstruction.

Accuracy and cost Cost includes the controller and internal visual-model calls, including termination review. We report tokens, VLM calls, elapsed time, and the fraction of queries with an actually executed RGB-D reconstruction; a skipped repair request does not count as execution. Timing in-

Table 2: Comparison of 3D visual grounding methods on Nr3D. Easy/Hard and Dep./Indep. denote the standard difficulty and view-dependence splits, respectively.

Method	Agent	Venue	Difficulty		View Dependence		Overall↑
			Easy↑	Hard↑	Dep.↑	Indep.↑	
Fully-Supervised Methods							
ViL3DRel Chen et al. (2022)	–	NeurIPS'22	70.2	57.4	62.0	64.5	64.4
MiKASA Chang et al. (2024)	–	CVPR'24	69.7	59.4	65.4	64.0	64.4
SceneVerse Jia et al. (2024)	–	ECCV'24	72.5	57.8	56.9	67.9	64.9
Zero-Shot Methods							
ZSVG3D Yuan et al. (2024)	GPT-4turbo	CVPR'24	46.5	31.7	36.8	40.0	39.0
VLM-Grounder Xu et al. (2025)	GPT-4V	CoRL'24	55.2	39.5	45.8	49.4	48.0
SeeGround Li et al. (2025)	Qwen2-VL-72B	CVPR'25	54.5	38.3	42.3	48.2	46.1
AgentGrounder* Huynh et al. (2026)	Qwen3-VL-32B	ICRA'26	58.1	43.0	42.7	56.5	51.2
TAB* Wang et al. (2026)	Qwen3-VL-32B	EMNLP'26	51.5	32.5	38.5	45.4	45.2
GEAR (Ours)	Qwen3-VL-32B	–	**59.6**	**51.8**	**51.0**	**59.1**	**56.0**

Table 3: Comparison of inference costs between our agent and TAB-agent. All statistics are averaged per query. The colored subscripts denote the relative cost with respect to TAB-agent.

Method	Token Cost			API Calls↓	Time Cost		Agent Steps↓
	Input (k)↓	Output (k)↓	Total (k)↓		VLM (s)↓	Total (s)↓	
TAB-agent	698.48	4.56	703.04	256.42	105.71	982.18	14.93
Ours	**372.47** $_{0.53\times}$	**2.70** $_{0.59\times}$	**375.17** $_{0.53\times}$	**55.41** $_{0.22\times}$	**104.75** $_{0.99\times}$	**472.83** $_{0.48\times}$	**7.62** $_{0.51\times}$

cludes retries and GPU waiting within the measured interval. Direct Finish rate counts valid explicit or automatic agent completion, independently of grounding correctness.

4.2 Comparison with Existing Methods

Tables 1 and 2 compare GEAR with existing 3D visual grounding methods on ScanRefer and Nr3D. GEAR achieves the best overall performance among zero-shot methods, reaching 60.4% Acc@0.25 and 52.4% Acc@0.5 on ScanRefer and 56.0% accuracy on Nr3D. On ScanRefer, GEAR surpasses the previous state-of-the-art TAB-agent by 1.2 and 1.6 points at Acc@0.25 and Acc@0.5, respectively, while achieving stronger performance on the Multiple split, where missing or ambiguous candidates are more challenging. On Nr3D, GEAR consistently improves over prior zero-shot agents across both difficulty and view-dependence splits, demonstrating robust grounding under diverse language descriptions. These results show that evidence-adaptive grounding combines the efficiency of proposal-based reasoning with the recoverability of selective visual evidence acquisition.

4.3 Inference Cost

Table 3 reports per-query resource usage. GEAR uses 375.17k tokens compared with TAB's 703.04k, a 46.6% reduction, and its reported total time decreases from 982.18 to 472.83 seconds, a 51.9% reduction. The recorded API-call count falls by 78.4%, while the number of agent steps falls from 14.93 to 7.62. The LLM-time column changes by only 0.9%, however; the total-time reduction must not be interpreted as halving model-response latency. The two systems expose different tool granularities, so a step is not a common unit of computation across systems.

4.4 Component Analysis

Component ablation. Table 4 evaluates the contribution of the major components in GEAR. We first remove VLM Controller, and replace the VLM-based action selection with a rule-based policy that selects grounding operations according to predefined evidence conditions. This variant reduces Acc@0.25 from 60.4% to 56.4% and Acc@0.5 from 52.4% to 49.2%, suggesting that fixed routing captures part of the evidence-adaptive behavior but is less effective than query- and history-conditioned operation selection. Removing Spatial Memory further degrades grounding accuracy and substantially increases token usage and inference time, despite a similar number of agent steps, indicating that persistent candidate state improves the efficiency and consistency of evidence accumulation. Without structured Observation, accuracy also drops notably, showing the importance

Table 4: Component ablations of the full agent. Each variant modifies one mechanism while using the same evaluation queries and disabling evaluator recovery.

Variant	ScanRefer		Inference Cost		
	Acc@0.25↑	Acc@0.5↑	Agent Steps↓	Infer Time (s)↓	Tokens (k)↓
w/o VLM Controller	56.4	49.2	10.2	465.5	137.2
w/o Spatial Memory	53.2	46.0	9.9	783.0	588.2
w/o Observation	51.2	45.6	10.0	374.4	339.0
w/o Terminal Validation	44.0	40.4	2.1	75.1	25.5
Ours (Full Model)	**60.4**	**52.4**	7.6	472.8	375.2

Table 5: Analysis of different strategies in our framework.

Component	Setting	Acc@0.25↑	Acc@0.5↑	mIoU↑
Observation	w/o Policy Hint	60.0	52.4	50.3
Object Analyzer	w/o View Probe	47.2	41.2	40.0
	w/o RGB-D Repair	52.8	47.2	45.3
Geometric Solver	w/o Compound Ordinal Ranking	59.2	49.2	48.2
	w/o Local-Anchor Grouping	58.8	49.2	48.9
Terminal Validation	w/o Finish Readiness Checks	44.4	40.8	38.6
	w/o Final Relation Review	59.2	51.6	49.8
Ours (Full Model)		**60.4**	**52.4**	**50.4**

of exposing actionable evidence and unresolved conditions to subsequent agent decisions. Finally, removing Terminal Validation causes the largest performance degradation while sharply reducing inference cost, indicating that the apparent efficiency gain largely results from premature termination without sufficient evidence checking. Overall, the full model achieves the highest grounding accuracy with a moderate inference budget, supporting the complementary roles of adaptive control, persistent memory, structured feedback, and guarded termination.

Different strategies. Table 5 further examines individual mechanisms within GEAR. Removing the policy hint from Observation has little effect on grounding accuracy, suggesting that the controller can largely infer the next operation from structured tool feedback and interaction history. Within Object Analyzer, removing the view probe causes a substantial performance drop, highlighting the importance of candidate-specific visual verification for resolving ambiguous hypotheses. Disabling RGB-D repair also consistently degrades performance, confirming the importance of on-demand reconstruction for recovering missing or unreliable candidates. For Geometric Solver, removing either compound ordinal ranking or local-anchor grouping consistently reduces grounding performance, demonstrating the benefit of explicitly resolving structured spatial constraints rather than relying on generic relation scoring. For Terminal Validation, removing finish-readiness checks results in the largest degradation, indicating that explicit evidence requirements are critical for preventing unsupported or premature grounding decisions. In contrast, removing the final relation review leads to only a modest drop, suggesting that it primarily serves as a complementary safeguard for difficult cases. Overall, these results support the complementary roles of selective visual verification, candidate repair, structured geometric reasoning, and guarded termination in GEAR.

5 Conclusion

We presented GEAR, an evidence-adaptive ReAct agent for zero-shot 3D visual grounding that combines proposal-based reasoning, selective visual verification, and bounded RGB-D repair. Candidate-bound memory preserves evidence consistency, while role-aware geometric reasoning and terminal validation support candidate selection and answer acceptance. Experiments on 250-query subsets of ScanRefer and Nr3D show competitive grounding accuracy, with lower measured token usage and total runtime than the reproduced TAB baseline. Ablations support complementary roles for visual evidence acquisition, persistent state, and acceptance checking.

AI USE STATEMENT

manuscript, including improving grammar, clarity, and readability, and to create illustrative elements used in the framework figures. All AI-assisted content was reviewed by the authors to ensure technical accuracy and consistency with the intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our method and experimental protocol are described in Sections 3 and 4.1. Appendix A.3 describes prompt construction and provides representative excerpts; the accompanying materials contain the complete recorded controller prompt, tool schemas, and source and artifact manifests. The sample identities, recorded trajectories, and evaluation outcomes in Appendix A.1 connect each visualization to its underlying execution. Appendix A.2 discusses the scope of the evidence and the remaining dependencies on sensing inputs, model services, and execution environments.

REFERENCES

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

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Chun-Peng Chang, Shaoxiang Wang, Alain Pagani, and Didier Stricker. Mikasa: Multi-key-anchor & scene-aware transformer for 3d visual grounding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14131–14140. IEEE, 2024.

Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *European Conference on Computer Vision*, 2020.

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.

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.

Cuong Huynh, Maxim Popov, Denis Gridusov, and Sergey Kolyubin. Agentgrounder: Zero-shot 3d visual pointcloud grounding using multimodal language models. *arXiv preprint arXiv:2605.25901*, 2026.

Ayush Jain, Nikolaos Gkanatsios, Ishita Mediratta, and Katerina Fragkiadaki. Bottom up top down detection transformers for language grounding in images and point clouds. In *European Conference on Computer Vision*, 2022.

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*, pp. 289–310. Springer, 2024.

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. *Advances in Neural Information Processing Systems*, 38:165549–165576, 2026.

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*, pp. 3707–3717, 2025.

Jiawen Lin, Shiran Bian, Yihang Zhu, Wenbin Tan, Yachao Zhang, Yuan Xie, and Yanyun Qu. Seqvlm: Proposal-guided multi-view sequences reasoning via vlm for zero-shot 3d visual grounding. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 3094–3103, 2025.

Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 157277–157302, 2026.

Zhipeng Qian, Yiwei Ma, Zhekai Lin, Jiayi Ji, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji. Multi-branch collaborative learning network for 3d visual grounding. In *European conference on computer vision*, pp. 381–398. Springer, 2024.

Ozan Unal, Christos Sakaridis, Suman Saha, and Luc Van Gool. Four ways to improve verbo-visual fusion for dense 3d visual grounding. In *European Conference on Computer Vision*, pp. 196–213. Springer, 2024.

Haibo Wang, Zihao Lin, Zhiyang Xu, and Lifu Huang. Think, act, build: An agentic framework with vision language models for zero-shot 3d visual grounding. *arXiv preprint arXiv:2604.00528*, 2026.

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 *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3961–3985, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20623–20633, 2024.

Duo Zheng, Shijia Huang, and Liwei Wang. Video-3d llm: Learning position-aware video representation for 3d scene understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8995–9006. IEEE, 2025.

A Appendix

A.1 Qualitative Results

Figure 3 presents six recorded ScanRefer episodes illustrating how GEAR selects and revisits grounding operations as evidence changes. Cases A–E achieve IoU ≥ 0.5 under the proposal-matching protocol described in Section 4.1. All Five episodes complete through explicit or automatic `Finish`, without evaluator recovery. We report IoU to distinguish the quality of the predicted box.

A: Direct geometric completion. For the refrigerator query, spatial snapping retrieves one target candidate and three anchors. Geometric solving selects the refrigerator and triggers automatic completion after two agent steps, with IoU approximately 1.0. No semantic verification, view probe, or RGB-D repair is invoked. This example illustrates a short inference path when the retrieved candidates and their geometry support an accepted decision. The displayed RGB image provides post-hoc scene context; it was not inspected by the agent during this episode.

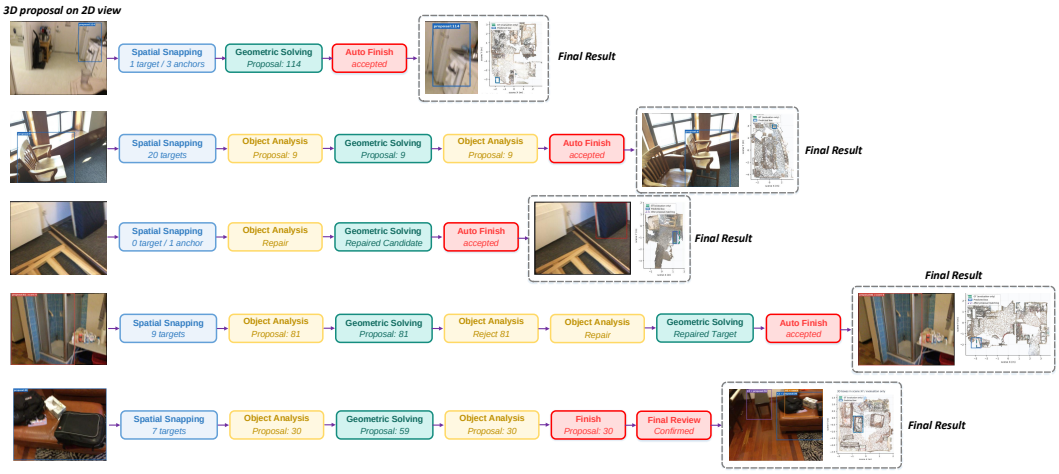

Figure 3: **Recorded grounding trajectories of GEAR.** (A) Direct geometric completion. (B) Selective visual verification. (C) Repair after empty target retrieval. (D) Candidate conflict followed by repair and renewed geometric solving. (E) Explicit completion with final review.

B: Selective visual verification. For the wooden-chair query, semantic analysis and geometric solving initially favor different candidates. The geometric winner has a ranking margin of 0.071, below the required 0.12, leaving the decision unresolved. A subsequent view probe supplies appearance evidence for `proposal:9`, after which the agent completes with IoU approximately 1.0. This trajectory illustrates acquiring additional evidence for an existing hypothesis without invoking reconstruction. The auxiliary relation review returns `unknown` because distinct views are insufficient; the episode therefore does not establish successful final relation verification.

C: Repair after empty target retrieval. For the black-mattress query, initial retrieval returns no target candidate but identifies a radiator anchor. The agent invokes RGB-D repair, which produces a new candidate that passes the recorded trust and drift checks, and then evaluates it geometrically. The predicted box has IoU 0.406, increasing to 0.845 after proposal matching. An overlapping proposal labelled as a door exists in the full proposal table, although it was absent from the retrieved target set. This case illustrates recovery from a target-retrieval gap. Its final localization score reflects both the reconstructed hypothesis and subsequent proposal matching.

D: Revisiting geometry after conflicting evidence. For the cream-colored-door query, semantic analysis favors `proposal:1`, whereas geometric solving favors `proposal:81`. A view probe assigns zero appearance support to `proposal:81` in all three inspected views. The agent then invokes candidate repair and returns to geometric solving before completing. This sequence illustrates how contradictory visual evidence can lead to another evidence-acquisition step and a repeated geometric evaluation. The correct door proposal was already present in the initial candidate set, so the episode concerns conflicting candidate evidence rather than missing proposal coverage. The reconstructed prediction has IoU 0.397, and proposal matching raises the evaluated IoU to 0.937.

E: Explicit completion followed by final review. For the ottoman query, semantic and visual evidence favor `proposal:30`, while an earlier geometric ranking favors `proposal:59`. The controller explicitly selects `proposal:30` through `Finish`. One final-review model call records a `confirmed` verdict and retains the same answer, yielding IoU approximately 1.0. This example documents the execution of the review path without an answer replacement. However, the recorded anchor assignment for the piano-related constraint is inconsistent with the accompanying explanation. The positive review verdict therefore should not be interpreted as evidence that every queried relation was correctly verified.

A.2 Limitation and Future Work

Our study has several limitations that motivate future work. First, following prior zero-shot 3D grounding protocols, we evaluate on 250-query subsets of ScanRefer and Nr3D. While all repro-

duced baselines use the same evaluation queries and protocol, evaluation on larger query sets and more diverse 3D environments would provide a more comprehensive assessment of generalization.

Second, the current geometric solver primarily focuses on spatial relations that can be explicitly evaluated from 3D object geometry. More complex functional, interaction-centric, or temporally grounded relations are not explicitly modeled in the current implementation. Extending the structured reasoning interface to richer relation types would broaden the applicability of the framework.

Third, although GEAR enables evidence-adaptive decision making through a VLM controller, the current routing mechanism relies on the reasoning capability of the underlying foundation model rather than an explicitly learned policy. Consequently, stronger controllers may further improve evidence acquisition and tool selection. Developing more efficient and adaptive routing strategies, such as lightweight learned controllers or specialized decision modules, remains an interesting direction for future work.

A.3 Prompts of GEAR

To ensure full transparency and facilitate the reproducibility of our work, this section comprehensively details the exact textual prompts employed across the various modules of GEAR.

A.3.1 Prompt for VLM controller

You are a Skill Dispatcher. Your goal is to select the most appropriate expert skill for the user's query.

YOUR EXPERT SKILLS
Expert Skill: Atomic 3D Visual Grounding

AVAILABLE TOOLS
The available tools and their STRICT parameter definitions are as follows:
{tool_descriptions}
RESPONSE FORMAT
Please respond in a step-by-step loop. For each step, output a Thought followed by an Action, strictly according to the following format:

Thought:
- First, recall "YOUR EXPERT SKILLS" to decide which skill to use and the current pipeline stage.
- Analyze the previous Observation.
- Plan the next step accordingly.
- CRITICAL: Check if the previous observation indicates failure. If so, consider retrying with looser constraints or ABORTING.

Action: The action you decide to take. It MUST be formatted EXACTLY as one of the following:

1. Tool Call:
* <tool_name >(<tool_input_json >)
* <tool_name >must be one of: [tool_names]
* <tool_input_json >must be a valid JSON string containing arguments matching the tool's schema.
* Example: filter_by_mask("threshold": 0.5)

2. Finish Task:
Finish[<your final answer here>]
* Use this when you have obtained the final answer or completed the task successfully.
* Example: Finish[The 3D bounding box for the small wooden cabinet is ...]

3. Abort Task:
Abort[<failure_reason>]
* Use this when the task CANNOT be completed.
* Example: Abort[No images found after mask filtering]

Important Reminders
1. Every response **MUST** include exactly one 'Thought:' block and exactly one 'Action:' block.
2. 'Finish' and 'Abort' ARE Actions. They MUST be written directly after 'Action: '.

3. **NEVER** output a separate "Final Answer:" section.
4. **NEVER** output "Action: None". If you have the answer, your action is 'Finish[...]'.
5. Tool calls must strictly follow the format: tool_name(tool_input_json).
6. Only use 'Finish' when you are certain you have sufficient information to answer the question.
7. Only use 'Abort' when you are certain you cannot complete this task with current tools and information.
8. If the information returned by a tool is insufficient, continue using other tools or the same tool with different parameters.
9. Treat 'space_snapper', 'object_analyzer', and 'geometry_solver' as the only controller-visible atomic operations when they are the registered tools: - 'space_snapper': use this to build/update spatial memory from object proposals, scene graph nodes, target candidates, anchor candidates, and scope summaries. - 'object_analyzer': use this to update spatial memory with semantic/attribute evidence, run a bounded visual attribute check with 'mode="view_probe"', compare candidates, or create a repaired candidate with 'mode="instance_repair"'/'mode="video_lift"' when proposals are missing, target_drift/failed evidence is reported, repeated lookup/geometry evidence is stuck with unresolved close rivals, or 'recommended_next'/'finish_policy' explicitly asks for instance repair. - 'geometry_solver': use this to rank active memory candidates under spatial constraints. If you omit 'subjects' or 'anchors', it will use 'active_candidate_ids' and 'anchor_ids' from spatial memory when available. Put anchor category words in 'anchors' or 'anchor_terms'; do not pass relation anchors to 'object_analyzer'.
10. Do not ask for hidden low-level actions such as query_parse, read_image_files, masks_filter, VLM scoring, SAM segmentation, reconstruction, or bbox calculation unless they are explicitly listed in AVAILABLE TOOLS. In atomic mode those details are internal to the three tools.
11. This is not a fixed pipeline. Choose a dynamic sequence based on the spatial_memory_summary, risk_assessment, finish_policy, and recommended_next:
- low-risk geometry query: 'space_snapper → geometry_solver → Finish[memory_id]'
- high-risk multiple/ordinal/attribute query: 'space_snapper → object_analyzer(mode="semantic_attribute", targets=<recommended target ids>) → geometry_solver → Finish[memory_id]'
- attribute-only query with close compatible proposal rivals: 'space_snapper → object_analyzer(mode="semantic_attribute") → object_analyzer(mode="view_probe", targets=<top compatible ids>) → Finish[memory_id]'
- proposal-missing, repair-required, or lookup-stalled query: 'space_snapper(scope="around anchor") → object_analyzer(mode="instance_repair", route="fast", max_filtered_num=12) → geometry_solver → Finish[memory_id]'
12. Finish from a candidate already stored in spatial memory. Prefer 'Finish[proposal:<id>]' or 'Finish[repaired:<id>]' using the 'best_candidate'/'memory_candidate_ids' returned by the atomic tools. Only finish when 'geometry_solver.ready_to_finish' is true, 'category_validation.compatible' is true, and 'finish_policy.blockers' is empty. If an observation recommends 'object_analyzer(semantic_attribute)' or 'object_analyzer(instance_repair)', call 'object_analyzer' with that mode and the provided 'recommended_object_analyzer_args' before finishing. The original target noun controls the answer: context/accessory words such as a painting next to a door, a bag on top of a trash can, or a tissue box on a table cannot replace the door/trash can/table itself.
13. Do not call 'object_analyzer(mode="instance_repair")' only because a semantic compare margin is small. For close proposal rivals, use 'geometry_solver' to fuse/rerank active candidates first, or finish if the latest observation says 'recommended_next' is 'Finish' and no geometry constraints remain.
14. If 'space_snapper' has active target candidates and no latest 'geometry_solver' observation has ranked them, do not call target 'instance_repair' yet. Run 'geometry_solver' over the active target candidates and anchors first. Use target 'instance_repair' only after proposals are missing, the latest 'finish_policy'/'recommended_next' asks for repair, or geometry/view evidence remains blocked; anchor repair with an explicit non-target 'target_category' can still be used for missing relation anchors.
15. Do not reinterpret a later context noun as the target after the original target is parsed. For "black door ... painting on the wall next to it", the target remains the door; the painting is only context evidence. If a target candidate has already been verified by 'view_probe', run 'geometry_solver'/Finish instead of spending video-lift budget on a non-target context object.
16. For attributes like "covered with a blanket", "with a bag on it", or "with a tissue box on it", the target noun remains the object before the attribute. Treat the accessory as an anchor/evidence object; do not finish with the accessory. For "covered with X", the target should be below/overlapped by X, not on top of X.
17. For 3D bounding boxes, do not judge validity from the sign or magnitude of center coordinates alone. ScanNet scene coordinates can legitimately contain negative x/y/z values. A valid candidate bbox has exactly six finite numbers and positive box sizes.

Now, please start solving the following question.

A.3.2 3D VISUAL GROUNDING SKILL

Expert Skill: Atomic 3D Visual Grounding

When to use this skill
Use this skill when the user asks for the 3D bounding box of an object in a ScanNet-style RGB-D scene video. The user normally provides a ‘scene$_i d$‘$and a natural - language query$.

Core Principle
This is a ReAct controller over three atomic operations, not a fixed pipeline and not a free-form low-level tool scheduler. The controller repeatedly decides which evidence is missing, calls one atomic tool, reads the observation, and updates its working hypothesis.

The only intended controller-visible tools are:

1. ‘space_snapper‘: Where should I look?
2. ‘object_analyzer‘: What is this object or local region?
3. ‘geometry_solver‘: Does it satisfy the spatial constraints?

Low-level implementation details such as query parsing, object lookup, view selection, SAM segmentation, VLM frame scoring, multi-view expansion, point-cloud reconstruction, and bbox calculation are hidden inside the atomic tools when needed. Do not call those low-level tools unless they are explicitly listed in AVAILABLE TOOLS.

Important: this method is memory-based. Proposals, anchors, repaired instances, relations, and visual evidence are accumulated in ‘spatial_memory.json‘; the final answer should select an existing memory candidate.

Atomic Tool Roles
‘space_snapper‘
Use this first unless a previous observation already contains a fresh spatial snapshot. It creates or updates object-centric spatial memory for the scene or a narrower scope.

Call it to:
- Parse the query into target, attributes, anchors, and spatial constraints.
- Retrieve target candidates from 3D object proposals or ScanNet annotations.
- Retrieve anchor candidates mentioned by relations.
- Narrow the scope, for example ‘entire_scene‘, ‘around monitor‘, ‘near candidate 12‘, ‘left side of room‘, or ‘candidate_set‘.

Useful arguments:
- ‘query‘: raw query.
- ‘scene_id‘: scene id.
- ‘scope‘: current spatial scope.
- ‘granularity‘: ‘coarse‘ first, ‘fine‘ when revisiting an uncertain region.
- ‘parsed_query‘: pass the previous parsed query when available.

‘object_analyzer‘
Use this when the active memory candidates need semantic/attribute evidence, rival comparison, or proposal repair.

Call it to:
- Verify that candidates match the target category and attributes.
- Compare multiple rival candidates.
- Use selected views when available to verify color, material, shape, state, or small details.
- Recover and localize a missing target instance by setting ‘mode‘ to ‘video_lift‘ or ‘instance_repair‘.

Important modes:
- ‘video_lift‘: default mode. Runs internal frame retrieval, SAM/VLM reference selection, temporal expansion, RGB-D reconstruction, and bbox calculation.
- ‘instance_repair‘: same video-based recovery path, used when proposals are missing or unreliable.
- ‘semantic_attribute‘: proposal-side semantic/attribute ranking. This updates active memory candidates and can be enough for simple attribute-only queries when rivals are eliminated.

- ‘compare‘: contrast rival candidates.

Use ‘instance_repair‘/‘video_lift‘ when proposals are missing, too ambiguous, or visibly unreliable. Otherwise, keep reasoning over spatial-memory proposals.

‘geometry_solver‘
Use this whenever the query contains spatial, ordinal, proximity, support, or anchor-based constraints, or when active memory candidates need ranking.

Call it to:
- Score relations such as ‘left_of‘, ‘right_of‘, ‘above‘, ‘below‘, ‘on‘, ‘near‘, ‘between‘, ‘front_of‘, and ‘behind‘.
- Retrieve or use anchor candidates.
- Rank subjects under spatial constraints.
- Eliminate rivals before final answer.

Pass:
- ‘subjects‘: memory ids, candidate ids, files, or candidate lists. If omitted, ‘geometry_solver‘ uses ‘active_candidate_ids‘ from spatial memory.
- ‘anchors‘: anchor ids, anchor file, candidate list, or anchor category terms. - ‘constraints‘: the spatial constraints from ‘space_snapper.parsed_query.conditions‘ or a focused subset.

Controller Policy

Default loop:
‘space_snapper → geometry_solver/object_analyzer as needed → Finish[memory_id]‘

This is not a fixed order. Choose the next atomic action by missing evidence:

- No spatial memory yet: call ‘space_snapper(scope=”entire_scene”)‘.
- No target candidates: narrow scope with ‘space_snapper‘; if still missing, call ‘object_analyzer(mode=”instance_repair”)‘.
- Target category or attributes are uncertain: call ‘object_analyzer(mode=”semantic_attribute”)‘ over the active candidate ids.
- Spatial relations are present and unverified: call ‘geometry_solver‘.
- Multiple candidates remain: use ‘geometry_solver‘ for spatial constraints and ‘object_analyzer(mode=”semantic_attribute”)‘ for visual/semantic constraints; call ‘instance_repair‘ only when proposal evidence is insufficient.
- Evidence is sufficient and a memory candidate is selected: ‘Finish[proposal:<id>]‘ or ‘Finish[repaired:<id>]‘.

Evidence Standard

Only finish when the selected candidate exists in spatial memory and all required target/category/spatial evidence has been checked or is unnecessary for the query.

The final action should contain the selected memory candidate id:
‘Finish[proposal:12]‘

If you output bbox numbers instead, they must be the bbox of an existing memory candidate:
‘Finish[[cx, cy, cz, dx, dy, dz]]‘

If an atomic observation reports an explicit error, continue with a different atomic action or abort with a clear reason.