QGround: Condition-Wise Evidence Aggregation for 3D Grounding with 2D VLMs

Fengyun Wang[1] **Jian Wang**[2] **Dingwei Zhang**[3] **Jinhui Tang**[3] **Qianru Sun**[1,*]
[1]School of Computing and Information Systems, Singapore Management University
[2]School of Computer Science, Nanjing University of Posts and Telecommunications
[3]School of Computer Science and Engineering, Nanjing University of Science and Technology
[*]Corresponding author: qianrusun@smu.edu.sg

Abstract

Using 2D vision-language models (VLMs) for 3D grounding raises a decision-formulation problem: how should image-level evidence be turned into a 3D object decision? This problem matters as pretrained 2D VLMs provide strong visual-semantic evidence, while 3D grounding requires disambiguating objects in cluttered scenes. Current VLM-based formulations make this conversion holistically: the model either selects one object from a rendered candidate set or assigns one overall match judgment to each candidate. Such decisions obscure the evidence needed for disambiguation, since different candidates may satisfy different subsets of the category, attribute, and relational cues. In this work, we reformulate 3D grounding with 2D VLMs as condition-wise evidence aggregation. The reformulation follows three principles: use natural object-centric images rather than rendered candidate images; read the referring expression condition by condition rather than as one holistic query; and use model preference rather than a hard match output as grounding evidence. We instantiate these principles in QGround, a training-free implementation of this decision formulation. Experiments on ScanRefer and Nr3D show improved grounding performance, with analyses demonstrating stronger candidate disambiguation and more interpretable decisions. Code is available at https://github.com/fereenwong/QGround.

1 Introduction

Using 2D vision-language models (VLMs) for 3D grounding raises a decision-formulation problem. A 2D VLM can provide strong image-level visual-semantic evidence, but 3D grounding requires deciding which object in a 3D scene matches the referring expression. This decision is difficult in cluttered scenes, where multiple objects may share the same category, attributes, or spatial context. The central question is therefore how evidence from a 2D VLM should be organized, compared, and converted into a reliable 3D grounding decision.

Existing formulations use an overly holistic decision abstraction. Existing formulations typically answer this question with holistic decisions. SeeGround Li et al. [2025a], for example, renders a candidate set and asks the VLM to select one object in a single step. A natural candidate-wise baseline keeps candidates separate, but still asks for one global match judgment per candidate. These interfaces are simple, but they ask the VLM for the final decision before making the supporting evidence explicit. This is fragile in cluttered scenes: a distractor may share the target category, match a salient attribute, or satisfy part of the spatial context. As illustrated in Fig. 1, one-shot set selection entangles visual grounding with ID selection over a rendered candidate set, while holistic matching can give the same outcome to candidates supported by different cues. In both cases, the VLM's fine-grained judgments are not organized as comparable evidence across candidates. The missing layer is a decision interface that reads this evidence before the final ranking is made.

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

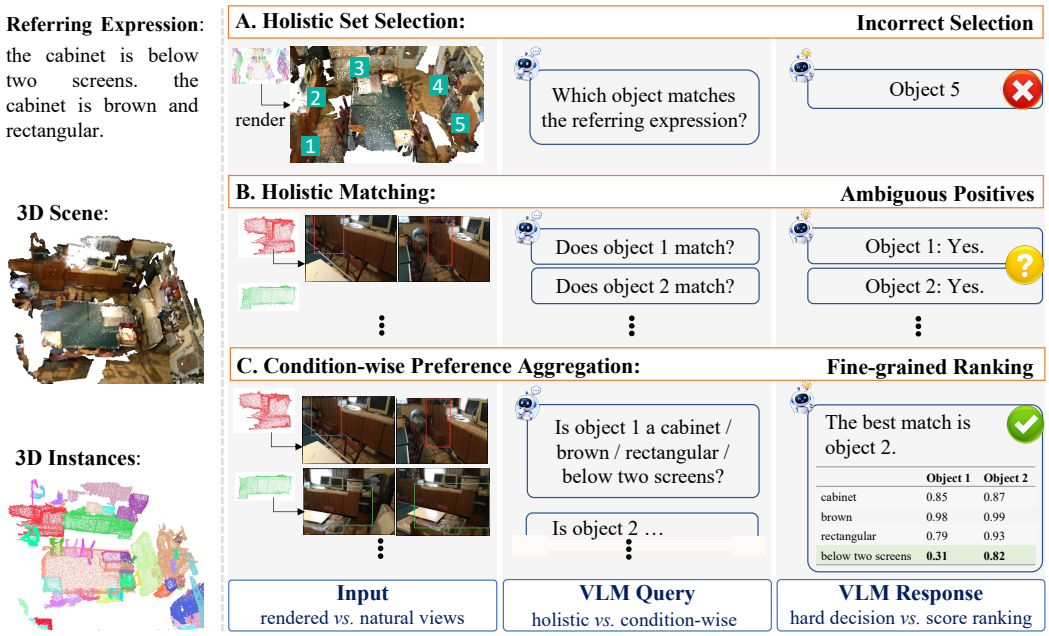


Figure 1: **Decision interfaces for training-free 3D grounding with 2D VLMs.** Holistic set selection makes one decision over a rendered candidate set, and holistic matching scores each candidate with one overall match judgment. QGround instead treats grounding as condition-wise evidence aggregation: the VLM reads object-centric views, evaluates explicit conditions separately, and turns condition-level preferences into the final candidate ranking.

QGround reformulates grounding as condition-wise evidence aggregation. QGround builds this evidence layer through three design rules. (1) *Natural inputs*: the model should read visual evidence that stays close to its native input distribution, rather than rendered candidate sets designed for one-shot selection. (2) *Condition-wise verification*: evidence should be read condition by condition, so salient but incomplete cues do not dominate the entire grounding decision. (3) *Preference reading*: the model output should be used as evidence of support and preference, rather than treated as a hard grounding decision taken directly from next-token prediction. QGround instantiates these rules by verifying each candidate from selected object-centric views, decomposing the referring expression into category, attribute, and relational conditions, and reading candidate preference from the resulting pattern of support and rejection. As shown in Fig. 1, this changes both the decision unit and the model interface: grounding becomes condition-wise preference aggregation over candidates, yielding a more interpretable decision process.

Contributions. In summary, our main contributions are as follows:

- We reformulate training-free 3D grounding with 2D VLMs from holistic decision making into candidate-wise evidence aggregation, where grounding is decided by comparable support across visual and relational cues.
- We introduce *QGround*, an instantiation of this reformulation built on three design rules: natural inputs, condition-wise verification, and preference reading.
- We demonstrate that *QGround* achieves state-of-the-art performance on ScanRefer and Nr3D, and provide analyses showing how the reformulation improves decision interpretability.

2 Related Work

3D Visual Grounding. 3D visual grounding aims to localize a target object in a 3D scene from a natural-language description. Most existing methods learn this ability from task-specific 3D-language supervision. *3D-based methods* operate directly on point clouds, voxels, or object proposals and learn joint 3D-language representations from annotated data. Early supervised methods such as ScanRefer Chen et al. [2020] and ReferIt3DNet Achlioptas et al. [2020] encode geometry and

language jointly, while later transformer- or graph-based designs (*e.g.*, 3DVG-T Zhao et al. [2021], TransRefer3D He et al. [2021], TGNN Huang et al. [2021], BUTD-DETR Jain et al. [2022]) further improve relation modeling and spatial reasoning. To reduce annotation cost, weakly supervised methods such as WS-3DVG Wang et al. [2023] and 3DWG Li et al. [2025b] exploit weaker scene-level signals, but they still require task-specific 3D-language learning. In contrast, our setting is training-free: pretrained 2D VLMs provide semantic evidence, and the central question is how to convert that evidence into a reliable 3D grounding decision.

Training-Free 3D Grounding with 2D VLMs. Training-free methods use pretrained foundation models to interpret referring expressions and reason about objects in 3D scenes. LLM-Grounder, ZSVG3D, and LaSP translate language into tool calls or executable spatial reasoning over 3D objects Yang et al. [2024], Yuan et al. [2024], Mi et al. [2025]. VLM-based methods focus on constructing visual inputs that expose relevant scene evidence: VLM-Grounder uses stitched image sequences Xu et al. [2024], SeeGround combines query-aligned rendered views with spatial text Li et al. [2025a], SeqVLM constructs proposal-guided multi-view image inputs Lin et al. [2025], and SPAZER integrates rendered scene views and retrieved images in a progressive agent pipeline Jin et al. [2025]. These designs improve the coverage and organization of visual evidence, while typically leaving the final target selection to the VLM or agent's joint interpretation of the referring expression. Even when attributes and relations are parsed explicitly, they are generally supplied together to the grounding step, so their individual support or rejection of each candidate remains implicit in the final judgment. This distinction matters when same-category distractors satisfy some conditions but violate others: a holistic judgment can favor a salient partial match without preserving the evidence needed to distinguish it from the target. QGround makes this evidence explicit by verifying each candidate condition by condition and aggregating the resulting signed preferences into the final ranking.

Foundation Models for 3D Scene Understanding. The use of pretrained foundation models in 3D extends beyond grounding. One direction transfers 2D visual priors into 3D perception, either by distilling image-based semantics into point-cloud models or by aligning 3D representations with open-vocabulary image-language features Yan et al. [2022], Peng et al. [2023], He et al. [2024], Wang et al. [2024b], Xu et al. [2025], Wu et al. [2024]. These methods support 3D semantic segmentation, open-vocabulary perception, and cross-domain generalization through dense semantic representations. Another direction builds 3D multimodal LLMs for broader scene reasoning, including 3D question answering, captioning, planning, and general 3D-language interaction Hong et al. [2023], Fu et al. [2024], Zheng et al. [2025], Huang et al. [2025b]. Such models learn interfaces between 3D scenes and language for a broad range of scene-understanding tasks.

Closely Related Ideas. Related studies have explored structured language processing and VLM-based scoring across grounding and image–text evaluation tasks. ViewSRD Huang et al. [2025a] decomposes multi-anchor descriptions to learn textual–scene representations for supervised 3D grounding, while VLM-Grounder Xu et al. [2024] extracts attributes and conditions to guide agent-based view and instance selection. In other tasks, VQAScore Lin et al. [2024] uses positive-answer probabilities to evaluate image–text alignment in text-to-visual generation, and True/False Verification Liu and Hu [2025] judges each 2D box against a complete referring expression. In these works, decomposition supports representation learning or agent guidance, while evaluation and matching remain holistic. QGround instead makes condition-wise verification the basis of the grounding decision: each candidate's signed preferences are aggregated across conditions to determine the final 3D candidate ranking.

3 Method

3.1 Grounding as Candidate Disambiguation

The formulation asks how evidence from a 2D VLM should rank a set of 3D object candidates, obtained from an off-the-shelf 3D proposal method, against the referring expression. Formally, given a referring expression $\mathcal{T}$ and candidates $\mathcal{O} = \{o_1, \ldots, o_N\}$, the goal is to assign each candidate a grounding score $\mathrm{Score}(o_i, \mathcal{T})$ and select the one with the strongest support:

$$\hat{o} = \arg\max_{o_i \in \mathcal{O}} \mathrm{Score}(o_i, \mathcal{T}). \tag{1}$$

The key design choice is what this score represents. QGround treats it as accumulated evidence for how a candidate satisfies the referring expression, rather than a holistic match response. This yields

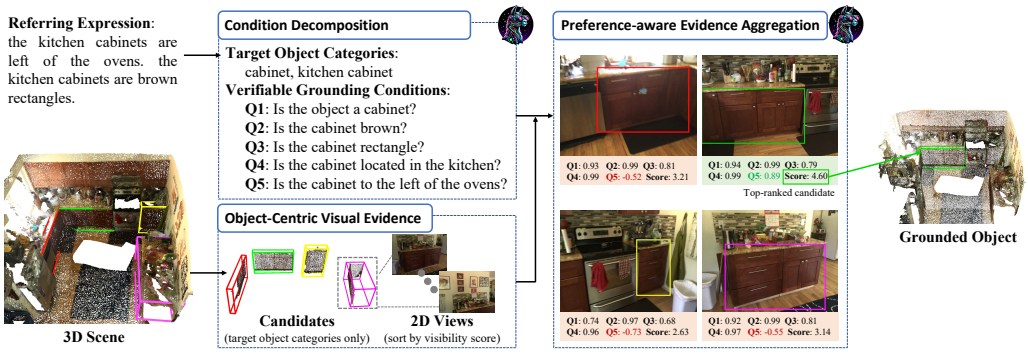


Figure 2: **QGround framework.** QGround turns a referring expression into explicit grounding conditions, verifies each 3D candidate from object-centric natural views, and reads condition-level preferences from a 2D VLM. These preferences are aggregated into a fine-grained candidate ranking for the final grounding decision.

three rules for the decision interface: use object-centric evidence close to the 2D VLM's native image distribution, verify explicit conditions instead of one holistic judgment, and read token preferences instead of a hard decision.

3.2 Object-Centric Visual Evidence

Object-centric evidence defines the visual interface between a 3D candidate and the 2D VLM. QGround uses object-centric evidence from natural scene views as the shared visual evidence for later verification. For each candidate o_i, QGround projects its points into each view and ranks the views by how many candidate points remain visible:

$$\mathrm{Vis}(o_i, v_k) = \frac{n_{\text{visible}}}{n_{\text{total}}}, \tag{2}$$

where n_{visible} counts candidate points that project into the image and remain unoccluded under the view depth ordering, and n_{total} is the total number of points in o_i. The top-K views with the highest visibility score are used as the evidence for o_i, and the candidate region is highlighted in each selected view to center the verification on the intended object while preserving the surrounding scene context.

3.3 Condition Decomposition

Condition decomposition makes the verifiable cues in a referring expression explicit. This allows each candidate to be evaluated by how well it satisfies the evidence in the expression. QGround rewrites a referring expression $\mathcal{T}$ as

$$\mathcal{T} \;\rightarrow\; \big(c, \mathcal{Q}\big), \qquad \mathcal{Q} = \{q_1, \ldots, q_M\}, \tag{3}$$

where c is the target category and each question q_j isolates one verifiable grounding condition, such as visual attributes, scene context, or spatial relations. These questions are used as explicit checks over visual evidence, not as free-form reasoning traces.

Category filtering and condition scoring serve different purposes. The category constraint narrows the search space to the correct semantic type before fine-grained verification. To account for open-vocabulary variation, we expand the target category into a small set of closely related labels $\mathcal{C}$ and keep only candidates whose predicted labels fall into $\mathcal{C}$. Filtering only determines which candidates enter verification; the final ranking is determined by condition-level evidence. This gives the refined decision form

$$\hat{o} = \arg \max_{o_i:\, c_i \in \mathcal{C}} \sum_{j=1}^{M} \mathrm{Score}(o_i, q_j). \tag{4}$$

For example, the expression *"The kitchen cabinets are left of the ovens. The cabinets are brown rectangles"* yields a category constraint centered on *cabinet*, together with verifiable grounding conditions covering appearance (*brown*, *rectangular*), scene context (*in the kitchen*), and spatial relation (*left of the ovens*).

3.4 Preference-Aware Evidence Aggregation

Condition-wise verification should preserve how strongly the VLM supports or rejects each condition. A hard answer loses this ranking signal: two candidates can receive the same positive decision even when one is much better supported by the visual evidence. QGround therefore reads the VLM's token preference as condition-level evidence, rather than using the generated token as the decision itself.

For each question q_j, let $\mathcal{V}_{\text{yes}}$ and $\mathcal{V}_{\text{no}}$ denote small verbalizer sets for yes and no. After the VLM processes the candidate views and the question, the next-token distribution at the answer position gives the model's support for possible responses. The support for each side is obtained by summing the probabilities of its verbalizer tokens:

$$p_{\text{yes}}(o_i, q_j) = \sum_{v \in \mathcal{V}_{\text{yes}}} p(v \mid o_i, q_j), \qquad p_{\text{no}}(o_i, q_j) = \sum_{v \in \mathcal{V}_{\text{no}}} p(v \mid o_i, q_j) \tag{5}$$

where $p(v \mid o_i, q_j)$ is the probability assigned to token v at the answer position. The condition then contributes the signed preference

$$\text{Score}(o_i, q_j) = p_{\text{yes}}(o_i, q_j) - p_{\text{no}}(o_i, q_j). \tag{6}$$

The signed preferences are summed over the shared set of conditions to rank candidates. Equal weighting requires no task-specific fitting. The score difference between two candidates then accumulates their condition-level evidence margins, so the ranking reflects their relative support and rejection across conditions. The aggregate score serves as ranking evidence rather than a calibrated probability. Table S4 compares equal summation with alternative aggregation strategies.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate QGround on two widely used 3D visual grounding benchmarks. *ScanRefer* Chen et al. [2020] contains over 51K free-form descriptions grounded in 800 ScanNet scenes, and is evaluated on *Unique* queries whose target category appears once in the scene and *Multiple* queries with same-category distractors. *Nr3D* Achlioptas et al. [2020], a subset of ReferIt3D, contains human-written queries collected from interactive reference games, with annotations for difficulty (*Easy* vs. *Hard*) and perspective sensitivity (*View-Independent* vs. *View-Dependent*).

Metrics. For ScanRefer, we report Acc@0.25 and Acc@0.5 following the standard evaluation protocol. For Nr3D, we report exact instance-level accuracy, since all ground-truth objects in each scene are included in the candidate set.

Implementation. QGround is evaluated with LLaVA-OV-7B, LLaVA-OV-72B Li et al. [2024], and Qwen2-VL-72B Wang et al. [2024a]. The two LLaVA-OV models assess scaling with model size, while Qwen2-VL-72B enables a backbone-matched comparison with SeeGround. Each reported backbone is used for the full pipeline, including condition decomposition and preference-aware verification. Unless otherwise noted, ablations and analyses use LLaVA-OV-7B. All experiments are training-free, with no task-specific fine-tuning. For ScanRefer, 3D candidates are generated by Mask3D Schult et al. [2022]; for Nr3D, the standard ground-truth-object candidate setting is used. For each candidate, QGround selects the top-$K = 3$ views by the visibility score defined in Sec. 3.

To keep query parsing consistent with prior work, condition decomposition adopts the initial structured parsing instructions of VLM-Grounder Xu et al. [2024] for extracting the target class, attributes, and grounding conditions. Each parsed condition is then converted into a binary question for candidate verification. For efficient candidate-wise verification, one candidate's visual evidence and all verifiable grounding conditions are processed in a single forward pass, with each condition realized as a separate question block and one preference score read at its corresponding answer position. Masked attention prevents cross-condition interference: the answer position for condition q_j can attend only to the visual tokens and the tokens of q_j. The preference score uses the aggregated probabilities of a *yes* token set {`"YES"`, `"Yes"`, `"yes"`, `" YES"`, `" Yes"`, `" yes"`, `"Y"`, `" Y"`} and a *no* token set {`"NO"`, `"No"`, `"no"`, `" NO"`, `" No"`, `" no"`, `"N"`, `" N"`}. On ScanRefer, the final score sums the category-verification score and the condition-verification scores; on Nr3D, category verification is omitted because ground-truth object categories are available. The exact prompts are provided in Supplementary Section S1.1.

Table 1: **3D grounding results on the *ScanRefer* Chen et al. [2020] validation set.** Gray shading indicates results evaluated on 250 selected samples. **Bold** indicates the best training-free result.

Method	Supervision	Agent	Unique		Multiple		Overall	
			Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
ScanRefer Chen et al. [2020]	Fully	-	67.6	46.2	32.1	21.3	39.0	26.1
3DVG-T Zhao et al. [2021]	Fully	-	77.2	58.5	38.4	28.7	45.9	34.5
BUTD-DETR Jain et al. [2022]	Fully	-	84.2	66.3	46.6	35.1	52.2	39.8
3D-VisTA Zhu et al. [2023]	Fully	-	81.6	75.1	43.7	39.1	50.6	45.8
ConcreteNet Unal et al. [2025]	Fully	-	86.4	82.1	42.4	38.4	50.6	46.5
SeeGround Li et al. [2025a]	Training-Free	Qwen2-VL-72b	75.7	68.9	34.0	30.0	44.1	39.4
ZSVG3D Yuan et al. [2024]	Training-Free	GPT-4 turbo	63.8	58.4	27.7	24.6	36.4	32.7
VLM-Grounder Xu et al. [2024]	Training-Free	GPT-4V	66.0	29.8	48.3	33.5	51.6	32.8
SeqVLM Lin et al. [2025]	Training-Free	Doubao-1.5-pro	77.3	72.7	47.8	41.3	55.6	49.6
SPAZER Jin et al. [2025]	Training-Free	GPT-4o	80.9	72.3	**51.7**	43.4	57.2	48.8
QGround (Ours)	Training-Free	Qwen2-VL-72B	78.6	72.0	46.6	41.4	52.8	47.3
QGround (Ours)	Training-Free	LLaVA-OV-7b	79.7	74.3	47.9	42.9	54.1	49.0
QGround (Ours)	Training-Free	LLaVA-OV-72b	**83.0**	**77.6**	51.6	**46.0**	**57.7**	**52.2**

Table 2: **3D grounding results on the *Nr3D* Achlioptas et al. [2020] validation set.** Gray shading indicates results evaluated on 250 selected samples. **Bold** indicates the best training-free result.

Method	Supervision	Agent	Difficulty		Viewpoint		Overall
			Easy	Hard	Dep.	Indep.	
ReferIt3DNet Achlioptas et al. [2020]	Fully	-	43.6	27.9	32.5	37.1	35.6
3DVG-T Zhao et al. [2021]	Fully	-	48.5	34.8	34.8	43.7	40.8
BUTD-DETR Jain et al. [2022]	Fully	-	60.7	48.4	46.0	58.0	54.6
ViL3DRel Chen et al. [2022]	Fully	-	70.2	57.4	62.0	64.5	64.4
ZSVG3D Yuan et al. [2024]	Training-Free	GPT-4 turbo	46.5	31.7	36.8	40.0	39.0
SeeGround Li et al. [2025a]	Training-Free	Qwen2-VL-72B	54.5	38.3	42.3	48.2	46.1
VLM-Grounder Xu et al. [2024]	Training-Free	GPT-4V	55.2	39.5	45.8	49.4	48.0
SeqVLM Lin et al. [2025]	Training-Free	Doubao-1.5-pro	58.1	47.4	51.0	54.5	53.2
SPAZER Jin et al. [2025]	Training-Free	GPT-4o	68.0	**58.8**	**59.9**	66.2	63.8
QGround (Ours)	Training-Free	LLaVA-OV-7B	69.0	49.6	50.7	63.3	59.1
QGround (Ours)	Training-Free	LLaVA-OV-72B	**73.6**	56.2	57.6	**68.2**	**64.7**

4.2 Main Results

Results on ScanRefer. Table 1 demonstrates the effectiveness of QGround's condition-wise verification for ScanRefer grounding. The clearest comparison is with SeeGround Li et al. [2025a]: both use Qwen2-VL-72B and Mask3D proposals, but SeeGround makes a holistic selection over rendered candidates. Under this matched setting, QGround reaches 52.8%/47.3% overall Acc@0.25/Acc@0.5, improving over SeeGround by 8.7/7.9 percentage points. These gains support ranking candidates by their aggregated condition-level evidence rather than relying on a holistic selection. The gains also extend to the *Multiple* split, where same-category distractors make grounding a disambiguation problem: QGround with LLaVA-OV-7B exceeds SeeGround by 13.9/12.9 percentage points, despite using a smaller backbone. Finally, replacing the 7B VLM with LLaVA-OV-72B raises the overall result to 57.7%/52.2%, the best overall training-free performance in the table. This scaling trend suggests that the formulation provides a direct path for converting stronger 2D VLM judgments into better 3D grounding decisions.

Results on Nr3D. Table 2 reports results on Nr3D, where grounding is performed over ground-truth object candidates. With proposal recall excluded, this setting directly tests whether the 2D VLM can rank the correct object from language-conditioned evidence. QGround with LLaVA-OV-7B reaches 59.1% overall accuracy, already surpassing SeeGround and recent training-free methods such as VLM-Grounder and SeqVLM. Scaling the same formulation to LLaVA-OV-72B further improves the overall result to 64.7%, giving the best training-free overall accuracy.

Qualitative comparison. Figure 3 compares QGround with SeeGround Li et al. [2025a] on challenging ScanRefer examples under the same Mask3D Schult et al. [2022] candidate set. Using the same proposals allows the examples to illustrate how the two methods select targets from a shared candidate set. For each case, QGround breaks the description into explicit conditions, reads the VLM's preference for each condition, and aggregates the resulting evidence into the final ranking. The



Figure 3: **Qualitative comparison with condition-wise verification.** The top rows compare the final grounding results of SeeGround and QGround. The bottom rows expose QGround's condition-level evidence: each candidate is checked against decomposed conditions, and the resulting preferences show why distractors are rejected before the final ranking is formed.

condition-level scores identify the attributes and relations that support the target or reject distractors. More results are provided in Supplementary Section S4.

4.3 Ablation Study

Ablation on the design rules. Table 3 starts from the most direct candidate-wise baseline: each candidate is observed through native object-centric evidence, but the VLM still makes one holistic match judgment. Adding condition-wise verification improves the overall result by 9.3/8.2 points, while preference-aware scoring gives a similar gain of 8.7/7.9 points. These two decision rules address different failures: condition-wise verification preserves cue-level evidence, and preference reading makes hard outcomes comparable. They are strongest together on *Multiple* scenes (+15.0/+13.2 points), where the target must be separated from same-category distractors. The native-input rule is tested by replacing native images with rendered inputs under the same aggregation design; the 10.7/10.2-point drop shows that evidence aggregation depends on whether the VLM receives images close to its expected input distribution. Supplementary Fig. S1 shows that preference reading breaks ties among candidates with the same hard condition pattern.

Open-vocabulary category filtering. The final ablation studies how the open-vocabulary category set $\mathcal{C}$ is constructed before verification. As defined in Sec. 3, candidates are retained when their predicted instance label c_i falls into $\mathcal{C}$. The verification scores are fixed from the final QGround pipeline, so the ablation changes only the retained candidate set. Table 4 compares three constructions of $\mathcal{C}$: (1) *Exact Match*, where $\mathcal{C}$ contains only the parsed category c; (2) *Fuzzy Match*, where $\mathcal{C}$ is expanded by partial lexical overlap such as "cabinet" vs. "kitchen cabinet"; (3) *Semantic Match*, where $\mathcal{C}$ is expanded with labels that a 2D VLM judges semantically equivalent to c. Semantic

Table 3: **Ablation on the three design rules of QGround.** The baseline keeps the native input but uses holistic matching. The last row replaces the native input with rendered images while keeping the other two rules enabled. "+"/"-" denote change relative to the baseline/QGround.

Variant	Native input	Cond.-wise	Pref.-aware	Unique		Multiple		Overall	
				Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Baseline	✓			68.2	63.7	32.9	29.7	39.8	36.3
+Cond	✓	✓		77.2$^{+9.0}$	72.0$^{+8.3}$	42.3$^{+9.4}$	37.9$^{+8.2}$	49.1$^{+9.3}$	44.5$^{+8.2}$
+Pref	✓		✓	75.0$^{+6.8}$	69.7$^{+6.0}$	42.2$^{+9.3}$	38.1$^{+8.4}$	48.5$^{+8.7}$	44.2$^{+7.9}$
QGround	✓	✓	✓	79.7$^{+11.5}$	74.3$^{+10.6}$	47.9$^{+15.0}$	42.9$^{+13.2}$	54.1$^{+14.3}$	49.0$^{+12.7}$
replace input with rendered images				72.4$^{-7.3}$	66.5$^{-7.8}$	36.4$^{-11.5}$	32.1$^{-10.8}$	43.4$^{-10.7}$	38.8$^{-10.2}$

Table 4: **Ablation on category matching strategies.** Comparison of *Exact Match*, *Fuzzy Match*, and *Semantic Match*. *Exact Match* keeps only the parsed category label, *Fuzzy Match* allows lexical overlap, and *Semantic Match* expands the category set using 2D VLM judgments. "+" denotes the gain over *Exact Match*.

Method	Unique		Multiple		Overall	
	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Exact Match	75.9	70.9	43.1	38.7	49.5	45.0
Fuzzy Match	77.8$^{+1.9}$	72.3$^{+1.4}$	45.7$^{+2.6}$	41.0$^{+2.3}$	52.0$^{+2.5}$	47.1$^{+2.1}$
Semantic Match	79.7$^{+3.8}$	74.3$^{+3.4}$	47.9$^{+4.8}$	42.9$^{+4.2}$	54.1$^{+4.6}$	49.0$^{+4.0}$

matching makes category filtering more robust to open-vocabulary label variation, improving the overall result over exact matching by 4.6/4.0 points.

4.4 Discussion

Beyond accuracy, QGround raises practical questions about reasoning control, controllable inference cost, and the bottlenecks that still limit candidate-wise grounding.

Explicit verification vs. thinking-based VLM reasoning. Thinking-based VLMs offer a direct alternative to QGround: the model can generate a reasoning trace before making the grounding decision. To compare this alternative under a controlled setting, both variants use the same Qwen3-VL model family, with Qwen3-VL-Thinking-8B for free-form thinking and Qwen3-VL-Instruct-8B as the agent in QGround. The reasoning prompt is provided in Supplementary Section S3.3. Table 5 reports QGround's results on the full ScanRefer validation set, reaching 56.6%/51.2% overall Acc@0.25/Acc@0.5. The two approaches differ in how they structure grounding decisions. (1) *Controlled reasoning.* Thinking-based grounding leaves the decision inside a long free-form trace, where useful cues can mix with transitional text, irrelevant statements, or hallucinated details. QGround instead keeps the decision explicit: each condition contributes its own support or rejection signal. (2) *Lower visual-token cost.* Thinking-based inference autoregressively generates the reasoning trace while attending to visual tokens. QGround keeps decomposition text-only and reads candidate preferences from a single masked-attention forward pass. Figure 4 illustrates this difference on a challenging example.

Table 5: **Comparison with thinking-based VLM reasoning on the ScanRefer validation set.** Both methods use the same Qwen3-VL model family. "+" denotes the gain over the thinking-based baseline.

Method	Agent	Unique		Multiple		Overall	
		Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Qwen3-VL-Thinking	Qwen3-VL-Thinking-8B	70.0	64.9	38.0	34.2	44.2	40.1
QGround	Qwen3-VL-Instruct-8B	82.7$^{+12.7}$	77.2$^{+12.3}$	50.3$^{+12.3}$	45.0$^{+10.8}$	56.6$^{+12.4}$	51.2$^{+11.1}$

Efficiency analysis. Table 6 breaks down QGround's inference cost and its controllable knobs. Timing is measured on 200 queries with LLaVA-OV-7B on an NVIDIA L40 GPU. Description decomposition and category-set construction are query-level language steps, taking 3.12 and 0.70

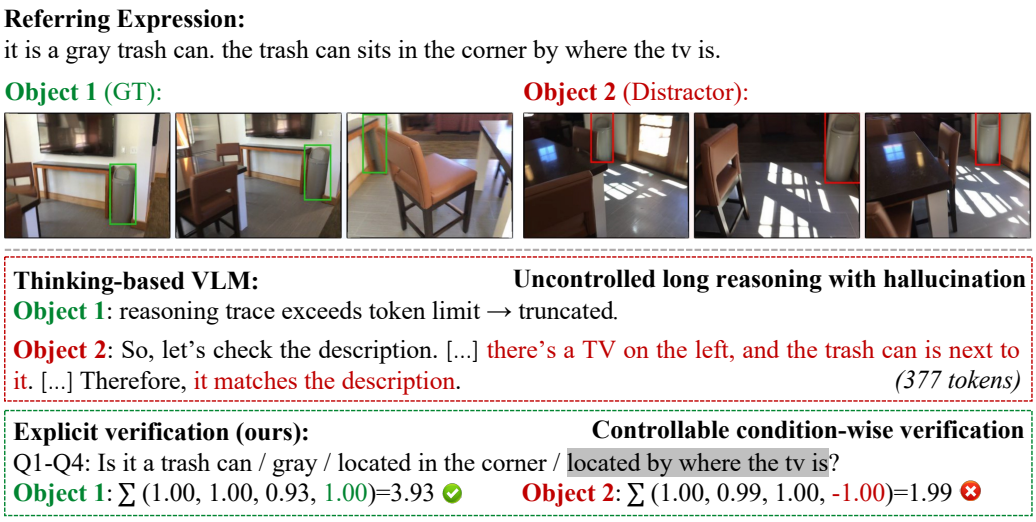


Figure 4: **Explicit verification provides a more controlled alternative to thinking-based VLM reasoning.** Given the same candidate evidence, thinking-based reasoning can produce long free-form traces and hallucinated cues, leading the model to favor a distractor. QGround instead verifies explicit conditions independently and reads condition-level preferences, yielding an inspectable ranking that supports the GT object and rejects the distractor.

seconds, respectively. Verification is candidate-dependent: with the default $K=3$ views, each candidate takes 0.33 seconds as QGround reads answer-token preferences from one VLM forward pass. Category filtering keeps 8.16 candidates per query on average, giving an average latency of 6.51 seconds. The variable cost is the number of retained candidates: filtering leaves fewer than 5 candidates for 53.4% of queries and at least 20 candidates for only 10.5%. For this long tail, a top-20 cutoff gives a calculated single-GPU latency of 10.42 seconds with only a 0.21/0.01-point drop in Acc@0.25/Acc@0.5. Since candidates are verified independently, this verification cost can be divided across GPUs; for example, an ideal four-way parallel verification split would reduce the top-20 verification time from 6.60 seconds to about 1.65 seconds, excluding scheduling and communication overhead; this is an estimate rather than a measured multi-GPU result. Supplementary Table S7 compares SeeGround and QGround under matched backbone and hardware settings.

Table 6: **Efficiency analysis on ScanRefer.** Timing is measured on 200 queries with LLaVA-OV-7B on an NVIDIA L40 GPU. The total uses the average number of retained candidates after filtering.

Stage	Decomposition	Category Set	Verification	Total
Time	3.12 s / query	0.70 s / query	0.33 s / cand. × 8.16 cand. (avg.)	6.51 s / query

Bottleneck: condition-level verification remains imperfect. Condition-wise evidence aggregation depends on whether the 2D VLM can reliably verify each decomposed condition. To isolate errors made by the 2D VLM itself, the ground-truth object is used as the candidate and each decomposed condition is verified independently. Here, rejecting the GT candidate indicates a condition-level misjudgment. The rejection rate is low for category conditions (4.3%), higher for attribute conditions (9.2%), and highest for relation conditions (23.9%). These errors expose a joint limit of the interface: the 2D VLM may misjudge the evidence, and the selected views may not make the relevant evidence sufficiently observable, especially for spatial or object-to-object relations.

Bottleneck: proposal quality limits the ceiling. QGround operates after 3D instance proposals are generated. It can compare and rank candidate objects, but it cannot recover a target that is missing from the proposal set. With Mask3D Schult et al. [2022] proposals, 72.2% of ground-truth targets have a candidate with both the correct semantic class and sufficient spatial overlap (IoU ≥ 0.5), which forms a proposal-level ceiling for candidate-based grounding.

Supplementary Secs. S2–S3 provide further ablations and analyses of pipeline bottlenecks and inference cost.

5 Conclusions

This paper studies how 2D VLM evidence should be converted into a 3D grounding decision. The central issue is not only whether the VLM can recognize visual cues, but whether its fine-grained judgments are preserved as comparable evidence across candidates. QGround addresses this issue by reformulating grounding as condition-wise evidence aggregation: candidates are verified from natural object-centric views, referring expressions are decomposed into explicit conditions, and model preferences are aggregated into the final ranking. Experiments on ScanRefer and Nr3D show that this formulation improves candidate disambiguation, especially in cluttered scenes with same-category distractors, while providing a more structured and interpretable decision process. The remaining bottlenecks suggest that stronger 3D proposal generation and more reliable relation verification are complementary to the proposed decision formulation.

Acknowledgments

The authors gratefully acknowledge the support from the DSO research grant awarded by DSO National Laboratories, Singapore. This project is also partially supported by the Ministry of Education, Singapore, under its Tier-1 Academic Research Fund (No. 24-SIS-SMU-040).

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*, pages 422–440. Springer, 2020.

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*, pages 202–221. Springer, 2020.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Language conditioned spatial relation reasoning for 3d object grounding. In *Advances in Neural Information Processing Systems*, 2022.

Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-llm: Extending language model for 3d visual understanding and reasoning. *arXiv preprint arXiv:2403.11401*, 2024.

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.

Qingdong He, Jinlong Peng, Zhengkai Jiang, Kai Wu, Xiaozhong Ji, Jiangning Zhang, Yabiao Wang, Chengjie Wang, Mingang Chen, and Yunsheng Wu. Unim-ov3d: Uni-modality open-vocabulary 3d scene understanding with fine-grained feature representation. *arXiv preprint arXiv:2401.11395*, 2024.

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

Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3d instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1610–1618, 2021.

Ronggang Huang, Haoxin Yang, Yan Cai, Xuemiao Xu, Huaidong Zhang, and Shengfeng He. Viewsrd: 3d visual grounding via structured multi-view decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9726–9736, 2025a.

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

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*, pages 417–433. Springer, 2022.

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. *arXiv preprint arXiv:2506.21924*, 2025.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

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 Computer Vision and Pattern Recognition Conference*, pages 3707–3717, 2025a.

Xiaoqi Li, Jiaming Liu, Nuowei Han, Liang Heng, Yandong Guo, Hao Dong, and Yang Liu. 3dwg: 3d weakly supervised visual grounding via category and instance-level alignment. *arXiv preprint arXiv:2505.01809*, 2025b.

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. pages 3094–3103, 2025.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024.

Jeffrey Liu and Rongbin Hu. Zero-shot referring expression comprehension via visual-language true/false verification. *arXiv preprint arXiv:2509.09958*, 2025.

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.

Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 815–824, 2023.

Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. *arXiv preprint arXiv:2210.03105*, 2022.

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*, pages 196–213. Springer, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Pengfei Wang, Yuxi Wang, Shuai Li, Zhaoxiang Zhang, Zhen Lei, and Lei Zhang. Open vocabulary 3d scene understanding via geometry guided self-distillation. In *European Conference on Computer Vision*, pages 442–460. Springer, 2024b.

Zehan Wang, Haifeng Huang, Yang Zhao, Linjun Li, Xize Cheng, Yichen Zhu, Aoxiong Yin, and Zhou Zhao. Distilling coarse-to-fine semantic matching knowledge for weakly supervised 3d visual grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2662–2671, 2023.

Yao Wu, Mingwei Xing, Yachao Zhang, Xiaotong Luo, Yuan Xie, and Yanyun Qu. Unidseg: Unified cross-domain 3d semantic segmentation via visual foundation models prior. *Advances in Neural Information Processing Systems*, 37:101223–101249, 2024.

Jingyi Xu, Weidong Yang, Lingdong Kong, Youquan Liu, Qingyuan Zhou, Rui Zhang, Zhijun Li, Wen-Ming Chen, and Ben Fei. Visual foundation models boost cross-modal unsupervised domain adaptation for 3d semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024.

Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European conference on computer vision*, pages 677–695. Springer, 2022.

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7694–7701. IEEE, 2024.

Wenlong Yu, Qilong Wang, Chuang Liu, Dong Li, and Qinghua Hu. CoE: Chain-of-explanation via automatic visual concept circuit description and polysemanticity quantification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4364–4374, 2025.

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*, pages 20623–20633, 2024.

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*, pages 2928–2937, 2021.

Duo Zheng, Shijia Huang, and Liwei Wang. Video-3d llm: Learning position-aware video representation for 3d scene understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8995–9006, 2025.

Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2911–2921, 2023.

Supplementary Material for "QGround: Condition-Wise Evidence Aggregation for 3D Grounding with 2D VLMs"

This supplementary includes:

- Implementation details (Sec. S1), including QGround prompts;
- More ablation studies (Sec. S2), covering the decision rules, view count, VLM backbone, condition types, aggregation, and proposal/label quality;
- Additional discussion (Sec. S3), covering stage-wise pipeline diagnostics, a matched-backbone efficiency comparison, and details of the thinking-based comparison;
- More visualization results (Sec. S4), including a full demo of QGround and additional qualitative comparisons with SeeGround.

S1 Implementation Details

S1.1 QGround Prompts

This supplementary is for Section 3 of the main paper. QGround uses three language-interface prompts: 1) decomposing the referring expression $\mathcal{T}$ into a target category c and verifiable grounding conditions $\mathcal{Q}$; 2) constructing a category set $\mathcal{C}$ from c for open-vocabulary candidate filtering; and 3) reading condition-level preferences for each retained candidate from its object-centric views. Table S1 provides the corresponding prompt templates. The initial instructions for parsing the target class, attributes, and grounding conditions are adapted from VLM-Grounder Xu et al. [2024]. QGround extends the parsed output by turning each grounding condition into an independently verifiable question whose signed evidence contributes directly to candidate ranking.

Table S1: **Prompts of QGround.** The {placeholders} denote variables used in the input templates. In particular, {View v_k^i} refers to the view used to verify instance o_i.

Condition_Decomposition_Prompt

You are working on a 3D visual grounding task, which involves receiving a query that specifies a particular object by describing its attributes and grounding conditions to uniquely identify the object. Here, attributes refer to the inherent properties of the object, such as category, color, appearance, function, etc. Grounding conditions refer to considerations of other objects or other conditions in the scene, such as location, relative position to other objects, etc. Your tasks are:

1. Parse the query and extract the target object category, attributes, and grounding conditions. Each attribute and condition should be returned individually. Sometimes the object's category is not explicitly specified, and you need to deduce it through reasoning.

2. Generate a list of verification questions that can be used to verify if an object in object-centric views matches this description.

Each verification question should be:

- Clear and specific

- Answerable with YES/NO by examining the object in views

- Focused on one aspect (class, single attribute, or single condition)

Your response should be formatted using JSON contained in ```` ```json``` ````. Here are some examples:

Input:

Query: this is a brown cabinet. it is to the right of a picture.

Output:

{

"target_class": "cabinet",

"attributes": ["it's brown"],

"conditions": ["it's to the right of a picture"],

"verification_questions": ["Is the object in the red bounding box a cabinet?", "Is the cabinet in the red bounding box brown?", "Is the cabinet in the red bounding box located to the right of a picture?"]

}

...(two more examples)

Ensure your response adheres strictly to this JSON format, as it will be directly parsed and used. The verification_questions list should include exactly one question for the target_class, one question for each attribute, and one question for each condition.

Query: {Description $\mathcal{T}$}

Category_Set_Construction_Prompt

You are a strict semantic label matcher. Rule:

- Return "YES" only if the instance label and target class refer to the SAME physical object type.

- Otherwise, return "NO".

Guidelines:

YES examples: chair ↔ chair, shoe ↔ shoes, cabinet ↔ kitchen cabinet

NO examples: desk ≠ chair, chair ≠ table, tv ≠ monitor

Output JSON: {"label": "YES" or "NO"} with no extra text.

Query:

Target class: {target_class c}

Instance labels: [{instance_label c_1}, {instance_label c_2}, ...]

Preference_Aware_Verification_Prompt

You are an assistant for visual grounding and object verification. You will be shown multiple views of the SAME object from different viewpoints, with the object marked by a RED BOUNDING BOX in each view.

IMPORTANT: Focus ONLY on the object inside the red bounding box.

Your task is to answer verification questions about the marked object. Answer each question separately with YES or NO.

Instructions:

1. Carefully examine ONLY the object inside the red bounding box across all views.

2. Answer EACH question separately with YES or NO.

3. For spatial relationship questions, check if the relationships are visible in the views.

4. If uncertain about any question, answer NO.

Output format:

Each question is followed by an answer position, where the YES/NO probabilities are read.

Query:

<|vision_start|>{View v_k^i}<|vision_end|> (repeat for K red-boxed views)

...

Q1: {verifiable grounding condition q_1}

Answer:<|im_end|>\n<|im_start|>assistant\n (read the answer probability for q_1 here)

Q2: {verifiable grounding condition q_2}

Answer:<|im_end|>\n<|im_start|>assistant\n (read the answer probability for q_2 here)

...

Condition_Decomposition_Prompt converts a free-form referring expression $\mathcal{T}$ into the structured form $(c, \mathcal{Q})$ used in the main paper. The prompt identifies the target category c and extracts verifiable grounding conditions, including object attributes (*e.g.*, color, shape, appearance, function), scene context, and spatial relations to other objects. Each extracted cue is written as a clear YES/NO question so that candidates can be checked condition by condition. In QGround, c is used to construct the category set $\mathcal{C}$, while $\mathcal{Q}$ defines the condition-level evidence used for candidate scoring.

Category_Set_Construction_Prompt expands the parsed target category c into a small set of candidate labels $\mathcal{C}$. The prompt keeps labels that refer to the same object type as c (*e.g.*, *cabinet* and *kitchen cabinet*) and rejects labels from different object types. The resulting category set is used only for candidate filtering; final ranking is still determined by condition-level evidence.

Preference_Aware_Verification_Prompt evaluates how well a retained candidate satisfies each condition in $\mathcal{Q}$ from its object-centric views. The prompt states that the selected K views show the same candidate from different viewpoints and that the candidate is marked by a red bounding box. Each condition is placed in an independent question block followed by `Answer:<|im_end|>\n<|im_start|>assistant\n`, so the next-token distribution at this position naturally corresponds to the beginning of the answer. QGround sums the probabilities of the yes/no verbalizer tokens from this distribution instead of generating and parsing a full response. This is faster and preserves a continuous preference score for each condition. This exposes category, attribute, context, and relation cues as separate evidence terms for the final candidate ranking.

S2 More Ablation Studies

This section supplements the ablation study in Sec. 4.3. Unless otherwise specified, ablations are conducted on ScanRefer with LLaVA-OV-7B.

S2.1 Cross-Backbone Analysis of the Decision Rules

This section examines whether the gains from the decision rules in Table 3 extend to a different VLM backbone. The same ablation is repeated with Qwen2-VL-7B, keeping the candidate set and native views fixed across the first four variants in Table S2. The last variant replaces native views with rendered images while retaining both decision rules.

Table S2: **Backbone transfer of the three design rules on ScanRefer.** All variants use Qwen2-VL-7B. The baseline uses native input with holistic matching; the last row changes only the input to rendered images. "+"/"-" denote change relative to the baseline/QGround.

Variant	Native input	Cond.-wise	Pref.-aware	Unique		Multiple		Overall	
				Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Baseline	✓			74.3	69.2	39.1	35.0	46.0	41.6
+Cond	✓	✓		77.6$^{+3.3}$	72.1$^{+2.9}$	41.2$^{+2.1}$	36.7$^{+1.7}$	48.2$^{+2.2}$	43.5$^{+1.9}$
+Pref	✓		✓	75.7$^{+1.4}$	70.1$^{+0.9}$	44.3$^{+5.2}$	39.7$^{+4.7}$	50.4$^{+4.4}$	45.6$^{+4.0}$
QGround	✓	✓	✓	79.7$^{+5.4}$	73.8$^{+4.6}$	45.1$^{+6.0}$	40.4$^{+5.4}$	51.8$^{+5.8}$	46.9$^{+5.3}$
replace input with rendered images				70.6$^{-9.1}$	65.3$^{-8.5}$	34.8$^{-10.3}$	30.7$^{-9.7}$	41.7$^{-10.1}$	37.4$^{-9.5}$

Condition-wise verification improves the baseline by 2.2/1.9 points at Acc@0.25/Acc@0.5, preference-aware scoring improves it by 4.4/4.0 points, and their combination is best at 51.8%/46.9%. Replacing native scene images with rendered inputs reduces this result by 10.1/9.5 points. The gains differ in magnitude from those with LLaVA-OV-7B, but the ordering is consistent: both decision rules help, their combination is strongest, and native visual evidence remains important.

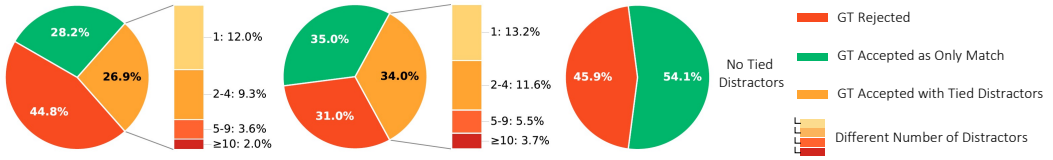
S2.2 Mechanism Analysis of the Decision Rules

Figure S1 explains why the two decision rules in QGround are complementary. Condition-wise verification first prevents useful evidence from being lost in a holistic judgment; preference-aware scoring then makes that evidence comparable across candidates.

(1) Condition-wise verification: recovering valid targets from mixed evidence. Holistic matching gives the VLM only one output for a candidate. When the evidence is mixed, this single outcome can be dominated by an uncertain cue: the ground-truth object may be rejected even though most conditions support it. Condition-wise verification changes the decision unit from an overall match judgment to explicit condition checks. As a result, supporting cues remain visible and can still contribute to the final decision. In Figure S1, this reduces the ground-truth rejection rate from 44.8% to 31.0%.

(2) Preference-aware scoring: breaking ties among candidates with similar hard evidence. Condition-wise verification alone still produces hard condition outcomes. This can make several candidates indistinguishable when they share the same satisfied/violated pattern, which increases tied-distractor cases from 26.9% to 34.0%. Preference-aware scoring resolves this by reading the

strength of support and rejection for each condition rather than only its hard outcome. The continuous evidence separates candidates with the same binary pattern: QGround produces no tied distractors and accepts the ground truth as the only match in 54.1% of cases.



(a) Baseline (b) Baseline + Cond (c) QGround

Figure S1: **Detailed analysis of the two decision rules in Table 3**, based on Acc@0.25. Condition-wise verification rejects fewer ground-truth objects by checking explicit evidence separately (smaller in (b)). Preference-aware scoring converts hard matches into ranked candidate preferences, removing tied distractors in the final decision (no in (c)).

S2.3 Effect of the Number of Views

This section studies how many views should be retained for each candidate.

Figure S2 examines the accuracy–latency trade-off controlled by the number of selected views. Using more than one view is important: moving from $K{=}1$ to $K{=}3$ substantially improves grounding accuracy by exposing complementary evidence for the same candidate. After this point, the accuracy curve begins to saturate, whereas the verification latency on the right axis continues to grow with each additional view. Thus, larger view sets provide increasingly limited evidence gain at a steadily higher per-candidate cost. We use $K{=}3$ by default because it captures the main multi-view benefit while keeping verification efficient.

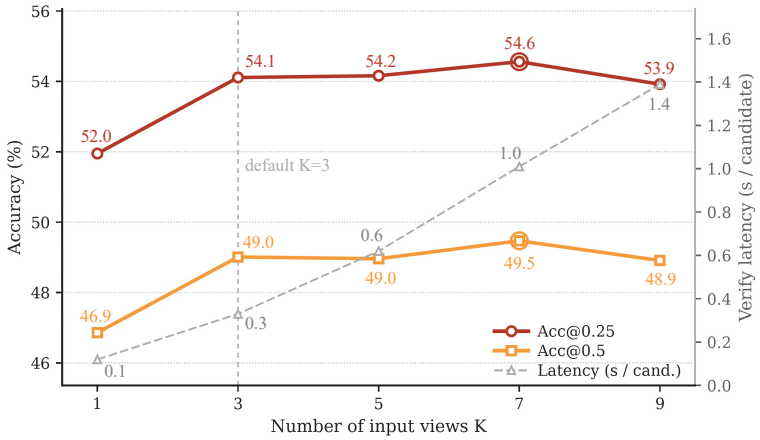


Figure S2: **Effect of the number of selected views.** The left axis reports grounding accuracy, and the right axis reports verification latency in seconds per candidate. Accuracy improves clearly from $K{=}1$ to $K{=}3$, while larger K brings smaller gains with steadily higher latency. This trade-off motivates the default choice of $K{=}3$.

S2.4 Contribution of Condition Types

This section examines how category, attribute, and context/relation conditions contribute to grounding. All variants follow the default QGround pipeline, with candidate construction, category filtering, view selection, and preference scoring held fixed. *Category only* ranks candidates using the category-verification score alone; + *Attribute* adds the scores of attribute conditions. *Full conditions* sums the scores of the complete condition set, further including context and relations.

Table S3: **Ablation of condition types on ScanRefer.** All variants use LLaVA-OV-7B and the same candidate-ranking pipeline. Full conditions add context and relation evidence.

Condition set	Unique		Multiple		Overall	
	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Category only	78.8	72.9	33.8	30.1	42.5	38.4
+ Attribute	78.7	72.9	38.1	34.1	46.0	41.6
Full conditions	79.7	74.3	47.9	42.9	54.1	49.0

Table S3 shows that category evidence alone is effective on *Unique* queries but insufficient on *Multiple* queries. Adding attributes improves *Multiple* Acc@0.25/Acc@0.5 by 4.3/4.0 points; including context and relations adds a further 9.8/8.8 points. The smaller changes on *Unique* are consistent with ScanRefer's dataset structure: *Unique* targets have no same-category distractors, whereas *Multiple* targets do. Under this setting, adding attributes and relations yields larger gains on queries with same-category distractors. The full condition set achieves the highest accuracy on both query groups.

S2.5 Alternative Aggregation Rules

This section examines common aggregation alternatives to QGround's summation, including *Hard voting*, *Rank aggregation*, and *Log-odds sum*. All variants reuse the same LLaVA-OV-7B YES/NO probabilities for each candidate–condition pair, without rerunning the VLM. *Hard voting* gives a candidate one vote for each question with $\mathrm{Score}(o_i, q_j) > 0$ and sums its votes. *Rank aggregation* assigns each candidate per-question ranking points from 1 (lowest preference) to N (highest among N candidates), averages points for ties, and sums the points across questions. *Log-odds sum* sums $\ln[p_{\text{yes}}(o_i, q_j)/p_{\text{no}}(o_i, q_j)]$ across questions, flooring each probability at 10^{-6}. *Sum (Ours)* directly sums $\mathrm{Score}(o_i, q_j)$ over all questions, without weighting or normalization. Each rule selects the candidate with the highest aggregate score.

An additional oracle-style *Type-weighted sum* tests whether optimally chosen weights can substantially improve on summation. It weights three positional groups: the first question (category), the second question when attributes are parsed, and all remaining questions. The group preferences are summed before weighting. The first two weights are searched from 0 to 4 in steps of 0.1, with the third set to 0 or 1. The reported weights $(1.7, 0.9, 1.0)$ are selected to maximize accuracy on the evaluation set itself, providing an oracle-style reference rather than a held-out result.

Table S4 shows that summation reaches 54.1%/49.0%, outperforming the other fixed rules. Even the oracle-style weighted variant improves accuracy by only 0.3/0.3 points, suggesting limited benefit from type weighting in this setting. Summation, the default setting of QGround, achieves comparable performance without tuning weights.

Table S4: **Alternative aggregation rules on ScanRefer with LLaVA-OV-7B.** The gray row is an oracle-style reference with weights selected on the evaluation set; all other rules use fixed aggregation.

Aggregation	Acc@0.25	Acc@0.5
Hard voting	49.1	44.5
Rank aggregation	50.4	45.8
Log-odds sum	53.2	48.3
Sum (Ours)	54.1	49.0
Type-weighted sum (Oracle)	54.4	49.3

S2.6 Proposal and Label Quality

This section examines how proposal quality and category labels affect grounding accuracy. The variants in Table S5 retain QGround's condition decomposition and decision rules while changing the candidate inputs or category-matching strategy. With Mask3D proposals, *Exact* matches candidate labels only to the parsed target category, whereas *Semantic* uses QGround's default open-vocabulary category matching. The two *GT* variants supply ground-truth labels with Mask3D proposals, or ground-truth boxes and labels together.

Semantic matching improves over exact matching by 4.6/4.0 points at Acc@0.25/Acc@0.5, mitigating mismatches between proposal labels and the target category. Ground-truth labels improve accuracy further, indicating that label-related errors remain despite semantic matching. To examine proposal quality separately from label correctness, the *Mask3D / GT* and *GT / GT* rows use ground-truth labels in both settings. Replacing Mask3D proposals with ground-truth boxes improves Acc@0.5 by 13.3 points on *Unique* and 10.6 points on *Multiple*; in both groups, the gain is larger than at Acc@0.25. Correcting labels alone is therefore insufficient: proposal quality remains a substantial bottleneck, especially at the stricter IoU threshold.

Table S5: **Effect of proposal and label quality on ScanRefer.** "Semantic" is QGround's open-vocabulary label matching; "Exact" keeps only the parsed label; and "GT" supplies ground-truth boxes or labels as indicated.

Proposal	Label	Unique		Multiple		Overall	
		Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
Mask3D	Exact	75.9	70.9	43.1	38.7	49.5	45.0
Mask3D	Semantic	79.7	74.3	47.9	42.9	54.1	49.0
Mask3D	GT	86.0	77.2	53.2	47.0	59.6	52.9
GT	GT	91.2	90.5	57.8	57.6	64.3	64.0

S3 Additional Discussion

This section provides stage-wise pipeline diagnostics, a matched-backbone efficiency comparison, and details of the thinking-based comparison. The pipeline diagnostics use the full ScanRefer validation set with LLaVA-OV-7B; the comparison settings are specified in their respective subsections.

S3.1 Stage-Wise Pipeline Diagnostics

Table S6 separates candidate availability, condition evidence, and final selection to locate grounding bottlenecks. Each entry reports the percentage of validation queries meeting the criterion at the specified IoU threshold. Proposal coverage requires only sufficient overlap with the target box; the correct-category reference additionally requires agreement with the ground-truth category. The actual pipeline instead uses the parsed category and semantic expansion to filter proposals, then selects observable views. For the retained target-matching candidates, rows 4–6 cumulatively check positive signed preferences for category, attributes, and all conditions. These diagnostic checks are separate from the final QGround prediction, which ranks candidates by their summed preferences without imposing all-positive requirements. The holistic baseline uses the same VLM, candidates, and views.

Table S6: **Stage-wise ScanRefer pipeline diagnostics.** Values are percentages of validation queries meeting each criterion. Rows 4–6 are diagnostic positive-evidence checks; row 7 reports default QGround accuracy.

Diagnostic criterion	Unique		Multiple		Overall	
	IoU≥0.25	IoU≥0.5	IoU≥0.25	IoU≥0.5	IoU≥0.25	IoU≥0.5
1. Proposal covers target	95.8	88.9	92.7	83.8	93.3	84.8
1b. + Correct proposal category	85.3	79.6	77.4	70.4	78.9	72.2
2. Semantic category filter	84.3	78.9	80.6	73.2	81.3	74.3
3. + Observable selected view	84.2	78.8	80.5	73.0	81.2	74.2
4. + Positive category evidence	83.3	78.0	79.0	71.8	79.8	73.0
5. + Positive attribute evidence	82.4	77.2	77.3	70.2	78.3	71.5
6. + Positive evidence for all conditions	81.6	76.4	76.2	69.2	77.2	70.6
7. Final QGround prediction	79.7	74.3	47.9	42.9	54.1	49.0
Holistic matching baseline	68.2	63.7	32.9	29.7	39.8	36.3

The reduction from proposal coverage to the correct-category reference shows that box overlap alone does not ensure that the target has a correct proposal label. Relative to this reference, semantic filtering slightly lowers retention on *Unique* but raises it on *Multiple*. The filter can exclude correctly

labelled proposals when the parsed category is wrong, while semantic expansion can recover target proposals assigned related labels. The comparison thus reflects both query parsing and semantic label matching, rather than an isolated gain from expansion.

Requiring an observable view removes few additional targets, but view availability does not guarantee reliable condition evidence. Likewise, positive condition evidence does not guarantee correct selection: on *Multiple*, the all-positive diagnostic rate exceeds final accuracy by 26.3 points at IoU 0.5. A target can receive support for every condition yet lose to a competing candidate with a higher aggregate score.

The stage-wise analysis suggests that proposal coverage and category filtering constrain target availability, while using condition evidence to distinguish the target from same-category candidates remains a key downstream challenge.

S3.2 Efficiency Comparison

This section supplements the main efficiency analysis with a backbone- and hardware-matched comparison of SeeGround and QGround. Both use Qwen2-VL-7B on one NVIDIA L40. Table S7 separates query processing from VLM grounding; reported totals sum these costs, with QGround's candidate verification counted sequentially. SeeGround generates an object selection through autoregressive decoding. QGround instead reads YES/NO token probabilities directly as condition preferences, avoiding generation of a textual judgment. All conditions for one candidate share a single masked-attention forward pass, rather than requiring a separate pass per condition.

Table S7: **Backbone- and hardware-matched efficiency comparison.** SeeGround and QGround use Qwen2-VL-7B on one NVIDIA L40. Total time sums query processing and VLM grounding, with sequential candidate verification for QGround.

Item	SeeGround	QGround
Render / stitch	render 1 view	none
VLM grounding calls	1	1 / candidate (8.16 avg.)
Grounding decode tokens	about 70	0
Query processing time	0.5 s	3.5 s
VLM grounding time / query	2.5 s	$0.27\text{ s} \times 8.16 \approx 2.2\text{ s}$
Total time	3.0 s	5.7 s

Decoding-free verification takes 0.27 seconds per candidate, with all conditions scored together. For the average 8.16 retained candidates, sequential verification costs approximately 2.2 seconds per query, slightly below SeeGround's 2.5-second grounding time. Query processing takes 3.5 seconds for QGround versus 0.5 seconds for SeeGround, accounting for QGround's higher total time. The comparable grounding-stage cost with Qwen2-VL-7B complements the accuracy gains reported in Table 1: with both methods using Qwen2-VL-72B, QGround improves Acc@0.25/Acc@0.5 by 8.7/7.9 points over SeeGround.

S3.3 Details of the Thinking-Based Comparison

For the comparison in Table 5, the thinking-based baseline uses the same candidate objects and multi-view scene inputs as QGround, with each evaluated candidate marked by a red bounding box. Qwen3-VL-Thinking-8B uses its built-in thinking mode to generate reasoning and return one candidate-level YES/NO judgment. To match the model family and parameter scale, QGround uses the corresponding Qwen3-VL-Instruct-8B variant for explicit condition verification and preference scoring. The prompt below requests concise checks of category, attributes, and spatial relations in order, but does not require a separate output for each individual condition in the description.

Table S8: **Prompt for the thinking-based VLM baseline.** The prompt asks the thinking model to verify one red-boxed candidate against the referring description and output a final YES/NO decision. The {placeholders} denote variables in the input template.

Thinking_Based_Verification_Prompt

You are a visual grounding judge. You will see ONE candidate object marked with a RED bounding box across multiple viewpoint images of an indoor 3D scene.

Your task: decide whether this candidate matches the natural-language description. Verify each of these in order, but be CONCISE – one short sentence per check is enough:
1. Category – is the candidate the same object class the description refers to? If not, answer NO.
2. Attributes – color, material, shape, size, state. Do the visible attributes match what the description says, or at least not contradict it?
3. Spatial relations – if the description mentions location or relations to other objects, do the surrounding views support that?
Keep your reasoning short (a few sentences total, no bullet lists, no repetition). Then end with EXACTLY one line:
Final answer: YES or Final answer: NO

Query:
{Description $\mathcal{T}$}
Candidate object (red bounding box), shown in {N} views:
{View v_k^i} (repeat for N red-boxed views)
Verify briefly: (1) category, (2) visible attributes (color/material/shape/size), (3) spatial relations from the description. Then end with: Final answer: YES or Final answer: NO

S4 More Visualization Results

This supplementary is for Section 4 of the main paper.

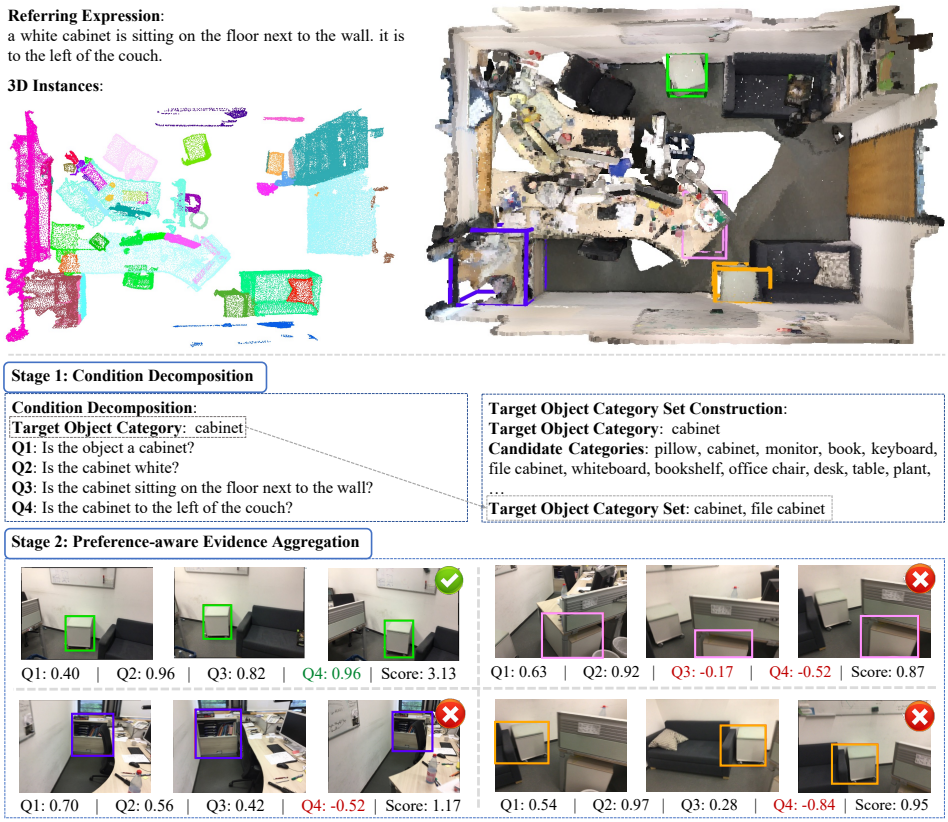


Figure S3: **A full demo of QGround. Top:** Input referring expression, 3D scene, and Mask3D Schult et al. [2022] instance proposals. **Bottom:** QGround first decomposes the expression into a target category c and verifiable grounding conditions $\mathcal{Q}$, then expands c into a category set $\mathcal{C}$ for open-vocabulary candidate filtering. Retained candidates are evaluated from object-centric views with preference-aware condition verification, and the final object is selected by aggregating condition-level support and rejection.

A Full Demo of QGround. Figure S3 presents a complete example of QGround on a real 3D indoor scene. Mask3D Schult et al. [2022] first produces 74 instance proposals. QGround parses the referring expression into a target category and verifiable grounding conditions, then expands the

category into a small category set containing *cabinet* and *file cabinet*. This filtering step keeps 7 candidate objects. For representative candidates, the figure shows the selected object-centric views and the condition-level preference signals read from the 2D VLM. The final prediction is obtained by aggregating support and rejection across conditions, rather than by counting hard "YES" responses.

More Qualitative Comparison. Figure S4 provides additional qualitative comparisons between QGround and SeeGround on challenging referring expressions. Both methods operate on the same Mask3D candidate set, but they expose different decision behaviors: SeeGround makes a holistic set-selection decision, whereas QGround builds the ranking from condition-level support and rejection. Visual explanations help connect model decisions to the image evidence supporting them Yu et al. [2025]. The visualized evidence shows which attributes or relations support the target and which cues reject distractors, making the final decision easier to inspect.

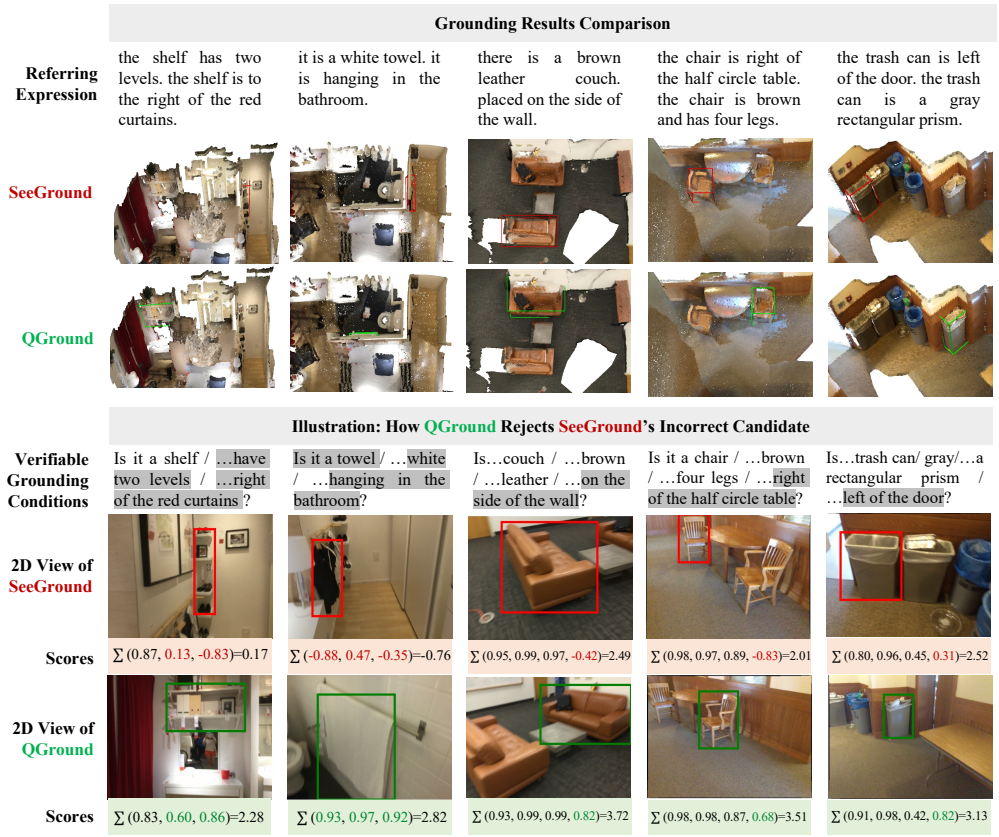


Figure S4: **Additional qualitative comparison with condition-wise preference aggregation.** The top rows compare the final grounding results of SeeGround and QGround under the same Mask3D Schult et al. [2022] candidate set. The bottom rows show QGround's condition-level evidence: decomposed conditions produce support and rejection signals for each candidate, making the final ranking more interpretable than a holistic grounding decision.

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 state the central claims of the paper: 3D grounding with 2D VLMs requires converting fine-grained visual-semantic judgments into comparable candidate evidence, existing holistic formulations are mismatched to this difficulty, and QGround addresses it through condition-wise evidence aggregation. These claims are supported by the method design and experiments on ScanRefer and Nr3D.

   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 paper discusses the main limitations of the current approach, including condition-level verification errors, dependence on proposal quality, and test-time inference cost. The experiments are conducted on ScanRefer and Nr3D, so broader generalization beyond these benchmarks remains an empirical question.

   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 does not present theoretical results requiring formal assumptions 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: The paper reports the datasets, metrics, backbone model, and key inference settings. Code and instructions for reproducing the main experimental results are provided in the public repository linked in the abstract.

   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 public repository linked in the abstract provides code and instructions for reproducing the main experiments. The paper uses public benchmarks and existing pretrained models.

   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 paper specifies the datasets, evaluation metrics, no-finetuning setting, backbone models, and key inference choices, and additional implementation details are provided in the supplemental material.

   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: [No]

   Justification: The paper reports single-run benchmark results following standard evaluation protocols, but it does not provide error bars, confidence intervals, or formal statistical significance tests.

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: [No]

   Justification: The paper reports timing for the efficiency analysis on an NVIDIA L40 GPU, but it does not provide full information about memory usage, per-experiment runtime, or total compute budget for all reported experiments.

   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 conforms to the NeurIPS Code of Ethics. The work uses public benchmarks and existing pretrained models and does not involve human subjects or private data collection by the authors.

   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: [No]

    Justification: The current version does not include a dedicated discussion of broader societal impacts. As a foundational 3D grounding method, the work may support applications in robotics and embodied systems, but errors in grounding could also pose risks in safety-critical or surveillance-related settings.

    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 paper does not release high-risk assets such as generative models, scraped datasets, or dual-use systems that require special release safeguards.

    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: [No]

Justification: The paper cites the public datasets, pretrained models, and prior methods used in the experiments, but it does not yet explicitly document the corresponding licenses and terms of use.

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: The released QGround code is documented in the public repository linked in the abstract. Its README provides installation, data preparation, inference, and evaluation instructions.

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 paper does 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 paper does 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.
    - 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: Pretrained multimodal language models / VLMs are core components of the method. LLaVA-OV models are used for condition decomposition and candidate verification, and Qwen3-VL models are used in the thinking-based comparison, as described in the paper.

    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.