WHERE AND WHY IS GROUNDED REASONING USEFUL?

Anonymous authors
Paper under double-blind review

ABSTRACT

Grounded reasoning, where a model outputs text coordinates of relevant locations in chain-of-thought reasoning, is a common strategy for solving complex visual reasoning problems. However, the benefits and limitations of grounded reasoning remain understudied. We study where and why grounded reasoning is most useful. To do so, we introduce a chain-of-thought analysis to measure how much a model uses its outputted grounding in its thinking trace and perform a detailed ablation study of a grounded reasoning training recipe. We find that grounded reasoning improves counting but shows no significant gain over baselines without coordinate grounding across a wide range of spatial reasoning and perception benchmarks. Using several synthetic tasks, we show that grounded reasoning is most useful when a task requires undescribable references to visual concepts that are difficult to describe in language. However, we observe that many existing visual reasoning benchmarks are solvable with only describable text references. To close this gap, we introduce BrickBench, a new benchmark that requires undescribable visual references to solve.

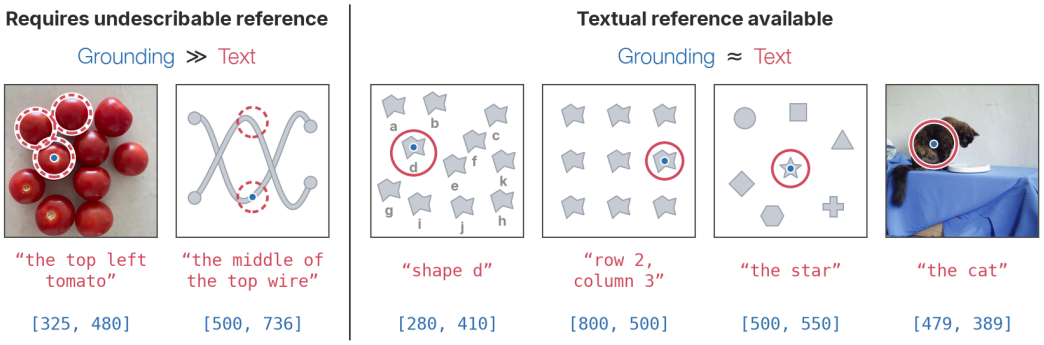


Figure 1: **Visual grounding is useful as an undescribable reference**. When objects in a task are uniquely identifiable in text (e.g. through a label, descriptor, or position), we say that a *textual reference* is available. When textual references are available, grounded reasoning performs no better than text reasoning. When no unique text description is available (e.g. in dense counting or tracing tasks), we say the task requires maintaining *undescribable references*. Grounded reasoning significantly outperforms text reasoning when the task requires undescribable references.

1 INTRODUCTION

While solving visual tasks, humans shift their gaze to objects pertaining to the problem at hand. Grounded reasoning models mimic this behavior by enabling a vision language model (VLM) to output text coordinates while reasoning. Grounded reasoning has a number of appealing properties. It can reduce hallucinations by anchoring the VLM to certain regions of the image and serves as an interpretable way of diagnosing model failures. Moreover, recent coordinate-based grounded

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

reasoning does not require reprocessing of cropped images or replaying of visual features[1]. For these reasons, training recipes for grounded reasoning have received broad interest (Lu et al., 2026; Wang et al., 2026a; Zhang et al., 2026b; Sarch et al., 2026).

However, it remains unclear where and why grounded reasoning could be useful. For instance, recent results from iVGR (Zhang et al., 2026a) suggest that, on certain tasks, text-only reasoning modes can outperform grounded reasoning. As such, an important goal is to understand which tasks necessitate the additional complexity of grounded reasoning and which domains can be solved with natural language reasoning. In this work, we investigate the following question: *where and why is grounded reasoning actually necessary to solve a task*?

We first perform a training-free study of whether existing models actually rely on their intermediate grounding outputs to solve tasks. Existing chain-of-thought (CoT) interpretability techniques focus on editing the chain-of-thought after generation and examining if the model answer changes (Lanham et al., 2023). However, simply removing or randomizing points after the CoT is generated is an insufficient test, because the model might have derived information from the coordinates in subsequent text reasoning. We therefore propose a test for measuring grounding usage that randomizes the CoT's emitted points during inference (Figure 2). Our method shows that many existing open-source grounded reasoning models do not rely on grounding for perception and spatial reasoning tasks, with performance remaining generally unchanged after coordinate randomization (Table 1).

In order to understand this phenomenon, we train our own grounded reasoning models and carefully study the effect of different training recipes on downstream performance. We find that the usage of model grounding is sensitive to the accuracy of the grounding coordinates during SFT, but is also generally uncorrelated with task performance. Moreover, even with high-quality reasoning traces and grounding annotations, we find that grounded reasoning is not helpful when compared to a strong text-only reasoning baseline on a wide range of perception and reasoning benchmarks (Table 3). In contrast, we find that grounded reasoning is helpful for counting benchmarks. How can we explain why grounding is helpful for counting tasks but not for spatial reasoning or perception?

We hypothesize that the major value of grounded reasoning is when a task requires maintaining *undescribable references* to concepts that are difficult to describe in language (Figure 1). To demonstrate this, we develop a suite of evaluations across tracing, counting, and tracking tasks. For each benchmark, we train a model on high quality, in-domain chain of thought traces for both text-only and grounded reasoning. When visual concepts are hard to describe with text, the benefit of grounding is large. Conversely, when labels are added to the task, the marginal value of grounding over pure-text reasoning drops significantly (Figure 4).

We observe that most existing visual reasoning benchmarks struggle to demonstrate the benefit of grounded reasoning because they are mostly solvable with describable references. To close this gap, we use insights from our study to design BrickBench, a challenging new visual reasoning benchmark that requires maintaining undescribable references. On BrickBench, models such as GPT-5.6 Sol and Gemini 3.8 Flash score just 18.9% and 15.6%, respectively (Table 4). Our main contributions are as follows:

1. We introduce a chain-of-thought faithfulness technique to measure grounding usage in existing reasoning models and find that many open models do not use their generated coordinates when reasoning.

2. We perform a detailed ablation study of the grounding training recipe across spatial reasoning, fine-grained perception, and counting tasks. We find that grounded reasoning outperforms natural language reasoning on counting and other tasks that require undescribable references to visual concepts that are difficult to describe in language.

3. We introduce BrickBench, a challenging benchmark designed to measure a model's capability to reason with undescribable visual references. Many of the strongest foundation models today perform poorly on BrickBench, scoring under 20%.

[1]Throughout this paper, we use "grounded reasoning" to refer to models and techniques that only output text coordinates during reasoning, as opposed to adding cropped images or features into context.

2 Related Work

Grounded reasoning models. A longstanding issue for VLMs has been over-reliance on textual priors (Asadi et al., 2026; Chen et al., 2024; Vo et al., 2026; Bai et al., 2025b; Kancheti et al., 2026), which weakens perception and visual reasoning performance. As such, a large variety of techniques exist to enable models to reground to an image during reasoning. One common approach is to format the re-grounding operation as a tool call, which then runs a crop or zoom operation (Zheng et al., 2026). However, repeatedly processing images through tool calls adds significant overhead. An alternative approach is to replay a subset of the original image's features into the context (Wang et al., 2026b). This method requires specialized training and modules to process image tokens, adding significant complexity to model design. Textual grounding approaches, where a model simply interleaves text coordinates during reasoning, enable a model to reason over grounding without accumulating image tokens or requiring any bespoke architectural changes. These properties are highly appealing, and as such, a number of recent works have focused on developing training recipes for grounded reasoning (Lu et al., 2026; Wang et al., 2026a; Zhang et al., 2026a;b; Dohmann & PerceptronAI, 2026; Sarch et al., 2026). For these reasons, we opt to primarily study textual grounding over other visual grounding variants.

Analysis of textual grounding. Recent works have provided some insight into the strengths and weaknesses of grounding. iVGR (Zhang et al., 2026a) finds that when grounded reasoning models are prompted not to output coordinates, their performance generally increases across a large range of benchmarks. They attribute this phenomenon to the model making errors in grounding. We find contradictory results: model performance generally remains stable even when all grounding coordinates are forced to be incorrect by our decode randomization. Dohmann & PerceptronAI (2026) performs a similar analysis of the textual grounding recipe but only considers the single synthetic task of Set. We benchmark grounding across a variety of real-world evaluation benchmarks and develop several synthetic tasks to illustrate the strength of grounding. Lu et al. (2026) claims that textual grounding is helpful for topological reasoning tasks like wire-tracing but does not compare against pure-text reasoning baselines. We provide controlled experiments against text reasoning and propose a more general explanation for where textual grounding is most useful.

Visual reasoning benchmarks. Precise perception and spatial understanding are widely observed weaknesses of VLMs. As such, a wide range of reasoning benchmarks seek to evaluate VLMs on such capabilities (Liu et al., 2025; Lee et al., 2026; Zhang et al., 2025; Du et al., 2024; Jia et al., 2026). Lu et al. (2026), Chen et al. (2026), and Berman & Deng (2025) also consider the tracing task. However, we use it as a controlled training environment to demonstrate the effectiveness of grounded reasoning over text reasoning and direct fine-tuning, as opposed to an evaluation of the zero-shot capabilities of VLMs.

Most similar to our work is BabyVision (Chen et al., 2026), which also considers visual tasks that are difficult to describe in language. However, many BabyVision tasks accept text answer formats and therefore contain textualizable references for objects (e.g., "which puzzle piece in the image, labeled A, B, C, D, would fit into the gap"). In contrast, objects in BrickBench are unlabeled and placed in irregular locations; as such, undescribable references are necessitated. Moreover, BrickBench uses points as the answer format, which requires greater precision compared to a multiple-choice question and yields a more interpretable analysis of failure modes.

3 Where Is Grounding Useful?

3.1 Do Models Use Their Grounding?

Decode-time Grounding Randomization. Our goal is to determine whether a grounded reasoning model uses its grounding in its reasoning. One approach would be to have the model emit its full reasoning trace and answer, and then measure if the model changes its answer if the coordinates are randomized. However, this approach doesn't account for information that the model may have already derived from the grounding during further text reasoning. For example, a model that emits the coordinate of two objects may use those coordinates to reason "object A is at [x1,y1] and object B is at [x2, y2], therefore A is to the left of B." Randomizing just the coordinates after the full reasoning trace is generated would still preserve "therefore A is to the left of B," which the model could still use to solve the problem.

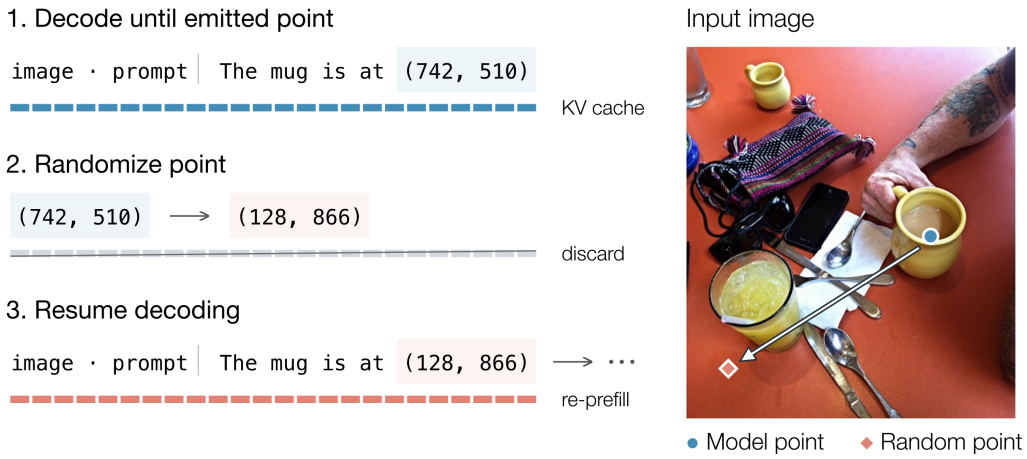


Figure 2: **Decode Grounding Corruption**. We introduce a CoT analysis technique to analyze whether a model uses its grounding. Instead of editing CoT after generation, we randomize the model's outputted points during decoding and reset the KV-cache.

Therefore, to measure whether existing model checkpoints actually use their grounding coordinates, we introduce a chain-of-thought faithfulness test to corrupt coordinates *during decoding*. Whenever a model generates a coordinate, we randomize the coordinate and reset the KV-cache using the edited prefix. As such, all future generation is conditioned on a randomized coordinate, which gives more accurate signal on the effect of the coordinates during reasoning. This method is similar to the "Adding Mistakes" intervention in Lanham et al. (2023), but is instead executed programmatically and performed multiple times per completion. For each question, we sample 20 completions at temperature 1.0, and report the change in mean performance (% acc) after randomization.

Model	MultihopSpatial	SpatialMQA	V*	CountQA
Open models				
TreeVGR-7B-CI	-0.62 ± 0.33	-0.50 ± 0.28	$+0.55 \pm 0.71$	-0.05 ± 0.31
TreeVGR-7B-RL	-0.17 ± 0.22	-0.21 ± 0.19	-0.08 ± 0.35	-2.85 ± 0.28
iVGR-CI	-0.93 ± 0.29	-0.41 ± 0.32	-0.99 ± 0.51	-2.68 ± 0.36
DeepEyes-7B	$+0.63 \pm 0.24$	$+0.02 \pm 0.26$	-0.68 ± 0.38	-0.32 ± 0.28
Ours (format type)				
Appended	-1.64 ± 0.31	-0.61 ± 0.35	-2.36 ± 0.59	-0.38 ± 0.38
Reflection	-3.63 ± 0.30	-2.50 ± 0.35	-4.16 ± 0.62	-4.44 ± 0.37

Table 1: **Grounding usage experiments.** We report change in mean performance for our coordinate corruption experiments. Many open models don't rely on their grounding for spatial reasoning and perception. Our retrained models (below) are more reliant on their grounding outputs.

Existing recipes don't necessarily lead to grounding usage. We use our grounding randomization technique to analyze grounded reasoning models, such as TreeVGR, iVGR, and DeepEyes, across several benchmarks. We observe that, in general, open models do not significantly use their grounding (Table 1), except on counting tasks. This finding re-contextualizes results from iVGR, which finds that models prompted not to use grounding see performance increases. Zhang et al. (2026a) posits that this difference can be attributed to models making grounding errors during reasoning, which the models then incorrectly reason over. However, our results demonstrate that models don't experience significant performance degradation even when all grounding is forced to be incorrect in our corruption experiments, suggesting that this explanation is incomplete.

3.2 Studying the Grounded Reasoning Recipe

Problem setting and evaluation. Our goal is to study various aspects of the training recipe for textual grounding. We train models across many variations of a training recipe and evaluate them

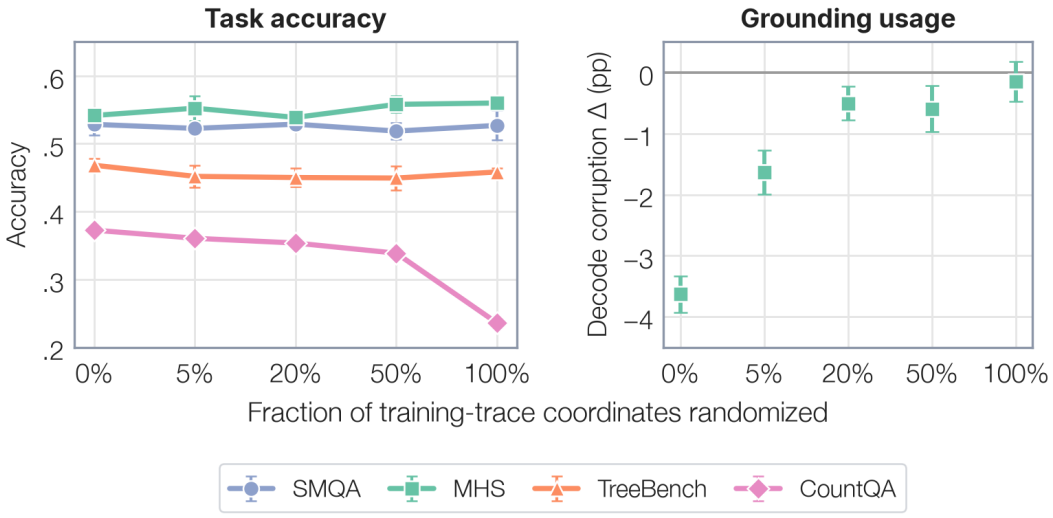


Figure 3: **Training Coordinate Randomization**. We randomize varying percentages of coordinates in the SFT set and report task performance and grounding usage. While grounding usage decreases with more randomization, task performance remains relatively flat, except on counting.

over a wide range of public benchmarks where we expect grounding might help. We categorize each benchmark into one of the following broad task types: spatial reasoning, perception, and counting. For spatial reasoning, we evaluate on SpatialMQA (SMQA) (Liu et al., 2025), MultihopSpatial (MHS) (Lee et al., 2026), EmbSpatial (Du et al., 2024), VSR (Liu et al., 2023), OmniSpatial (Jia et al., 2026), and ERQA (Team et al., 2025). For perception, we evaluate on V* (Wu & Xie, 2023), MME-RW (Zhang et al., 2025), and TreeBench (Wang et al., 2026a). For counting, we evaluate on CountQA (Tamarapalli et al., 2025) and FSC-147 (Ranjan et al., 2021).

Unless otherwise specified, we train 3 checkpoints with random seeds. For each trained checkpoint, we evaluate 3 times with separate random seeds at a temperature of 1.0 and report the mean result for each benchmark. To evaluate the differences between grounded and ungrounded checkpoints, we report the mean performance delta of the grounded performance minus the ungrounded performance (denoted $\bar{\Delta}$). We then compute a 95% confidence interval for the delta (see Appendix A for more detail). We consider a delta significant if the confidence interval does not include zero.

Dataset construction. We observe that existing open datasets have incorrect or low-quality grounding and/or reasoning traces. For instance, the TreeVGR SFT dataset has bounding boxes that are extremely large and generally inaccurate (Appendix 7). On the other hand, datasets from Thinking with Visual Grounding (Zhang et al., 2026b) have accurate grounding but excessively long thinking traces. As such, we construct our own high-quality SFT datasets based on Zhang et al. (2026b)'s annotated SpatialMQA and MultihopSpatial sets. We first generate ungrounded thinking traces by prompting Gemini 3.5 Flash (Google DeepMind, 2026a) to produce a rationale for the solution without grounding. We then provide the QA, trace, and ground truth grounding annotations to Gemini 3.5 Flash and prompt the model to edit the trace to incorporate the grounding annotations. We choose to use point annotations over boxes in our grounded reasoning data, as we find points generally perform better than boxes across most benchmarks (Appendix 6). Additionally, we filter out samples that contain incorrect coordinates or large quantities of irrelevant grounding coordinates. This yields a high-quality grounded SFT dataset of 6,368 samples. For RL, we use the TreeVGR RL dataset, which is composed of questions from the V^* train set (Wu & Xie, 2023) and VisDrone (Zhu et al., 2021). We additionally supplement the RL training set by rolling out a grounded reasoning SFT checkpoint on the SFT QA set and adding any question with pass@8 $\in [0.2, 0.8]$.

Training. Following prior work (Wang et al., 2026a; Zhang et al., 2026a), we first perform a cold-start SFT stage before reinforcement learning. Unless otherwise specified, we initialize from the Qwen3-VL 8B (Bai et al., 2025a) instruct model. More recent models, such as Qwen3.5 (Qwen Team, 2026b), are explicitly trained for grounded reasoning. To isolate the effect of our training

Benchmark	n	SFT Ungrounded	SFT Grounded	SFT Δ	RL Ungrounded	RL Grounded	RL Δ
Spatial reasoning							
SMQA	1076	0.537	0.527	-1.0 ± 2.1	0.571	0.562	-0.9 ± 1.8
MHS	1500	0.551	0.539	-1.3 ± 1.3	0.615	0.588	$\mathbf{-2.7 \pm 1.9}$
EmbSpatial	3640	0.730	0.733	$+0.2 \pm 0.7$	0.791	0.775	$\mathbf{-1.7 \pm 0.9}$
VSR	1222	0.803	0.801	-0.2 ± 1.5	0.789	0.733	$\mathbf{-5.5 \pm 1.3}$
OmniSpatial	1533	0.473	0.470	-0.3 ± 1.2	0.510	0.518	$+0.9 \pm 1.5$
ERQA	400	0.393	0.396	$+0.3 \pm 2.7$	0.432	0.423	-0.9 ± 3.0
Fine-grained perception							
V*	191	0.834	0.844	$+1.0 \pm 2.7$	0.876	0.898	$+2.3 \pm 2.7$
MME-RW	1919	0.533	0.531	-0.3 ± 1.3	0.569	0.575	$+0.6 \pm 1.0$
TreeBench	405	0.449	0.467	$+1.8 \pm 3.1$	0.484	0.483	-0.1 ± 2.7
Counting							
CountQA	1001	0.307	0.364	$\mathbf{+5.7 \pm 2.7}$	0.302	0.335	$\mathbf{+3.3 \pm 3.2}$
FSC-147	1190	0.077	0.103	$\mathbf{+2.6 \pm 1.3}$	0.067	0.088	$\mathbf{+2.1 \pm 1.4}$

Table 3: **Training recipe comparison**. We report the performance of grounded reasoning and text-only reasoning training recipes after SFT and RL training phases. Deltas that exclude 0 from the confidence interval are highlighted. We find that grounding provides a statistically significant benefit on counting, but not on spatial reasoning or perception.

recipe, we use an older model. For RL, we apply a format, correctness, and grounding reward. We do not apply the grounding reward for text reasoning checkpoints. We train using LlamaFactory (Zheng et al., 2024) and veRL (Sheng et al., 2024). Our RL training setup is the same as TreeVGR, but we use a slightly altered grounding reward from Jiang et al. (2025) (see Appendix B).

3.2.1 Results

Coordinate accuracy affects grounding usage, but not accuracy. We first study how the accuracy of the grounding annotations in training affects downstream performance and grounding usage. We randomize varying percentages of the coordinates in our SFT training set and examine the downstream SFT checkpoint performance on SpatialMQA, MultihopSpatial, TreeBench, and CountQA. We also perform the coordinate randomization test on each checkpoint on MultihopSpatial. We report our results in Figure 3. We find that grounding usage decreases significantly as a larger percentage of coordinates is randomized, with the delta going from -3.63 at 0% randomized to -0.14 at 100% randomized. This result suggests that noise in the training data may cause the model not to use its grounding. It also supports design decisions in Lu et al. (2026), which note the importance of having high-quality ground truth in the SFT stage. Furthermore, while grounding usage decreases, we find that performance generally remains flat across spatial reasoning and fine-grained perception benchmarks, but decreases monotonically for CountQA. The lack of correlation between grounding usage and performance suggests that existing perception and spatial reasoning benchmarks do not benefit from better grounded reasoning. On the other hand, counting benefits significantly from effective grounded reasoning.

Benchmark	Appended	Reflection	Δ
SMQA	0.529	0.527	-0.2 ± 1.2
MHS	0.544	0.539	-0.5 ± 1.2
V*	0.839	0.844	$+0.5 \pm 4.5$
MME-RW	0.533	0.531	-0.2 ± 1.0
TreeBench	0.451	0.467	$+1.6 \pm 2.0$
CountQA	0.320	0.364	$\mathbf{+4.4 \pm 2.8}$
FSC-147	0.073	0.103	$\mathbf{+3.0 \pm 1.6}$

Table 2: **Appended vs reflection performance**. The reflection reasoning style yields significantly better performance on counting.

Reasoning templates affect grounding usage. We also ablate various strategies commonly used to insert grounding data in reasoning traces. We consider two approaches: *Appended*, where grounding coordinates are appended when relevant objects are mentioned during reasoning, and *Reflection*, where the traces contain a reflection phrase "Let's look at [coordinate]" (see Appendix 6 for examples). We find that reflection results in significantly more grounding usage than appending grounding

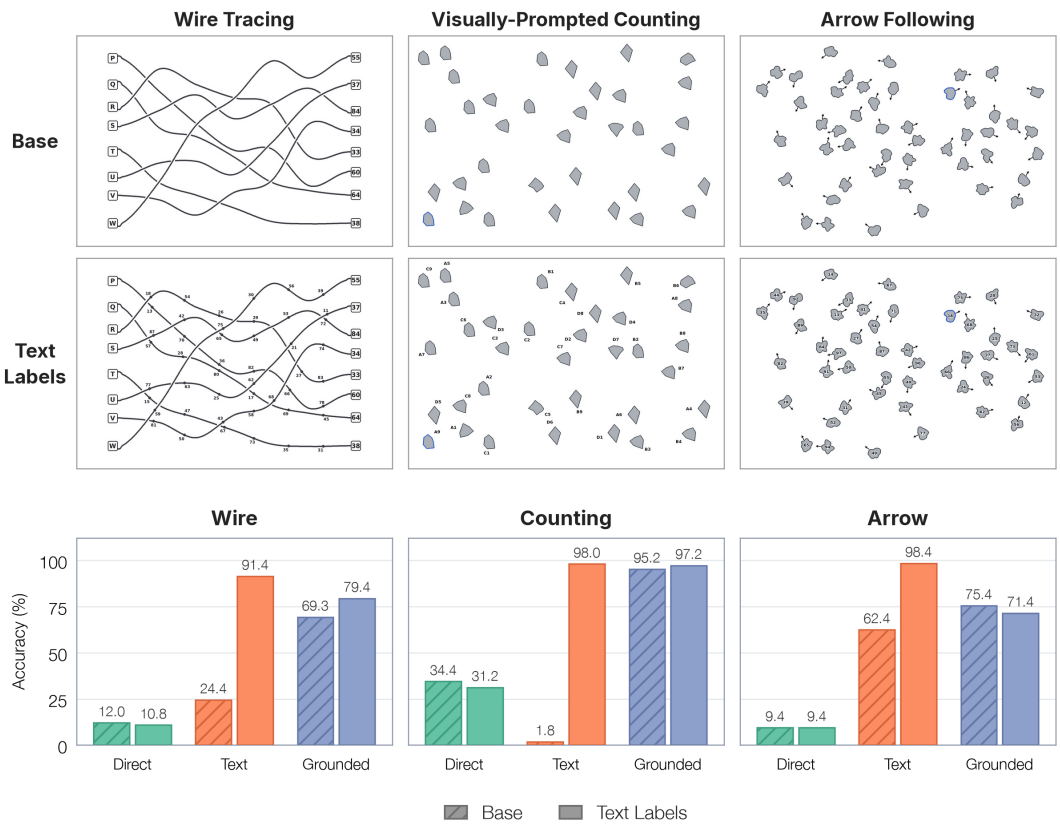


Figure 4: **Synthetic Probe Tasks**. We use three synthetic tasks to study where grounded reasoning improves over pure text reasoning. We find that grounding performs significantly better than text reasoning when no text-labels are available, but worse when given text labels. Zoom in for detail.

coordinates (Table 1) and also much better performance on counting (Table 2). These results show that "hinting" the model to use its grounding can be an effective strategy for improving performance. As such, we opt to use the reflection format in all other experiments.

Grounding recipes are not generally better than text-only recipes. We report the performance of grounded and ungrounded training recipes after SFT and after RL in Table 3. After SFT, the delta between grounded and ungrounded recipes is positive and significant for both counting benchmarks (+5.7 on CountQA and +2.6 on FSC-147), but is not significant on any spatial reasoning or fine-grained perception benchmarks. Similarly, after RL, only the counting benchmarks retain a significant positive delta, while we see regressions for MultihopSpatial, EmbSpatial, and VSR. These results suggest that even with high-quality training data and high grounding usage, grounded reasoning is generally helpful for counting tasks but not for spatial reasoning and perception. Based on these results, we seek to understand why grounded reasoning benefits some tasks, but not others.

4 Why Is Grounding Useful?

We hypothesize that grounding is primarily beneficial when the task requires an *undescribable reference*. To illustrate this claim, we develop several synthetic tasks to probe where grounded reasoning improves over pure text reasoning. These tasks are designed to represent situations that are particularly difficult to solve with textual references. In particular, we use arbitrary shapes and a uniform color to prevent textual identification by common shape or color names. We also non-uniformly scatter objects, which makes maintaining the position of objects difficult using plain text.

Probe tasks. We consider three tasks, shown in Figure 4. In *wire-tracing*, the model is given a left terminal and must determine which right terminal its wire reaches. In *visually prompted*

counting, one polygon is outlined, and the model must count how many shapes are identical to it. In *arrow following*, the model starts at an outlined blob and follows the arrows for several hops, before pointing to the final blob. Each task requires tracking undescribable visual references, such as the current position along a wire, the set of shapes already counted, or the current blob in the chain. Generation parameters and answer formats are given in Appendix D.

Experimental setup. For each task, we finetune Qwen3-VL-8B on 3 different reasoning variations: direct, text CoT, and grounded CoT. In the "direct" setting, no chain-of-thought is provided, and the model directly predicts the correct answer. In "text CoT," we use plain text chain-of-thought, where positions of objects are described using language. In "grounded CoT," we represent the positions of objects using [x, y] coordinates instead. To examine the benefit of grounding when positions in the image can be localized with language, we also re-render each task with text labels on objects or positions relevant for solving the task, as demonstrated in Figure 4. For these re-renders, we adjust the "text CoT" traces to refer to the precise text labels.

Grounded reasoning is helpful when text references are unavailable. We report our results in Figure 4. On the unlabeled scenes, we see that grounded reasoning improves significantly over text reasoning (69.3 vs 24.4 on wire-tracing) and direct fine-tuning (95.2 vs 34.4 on counting). However, once labels are added, the value of grounding over text reasoning drops significantly. We observe that text reasoning outperforms grounded reasoning on all three tasks when given text labels. As such, we conclude that grounding is primarily helpful when no textual references are available to solve a task.

5 BRICKBENCH

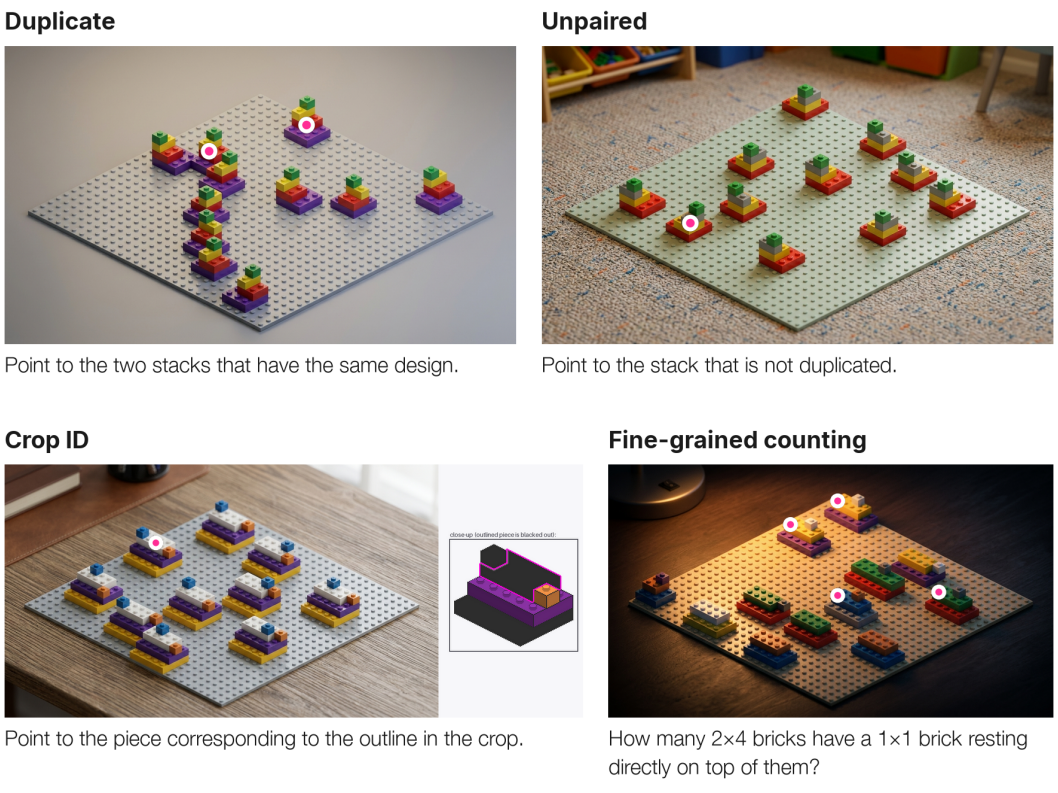


Figure 5: **BrickBench**. Samples from each task distribution of our BrickBench evaluation set. BrickBench tasks involve iterative comparison across objects difficult to describe in language, and require precise answers in the form of a point or integer.

Our study in Section 4 demonstrates that grounding capabilities are best illustrated on benchmarks that require undescribable references. However, because existing benchmarks have objects that are uniquely describable in text, they are easy to solve using textual references. To address this gap, we use our insights from Section 4 to create BrickBench, a benchmark that requires undescribable references to solve.

Benchmark construction. BrickBench contains four different task types, described in Figure 5. Each task type has 250 samples, for a total of 1000 questions in the benchmark. Each task is designed to require iterative comparison over every pile of bricks. Both the precise geometry and position of the piles are difficult to describe in language, which necessitates the effective use of undescribable visual references.

To create BrickBench, we combine procedural generation with generative style transfer. For each task type, we create a procedural generator of isometric 2D scenes. We then pass this synthetic render into Nano-Banana (Google, 2026) to restyle the image and use the original segmentation map as ground-truth for the tasks that require point output (Appendix 9). In order to use the original segmentation as ground-truth, the restyled scene must be aligned with the original synthetic render. However, we find that passing just the synthetic render to Nano-Banana results in misalignment of the segmentation masks, as shown in Appendix 10. As such, we prompt the model with an additional render featuring bounding boxes of the block stacks and fix the aspect ratio during generation.

Model	Crop ID	Fine-grained counting	Duplicate	Unpaired	Overall
GPT-5.6 Sol (OpenAI, 2026)	35.2	8.3	11.6	20.4	18.9
Gemini 3.8 Flash (Google DeepMind, 2026b)	20.4	15.5	8.8	17.6	15.6
Claude Opus 5 (Anthropic, 2026)	26.8	10.7	8.4	11.6	14.4
Qwen 3.8 Max (Qwen Team, 2026a)	6.4	2.0	13.2	12.4	8.5
Qwen3.5 9B (base) (Qwen Team, 2026b)	1.5	8.5	2.4	8.4	5.2
FT Direct	8.1	16.1	8.3	7.7	10.1
FT Text Reasoning	85.2	66.5	39.1	54.1	61.2
FT Grounded Reasoning	91.6	95.4	64.7	79.9	82.9

Table 4: **VLM performance on BrickBench**. Frontier models struggle on BrickBench, suggesting limited ability to maintain undescribable visual references. Models trained with in-domain grounding data perform better than pure-text reasoning, demonstrating the benefit of grounded reasoning.

BrickBench Evaluations. We evaluate a variety of models on BrickBench and report our results in Table 4. We find that existing models struggle on BrickBench, with strong multimodal models such as Gemini 3.8 Flash and GPT-5.6 Sol scoring just 15.6% and 18.9%, respectively. We observe that Qwen 3.8 Max often points to empty regions of the image, unlike other frontier models which always point to a valid pile, which may suggest weaker overall grounding capabilities. On fine-grained counting, GPT-5.6 Sol over-counts on 71.8% of questions (mean errors of $+2.2$), suggesting that the model fails to filter out piles that don't fulfill the condition. On the Duplicate task, Gemini 3.8 fails to find either twin on 44.4% of tasks. More broadly, these results suggest that the existing frontier models have limited capability to reason over undescribable references.

Finally, to validate that the benchmark benefits from grounded reasoning, we fine-tune Qwen-3.5 9B (Qwen Team, 2026b) on the pre-style-transfer images and ablate grounded and pure text reasoning strategies. We find that grounded reasoning significantly outperforms text-only reasoning (82.9 vs 61.2). Note that these fine-tuned results are intended only as a diagnostic, as BrickBench is designed to evaluate models that have not been trained directly for the task.

6 Conclusion

We have provided both a detailed analysis of *where and why textual grounding is most useful* and a benchmark to help guide future research on grounded reasoning. However, we note an interesting conundrum: grounded reasoning is most useful on tasks that are difficult to describe in language, but automatic methods of acquiring grounding rely on text referring expressions. So how can we acquire labels for grounded reasoning at scale? We hope future work will shed light on this question.

AI USE STATEMENT

We used LLMs to aid or polish writing; for instance, we checked for grammatical errors in the manuscript with LLMs. We used LLMs for research execution, and wrote portions of the experimental code using LLMs. We also used LLMs to generate the SFT dataset used in our experiments.

REPRODUCIBILITY STATEMENT

We use open, academic datasets as the base for our experiments. We plan to release training code, experiment code, benchmark, and will open-source all modified datasets (e.g. our SFT dataset) used in the paper. We also provide detailed examples of our generation pipeline and chain-of-thought traces in the Appendix.

REFERENCES

Anthropic. System card: Claude Opus 5. https://www-cdn.anthropic.com/c5fbac3f0b1280a933ebd26d3cb8bb9f5bdeaf48/Claude%20Opus%205%20System%20Card.pdf, July 2026.

Mohammad Asadi, Jack W. O'Sullivan, Fang Cao, Tahoura Nedaee, Kamyar Rajabalifardi, Fei-Fei Li, Ehsan Adeli, and Euan Ashley. Mirage: The illusion of visual understanding, 2026. URL https://arxiv.org/abs/2603.21687.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. Hallucination of multimodal large language models: A survey, 2025b. URL https://arxiv.org/abs/2404.18930.

Shmuel Berman and Jia Deng. Vlms have tunnel vision: Evaluating nonlocal visual reasoning in leading vlms. In *Advances in Neural Information Processing Systems 38*, NeurIPS 2025, pp. 87690–87711. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-2642. URL http://dx.doi.org/10.52202/085713-2642.

Edwin E. Catmull and Raphael Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, pp. 317–326, 1974. URL https://api.semanticscholar.org/CorpusID:118383557.

Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Y. charles, Yiping Bao, Yuantao Fan, Guopeng Li, Haiyang Shen, Xuanzhong Chen, Wendong Xu, Shuzheng Si, Zefan Cai, Wenhao Chai, Ziqi Huang, Fangfu Liu, Tianyu Liu, Baobao Chang, Ming Wu, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. Babyvision: Visual reasoning beyond language, 2026. URL https://arxiv.org/abs/2601.06521.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models?, 2024. URL https://arxiv.org/abs/2403.20330.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin

Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. URL https://arxiv.org/abs/2409.17146.

Jeremy Dohmann and PerceptronAI. Teaching vlms to think visually. *PerceptronAI Research Blog*, 2026. https://perceptron.inc/blog/teaching-vlms-to-think-visually.

Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models, 2024. URL https://arxiv.org/abs/2406.05756.

Google. Build with Nano Banana 2, our best image generation and editing model. https://blog.google/innovation-and-ai/technology/developers-tools/build-with-nano-banana-2/, 2026.

Google DeepMind. Gemini 3.5 Flash model card. https://deepmind.google/models/model-cards/gemini-3-5-flash/, 2026a.

Google DeepMind. Gemini 3.8 Flash model card. https://deepmind.google/models/model-cards/gemini-3-8-flash/, 2026b.

Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models, 2026. URL https://arxiv.org/abs/2506.03135.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction, 2025. URL https://arxiv.org/abs/2510.12798.

Sai Srinivas Kancheti, Aditya Sanjiv Kanade, Vineeth N. Balasubramanian, and Tanuja Ganu. Chain-of-thought degrades visual spatial reasoning capabilities of multimodal llms, 2026. URL https://arxiv.org/abs/2604.16060.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Youngwan Lee, Soojin Jang, Yoorhim Cho, Seunghwan Lee, Yong-Ju Lee, and Sung Ju Hwang. Multihopspatial: Multi-hop compositional spatial reasoning benchmark for vision-language model, 2026. URL https://arxiv.org/abs/2603.18892.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning, 2023. URL https://arxiv.org/abs/2205.00363.

Jingping Liu, Ziyan Liu, Zhedong Cen, Yan Zhou, Yinan Zou, Weiyan Zhang, Haiyun Jiang, and Tong Ruan. Can multimodal large language models understand spatial relations? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 620–632. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.31. URL http://dx.doi.org/10.18653/v1/2025.acl-long.31.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Ruijie Lu, Yiyang Ma, Xiaokang Chen, Lingxiao Luo, Zhiyu Wu, Zizheng Pan, Xingchao Liu, Yutong Lin, Hao Li, Wen Liu, Zhewen Hao, Xi Gao, Shaoheng Nie, Yixuan Wei, Zhenda Xie, Ting Chen, and Gang Zeng. Thinking with visual primitives. 2026.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations, 2024. URL https://arxiv.org/abs/2411.00640.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026. Covers GPT-5.6 Sol, Terra, and Luna.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026a. URL https://qwen.ai/blog?id=qwen3.8.

Qwen Team. Qwen3.5-omni technical report, 2026b. URL https://arxiv.org/abs/2604.15804.

Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything, 2021. URL https://arxiv.org/abs/2104.08391.

Gabriel Sarch, Snigdha Saha, Naitik Khandelwal, Ayush Jain, Michael J. Tarr, Aviral Kumar, and Katerina Fragkiadaki. Grounded reinforcement learning for visual reasoning, 2026. URL https://arxiv.org/abs/2505.23678.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Jayant Sravan Tamarapalli, Rynaa Grover, Nilay Pande, and Sahiti Yerramilli. Countqa: How well do mllms count in the wild?, 2025. URL https://arxiv.org/abs/2508.06585.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased, 2026. URL https://arxiv.org/abs/2505.23941.

Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Jiani Zheng, Sule Bai, Zijian Kang, Jiashi Feng, Zhuochen Wang, and Zhaoxiang Zhang. Traceable evidence enhanced visual grounded reasoning: Evaluation and methodology, 2026a. URL https://arxiv.org/abs/2507.07999.

Jiacong Wang, Zijian Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, and Jun Xiao. Vgr: Visual grounded reasoning, 2026b. URL https://arxiv.org/abs/2506.11991.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms, 2023. URL https://arxiv.org/abs/2312.14135.

Chang-Bin Zhang, Yujie Zhong, Qiang Zhang, and Kai Han. ivgr: Internalizing visually grounded reasoning for mllms with reinforcement learning, 2026a. URL https://arxiv.org/abs/2605.31096.

Junkai Zhang, Yihe Deng, Kai-Wei Chang, and Wei Wang. Thinking with visual grounding, 2026b. URL https://arxiv.org/abs/2606.16122.

Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans?, 2025. URL https://arxiv.org/abs/2408.13257.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning, 2026. URL https://arxiv.org/abs/2505.14362.

Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge, 2021. URL https://arxiv.org/abs/2001.06303.

A Evaluations

Confidence Interval Calculations. Following Miller (2024), we compute the question-level uncertainty (denoted $\mathrm{SE_q}$) using a paired standard error over per-question differences, averaging over checkpoints and clustering by image. We compute run-level uncertainty (denoted s_{run}) by taking the standard deviation of the delta over the three seeds. The final confidence interval is computed as $\bar{\Delta} \pm 1.96\sqrt{\mathrm{SE_q^2} + s_{\text{run}}^2/3}$.

B Training Details

Table 5 lists the hyperparameters for the SFT and reinforcement learning stages. All runs are initialized from Qwen3-VL-8B Instruct unless otherwise noted. Following Wang et al. (2026a), we use the AdamW (Loshchilov & Hutter, 2019) optimizer for all stages and GRPO (Shao et al., 2024) as the RL algorithm.

	SFT	RL
Epochs	1	5
Batch	64	512×8
LR	10^{-5}	10^{-5}
Warmup	cosine, 3%	constant
Weight decay	0	0.01
KL / clip	–	0.01 / 0.2

Table 5: **Training hyperparameters**.

Our grounding reward is defined as follows:

$$r_j = \max_i s(p_i, b_j), \quad \mathrm{Rec} = \frac{1}{M}\sum_{j=1}^{M} r_j, \quad \mathrm{Prec} = \frac{1}{N}\sum_{j=1}^{M} r_j, \quad R_{\text{gnd}} = \frac{2\,\mathrm{Prec}\cdot\mathrm{Rec}}{\mathrm{Prec}+\mathrm{Rec}}, \tag{1}$$

where r_j is the credit for the j-th annotated box, M is the number of annotated boxes b_j, N is the number of distinct coordinates p_i emitted in y. Because the ground truth is only annotated with boxes, instead of segmentation, we use a binary point-in-box credit during RL on pointing experiments. Otherwise, we use an IoU credit.

C Grounded SFT Data

Quality of existing data. Figure 7 shows TreeVGR SFT samples with boxes that are oversized or cover the wrong objects.

Points versus boxes. Table 6 compares SFT checkpoints trained on point and box grounding with otherwise identical traces. Boxes performed significantly worse on SMQA, MHS, and MME-RW, so we use points in our SFT data. However, boxes are better than points on counting.

Trace format. Figure 6 compares the Appended and Reflection trace formats.

D Probe Tasks

Training details. For wire-tracing and visually prompted counting, we finetune on training sets of 5k QA samples. Empirically, we found arrow-following to be a more difficult task and instead generate 50k QA samples for training. In each experiment, we train for 1 epoch.

Task generation. We describe the generation procedure and answer format for each probe task.

Wire-tracing. Each image has eight labeled terminals on both the left and right sides. Wires between terminals are generated with Catmull-Rom splines (Catmull & Rom, 1974). Given the letter of a left terminal, the model must identify the right terminal that its wire reaches.

Bench	n	Point	Box	Box − Point
Spatial Reasoning				
SMQA	1076	0.527	0.509	$\mathbf{-1.9 \pm 1.5}$
MHS	1500	0.539	0.525	$\mathbf{-1.4 \pm 1.2}$
EmbSpatial	3640	0.733	0.728	-0.4 ± 1.2
VSR	1222	0.801	0.797	-0.4 ± 2.1
OmniSpatial	1533	0.470	0.467	-0.3 ± 1.7
ERQA	400	0.396	0.395	-0.1 ± 2.1
Fine-grained Perception				
V*	191	0.844	0.848	$+0.3 \pm 2.3$
MME-RW	1919	0.531	0.521	$\mathbf{-0.9 \pm 0.9}$
TreeBench	405	0.467	0.449	-1.8 ± 2.4
Counting				
CountQA	1001	0.364	0.391	$+2.6 \pm 2.8$
FSC-147	1190	0.103	0.115	$+1.2 \pm 2.0$

Table 6: **Point versus box grounding after SFT.** We report the box minus point accuracy delta with 95% confidence intervals computed as in Appendix A. Bold deltas exclude 0.



Question Where is the knife located relative to the cake?
(a) on/above (b) below (c) in front of (d) behind (e) left of (f) right of

Appended

Based on the image, I can see a large, round cake **[516, 472]** decorated with chocolate sprinkles sitting on a white cutting board. To the left side of the cake on the same cutting board, there is a silver knife **[81, 480]** lying flat. Since the knife is positioned to the left side of the cake, the correct option is (e).
\boxed{e}

Reflection

Based on the image, I find the cake at **[516, 472]**. **Looking at [516, 472]**, I see a large, round cake decorated with chocolate sprinkles sitting on a white cutting board. To the left side of the cake on the same cutting board, I find the knife at **[81, 480]**. **Looking at [81, 480]**, there is a silver knife lying flat. Since the knife at **[81, 480]** is positioned to the left side of the cake at **[516, 472]**, the correct option is (e).
\boxed{e}

Figure 6: **SFT format ablation**. We compare the appended and reflection SFT formats. We find that the reflection format generally leads to better grounding usage, and use this format in our SFT data.

Visually prompted counting. Each image contains $K \in \{29, 35, 41\}$ gray polygons. One polygon is outlined in blue and duplicated $M \in \{9, 10, 11, 12\}$ times. The model must output the number of shapes identical to the outlined one as an integer.

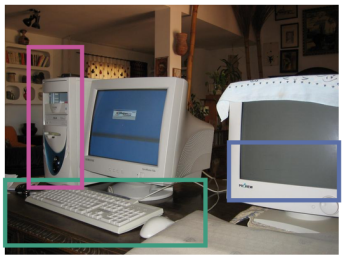
Arrow following. Each scene contains 48 blobs, each with one arrow pointing to another blob. The start blob is outlined in blue. The model must follow the arrows for $H \in \{2, 4, 6\}$ hops and point to the final blob, answering with an [x, y] coordinate normalized to [0, 1000] following standard practice (Deitke et al., 2024).

Chain-of-thought construction. Figure 8 shows paired text and grounded reasoning traces for arrow following.

E BrickBench

We consider four task types in BrickBench: Duplicate, Crop ID, Unpaired, and Fine-grained Counting.

(a) Incorrect box



Question Are there both keyboards and computers in this scene?

Trace There are two keyboards. One is in front of the left-side monitor, which is a standard white QWERTY keyboard ([0,420,480,576]). **The other is in front of the right-side monitor, identifiable by its shape despite being partially covered** ([536,330,800,468]).
For computers, the left setup has a tower CPU next to the monitor ([56,102,184,438]), indicating a computer system.

Answer Yes

(b) Incorrect box

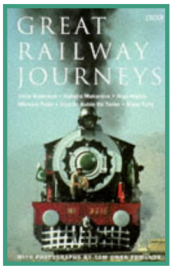


Question What vehicle is right of the fence?

Trace The question asks for the vehicle right of the [140,69,470,258] **fence**. The black metal fence is on the left-side of the image, separating the grassy area from the tracks. The [140,69,470,258] is on the tracks, and since the fence is on the left, the [140,69,470,258] is to the right of the fence.

Answer Train

(c) Box too large



Question Is this book related to Travel?

Trace The cover of the book titled "GREAT RAILWAY JOURNEYS" is shown. … Considering the title [0,0,299,475], the steam-train imagery [0,0,299,475], the subtitle [0,0,299,475], and the author's reputation, it is clear that the book is related to travel.

Answer Yes

Figure 7: **Incorrect TreeVGR samples**. Selected samples from the TreeVGR dataset illustrating errors in data such as incorrect box labeling or uninformative boxes.

Duplicate. Each scene has 10 to 12 stacks in the same colors, exactly two of which share a design, and the model must point to both. Every other stack differs from the shared design in the orientation or position of one or two bricks, so no single attribute separates the pair from the rest.

Crop ID. Each image shows a scene of 10 structures next to an enlarged inset of one of them. In the inset, two or three of the structure's five pieces are masked to silhouettes and one of the masked pieces is outlined. All structures use the same color for a given piece and differ only in the offset of a side marker and the position of the cap, so the model has to find the inset's structure in the scene by geometry and then point to the outlined piece.

Unpaired. Each scene has 10 to 13 stacks in the same colors. Every stack has an identical copy somewhere in the scene except one, and the groups of copies differ from each other only in geometry. The model must point to the stack that has no copy.

Fine-grained Counting. The model is asked how many 2×4 bricks satisfy a geometric predicate: having a 1×1 brick on top, touching a 1×1 at a short end, or resting crosswise on the brick below. The scenes also contain bricks of other shapes that satisfy the predicate, so the model must check each brick's shape as well as the predicate. Ground truth counts range from 2 to 7.

Figure 9 shows the pipeline used to generate the final benchmark images. A procedural generator renders an isometric scene and its segmentation map, and Nano-Banana restyles the render. Figure 10 shows the drift that occurs without a fixed aspect ratio and bounding-box conditioning.

Plain Scene **Labeled Scene**



Question This image shows 48 gray blobs. Each blob has one short arrow pointing at exactly one other blob. Start at the blue-outlined blob and follow the arrows hop by hop. Point to the blob you land on after 4 hops.

Text CoT	Grounded CoT	Text CoT with labels
Let me think about what I need to locate — I need to locate the blue-outlined starting blob, then follow the arrows one hop at a time, keeping track of where I am after each hop.		
The start is the blob at the middle left of the image (the sixth from the left).	The start is the blob at **[311, 632]**.	The start is **blob 37**.
Let's analyze these descriptions step-by-step.	Let's analyze these coordinates step-by-step.	Let's analyze these labels step-by-step.
Hop one: the arrow at the middle left of the image (the sixth from the left) points down and to the left, landing on the blob at the bottom left of the image (the first from the left). I am now at the bottom left of the image (the first from the left).	Hop one: the arrow at **[311, 632]** points down and to the left, landing on the blob at **[115, 776]**. I am now at **[115, 776]**.	Hop one: the arrow of **blob 37** points down and to the left, landing on **blob 16**. I am now at **blob 16**.
Hop two: the arrow at the bottom left of the image (the first from the left) points upward, landing on the blob at the middle left of the image (the first from the left). I am now at the middle left of the image (the first from the left).	Hop two: the arrow at **[115, 776]** points upward, landing on the blob at **[61, 493]**. I am now at **[61, 493]**.	Hop two: the arrow of **blob 16** points upward, landing on **blob 88**. I am now at **blob 88**.
Hop three: the arrow at the middle left of the image (the first from the left) points up and to the right, landing on the blob at the top left of the image (the third from the top). I am now at the top left of the image (the third from the top).	Hop three: the arrow at **[61, 493]** points up and to the right, landing on the blob at **[184, 266]**. I am now at **[184, 266]**.	Hop three: the arrow of **blob 88** points up and to the right, landing on **blob 45**. I am now at **blob 45**.
Hop four: the arrow at the top left of the image (the third from the top) points upward, landing on the blob at the top left of the image (the first from the top). I am now at the top left of the image (the first from the top).	Hop four: the arrow at **[184, 266]** points upward, landing on the blob at **[214, 130]**. I am now at **[214, 130]**.	Hop four: the arrow of **blob 45** points upward, landing on **blob 27**. I am now at **blob 27**.
After four hops I land on the blob at the top left of the image (the first from the top).	After four hops I land on the blob at **[214, 130]**.	After four hops I land on **blob 27**.
So the answer is the blob at the top left of the image (the first from the top).	So the answer is the point **[214, 130]**.	So the answer is **blob 27**.
\boxed{**[214, 130]**}		

Figure 8: **Probe reasoning traces**. We provide examples of our reasoning data used in our "Arrow Following" task to illustrate the difference between text traces and grounded traces.

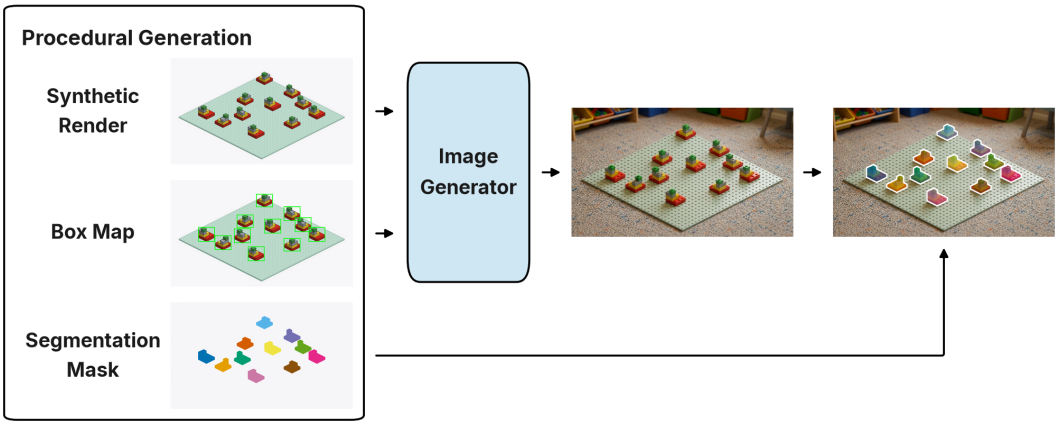


Figure 9: **Benchmark construction**. Our benchmark combines procedural generation with conditional image generation to produce realistic images aligned with the original synthetic ground truth.

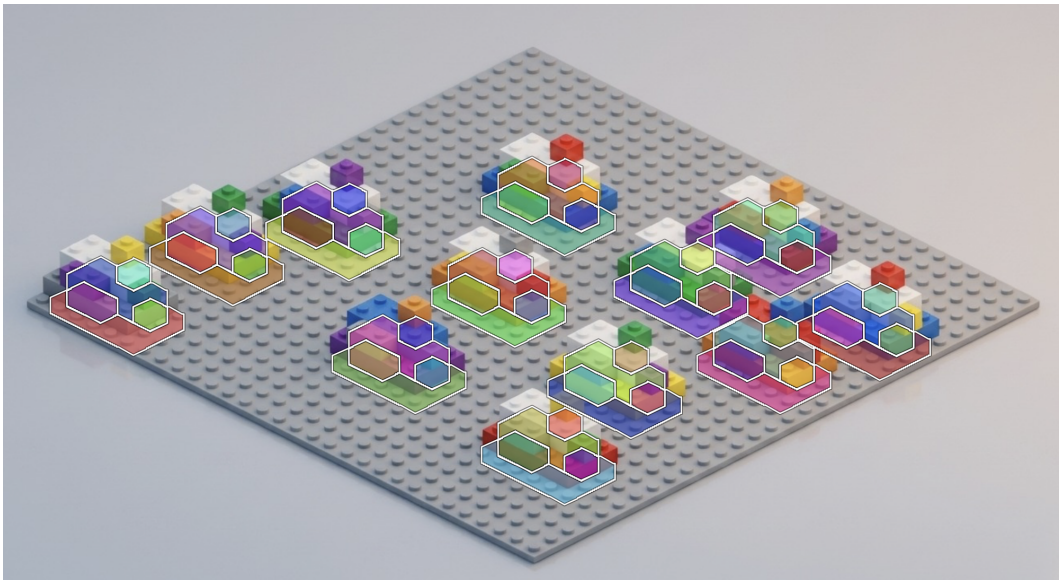


Figure 10: **Restyling drift**. Without locking the aspect ratio or prompting Nano-Banana with bounding boxes, we find that the restyled image drifts from the original, making the segmentation map ground truth (visualized as an overlay) incorrect.