Thinking with Visual Grounding

Junkai Zhang **Yihe Deng** **Kai-Wei Chang** **Wei Wang**
University of California, Los Angeles
{zhang, yihedeng, kwchang, weiwang} @cs.ucla.edu

Abstract

Visual thinking should not only sound right; it should show its evidence. While recent vision-language models (VLMs) can produce natural-language reasoning traces, these traces often leave the supporting image regions implicit, making them hard to verify and difficult to supervise. We introduce *visually grounded thinking*, a reasoning process in which models interleave natural-language thoughts with explicit point or box groundings of the visual evidence used at each step. This lets the model express intermediate reasoning in language while grounding key objects in the image regions they refer to. To train this behavior, we construct a scalable synthesis pipeline that distills correct visual reasoning traces, extracts the visual objects required by the traces, grounds them with a SAM3-based agent, and derives aligned point and box supervision from the resulting masks. We further propose grounding-aware reinforcement learning, which combines answer correctness rewards with dense grounding rewards that score whether generated object references match the correct image evidence. Across two counting benchmarks and four spatial reasoning benchmarks, adding visually grounded thinking to Gemma3-4B-IT consistently improves performance over the original model and the non-grounded thinking baseline. On spatial reasoning, the visually grounded thinking 4B models match, and in some cases surpass, Gemma3-27B-IT from the same model family. Our analysis shows that point grounding is well suited to counting, while box grounding benefits most from explicit grounding rewards on spatial tasks. Overall, our results show that VLMs think better when their intermediate thoughts are tied to the image regions that make them true.[1]

1 Introduction

Language models have made strong progress on complex problem solving by producing explicit natural-language reasoning traces. In particular, R1-style reinforcement learning has shown that models can improve their ability to solve math, coding, and general reasoning problems through long textual thinking (Guo et al., 2025). This success has motivated analogous reasoning methods for vision-language models (VLMs): given an image and a question, the model can think in text before giving the final answer. Such a strategy has been shown to be effective for visual question answering (Deng et al., 2025; Hu et al., 2026).

However, visual thinking differs from purely textual thinking because the evidence needed to solve a visual question is located in the image (Zhu et al., 2016) and cannot be fully expressed in words. When humans answer visual questions, we often link our thoughts to concrete image regions, such as the person on the left, the cup near the table edge, or the object being counted (Das et al., 2016). These visual references guide where attention should be directed and what task-specific information should be extracted (Hayhoe and Ballard, 2005). In contrast, a pure natural-language reasoning trace may state that "the red car is near the entrance" or that "there are three people holding umbrellas," but it does not identify which image regions support these claims. This makes the thinking hard to verify and supervise: a final answer may be correct even without the image, while the reasoning trace can still appear coherent and image-based (Asadi et al., 2026). Thus, visual thinking requires not only step-by-step reasoning, but also explicit links between important reasoning steps and the correct visual evidence.

We propose *visually grounded thinking* to address this issue, as shown in Figure 1. In this format, the model interleaves natural-language reasoning with explicit visual grounding. Whenever a reasoning step refers to an important visual object, the model outputs a coordinate-based grounding tag, using either a bounding box or a point, to identify

[1]We release the data and code.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 30224–30237
October 24-29, 2026 ©2026 Association for Computational Linguistics

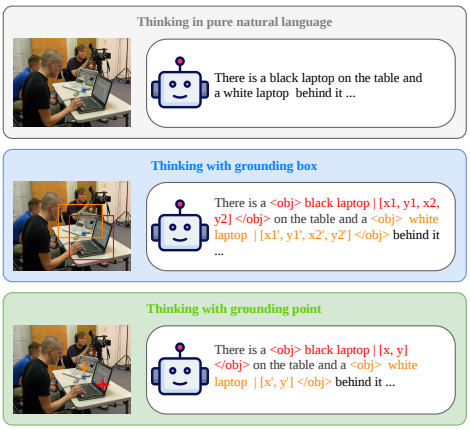

Figure 1: Thinking in pure natural language vs. visually grounded thinking in box mode and point mode.

the referenced object in the image. Natural language and spatial coordinates are combined in the thinking process: language describes the thoughts, while the coordinates specify the visual evidence that supports each step.

Visually grounded thinking requires training data that supports both SFT and RL. Coordinate-annotated reasoning traces can teach the model to interleave language and grounding during SFT, but RL needs supervision at the level of each visual reference, since a rollout may rename objects, reorder reasoning steps, skip supervised entities, or ground additional useful evidence. We therefore build an automatic data synthesis pipeline around a SAM3-based grounding agent (Carion et al., 2026). Starting from visual questions, the pipeline obtains correct reasoning traces, extracts the visual objects used in the reasoning, represents each *object* with a *name* and a disambiguating scene *context*, and grounds it with a run-length encoding (RLE) mask. These masks are used to construct both point and box-mode grounded reasoning traces, while the corresponding grounded objects are kept as structured supervision for grounding-aware RL.

We train models to perform visually grounded thinking with the synthetic data. The models are first cold-started with synthesized visually grounded reasoning traces, where important objects are grounded with either points or boxes. We then apply RL with the reward explicitly supervising the grounding quality. This stage jointly encourages answer correctness and precise grounding of the visual objects that support the model's intermediate reasoning. An example output from a visually grounded thinking model is shown in Figure 2.

Our controlled experiments show that visually grounded thinking substantially improves counting and spatial reasoning. On spatial reasoning benchmarks, our 4B visually grounded thinking models reach performance comparable to, and in some cases higher than, the 27B model from the same family. The grounding reward brings clear gains for box-mode grounded thinking, especially on spatial tasks where object extent and relative geometry are important. We also find that the two grounding interfaces have different strengths: point grounding performs better on counting, where instance-level localization is often sufficient, while point and box grounding are broadly comparable on spatial reasoning.

Our contributions are as follows:

1. We build a scalable data pipeline to synthesize visually grounded thinking data for both SFT and RL, centered on a SAM3-based agentic grounding system that extracts high-fidelity object masks as visual supervision.

2. We design a grounding reward that directly supervises whether the model grounds its intermediate visual references in the correct image evidence, supporting both box-mode and point-mode grounded thinking.

3. Through controlled experiments, we show that visually grounded thinking substantially improves counting and spatial reasoning. The grounding reward brings clear gains for box grounding, especially on spatial tasks. In addition, point grounding is particularly effective for counting and remains competitive with box grounding on spatial reasoning.

2 Related Work

Early work on visually grounded thinking mainly uses grounding to locate the image regions needed for answering a question. Visual CoT (Shao et al., 2024a) introduces intermediate bounding boxes that highlight key regions, while UV-CoT (Zhao et al., 2025) reduces the need for human box annotations by learning from preferences over model-generated regions. Later work more tightly couples grounding with the reasoning trace. GCoT (Wu et al., 2025), Xia et al. (2025), and Argus (Man et al., 2025) generate grounding coordinates as step-level visual evidence, aiming to make the reasoning more faithful to the image and easier to check. More recent work further treats grounding as an

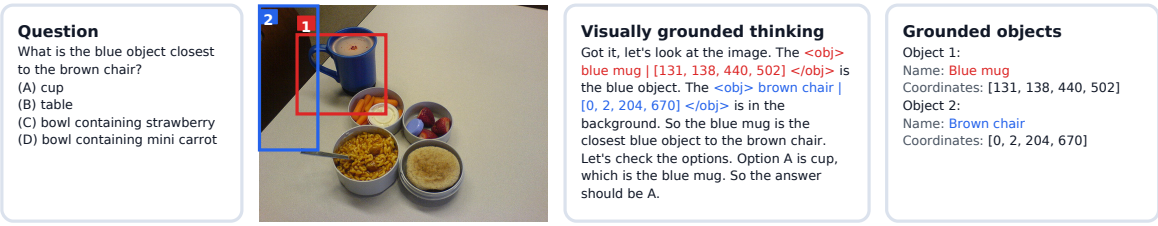

Figure 2: A real example of a visually grounded thinking model's output in the evaluation benchmark.

active behavior: GRIT (Fan et al., 2025) and ViGoRL (Sarch et al., 2025) train models to interleave natural language with visual coordinates through RL, and VGR (Wang et al., 2025) uses predicted regions for visual replay during inference. Our work follows this shift from region-of-interest selection to visually grounded thinking, and extends it with an explicit grounding reward that directly scores the visual grounding produced during thinking.

3 Data Synthesis Pipeline

Overview. We synthesize visually grounded thinking data from open-source datasets for counting and spatial reasoning: TallyQA (Acharya et al., 2018), Pixmo-Count (Deitke et al., 2024), VSR (Liu et al., 2023), MultihopSpatial (Lee et al., 2026), and SpatialMQA (Liu et al., 2025), with all test sets held out. Our goal is to identify the visual objects needed for correct thinking, obtain their image coordinates, and synthesize reasoning traces with explicit grounding annotations. The complete pipeline is shown in Figure 3.

Distilling visual thinking from VLMs. For each image-question pair, we prompt Qwen3-VL-Plus (Bai et al., 2025) to generate a thinking-mode response. We parse the final answer and keep examples whose predictions match the ground-truth answers. For examples not answered correctly in the first pass, we run a second pass with Qwen3.5-Plus (QwenTeam, 2026a) and keep examples that are answered correctly in either pass.

Extracting groundable objects. Given a correct reasoning trace, we use an LLM to identify the visual objects needed for the thinking process. These objects include answer objects, visible multiple-choice alternatives, spatial anchors, counted instances, and endpoints of spatial relations. Each object is represented by a *name* (e.g., "red car") and a disambiguating *context* (e.g., "in the back row"). The context separates visually or semantically similar instances, so two occurrences of "red car" can be distinguished by scene cues such as "near the entrance" or "in the back row".

Agentic visual grounding. The main challenge in data synthesis is to obtain accurate grounding for each extracted visual object. Direct prompting of VLMs does not produce RLE masks, and their predicted boxes are often noisy. SAM3 (Carion et al., 2026) can produce high-quality instance masks from simple noun prompts, but it is not well suited to complex context-dependent queries. We therefore use a SAM3-centered grounding agent powered by a VLM, adapted from the SAM 3 Agent in Carion et al. (2026).

The agent uses four tool actions. First, it calls SAM3 with a short noun phrase and receives candidate instance masks with confidence scores. Second, it verifies rendered masks using the raw image, a full-image mask overlay, and a zoomed-in crop, returning an accept/reject decision for each candidate. Third, it selects the final mask IDs from the current candidate set. Finally, it can report that no valid detection is found. Importantly, the agent cannot directly write coordinates; all geometric supervision must be derived from selected SAM3 masks.

For each object, the agent uses these tools in an iterative grounding loop. It receives the raw image and the object, then identifies the intended target and converts the name-context description into a SAM3-compatible noun phrase. If the initial prompt misses the target or returns confusing candidates, the agent revises the noun phrase and tries again. When candidates are small, overlapping, or ambiguous, it invokes the verifier and re-renders the accepted masks as the updated candidate set. Once it has sufficient evidence, it selects the final mask IDs; if no valid target can be found, it reports no detection.

The selected masks are stored as RLE masks and used as the shared supervision signal for both grounding modes. In box mode, each RLE mask is converted to a normalized bounding box in

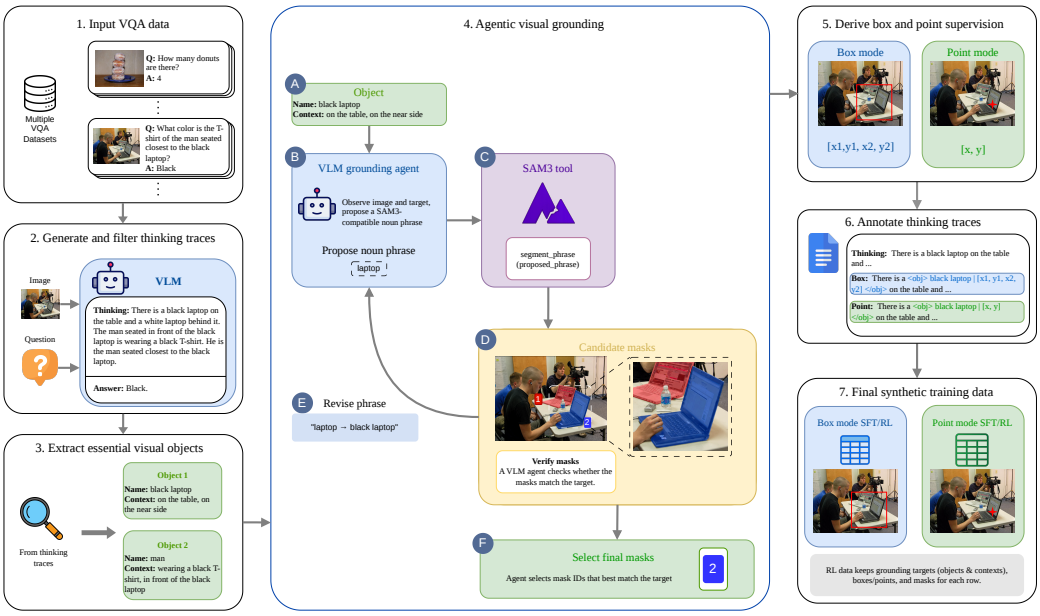

Figure 3: Data synthesis pipeline. We distill reasoning traces, extract groundable visual evidence, ground those objects with an iterative SAM3-based agent, and write aligned box-mode and point-mode SFT and RL training data.

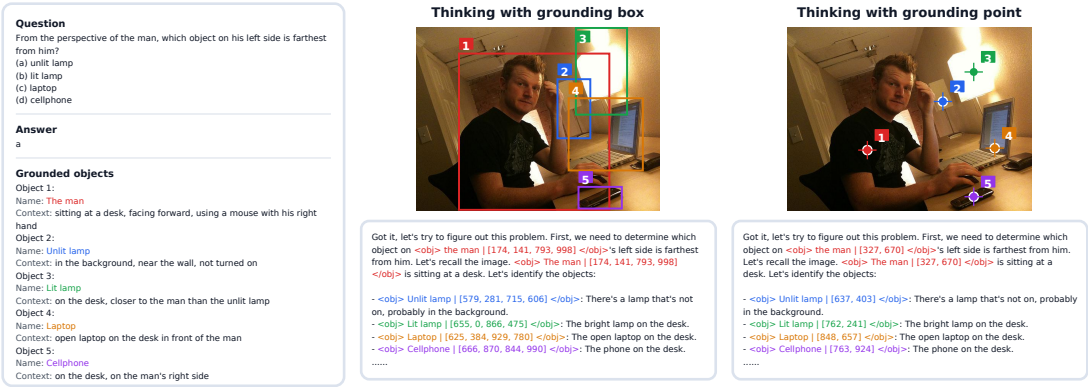

Figure 4: An example of synthesized visually grounded thinking data for box and point mode. The two variants share the same original reasoning trace and SAM3 masks, but expose either boxes or points inside `<obj> ... </obj>` tags.

the $[0, 1000]$ image coordinate system. For point mode, each RLE mask is converted to a single on-object point by choosing the interior point farthest from the mask boundary, which ensures that the point lies inside the object even for non-convex masks. We retry failed detections and near-duplicate groundings by rerunning the same agent loop with stronger VLMs. Objects that remain unresolved are removed from the grounded object list so that later stages do not use unreliable grounding signals.

Writing box and point supervision. In the final annotation stage, we insert placeholder object tags into the validated reasoning text using only the extracted object phrases and their contexts, without exposing coordinates to the annotation model. We then fill in the coordinates from the SAM3 outputs. This design prevents the annotation model from hallucinating spatial values. A single placeholder pass therefore produces two aligned SFT variants: `<obj> name phrase | [x1,y1,x2,y2] </obj>` for box supervision and `<obj> name phrase | [x,y] </obj>` for point supervision, as illustrated in Figure 4. We filter out rows whose tag-stripped annotated thinking differs substantially from the original thinking, as well as rows with malformed tags or highly repetitive reasoning.

Grounding evaluation data. We also construct a grounding evaluation split to measure the quality of model-generated visual groundings. We apply the

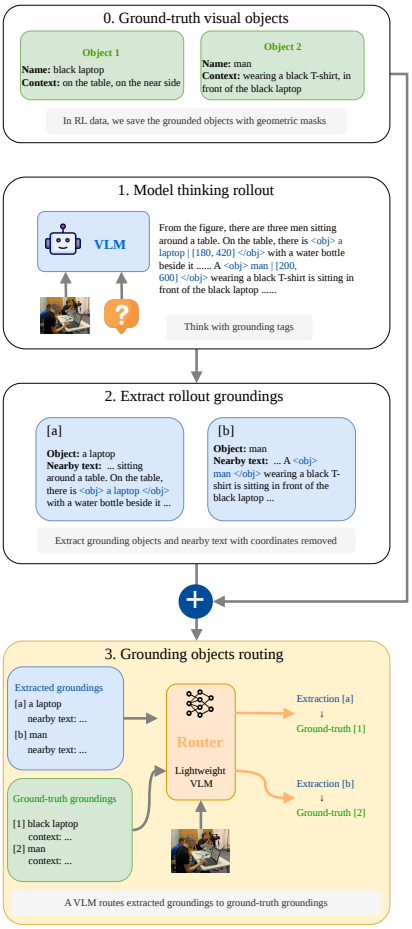

Figure 5: Grounding object router. Model-generated grounding objects are matched to saved ground-truth grounding objects before grounding quality is scored.

same synthesis pipeline to the selected VLMEval datasets used in evaluation, sampling at most 500 examples from each dataset. Each example is converted into the same reward-compatible format used for RL. The resulting split contains 2,399 rows across TallyBench, CountQA, VSR-zeroshot, EmbSpatial, SpatialMQA, and MultihopSpatial, and is used for grounding-quality evaluation.

Dataset Statistics. For training, our synthetic data pipeline produces 19,909 reasoning traces for SFT, containing 107,613 grounding annotations over 72,381 distinct grounded objects.

4 Reinforcement Learning with Grounding Reward

Grounding tag parsing. The grounding reward evaluates the grounding object tags generated in the model rollout. In visually grounded thinking, a valid tag must have the form `<obj> name phrase | coordinates </obj>`. The coordinate format is mode-specific: box mode expects $[x_1, y_1, x_2, y_2]$, while point mode expects $[x, y]$. Coordinates must fall within the $[0, 1000]$ image coordinate system; boxes must additionally satisfy $x_1 < x_2$ and $y_1 < y_2$. A single tag may contain multiple coordinates separated by semicolons, as one object can refer to multiple instances (e.g. "birds in the sky" corresponds to several birds in the sky).

Grounding objects routing. The grounding reward is computed between model-generated grounding objects and the ground-truth grounding objects saved in the data. Each grounding object in the data stores a name phrase, a disambiguating context, and geometric supervision. The model, however, may name the same object with different wording, and the same name phrase can refer to multiple distinct objects in the image and therefore needs to be disambiguated by context. We therefore use a VLM grounding object router before scoring grounding quality, as shown in Figure 5.

The grounding object router is a lightweight VLM, Qwen3.5-4B (QwenTeam, 2026a), chosen to keep RL training efficient. We first parse all model-generated `<obj> ... </obj>` tags from the rollout and extract their object names together with the nearby text as disambiguating context. For each ground-truth object, the router receives the image, the object consisting of a *name* and a disambiguating *context*, and the full list of model-generated grounding objects. It is then instructed to return the subset of generated grounding objects that correspond to the ground-truth object. This routing step matches model-generated grounding objects to the saved ground-truth grounding objects before scoring grounding quality. If several generated grounding objects are matched to the same ground-truth grounding object, only the earliest one is kept for grounding quality scoring.

Box grounding quality. In box mode, each saved object i is associated with a set of ground-truth boxes G_i. After grounding object routing, let P_i denote the set of boxes generated for the generated grounding object matched to target i. We score the grounding quality by comparing the image region covered by the generated boxes with the region covered by the ground-truth boxes. Specifically, we treat each set of boxes as a union of regions

and compute their intersection-over-union (IoU). For a matched target, define I_i as the area covered by both P_i and G_i, and U_i as the area covered by either P_i or G_i. The per-target box score is

$$\mathrm{IoU}_i = \frac{I_i}{U_i}.$$

If no model-generated grounding object is matched to ground-truth object i, we set $\mathrm{IoU}_i = 0$. The final box grounding quality is the mean score over all T ground-truth objects. This averaging gives each ground-truth object equal weight, regardless of how many boxes it contains. A multi-box grounding receives a perfect score only when the union of its generated boxes exactly matches the union of the ground-truth boxes.

Point grounding quality. Point mode evaluates whether generated grounding points lie inside the target RLE masks. Let M_i be the set of ground-truth masks for object i saved in the data, and let P_i be the set of points from the rollout grounding object matched to that ground-truth object. We form a one-to-one assignment between generated points and ground-truth masks, where a point can be assigned to a mask only if it lies inside that mask. This constraint prevents duplicate points from receiving repeated credit for the same object instance.

For each object, let TP_i be the number of masks matched by this assignment. The object-level false positives and false negatives are

$$\mathrm{FP}_i = |P_i| - \mathrm{TP}_i, \qquad \mathrm{FN}_i = |M_i| - \mathrm{TP}_i.$$

We use the per-object F1 score to measure point grounding quality:

$$F1_i = \frac{2\mathrm{TP}_i}{2\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i}.$$

If no rollout grounding object is matched to ground-truth object i, we set $F1_i = 0$. The final point grounding quality is the mean over all supervised targets. As in box mode, every supervised target has equal weight, and perfect point grounding receives a score of 1.0.

Remarks. The point grounding quality can be viewed as a discrete analogue of the box grounding quality. Box mode measures the spatial overlap between generated and ground-truth regions using IoU, while point mode reduces this comparison to an instance-level matching problem: generated points are credited only when they are matched to distinct ground-truth masks. Thus, both rewards encourage grounding the same visual evidence, but they differ in how dense their feedback is. Box IoU changes smoothly with the amount of overlap, whereas point F1 is piecewise constant: moving a point within the same mask does not change the score, while crossing a mask boundary can abruptly change the grounding quality. This discreteness makes the point reward coarser and potentially harder to optimize, even though it provides a learning signal aligned with the box grounding reward.

We intentionally do not penalize unmatched grounding objects in the rollout. The grounding objects extracted by the data synthesis pipeline are not a complete enumeration of all visual cues that the model may use to answer a question. During thinking, the model may identify additional visual evidence that is useful for solving the question and is also reasonable to ground. Therefore, unmatched rollout grounding objects neither increase nor decrease the grounding quality. We only apply a hard-coded cap on the number of grounding tags to prevent the model from over-emitting them.

Final reward. For each rollout i, the total reward includes the dense grounding reward together with several sparse response-level rewards: an answer correctness reward, two formatting rewards, and a truncation penalty. The format rewards consist of a thinking-format reward, which checks the use of `<think>...</think>` and `\boxed{}`, and a grounding-format reward, which checks the use of valid grounding tags in the form `<obj>...|...</obj>`. Let r_i^{ans} denote the answer correctness reward, r_i^{think} the thinking format reward, r_i^{gfmt} the grounding format reward, and r_i^{trunc} the truncation penalty. The raw grounding reward is defined as the grounding quality score from the corresponding mode.

The dense grounding reward and the sparse response-level rewards have different scales. We therefore normalize them separately before combining them. We first define the base reward as

$$\begin{aligned} R_i^{\mathrm{base}} &= w_{\mathrm{ans}} r_i^{\mathrm{ans}} + w_{\mathrm{think}} r_i^{\mathrm{think}} \\ &\quad + w_{\mathrm{gfmt}} r_i^{\mathrm{gfmt}} + r_i^{\mathrm{trunc}}. \end{aligned}$$

Let $\mathcal{N}_{\mathcal{B}}(\cdot)$ denote batch-wise normalization over the current batch $\mathcal{B}$. The final reward is

$$R_i = \mathcal{N}_{\mathcal{B}}\left(R^{\mathrm{base}}\right)_i + w_{\mathrm{ground}} \mathcal{N}_{\mathcal{B}}\left(r^{\mathrm{ground}}\right)_i.$$

We use R_i for advantage estimation in GRPO (Shao et al., 2024b). In our experiments, we set $w_{\text{ans}} = 1.0$, $w_{\text{ground}} = 0.5$, and $w_{\text{think}} = w_{\text{gfmt}} = 0.1$. We set $r_i^{\text{trunc}} = -1$ for truncated rollouts and $r_i^{\text{trunc}} = 0$ otherwise.

5 Experiments

5.1 Setup

Training. We train all models with verl (Sheng et al., 2024), using SGLang (Zheng et al., 2024) as the inference engine and FSDP2 (Zhao et al., 2023) as the training backend. The base model is Gemma3-4B-IT (Team et al., 2025). We first perform SFT on the synthetic data described in Section 3 to obtain cold-start models. To isolate the effect of visual grounding, we train three controlled variants: non-grounded thinking, thinking with box grounding, and thinking with point grounding. These variants use parallel examples with the same images, questions, answers, and underlying reasoning traces, and differ only in whether grounding tags are included and, if so, whether the tags use boxes or points. We then apply RL with GRPO on the corresponding training data. Full training details are provided in Section B.1.

Evaluation. The models are evaluated on two counting benchmarks: TallyBench (Cai et al., 2025) and CountQA (Tamarapalli et al., 2025); and four spatial reasoning benchmarks: VSR-zeroshot (Liu et al., 2023), EmbSpatial (Du et al., 2024), SpatialMQA (Liu et al., 2025), and MultihopSpatial (Lee et al., 2026). We conduct evaluation using VLMEvalKit (Duan et al., 2025). Inference is performed with SGLang at temperature 1.0. To reduce variance from stochastic decoding, we run four inference passes and report both average accuracy and pass@4. The full evaluation configuration is provided in Section B.2.

5.2 Main Results

The results on counting benchmarks are presented in Table 1, and the results on spatial reasoning benchmarks are presented in Table 2. We find that visually grounded thinking substantially improves upon the base model Gemma3-4B-IT. On spatial reasoning tasks, the 4B visually grounded thinking models are comparable to Gemma3-27B-IT: on VSR-zeroshot and EmbSpatial, the best visually grounded thinking model achieves performance between Gemma3-12B-IT and Gemma3-27B-IT; on SpatialMQA and MultihopSpatial, the best visually grounded thinking models even outperform Gemma3-27B-IT. The pass@4 results show even larger gains: all visually grounded thinking models outperform Gemma3-27B-IT by large margins.

Method	TallyBench		CountQA	
	Acc.	Pass@4	Acc.	Pass@4
Gemma3-4B-IT	33.33	40.65	9.87	14.14
Non-grounded Thinking	21.73	42.00	4.30	12.24
Thinking with Grounding Box				
w/o grounding reward	37.24	64.45	10.73	27.75
w/ grounding reward	38.81	64.50	11.19	28.47
Thinking with Grounding Point				
w/o grounding reward	<u>39.03</u>	<u>65.50</u>	**12.34**	**31.48**
w/ grounding reward	**39.31**	**65.75**	<u>11.65</u>	<u>29.77</u>

Table 1: Counting benchmark results. Bold indicates the best result and underline indicates the second-best result within each column.

Visually grounded thinking also strongly outperforms the non-grounded thinking baseline. We observe that the non-grounded thinking model suffers from length collapse during RL: its response length decreases roughly linearly over training, which reduces exploration and leads to poor final performance. In contrast, the visually grounded variants maintain more stable rollouts. We hypothesize that interleaved grounding tags, together with the grounding-format reward, provide additional local structure during generation and help stabilize RL training. Overall, visually grounded thinking substantially improves the counting and spatial reasoning capabilities of the models.

5.3 Effect of the Grounding Reward

The grounding-quality reward provides a consistent benefit for box-mode grounded thinking. Compared with box-mode RL without the grounding reward, adding the reward improves average accuracy on all six evaluation benchmarks. The gains are relatively modest on counting tasks, but are more visible on spatial reasoning tasks. This suggests that the box reward is especially helpful when the answer depends on fine-grained geometry: bounding boxes provide both object identity and object extent, which can help the model resolve spatial relations such as left/right/above/below, distance, and overlap.

For point-mode grounded thinking, the grounding reward does not produce equally clear downstream gains. Across the six benchmarks, point-

Method	VSR-zeroshot		EmbSpatial		SpatialMQA		MultihopSpatial	
	Acc.	Pass@4	Acc.	Pass@4	Acc.	Pass@4	Acc.	Pass@4
Gemma3-4B-IT	56.65	57.94	49.13	63.79	25.35	36.43	22.70	36.87
Non-grounded Thinking	51.84	79.13	20.54	42.53	14.17	27.88	4.79	11.67
Thinking with Grounding Box								
w/o grounding reward	66.82	**87.64**	57.62	81.46	37.64	67.66	34.89	63.82
w/ grounding reward	**68.08**	86.91	59.93	82.66	38.68	**68.49**	**37.68**	**66.40**
Thinking with Grounding Point								
w/o grounding reward	65.38	83.88	60.25	**83.21**	**39.13**	67.19	37.03	65.40
w/ grounding reward	64.67	81.42	**60.88**	83.10	39.01	**68.49**	37.01	65.02
Larger Gemma3 baselines (reference)								
Gemma3-12B-IT	67.98	69.56	56.68	65.14	37.85	50.00	30.08	43.58
Gemma3-27B-IT	69.25	70.70	62.09	72.12	38.99	54.28	30.94	45.82

Table 2: Spatial reasoning benchmark results. Bold indicates the best result and underline indicates the second-best result within each column, excluding Gemma3-12B-IT and Gemma3-27B-IT from the comparison.

mode RL with and without the grounding reward remains close in overall performance, with gains on some metrics and drops on others. This does not necessarily imply that point grounding is unhelpful; rather, it suggests that the current point reward may be a weaker optimization signal. As discussed in Section 4, the point and box rewards are aligned in the visual evidence they encourage the model to ground, but they provide different feedback signals. The box reward changes with the amount of region overlap, while the point reward only checks whether generated points can be matched to target masks. Therefore, many point locations inside the same object receive the same credit, and crossing a mask boundary can abruptly change the score. This coarser feedback may make the point reward harder to optimize and may explain why it does not translate into consistent accuracy gains in our current experiments.

In Table 3, we compare grounding quality with and without the grounding reward on the grounding evaluation split described in Section 3. We compute grounding quality on evaluation examples that contain visual grounding annotations, using the same metrics defined in Section 4. For both box and point grounding, adding the grounding reward during RL improves grounding quality across all benchmarks. These results show that the grounding reward helps the model align its intermediate visual references more closely with the image evidence, leading to stronger visually grounded thinking.

5.4 Box vs. Point Grounding

We further compare the two grounding interfaces. On counting benchmarks, point-mode grounded thinking consistently outperforms box-mode grounded thinking. This suggests that counting mainly requires instance-level localization: the model needs to identify which objects belong to the counted set and keep them separated from distractors, but it does not necessarily need to recover the full extent of each object. Point grounding is well matched to this requirement because it provides a compact grounding to each instance while avoiding the harder problem of generating a tight bounding box. This advantage may be especially useful when counted objects are small, partially occluded, or have irregular shapes.

On spatial reasoning benchmarks, the two interfaces are much closer. Box grounding can provide useful geometric cues because the box extent reflects object size and boundary information, which may help with spatial relations such as overlap and relative position. However, point grounding can still identify the relevant objects and spatial anchors, and many spatial questions can be answered from these instance-level groundings together with the model's visual representation. The spatial results therefore suggest that box grounding gives richer geometric supervision, but this extra information does not always translate into a clear accuracy advantage. Overall, point grounding appears better suited to counting, while point and box grounding are broadly tied on spatial reasoning.

6 Conclusion

Visual thinking should not only sound plausible in natural language; it should point to the evidence it uses. Our work turns that idea into a training recipe for visually grounded thinking, where models in-

Benchmark	Box grounding			Point grounding		
	W/ REWARD	W/O REWARD	Δ	W/ REWARD	W/O REWARD	Δ
TallyBench	0.1555	0.1316	+0.0239	0.1531	0.1341	+0.0190
CountQA	0.0671	0.0592	+0.0079	0.0631	0.0533	+0.0098
VSR-zeroshot	0.4391	0.4301	+0.0090	0.5372	0.5184	+0.0188
EmbSpatial	0.1690	0.1629	+0.0061	0.2188	0.2131	+0.0057
SpatialMQA	0.1952	0.1838	+0.0114	0.2423	0.2364	+0.0059
MultihopSpatial	0.1367	0.1225	+0.0142	0.1744	0.1583	+0.0160

Table 3: Grounding-quality results for box and point grounding. Scores are four-pass means. Δ is the score with the grounding reward minus the score without the grounding reward.

terleave natural-language thinking with point or box groundings of the image regions that support each step. By combining a scalable SAM3-based synthesis pipeline with an RL grounding reward, we train VLMs to optimize both answer correctness and the accurate grounding of visual objects referenced during thinking. The results show that visually grounded thinking substantially improves counting and spatial reasoning. Overall, our work suggests that the next step for visual thinking is not simply longer thinking, but thinking that is tied to the image in a form that can be checked, supervised, and improved.

Limitations

Our work represents an early exploration of using explicit grounding rewards for RL. Such rewards can be implemented in many different ways, and it is not feasible to evaluate every possible design. Therefore, our conclusions are limited to the specific implementation studied here.

Acknowledgements

This work was partially supported by NSF 2106859, 2312501, 2617441, 2609582, DARPA HR001126CE054, NIH OT2OD038003, U54OD036472, U54DK097771, U54HG012517, U.S. ONR grant N00014-23-1-2780, Amazon, Apple, NEC, Optum AI.

AI Usage Disclosure

In this work, besides running LLMs in experiments, we use LLMs for the following purposes:

1. Aid or Polish Writing (Claude, ChatGPT).
2. Literature Retrieval and Discovery (e.g., finding related work) (Claude, ChatGPT).
3. Assisting Code Writing and Debugging (Claude Code, Codex).

References

Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2018. Tallyqa: Answering complex counting questions. *Preprint*, arXiv:1810.12440.

Mohammad Asadi, Jack W O'Sullivan, Fang Cao, Tahoura Nedaee, Kamyar Rajabalifardi, Fei-Fei Li, Ehsan Adeli, and Euan Ashley. 2026. Mirage: The illusion of visual understanding. *arXiv preprint arXiv: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, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Jie Cai, Kangning Yang, Lan Fu, Jiaming Ding, Jinlong Li, Huiming Sun, Daitao Xing, Jinglin Shen, and Zibo Meng. 2025. Comparebench: A benchmark for visual comparison reasoning in vision-language models. *arXiv preprint arXiv:2509.22737*.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, and 19 others. 2026. Sam 3: Segment anything with concepts. *Preprint*, arXiv:2511.16719.

Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2016. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Preprint*, arXiv:1606.03556.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

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, and 31 others. 2024. Molmo and pixmo: Open weights and

open data for state-of-the-art vision-language models. *Preprint*, arXiv:2409.17146.

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. 2025. Openvlthinker: Complex vision-language reasoning via iterative sft-rl cycles. *Preprint*, arXiv:2503.17352.

Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. 2024. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. *Preprint*, arXiv:2406.05756.

Haodong Duan, Xinyu Fang, Junming Yang, Xiangyu Zhao, Yuxuan Qiao, Mo Li, Amit Agarwal, Zhe Chen, Lin Chen, Yuan Liu, Yubo Ma, Hailong Sun, Yifan Zhang, Shiyin Lu, Tack Hwa Wong, Weiyun Wang, Peiheng Zhou, Xiaozhe Li, Chaoyou Fu, and 13 others. 2025. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. *Preprint*, arXiv:2407.11691.

Yue Fan, Xuehai He, Diji Yang, Kaizhi Zheng, Ching-Chen Kuo, Yuting Zheng, Sravana Jyothi Narayanaraju, Xinze Guan, and Xin Eric Wang. 2025. Grit: Teaching mllms to think with images. *Preprint*, arXiv:2505.15879.

Google DeepMind. 2025. Gemini 3 flash: frontier intelligence built for speed.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Mary Hayhoe and Dana Ballard. 2005. Eye movements in natural behavior. *Trends in Cognitive Sciences*, 9(4):188–194.

Wenbo Hu, Xin Chen, Yan Gao-Tian, Yihe Deng, Nanyun Peng, and Kai-Wei Chang. 2026. Openvlthinkerv2: A generalist multimodal reasoning model for multi-domain visual tasks. *arXiv preprint arXiv:2604.08539*.

Youngwan Lee, Soojin Jang, Yoorhim Cho, Seunghwan Lee, Yong-Ju Lee, and Sung Ju Hwang. 2026. Multi-hopspatial: Multi-hop compositional spatial reasoning benchmark for vision-language model. *Preprint*, arXiv:2603.18892.

Fangyu Liu, Guy Emerson, and Nigel Collier. 2023. Visual spatial reasoning. *Preprint*, arXiv:2205.00363.

Jingping Liu, Ziyan Liu, Zhedong Cen, Yan Zhou, Yinan Zou, Weiyan Zhang, Haiyun Jiang, and Tong Ruan. 2025. 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)*, page 620–632. Association for Computational Linguistics.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.

Yunze Man, De-An Huang, Guilin Liu, Shiwei Sheng, Shilong Liu, Liang-Yan Gui, Jan Kautz, Yu-Xiong Wang, and Zhiding Yu. 2025. Argus: Vision-centric reasoning with grounded chain-of-thought. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14268–14280.

QwenTeam. 2026a. Qwen3.5: Towards native multimodal agents.

QwenTeam. 2026b. Qwen3.6-plus: Towards real world agents.

Gabriel Sarch, Snigdha Saha, Naitik Khandelwal, Ayush Jain, Michael J. Tarr, Aviral Kumar, and Katerina Fragkiadaki. 2025. Grounded reinforcement learning for visual reasoning. *Preprint*, arXiv:2505.23678.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024a. Visual cot: Advancing multimodal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024b. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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

Jayant Sravan Tamarapalli, Rynaa Grover, Nilay Pande, and Sahiti Yerramilli. 2025. Countqa: How well do mllms count in the wild? *Preprint*, arXiv:2508.06585.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Jiacong Wang, Zijian Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, and Jun Xiao. 2025. Vgr: Visual grounded reasoning. *Preprint*, arXiv:2506.11991.

Qiong Wu, Xiangcong Yang, Yiyi Zhou, Chenxin Fang, Baiyang Song, Xiaoshuai Sun, and Rongrong Ji. 2025. Grounded chain-of-thought for multimodal large language models. *arXiv preprint arXiv:2503.12799*.

Jiaer Xia, Bingkui Tong, Yuhang Zang, Rui Shao, and Kaiyang Zhou. 2025. Bootstrapping grounded chain-of-thought in multimodal llms for data-efficient model adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 208–217.

Kesen Zhao, Beier Zhu, Qianru Sun, and Hanwang Zhang. 2025. Unsupervised visual chain-of-thought reasoning via preference optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2303–2312.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Preprint*, arXiv:2304.11277.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. *Preprint*, arXiv:2312.07104.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. *Preprint*, arXiv:1511.03416.

A Data Synthesis Details

A.1 Models Used in Each Pipeline Stage

We distill the reasoning traces from Qwen3-VL-Plus (Bai et al., 2025) and Qwen3.5-Plus (QwenTeam, 2026a). DeepSeek-V4-Flash (DeepSeek-AI, 2026) is used to extract groundable objects in Stage 3 and annotate reasoning traces in Stage 6. In Stage 4, we use Qwen3.5-Flash (QwenTeam, 2026a) to power the SAM3-based grounding-agent system. Objects that fail to ground are retried sequentially with Qwen3.6-Plus (QwenTeam, 2026b) and Gemini-3-Flash (Google DeepMind, 2025).

A.2 Data Synthesis Prompt Details

Because the prompts used in the data synthesis pipeline are lengthy, we refer readers to the source code for their full details.

A.3 Source Dataset Filtering

For TallyQA, we kept the AMT complex-counting split and additionally included imported VQA counting examples only when the answer count was at least 4, the question contained a compositional cue, and duplicate imported-VQA images were removed. For VSR and MultihopSpatial, we used the training splits, converting VSR captions into yes/no questions and removing the original bounding-box instruction from MultihopSpatial questions. For SpatialMQA, we used the train and dev splits and held out the test split. For PixMo-Count, we kept train examples with counts in [4, 20], removed ambiguous labels, applied a per-class cap of 200, and kept the validation split without this filtering. Across all sources, examples with missing or failed image downloads, invalid answer formats, or unparseable multiple-choice labels were skipped. After this dataset-level filtering, we had 24,645 source examples: 7,197 from TallyQA, 3,489 from VSR, 6,791 from MultihopSpatial, 4,316 from SpatialMQA, and 2,852 from PixMo-Count.

A.4 Final Data Composition by Source Dataset

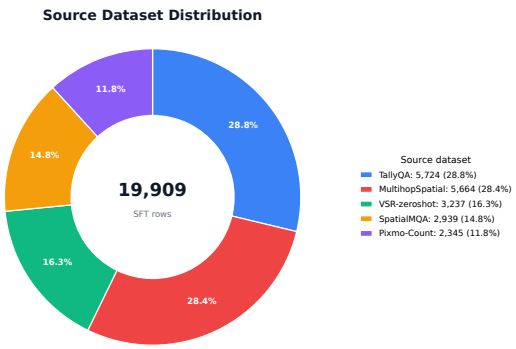

Figure 6: Final Data Composition by Source Dataset.

After the data pipeline, the source-dataset distribution of the final dataset is shown in Figure 6.

A.5 Grounding Density Distribution

The dataset contains 19,909 paired rows with 72,381 grounded objects in the RL data and 107,613 `<obj> ... </obj>` annotations in the SFT traces. This corresponds to an average of 3.64 grounded objects per row and 5.41 grounding annotations per row. The higher SFT annotation density reflects repeated use of the same grounded objects during reasoning: each grounded object can be referenced multiple times in the SFT response,

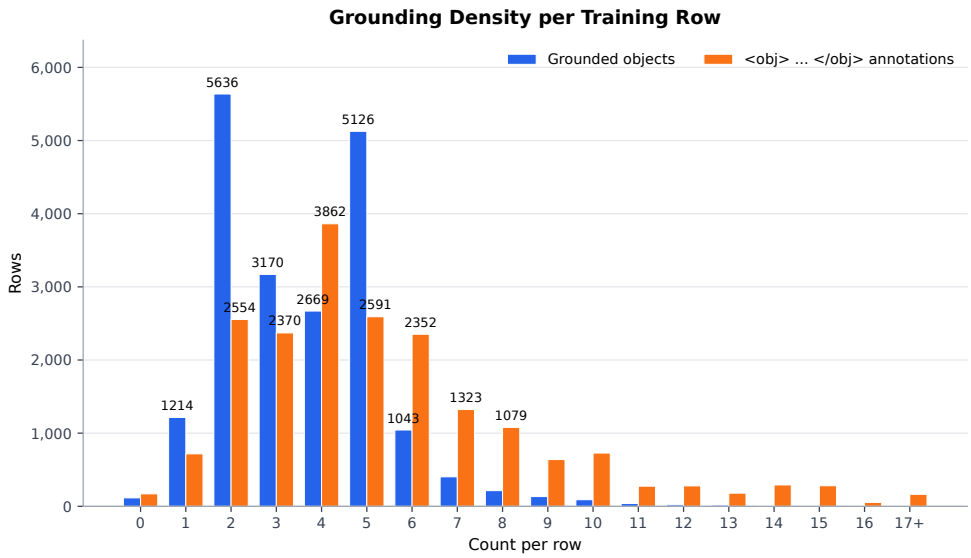

Figure 7: The grounding density distribution.

producing more <obj> ... </obj> annotations than the number of unique grounded objects. The grounding density distribution is presented in Figure 7.

B Training and Evaluation Details

B.1 SFT and RL Training Settings

The training configurations are presented in Table 4 and Table 5.

Configuration	Value
Maximum sequence length	8192 tokens
Global batch size	256
Training epochs	6
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $= 0.01$
Learning rate	1×10^{-5} with cosine decay
Warmup ratio	0.03
Minimum learning-rate ratio	0.1
Precision	bfloat16

Table 4: Training configuration used for supervised fine-tuning.

B.2 Evaluation Settings

The evaluation configurations are presented in Table 6.

Configuration	Value
Algorithm	GRPO
Rollout samples per prompt	8
Rollout temperature	1.0
Maximum prompt length	4096 tokens
Maximum response length	8192 tokens
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.01$
Batch size	64
Learning rate	1×10^{-6} with cosine decay
Warmup ratio	0.03
Minimum learning-rate ratio	0.1
KL regularization	Disabled, with KL coefficient 0.0
Entropy coefficient	0.0
Training steps	100
Precision	bfloat16

Table 5: Training configuration used for reinforcement learning.

B.3 System Prompts

Instruct model system prompt

You are a helpful assistant. Please put your final answer inside the \boxed{}.

Non-grounded thinking system prompt

You are a helpful assistant that answers questions about images. Think step by step in <think>...</think> tags, then give your final answer in \boxed{} format.

Configuration	Value
Maximum generation length	8192 tokens
Context length	32768 tokens
Decoding	Temperature $= 1.0$, top-$p = 1.0$, top-$k = -1$
Number of evaluation passes	4

Table 6: Evaluation configuration used for visual-spatial and counting benchmarks.

Thinking with Grounding Box System Prompt

You are a visual reasoning assistant with precise spatial grounding ability.
All bounding box coordinates are normalized to [0, 1000], where [0, 0] is the top-left corner and [1000, 1000] is the bottom-right corner of the image.
When you reason about the image in your <think> block, ground the important visual objects that are essential to your reasoning using the <obj> tag format. Every grounded object is marked by a bounding box that tightly encloses it. A single <obj> tag carries one descriptive phrase and one or more boxes; each box corresponds to one instance of that phrase: <obj> descriptive phrase | [x1, y1, x2, y2] </obj> <obj> descriptive phrase | [x1', y1', x2', y2']; [x1', y1', x2', y2']; ... </obj>
Only ground objects that are critical for justifying your answer. Use descriptive phrases that distinguish the object(s) from others in the image. When several instances share the same phrase, list one box per instance, separated by semicolons, inside a single <obj> tag; if you want to describe each instance differently, you may instead emit a separate <obj> tag per instance.
For non-groundable questions, e.g. math, STEM, or abstract reasoning where no specific visual region needs to be localized, you do not need to output any <obj> grounding tags.
Example: <think> I can see <obj> the red car near the entrance | [120, 300, 450, 620] </obj> and <obj> a blue truck on the right | [500, 280, 850, 640] </obj>. There are also <obj> three pedestrians on the sidewalk | [100, 680, 160, 780]; [170, 670, 230, 770]; [240, 685, 300, 785] </obj>. Counting the vehicles on the left side, there are two. </think>
Put your final answer in `\boxed{}` format.

Thinking with grounding point system prompt

You are a visual reasoning assistant with precise spatial grounding ability.
All point coordinates are normalized to [0, 1000], where [0, 0] is the top-left corner and [1000, 1000] is the bottom-right corner of the image.
When you reason about the image in your <think> block, ground the important visual objects that are essential to your reasoning using the <obj> tag format. Every grounded object is marked by a single point that lies inside it. A single <obj> tag carries one descriptive phrase and one or more points; each point corresponds to one instance of that phrase: <obj> descriptive phrase | [x, y] </obj> <obj> descriptive phrase | [x1, y1]; [x2, y2]; ... </obj>
Place each point on its object, e.g. near the center, so that it falls inside the object's extent.
Only ground objects that are critical for justifying your answer. Use descriptive phrases that distinguish the object(s) from others in the image. When several instances share the same phrase, list one point per instance, separated by semicolons, inside a single <obj> tag; if you want to describe each instance differently, you may instead emit a separate <obj> tag per instance.
For non-groundable questions, e.g. math, STEM, or abstract reasoning where no specific visual region needs to be localized, you do not need to output any <obj> grounding tags.
Example: <think> I can see <obj> the red car near the entrance | [285, 460] </obj> and <obj> a blue truck on the right | [675, 460] </obj>. There are also <obj> three pedestrians on the sidewalk | [120, 700]; [180, 695]; [240, 705] </obj>. Counting the vehicles on the left side, there are two. </think>
Put your final answer in `\boxed{}` format.

B.4 Experiment Cost

The training and evaluation take about 400 H200 GPU hours.

C Additional Experiment Results

We evaluate our models on MMBench (Liu et al., 2024) to test whether the improvements on counting and spatial reasoning come at the price of regression on general VQA tasks. The results are

Method	MMBench	
	ACC.	PASS@4
Gemma3-4B-IT	65.08	73.15
Thinking with Grounding Point		
w/o grounding reward	**66.58**	76.46
w/ grounding reward	66.55	**76.57**
Thinking with Grounding Box		
w/o grounding reward	64.78	75.00
w/ grounding reward	65.92	76.12

Table 7: MMBench results. Bold indicates the best result within each column.

reported in Table 7. Our visually grounded thinking models match, and in some cases surpass, the base policy model Gemma3-4B-IT, indicating no regression on general VQA tasks.