Perceive, Interact, Reason: Building Tool-Augmented Visual Agents for Spatial Reasoning

Changye Li[1], **Meng Lu**[2], **Yi Wu**[1], **Ligeng Zhu**[3]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Virginia Tech [3]NVIDIA

Code: `https://github.com/antoinegg1/PEDIA`
Model: `https://huggingface.co/Changyeli03/pedia_model`
Dataset: `https://huggingface.co/datasets/Changyeli03/pedia_data`

Abstract

While recent vision-language models (VLMs) demonstrate strong multimodal understanding, they remain limited in spatial reasoning tasks that require active evidence acquisition and multi-step visual interaction. This limitation suggests that relying solely on implicit visual representations from vision encoders is insufficient for recovering fine-grained spatial evidence. We introduce *PERception-Interaction-reason Agent* (**PERIA**), a tool-augmented visual agent for spatial reasoning tasks across map reasoning, visual probing, and vision reconstruction. PERIA uses two lightweight tool families: *vision perception tools* for exposing textual, symbolic, and spatial evidence, and *vision interaction tools* for manipulating visual context, tracing paths, and verifying spatial relations. To train PERIA, we develop a unified recipe that combines supervised tool-use trajectory synthesis, composite rewards, and Observation-Relaxed Group-in-Group Policy Optimization (OR-GIGPO) for effective multi-tool behavior. Experiments on 13 benchmarks from 8 datasets show that PERIA-8B improves over the Qwen3-8B backbone by 10.0 percentage points on in-distribution benchmarks and 4.4 percentage points on out-of-distribution benchmarks, while outperforming previous state-of-the-art baselines of similar size by 6.9–14.8 percentage points. It also achieves performance comparable to much larger models such as Qwen3-VL-235B-A22B-Thinking and GPT-5, demonstrating the effectiveness of PERIA in enhancing spatial reasoning capabilities.

1 Introduction

Vision-language models (VLMs) have made strong progress in multimodal understanding and vision-language reasoning [Bai et al., 2025, Comanici et al., 2025, Wang et al., 2025a, Singh et al., 2025]. Chain-of-thought prompting [Wei et al., 2023] and reasoning-oriented post training [Zhang et al., 2024] further enhance their ability to perform visual reasoning. However, many complex visual tasks require more than reasoning over a static visual representation, as models need to ground fine-grained evidence, extract task-relevant details, and reason about spatial relations through multiple intermediate steps [Yue et al., 2024, Ma et al., 2024, Stogiannidis et al., 2025]. Tool use provides a natural way to extend model capabilities. Inspired by LLM agents [Yao et al., 2023, Schick et al., 2023], recent VLMs have begun to use external visual tools for reasoning [OpenAI, 2025, Lu et al., 2025, Fu et al., 2025, Wu et al., 2026b].

Existing tool-augmented VLMs show promising gains on structured visual tasks such as chart understanding and mathematical problem solving, where answers often come from explicit evidence and mature expert tools such as ChartMoE [Xu et al., 2025]. Yet many spatial tasks cannot be solved by reading a local text span, chart element, equation, or detected object. For example, subway-map

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

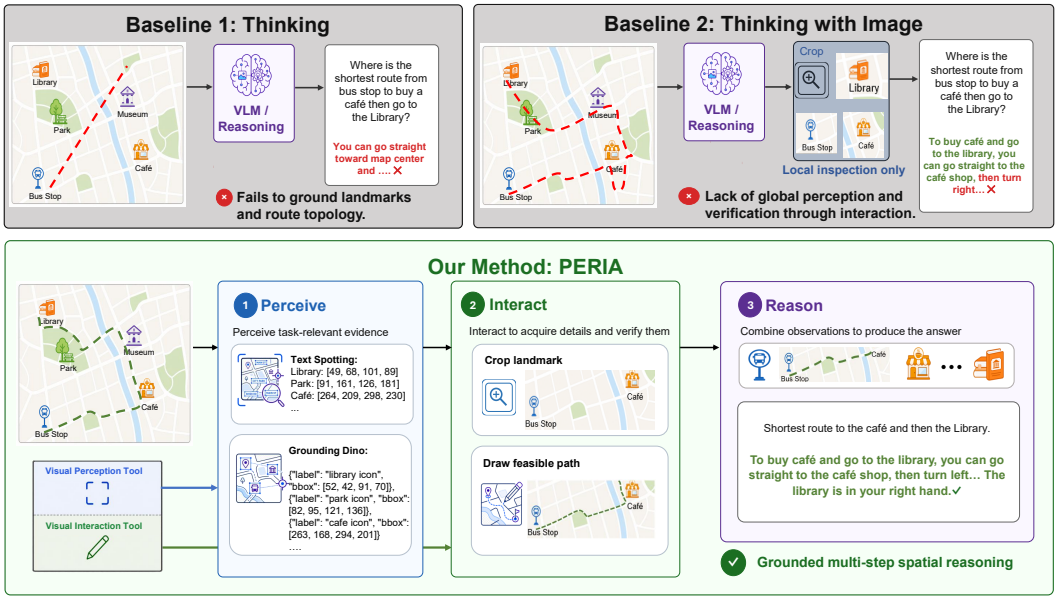

Figure 1: **Overview of our proposed *PERception-Interaction-reason Agent* (PERIA).** *PERception-Interaction-reason Agent* (**PERIA**) first uses perception tools to gather global spatial evidence, then applies interaction tools for fine-grained local analysis, and finally reasons over accumulated observations. This design addresses the limits of conventional methods, which either miss local visual details or fail to recover global spatial structure.

reasoning requires aligning station labels with route topology [Feng et al., 2025c,b], while MapTrace requires drawing valid paths over maps [Panagopoulou et al., 2025]. These tasks require a coherent spatial state that links entities across regions and supports topology-dependent reasoning.

The main limitation is a mismatch between current visual tools and spatial reasoning. Most tools read evidence or focus regions through OCR or cropping, but they do not by themselves build or update a spatial state across tool calls. Existing visual-tool methods also use limited tool spaces or domain-specific training data, which limits transfer to broader spatial reasoning [Fu et al., 2025, Lai et al., 2025]. Raw tool access is not enough either: Section 3 shows that tools provide useful evidence, but open-source VLMs often fail to use them reliably without tool-use training. Thus, the root issue is the lack of a trained policy that can compose perception and interaction tools, gather local evidence, recover global structure, and reason over accumulated observations.

This motivates our central question: *How can we train visual agents to leverage diverse visual tools within a unified reasoning process for spatial reasoning tasks?* We introduce *PERception-Interaction-reason Agent* (**PERIA**), a tool-augmented visual agent for spatial reasoning tasks. As shown in Figure 1, PERIA organizes visual problem solving into a unified process: the agent perceives task-relevant evidence, interacts with visual content for spatial planning and verification, and reasons over accumulated observations to produce the answer. It is supported by two complementary tool families: *vision perception tools* and *vision interaction tools*. For policy optimization, we further develop *Observation-Relaxed Group-in-Group Policy Optimization* (OR-GIGPO) to assign credit across multi-step visual tool-use trajectories. Empirically, PERIA-8B surpasses previous state-of-the-art baselines of similar size and reaches performance comparable to current leading models. Our contributions are summarized as follows:

- We introduce a diverse perception–interaction toolset for spatial reasoning and formulate PERIA as a general framework for tool-augmented visual agents that reason, plan, and verify over images. We further synthesize a diverse, high-quality trajectory dataset for supervised fine-tuning, enabling reliable tool invocation and tool-grounded reasoning.
- We develop a unified reinforcement training recipe centered on *Observation-Relaxed Group-in-Group Policy Optimization* (OR-GIGPO), our core optimization method for visual tool-use trajectories. Together with task-specific rewards, OR-GIGPO enables fine-grained credit

assignment and effective multi-tool composition. Ablation studies show that OR-GIGPO surpasses mainstream reinforcement learning algorithms in our tasks.

- We instantiate PERIA as a spatial reasoning agent across diverse task families, including map reasoning, visual probing, and vision reconstruction. Across 13 benchmarks from 8 datasets, PERIA-8B achieves consistent gains of 6.9–14.8 percentage points over strong VLM and tool-augmented baselines at a comparable scale, while approaching the performance of much larger models such as Qwen3-VL-235B-A22B-Thinking and GPT-5.

2 Related Work

Tool-Augmented Visual Reasoning Tool use is a common way to extend language agents beyond text generation. ReAct [Yao et al., 2023] shows that models can interleave reasoning traces with external actions, forming a simple recipe for multi-step problem solving and environment feedback integration. Recent VLM agents extend visual reasoning with tool calls in domains such as charts, mathematics, and tables, helping models read text, edit images, or call expert modules with strong empirical gains [Lu et al., 2025, Fu et al., 2025, Wu et al., 2026b, Zhao et al., 2025]. Another line of work explores interactive visual tools, where the DeepEyes series and Mini-o3 demonstrate the effectiveness of multi-turn image cropping, and Think3D studies 3D manipulation tools for spatial exploration [Zheng et al., 2026, Hong et al., 2026, Lai et al., 2025, Zhang et al., 2026]. However, the former line leaves spatial reasoning underexplored, while the latter often focuses on a small toolset or a narrow task family. In contrast, our work jointly trains perception and interaction tools within a unified framework, targeting general spatial reasoning agents that can read, inspect, trace, verify, and compose evidence across diverse task families.

Spatial Reasoning in VLMs. Spatial reasoning tests whether VLMs can connect visual evidence with relations, routes, layouts, and geometric changes. Prior work studies this ability across domains such as 3D spatial understanding [Chen et al., 2024, Ma et al., 2024], map and geospatial reasoning [Dihan et al., 2025, Xing et al., 2025, Feng et al., 2026], and visual search or probing in cluttered images [Wu and Xie, 2023b, Lai et al., 2025, Gavrikov et al., 2025]. Recent work further moves from isolated tasks toward broader spatial intelligence, including shared spatial skills, low-level visual primitives, simulation, and reconstruction [Stogiannidis et al., 2025, Jia et al., 2026, Chen et al., 2026, Wu et al., 2026a]. Together, these studies suggest that spatial reasoning requires global-local alignment, path tracing, relation checking, and multi-step planning, motivating our perception–interaction tool design that combines visual evidence acquisition with explicit visual interaction.

Reinforcement Learning for Tool-Use Agents. Reinforcement learning (RL) is widely used in post-training: PPO [Schulman et al., 2017] provides a general policy optimization framework, GRPO [Shao et al., 2024] estimates relative advantages from groups of sampled responses, and DAPO [Yu et al., 2025] improves large-scale RL with decoupled clipping and dynamic sampling. For tool-use agents, sparse rewards and long-horizon interactions make it difficult to evaluate intermediate decisions; RAGEN addresses this issue in multi-turn agent RL, while GIGPO assigns learning signals to both full trajectories and intermediate states [Wang et al., 2025b, Feng et al., 2025a]. Our setting further introduces visual observation variability from tool-agent outputs, motivating observation-relaxed policy optimization instead of exact state matching.

3 Motivation: The Gap Between Having Tools and Using Tools

Having visual tools does not mean using them well. We test this gap on 6 benchmarks from 4 datasets: MapTrace [Panagopoulou et al., 2025], ReasonMap [Feng et al., 2025c], ReasonMap-Plus [Feng et al., 2025b], and Visual Probing Easy/Medium/Hard [Lai et al., 2025]. The raw-tool setting gives each model access to 18 perception and interaction tools for evidence extraction, spatial annotation, and local inspection; Section 4.1 details the sandbox. As shown in Figure 2, GPT-5 improves with these tools, confirming that they expose useful spatial evidence. In contrast, Qwen3-VL-Thinking models often degrade, showing that open-source VLMs do not reliably turn tool access into better spatial reasoning without dedicated tool-use training.

We further diagnose the gap by inspecting 300 cases where GPT-5 succeeds with tools but Qwen3-VL-Thinking models fail. Table 1 shows that most errors are semantic tool-use failures rather

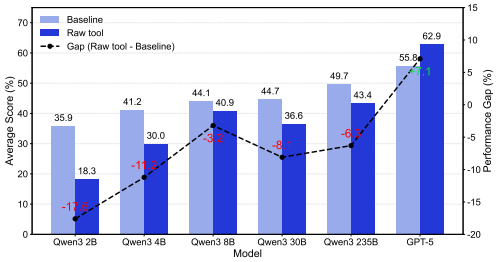

Figure 2: **Tool access does not guarantee tool competence.** We compare reasoning without tools (*Baseline*) with direct tool-augmented reasoning (*Raw tool*). GPT-5 benefits from raw tool access, but Qwen3-VL-Thinking models, abbreviated as Qwen3 in the figure, often degrade without tool-use training. The gap narrows with model size but remains clear.

Model	Error Count	Tool-Call Omission	Tool-Induced Error	Format Error	Other
Qwen3-2B	296 (98.7%)	**55.9%**	31.1%	2.0%	11.0%
Qwen3-4B	176 (58.7%)	**71.6%**	25.0%	2.3%	1.1%
Qwen3-8B	166 (55.4%)	**59.0%**	30.7%	1.8%	8.4%
Qwen3-30B-A3B	172 (57.3%)	**68.6%**	30.8%	0.6%	0.0%
Qwen3-235B	133 (44.3%)	38.3%	**60.9%**	0.8%	0.0%

Table 1: **Most failures come from not using tools well.** We analyze 300 cases from 6 benchmarks where GPT-5 solves the task with tools but Qwen3-VL-Thinking models, abbreviated as Qwen3 in the table, fail in most cases. Tool-call omission and tool-induced errors dominate, showing that current models mainly struggle with tool selection, invocation, and feedback integration. Larger models fail less often, but clear tool-use gaps remain.

than formatting failures. Models often skip needed tools or are misled by tool outputs during later reasoning. Thus, the gap is mainly a policy-learning problem: models must learn when to call tools, which tools to call, and how to integrate tool feedback. This motivates PERIA, which trains agents to coordinate perception and interaction tools throughout multi-step spatial reasoning.

4 Methodology

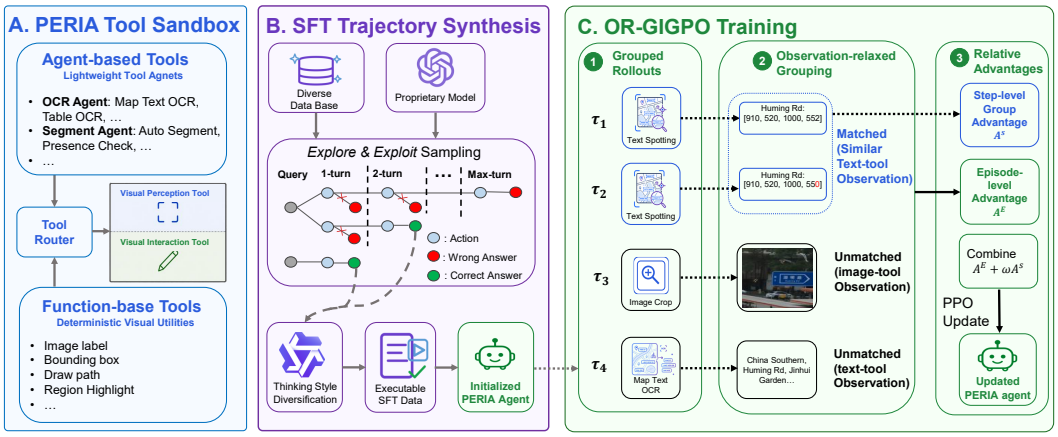

Figure 3: **PERIA learns tool use through sandbox construction, trajectory synthesis, and OR-GIGPO.** The pipeline first builds the PERIA Tool Sandbox, which organizes function-based visual utilities and lightweight agent-based tools into perception and interaction tools. It then synthesizes executable SFT trajectories with budgeted explore-and-exploit sampling and thinking-style diversification. Finally, OR-GIGPO groups rollouts by relaxed observations and combines step-level and episode-level advantages to assign credit across multi-step tool-use trajectories.

In this section, we describe the construction of PERIA, as illustrated in Figure 3. The overall design follows three stages. First, we formulate PERIA as a tool-augmented visual agent and build a modular tool sandbox that exposes both deterministic visual utilities and lightweight tool agents, organized into perception and interaction tools for acquiring evidence and manipulating visual context. Second, we synthesize executable tool-use trajectories for supervised fine-tuning (SFT): a proprietary model explores candidate tool-call paths under increasing turn budgets, and an expert model diversifies the intermediate reasoning styles. Third, we train the initialized agent with *Observation-Relaxed Group-in-Group Policy Optimization* (OR-GIGPO), which groups rollouts by relaxed observations and combines episode-level and step-level advantages for multi-step tool-use credit assignment. We

detail the PERIA formulation and tool sandbox in Section 4.1, trajectory synthesis in Section 4.2, and OR-GIGPO training with composite rewards in Section 4.3.

4.1 PERIA Formulation and Tool Sandbox

PERIA as a POMDP. Following prior formulations of tool-integrated and agentic reasoning [Yao et al., 2023, Lu et al., 2025], we formulate PERIA as a partially observable Markov decision process (POMDP) $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{I}$, $\mathcal{S}$, $\mathcal{O}$, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{R}$ denote the instruction, latent state, observation, action, transition, and reward spaces. Given an instruction $x \in \mathcal{I}$ and an initial visual observation o_0, we denote the task input as $q = (x, o_0)$. The agent iteratively generates a reasoning thought and an action:

$$g_t \sim \pi_\theta(\cdot \mid q, c_{t-1}), \qquad a_t \sim \pi_\theta(\cdot \mid q, c_{t-1}, g_t),$$

where $c_{t-1} = (g_1, a_1, o_1, \ldots, g_{t-1}, a_{t-1}, o_{t-1})$ is the interaction history, and a_t can be a perception-tool call, an interaction-tool call, or a final-answer action. Tool actions update the environment through $\mathcal{P}$ and return new observations, which are appended to the history. The resulting trajectory is

$$\tau = (g_1, a_1, o_1, \ldots, g_{T-1}, a_{T-1}, o_{T-1}, g_T, y) \sim \pi_\theta(\tau \mid q),$$

where $y = a_T$ is the final-answer action generated after the final reasoning step g_T.

PERIA Tool Sandbox. To support flexible tool-augmented spatial reasoning, we build a modular PERIA *Tool Sandbox* with 18 tools, implemented through function-based utilities and 3 lightweight tool agents. The sandbox provides a unified and extensible interface for tool invocation. We organize the tools into two categories: *visual perception tools*, which extract task-relevant evidence such as text, coordinates, and object presence, represented by `text_spotting`, `map_text_ocr`, and `auto_segment`; and *visual interaction tools*, which explicitly manipulate or annotate the visual scene, represented by `draw_path`, `bounding_box`, and `image_crop`. Our sandbox design draws inspiration from recent tool-augmented systems [Wu et al., 2026b, Fu et al., 2025, Lu et al., 2025], while introducing newly designed and task-specific tools tailored to spatial reasoning. Full tool definitions and sandbox design are provided in Appendix C.1.

4.2 Synthesizing Tool-Use Trajectories

Our preliminary analysis in Section 3 shows that directly enabling tool calls does not consistently improve model performance. We therefore synthesize tool-executing trajectories with a proprietary model (`GPT-5`) as supervised fine-tuning data to enhance the tool-calling capability of models.

Given a training instance (q, y^*), where $q = (x, o_0)$, we generate trajectories with an *explore-and-exploit* strategy. At round t, π_{pro} expands each current prefix by one reasoning–action step:

$$\{\tau_t^{(i)} = (\tau_{t-1}^{(i)}, g_t^{(i)}, a_t^{(i)})\}_{i=1}^{n_{\text{sample}}} \sim \pi_{\text{pro}}(\cdot \mid q, c_{t-1}^{(i)}),$$

where $a_t^{(i)}$ is either a tool call or a final-answer action; if it is a tool call, the returned observation $o_t^{(i)}$ is appended to the next prefix, while final-answer actions are denoted by $y_t^{(i)}$. We accept a trajectory once $y_t^{(i)} = y^*$; otherwise, we add error feedback and continue expansion until a valid trajectory is found or the turn budget T is reached.

To diversify reasoning, we rewrite only the intermediate thoughts $\{g_t\}$ with an open-weight expert model (`Qwen3-VL-235B-A22B-Thinking`), while keeping tool calls, observations, and answers fixed. Full details of SFT trajectory synthesis are provided in Appendix C.2.

4.3 Training Tool-Augmented Spatial Agents

We first fine-tune the model on the trajectories synthesized in Section 4.2, which equips the agent with basic tool-use abilities and the Perceive–Interact–Reason workflow. Since SFT mainly imitates existing trajectories and is insufficient for flexible tool composition, we further optimize the agent with reinforcement learning. To better handle multi-tool trajectories, we design OR-GIGPO for credit assignment and use a composite reward for spatial reasoning.

OR-GIGPO. Motivated by Group-in-Group Policy Optimization (GIGPO) [Feng et al., 2025a], we design *Observation-Relaxed Group-in-Group Policy Optimization* (OR-GIGPO) for multi-step spatial reasoning training. GIGPO combines episode-level and step-level advantages, comparing actions taken from the same anchor state. OR-GIGPO instead groups actions by their resulting tool observations within the same task. Since equivalent tool calls may return slightly different outputs, especially from tool agents, we relax exact observation matching to accommodate these variations.

OR-GIGPO retains the two-level advantage structure but uses action outcomes for step-level grouping. Following Section 4.1, let $q = (x, o_0)$ denote the instruction and initial visual observation. We define the episode-level rollout group and the step-level outcome group as

$$\mathcal{G}_E(q) = \{\tau_i : q_i = q\}_{i=1}^N, \mathcal{G}_S(q, \tilde{o}) = \big\{(a_t^{(i)}, R_t^{(i)}) : q_i = q, \underbrace{h(o_t^{(i)}) = h(\tilde{o}) \vee \mathrm{sim}(o_t^{(i)}, \tilde{o}) \geq \delta}_{\text{observation-relaxed matching}}\big\}.$$

Here τ_i is the i-th rollout, $a_t^{(i)}$ is the action generated conditioned on $(q^{(i)}, c_{t-1}^{(i)}, g_t^{(i)})$, and $o_t^{(i)}$ is the observation returned after executing that action. For step-level credit assignment, we associate each action with its discounted return $R_t^{(i)} = \sum_{k=t}^T \gamma^{k-t} r_k^{(i)}$, where T is the number of turns, γ is the discount factor, and $r_k^{(i)}$ is the reward at turn k. The episode group $\mathcal{G}_E(q)$ contains rollouts from the same input, while $\mathcal{G}_S(q, \tilde{o})$ groups action-return pairs by matching their resulting observations to $\tilde{o}$, without requiring identical pre-action states. Here $h(\cdot)$ is used for exact observation matching, and $\mathrm{sim}(\cdot, \cdot)$ matches semantically similar textual observations from tool agents, implemented with Python's `SequenceMatcher`.

For each rollout τ_i, let $R_i = R(\tau_i)$ be the trajectory-level reward. We compute the episode-level advantage A_E and the step-level advantage A_S as

$$A_E(\tau_i) = \frac{R_i - \mu_E(q)}{\sigma_E(q) + \epsilon_{\mathrm{adv}}}, \qquad A_S(a_t^{(i)}) = \frac{R_t^{(i)} - \mu_S(q, \tilde{o})}{\sigma_S(q, \tilde{o}) + \epsilon_{\mathrm{adv}}}.$$

Here $\mu_E(q)$ and $\sigma_E(q)$ are the mean and standard deviation of trajectory rewards over rollouts from the same task input $q = (x, o_0)$, and ϵ_{adv} is a small constant for numerical stability. The step-level statistics $\mu_S(q, \tilde{o})$ and $\sigma_S(q, \tilde{o})$ are computed over discounted returns in the same outcome group $\mathcal{G}_S(q, \tilde{o})$. In our trajectory-level reward setting, this step-level return propagates the final rollout reward back to intermediate tool decisions.

The final advantage combines trajectory-level credit and outcome-group credit:

$$A_t^{(i)} = A_E(\tau_i) + \omega A_S(a_t^{(i)}),$$

where ω controls the strength of step-level credit assignment. We optimize the policy with a clipped objective using the combined advantage, without an explicit KL penalty:

$$\mathcal{J}_{\text{OR-GIGPO}}(\theta) = \mathbb{E}\left[\frac{1}{\sum_{i=1}^N T_i} \sum_{i=1}^N \sum_{\ell=1}^{T_i} \min\left(\rho_\ell^{(i)} \hat{A}_\ell^{(i)}, \mathrm{clip}(\rho_\ell^{(i)}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}) \hat{A}_\ell^{(i)}\right)\right],$$

where θ denotes the policy parameters and ϵ_{clip} is the clipping threshold. Here N is the number of rollouts, T_i counts the generated thought and action tokens in rollout i, excluding tool observations, and $\hat{A}_\ell^{(i)} = A_t^{(i)}$ for a token generated at turn t. The token-level importance sampling ratio is

$$\rho_\ell^{(i)} = \frac{\pi_\theta(z_\ell^{(i)} \mid h_\ell^{(i)})}{\pi_{\theta_{\mathrm{old}}}(z_\ell^{(i)} \mid h_\ell^{(i)})},$$

where $z_\ell^{(i)}$ is the ℓ-th generated token and $h_\ell^{(i)}$ is its full preceding context, including the input, previous generated tokens, and tool observations. Further details on observation-relaxed grouping and OR-GIGPO implementation are provided in Appendix C.3.

Reward Design. For a generated rollout τ, we define a composite trajectory-level reward:

$$R(\tau) = R_{\mathrm{rep}}(\tau) + R_{\mathrm{format}}(\tau) + R_{\mathrm{correct}}(\tau).$$

The repetition penalty R_{rep} discourages repeated characters, repeated words, and repeated long text spans, and further applies turn-normalized penalties to sentence-level repetition to balance multi-turn

tool use. The format reward R_{format} encourages valid trajectories with explicit tags: intermediate tool-use turns should follow `<think>...</think><action>...</action>`, while the final turn should follow `<think>...</think><answer>...</answer>`. The correctness reward R_{correct} evaluates the final answer with normalized matching for general spatial QA, covering yes/no, numerical, range, and unordered-list answers. For MapTrace [Panagopoulou et al., 2025], we follow its NDTW-based metric and use a continuous reward:

$$R_{\text{correct}} = \min\{1, \max\{0, (d_{\text{high}} - d_{\text{NDTW}})/(d_{\text{high}} - d_{\text{low}})\}\}.$$

Here d_{NDTW} denotes the path distance, with $d_{\text{low}} = 0.3$ and $d_{\text{high}} = 0.8$ by default. For other tasks, R_{correct} is binary. Further details on reward penalties, correctness verification, and the LLM-judge fallback design are provided in Appendix C.4.

5 Experiments

Datasets and Metrics. Following prior visual reasoning evaluations [Feng et al., 2025b, Lai et al., 2025, Lu et al., 2025, Liu et al., 2026], we evaluate PERIA on a broad set of spatial reasoning tasks. For *in-distribution* evaluation, we select four task families from our training environment with held-out test splits: (1) MapTrace [Panagopoulou et al., 2025], (2) ReasonMap [Feng et al., 2025c], (3) ReasonMap-Plus [Feng et al., 2025b], and (4) Visual Probing tasks [Lai et al., 2025]. To assess *out-of-distribution* generalization, we evaluate additional benchmarks not used for training, including four tasks from Wu et al. [2026a] that span distinct spatial reasoning domains: (5) Ball Tracking, (6) Paper Folding, (7) Cube Three-View Reasoning, and (8) Real-world Spatial Reasoning, as well as (9) V* [Wu and Xie, 2023a], (10) MapEval [Dihan et al., 2025], and (11) BabyVision [Chen et al., 2026]. We report accuracy as the unified metric across all tasks. For MapTrace, we use NDTW to measure path alignment and report threshold-based accuracy, counting a prediction as correct when its distance is below 1.0. We further discuss how different evaluation criteria affect MapTrace scores in Appendix D.3. Our training data includes in-distribution trajectories and additional map QA data from MapQA [Chang et al., 2022]. Detailed information about the training data and evaluation datasets is provided in Appendix D.

Baselines. We compare PERIA with three groups of baselines: (i) *API-based proprietary VLMs*, including GPT-5 [Singh et al., 2025], Gemini-2.5-Flash [Comanici et al., 2025], and Gemini-2.5-Pro [Comanici et al., 2025]; (ii) *open-source VLMs*, including the `Qwen3-VL` series [Bai et al., 2025] with thinking enabled and the InternVL3.5 series [Wang et al., 2025a]; and (iii) *tool/reasoning-integrated VLMs*, including VTool-R1 series [Wu et al., 2026b], R1-OneVision-7B [Yang et al., 2025], and Mini-o3-7B-v1 [Lai et al., 2025]. Non-agentic baselines are evaluated without access to our tool sandbox, while tool-integrated baselines follow their original tool configurations.

Implementation Details. We instantiate PERIA using `Qwen3-VL-Thinking` backbones at 2B, 4B, and 8B scales. We conduct training with Verl-Tool [Jiang et al., 2025] and use vLLM [Kwon et al., 2023] for inference. For SFT, we train with a learning rate of 1×10^{-5} and a batch size of 32. For RL, we train with a learning rate of 1×10^{-6}, a batch size of 64, a group size of $N = 4$, and 3 epochs. Additional implementation details are provided in Appendix B.

5.1 Main Comparison

PERIA-8B achieves strong overall performance across 13 benchmarks. As reported in Table 2, PERIA-8B consistently improves over the Qwen3-8B backbone by 10.0 percentage points on in-distribution tasks and 4.4 percentage points on out-of-distribution tasks, and it exceeds prior state-of-the-art baselines at a comparable scale by 6.9–14.8 percentage points across 13 benchmarks from 8 datasets. On average, it further approaches larger open-weight and proprietary models such as Qwen3-VL-235B-A22B-Thinking and GPT-5. Across individual domains, including map-based reasoning, route tracing, visual probing, real-world spatial understanding, dynamic object tracking, object transformation, multi-view 3D reasoning, visual search, geospatial understanding, and language-free visual reasoning, PERIA-8B outperforms most similarly sized baselines and achieves performance comparable to much larger open-source models. These results indicate that effective visual tool-use training can provide gains beyond backbone scaling alone.

Methods (↓)	In-Distribution Dataset							Out-of-Distribution Dataset								Overall
Datasets (→)	ReasonMap Plus	ReasonMap	Visual Probing Easy	Visual Probing Medium	Visual Probing Hard	MapTrace	Avg.	MapEval	BabyVision	Ball Tracking	Paper Folding	Cube 3-View Reasoning	Real-world Spatial Reasoning	V*	Avg.	Avg.
Commercial Models																
GPT-5	89.7	53.8	62.4	21.6	25.4	82.2	55.8	72.7	22.9	25.9	11.0	32.9	38.3	70.1	39.1	46.8
Gemini-2.5-Flash	85.4	43.2	46.1	36.0	33.6	66.4	51.7	71.0	19.5	22.7	7.5	55.0	36.7	76.9	41.3	46.2
Gemini-2.5-Pro	81.3	55.1	62.1	38.7	32.0	80.4	58.2	71.0	27.3	28.3	16.8	55.0	40.0	78.5	45.2	51.3
Open-Source VLMs and Tool/Reason-Augmented Baselines																
InternVL3.5-8B	53.3	5.8	54.7	25.7	15.2	70.8	37.5	53.7	13.9	17.9	13.5	**51.0**	28.4	71.7	35.7	36.6
VTool-R1-7B	38.3	1.5	30.5	26.4	22.6	45.2	27.4	45.7	12.3	18.0	15.2	35.6	30.5	70.6	32.5	30.2
R1-OneVision-7B	52.1	8.2	39.2	21.7	17.3	41.9	30.0	45.2	14.4	15.7	12.9	33.7	28.2	53.9	29.1	29.6
Mini-o3	45.9	1.6	**70.2**	**52.9**	41.5	30.4	40.4	46.7	11.6	16.8	12.9	39.7	28.9	**88.2**	35.0	37.5
InternVL3.5-14B	51.3	4.4	58.1	12.0	19.8	76.2	36.9	54.0	14.6	16.2	11.0	46.6	26.8	69.6	34.1	35.4
Qwen3-30B-A3B	67.8	18.2	51.7	30.2	25.4	75.0	44.7	64.7	14.9	19.5	8.5	37.7	32.8	84.2	37.4	40.8
Qwen3-32B	68.8	20.6	57.4	32.0	24.5	75.5	46.4	67.2	**22.9**	20.3	8.1	48.1	**38.0**	85.8	**41.4**	43.8
Qwen3-235B	**76.7**	30.7	58.8	32.0	27.3	72.9	49.7	**69.0**	21.9	**20.9**	6.0	38.5	34.5	86.9	39.6	44.3
Our Method																
Qwen3-8B	65.7	12.3	54.6	31.3	26.6	74.1	44.1	54.0	13.0	17.8	11.0	21.2	27.7	77.5	31.7	37.4
PERIA-8B (ours)	72.6 +6.9	27.1 +14.8	61.7 +7.1	41.2 +9.9	**44.3** +17.7	77.7 +3.6	**54.1** +10.0	55.5 +1.5	15.5 +2.5	18.8 +1.0	**15.7** +4.7	36.3 +15.1	29.1 +1.4	82.2 +4.7	36.1 +4.4	**44.4** +7.0
w/o RL	66.2	21.1	51.0	37.3	32.0	74.2	46.9	54.8	12.4	18.1	14.6	39.0	29.6	74.4	34.7	40.4
w/o SFT	64.2	**45.1**	19.1	18.6	26.4	71.1	40.7	49.5	7.2	12.4	6.0	29.1	25.0	59.2	26.9	33.3
w/o Tool	72.4	13.7	36.1	21.2	19.8	**82.4**	40.9	59.0	13.9	19.6	15.0	30.4	28.3	76.4	34.7	37.6

Table 2: **PERIA closes much of the gap to larger proprietary and open-source models.** Scores are accuracy percentages, green gains are percentage points, and Qwen3 denotes Qwen3-VL-Thinking models. PERIA-8B substantially improves over strong open-source and tool/reasoning-augmented baselines, achieving the best overall average among non-commercial models and performance comparable to GPT-5 and Qwen3-235B. We also report ablations: *w/o RL* uses SFT alone, *w/o SFT* trains directly with RL from the base model, and *w/o Tool* removes tool access during training and inference.

SFT establishes tool competence, while the subsequent RL stage adds substantial gains. In Table 2, the backbone-to-SFT comparison (the *w/o RL* checkpoint) increases the six-ID average from 44.1% to 46.9% (+2.8), the OOD average from 31.7% to 34.7% (+3.0), and the overall average from 37.4% to 40.4% (+3.0). Adding the full RL stage then increases the ID average to 54.1% (+7.2), the OOD average to 36.1% (+1.4), and the overall average to 44.4% (+4.0). Thus, the OOD decrease when RL is removed is 1.4 percentage points; 4.0 percentage points is the overall decrease. The *w/o SFT* setting applies RL directly to the base model and answers whether tool use can be bootstrapped without supervised initialization; its 40.7% six-ID average is not an additive estimate of SFT's contribution. Within the matched RL optimizer comparison in Table 3, OR-GIGPO is the strongest tested optimizer, reaching 54.1% versus 51.3% for exact GIGPO, 51.7% for GRPO, and 43.0% for DAPO.

The PERIA tool sandbox and trajectory synthesis are essential for spatial reasoning. The *w/o Tool* ablation in Table 2 shows an imbalanced performance pattern, suggesting that training without tool access can lead to catastrophic forgetting in certain spatial domains and fails to support robust performance across diverse tasks. Meanwhile, the *w/o SFT* variant performs substantially worse than the full PERIA, showing that synthesized tool-use trajectories are critical for teaching reliable tool invocation before RL. The inference-time *Tool-use ablation* in Table 3 further confirms the complementary roles of perception and interaction tools. These results support our toolset design and show that PERIA benefits from both spatially targeted tools and supervised trajectory synthesis.

5.2 Ablation Studies

Observation-relaxed credit assignment improves multi-step tool learning. Table 3 compares OR-GIGPO with original GIGPO, GRPO, and DAPO on in-distribution benchmarks. OR-GIGPO achieves the best average accuracy, outperforming GIGPO by 2.8, GRPO by 2.4, and DAPO by 11.1 percentage points, which supports the value of observation-relaxed step-level advantages for fine-grained credit assignment in multi-step tool use. The *Average Step Adv Ratio* measures the fraction of trajectories with any nonzero step advantage: 9.3% for GIGPO and 62.3% for OR-GIGPO. This indicates that, in the setting of visual tool agents, original GIGPO struggles to match exactly identical observations and therefore largely degrades to GRPO-level behavior, while OR-GIGPO constructs relaxed step-level outcome groups that enable fine-grained advantage assignment. The similarity threshold must remain selective: lowering it from $\delta = 0.9$ to $\delta = 0.5$ further raises the step advantage ratio to 68.5% but reduces the average accuracy to 48.9%, as dissimilar observations are merged into the same outcome group. DAPO performs worse than the baseline, suggesting that its algorithmic design may not be well suited to our visual tool-use setting, where early-stage tool-calling ability is still unstable.

Datasets(→) Model (↓)	ReasonMap Plus	ReasonMap	Visual Probing Easy	Visual Probing Medium	Visual Probing Hard	MapTrace	Avg.
RL algorithm ablation							
Qwen3-8B	65.7	12.3	54.6	31.3	26.6	74.1	44.1
PERIA-8B (ours)							
w/ OR-GIGPO	**72.6**	**27.1**	**61.7**	41.2	**44.3**	**77.7**	**54.1**
w/ GIGPO	70.6	21.8	**61.7**	40.3	42.5	70.6	51.3
w/ GRPO	70.0	24.3	58.8	**41.5**	40.6	75.5	51.7
w/ DAPO	69.2	20.8	43.2	32.4	27.3	65.3	43.0
Model-size ablation							
Qwen3-2B	53.1	8.8	42.8	21.7	23.0	66.2	35.9
PERIA-2B	**58.2**	**16.7**	**43.0**	**26.0**	19.8	**72.2**	**39.3**
Qwen3-4B	64.2	13.7	46.1	25.3	26.4	71.6	41.2
PERIA-4B	**67.1**	**21.4**	**58.1**	**36.1**	**35.8**	**74.5**	**48.8**
Tool-use ablation							
Qwen3-8B Baseline	65.7	12.3	54.6	31.3	26.6	74.1	44.1
PERIA-8B (ours)	**72.6**	**27.1**	**61.7**	**41.2**	**44.3**	77.7	**54.1**
No Tool	67.6	26.6	46.9	23.9	25.5	75.9	44.4
w/ Previous Tool Set	65.8	22.1	58.9	36.2	25.5	77.6	47.6
w/o Perception Tools	46.0	11.2	57.5	34.7	34.9	76.4	43.5
w/o Interaction Tools	69.4	25.0	53.2	26.5	32.1	**78.5**	47.4

Table 3: **OR-GIGPO, model scale, and tool access all affect performance.** OR-GIGPO gives the best 8B result among RL algorithms, and the same recipe improves the 2B and 4B backbones. The full tool sandbox also outperforms no-tool, previous-tool, and tool-family removal settings, showing that perception and interaction tools are complementary. All scores are accuracy percentages.

PERIA's training recipe is effective beyond the 8B model. Table 3 also shows that the proposed training recipe transfers to smaller backbones. Compared with their Qwen3 baselines, PERIA-2B improves from 35.9% to 39.3%, and PERIA-4B improves from 41.2% to 48.8% on average. This indicates the generality of PERIA across model scales.

Both perception and interaction tools are needed for robust spatial reasoning. Table 3 evaluates PERIA-8B under different inference-time tool conditions. Removing all tools reduces the average score to 44.4%, showing that tool access is necessary for robust performance across diverse spatial reasoning domains. Using only the previous tool set reaches 47.6%, clearly below the full sandbox; this previous set contains six tools adopted from existing work: `image_crop`, `bounding_box`, `image_highlight`, `text_ocr`, `auto_segment`, and `grounding_dino`. This gap demonstrates that our PERIA tool design is more effective than generic visual utilities for spatial reasoning. Removing either perception or interaction tools also degrades performance, confirming their complementary roles in evidence acquisition, local verification, and spatial planning.

Additional ablations and analyses. We provide further details and studies in the appendix. Appendix B.1 reports the interaction-cap diagnostic, followed by detailed tool definitions (Appendix C.1) and the cost of thought diversification (Appendix C.2). Appendix D.3 examines the MapTrace evaluation criterion. Appendix E studies trajectory-teacher quality (Appendix E.1) and the rollout group size of OR-GIGPO (Appendix E.2). Appendix F then presents fine-grained tool-use analyses: tool-call distribution (Appendix F.1), zero-shot transfer of the PERIA sandbox to prior agents (Appendix F.2), map-oriented-tool removal and map/non-map balance (Appendix F.3), and tool effectiveness on Cube Three-View (Appendix F.4).

5.3 Qualitative Analysis

Figure 4 illustrates how PERIA combines perception and interaction for subway-map reasoning. Counting intermediate stops between Grand Av Arts/Bunker Hill and Jefferson/USC requires recovering the connecting station sequence and excluding the endpoints. Small labels and intersecting lines make this difficult to verify from the full map alone.

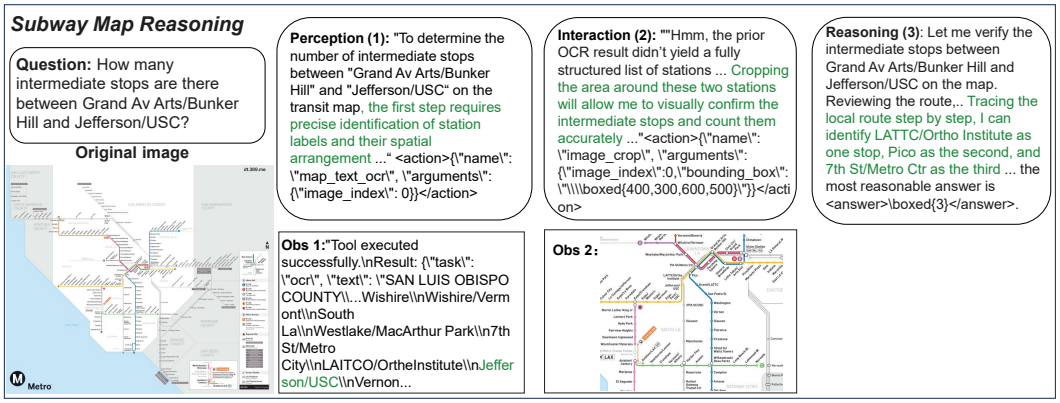

Figure 4: **PERIA grounds route tracing with visual tools.** In this qualitative example, PERIA uses perception tools to identify station labels, interaction tools to verify the relevant region, and accumulated observations to trace the route and answer correctly.

Perception identifies relevant entities; interaction verifies their spatial relationships. PERIA first invokes `map_text_ocr` to extract station labels. The observation contains relevant names but lacks a sufficiently structured sequence for reliable counting. The agent therefore invokes `image_crop` to inspect the queried region and verify the local route. It identifies LATTC/Ortho Institute, Pico, and 7th St/Metro Ctr as the three intermediate stops and returns 3. The tools play complementary roles: perception narrows the search, while interaction resolves local spatial uncertainty. The answer follows from reasoning over both observations rather than accepting the initial OCR output as a complete solution. More examples appear in Appendix G.

6 Conclusion

We presented *PERception-Interaction-reason Agent* (**PERIA**), a tool-augmented visual agent framework for general-purpose spatial reasoning. PERIA integrates perception and interaction tools into a unified Perceive–Interact–Reason process, enabling agents to acquire global spatial evidence, perform fine-grained local verification, and reason over accumulated observations. To train such agents, we introduced supervised tool-use trajectory synthesis and OR-GIGPO, an observation-relaxed policy optimization method for multi-step visual tool use. Experimental results show that PERIA consistently improves spatial reasoning performance, approaches much larger proprietary and open-source models, and benefits from both the designed tool sandbox and the two-stage training recipe. These results suggest that trained visual tool use is a promising direction for building more capable spatial reasoning agents. We discuss limitations and future work in Appendix A.

Broader impact. PERIA co-designs a heterogeneous spatial tool sandbox, executable trajectory supervision, and observation-relaxed credit assignment, making multi-tool spatial reasoning agents trainable. Our findings suggest that tool access alone does not yield tool competence, that executable and verified trajectories are needed to establish it, and that credit assignment should tolerate the surface-form variation of tool observations. We therefore expect future tool-augmented multimodal agents to benefit from designing the tool interface, trajectory supervision, and credit assignment jointly rather than in isolation.

Such agents may support visual assistance, map understanding, robotics, and navigation, but incorrect tool-grounded reasoning may mislead safety-sensitive decisions and the framework could be misused for surveillance; responsible deployment should include task-specific evaluation and human oversight.

Acknowledgments and Disclosure of Funding

This work was supported by research funds from Xiong'an AI Institute.

Beyond the primary affiliations listed above, the authors declare no financial relationships during the 36 months prior to submission that could reasonably be perceived to have influenced this work.

References

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, 2025. URL https://arxiv.org/abs/2511.21631.

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, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2026. URL https://arxiv.org/abs/2511.16719.

Shuaichen Chang, David Palzer, Jialin Li, Eric Fosler-Lussier, and Ningchuan Xiao. Mapqa: A dataset for question answering on choropleth maps, 2022. URL https://arxiv.org/abs/2211.08545.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

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, 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.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. Paddleocr-vl-1.5: Towards a multi-task 0.9b vlm for robust in-the-wild document parsing, 2026. URL https://arxiv.org/abs/2601.21957.

Mahir Labib Dihan, Md Tanvir Hassan, Md Tanvir Parvez, Md Hasebul Hasan, Md Almash Alam, Muhammad Aamir Cheema, Mohammed Eunus Ali, and Md Rizwan Parvez. Mapeval: A map-based evaluation of geo-spatial reasoning in foundation models, 2025. URL https://arxiv.org/abs/2501.00316.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025a. URL https://arxiv.org/abs/2505.10978.

Sicheng Feng, Kaiwen Tuo, Song Wang, Lingdong Kong, Jianke Zhu, and Huan Wang. Rewardmap: Tackling sparse rewards in fine-grained visual reasoning via multi-stage reinforcement learning. *arXiv preprint arXiv:2510.02240*, 2025b.

Sicheng Feng, Song Wang, Shuyi Ouyang, Lingdong Kong, Zikai Song, Jianke Zhu, Huan Wang, and Xinchao Wang. Can mllms guide me home? a benchmark study on fine-grained visual reasoning from transit maps. *arXiv preprint arXiv:2505.18675*, 2025c.

Sicheng Feng, Song Wang, Shuyi Ouyang, Lingdong Kong, Zikai Song, Jianke Zhu, Huan Wang, and Xinchao Wang. Reasonmap: Towards fine-grained visual reasoning from transit maps, 2026. URL https://arxiv.org/abs/2505.18675.

Xingyu Fu, Minqian Liu, Zhengyuan Yang, John Corring, Yijuan Lu, Jianwei Yang, Dan Roth, Dinei Florencio, and Cha Zhang. Refocus: Visual editing as a chain of thought for structured image understanding, 2025. URL https://arxiv.org/abs/2501.05452.

Paul Gavrikov, Wei Lin, M. Jehanzeb Mirza, Soumya Jahagirdar, Muhammad Huzaifa, Sivan Doveh, Serena Yeung-Levy, James Glass, and Hilde Kuehne. Visualoverload: Probing visual understanding of vlms in really dense scenes, 2025. URL https://arxiv.org/abs/2509.25339.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model, 2026. URL https://arxiv.org/abs/2511.05271.

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.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, et al. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search, 2025. URL https://arxiv.org/abs/2509.07969.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024. URL https://arxiv.org/abs/2303.05499.

Yuqi Liu, Tianyuan Qu, Zhisheng Zhong, Bohao Peng, Shu Liu, Bei Yu, and Jiaya Jia. Visionreasoner: Unified reasoning-integrated visual perception via reinforcement learning, 2026. URL https://arxiv.org/abs/2505.12081.

Meng Lu, Ran Xu, Yi Fang, Wenxuan Zhang, Yue Yu, Gaurav Srivastava, Yuchen Zhuang, Mohamed Elhoseiny, Charles Fleming, Carl Yang, Zhengzhong Tu, Yang Xie, Guanghua Xiao, Hanrui Wang, Di Jin, Wenqi Shi, and Xuan Wang. Scaling agentic reinforcement learning for tool-integrated reasoning in vlms, 2025. URL https://arxiv.org/abs/2511.19773.

Wufei Ma, Haoyu Chen, Guofeng Zhang, Celso M de Melo, Alan Yuille, and Jieneng Chen. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. *arXiv preprint arXiv:2412.07825*, 2024.

OpenAI. Openai o3 and o4-mini system card, April 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf. System card.

Artemis Panagopoulou, Aveek Purohit, Achin Kulshrestha, Soroosh Yazdani, and Mohit Goyal. Maptrace: Scalable data generation for route tracing on maps, 2025. URL https://arxiv.org/abs/2512.19609.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Ilias Stogiannidis, Steven McDonagh, and Sotirios A. Tsaftaris. Mind the gap: Benchmarking spatial reasoning in vision-language models, 2025. URL https://arxiv.org/abs/2503.19707.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025a. URL https://arxiv.org/abs/2508.18265.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025b. URL https://arxiv.org/abs/2504.20073.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Jialong Wu, Xiaoying Zhang, Hongyi Yuan, Xiangcheng Zhang, Tianhao Huang, Changjing He, Chaoyi Deng, Renrui Zhang, Youbin Wu, and Mingsheng Long. Visual generation unlocks human-like reasoning through multimodal world models. *arXiv preprint arXiv:2601.19834*, 2026a.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use, 2026b. URL https://arxiv.org/abs/2505.19255.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*, 2023a.

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

Shuo Xing, Zezhou Sun, Shuangyu Xie, Kaiyuan Chen, Yanjia Huang, Yuping Wang, Jiachen Li, Dezhen Song, and Zhengzhong Tu. Can large vision language models read maps like a human?, 2025. URL https://arxiv.org/abs/2503.14607.

Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu, Chun Yuan, and Jian Guo. Chartmoe: Mixture of diversely aligned expert connector for chart understanding, 2025. URL https://arxiv.org/abs/2409.03277.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, Bo Zhang, and Wei Chen. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization, 2025. URL https://arxiv.org/abs/2503.10615.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi, 2024. URL https://arxiv.org/abs/2311.16502.

Zaibin Zhang, Yuhan Wu, Lianjie Jia, Yifan Wang, Zhongbo Zhang, Yijiang Li, Binghao Ran, Fuxi Zhang, Zhuohan Sun, Zhenfei Yin, Lijun Wang, and Huchuan Lu. Think3d: Thinking with space for spatial reasoning, 2026. URL https://arxiv.org/abs/2601.13029.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models, 2024. URL https://arxiv.org/abs/2302.00923.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling, 2025. URL https://arxiv.org/abs/2507.07998.

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.

Appendix Contents

A Limitations and Future Work **16**

B Implementation Details **16**

B.1 Training and Inference 16

B.2 Evaluation Protocol 16

C Method Details **18**

C.1 Tool Sandbox 18

C.2 Trajectory Synthesis 21

C.3 OR-GIGPO 24

C.4 Rewards 24

D Dataset Details **25**

D.1 Training Data 25

D.2 Evaluation Domains 25

D.3 MapTrace Evaluation Criterion Ablation 26

E Additional Experiments **27**

E.1 Teacher Quality 27

E.2 Rollout Group Size 27

F Fine-grained Tool-use Analysis **27**

F.1 Tool-use Distribution 27

F.2 Zero-shot Interface Transfer 28

F.3 Map-oriented Tools and Map/Non-map Balance 28

F.4 Tool Effectiveness on Cube Three-View 29

G Qualitative Examples **29**

A Limitations and Future Work

Although PERIA shows consistent gains across model sizes, our main experiments focus on the 8B model due to computational constraints. We leave larger-backbone validation and a more detailed study of scaling with trajectory data size and tool-use diversity to future work. In addition, we restrict inference to at most 10 interaction turns. This setting covers the tasks studied in this paper, but longer-horizon spatial reasoning may require stronger memory, better exploration, and more stable credit assignment over extended tool-use trajectories.

Future work will extend PERIA to broader spatial reasoning domains, including realistic 3D reasoning, maze navigation, Sokoban-like planning, and embodied or environment-centric tasks that require stronger spatial state tracking. Building larger and more diverse trajectory datasets is also important for exploring the upper bound of PERIA.

B Implementation Details

B.1 Training and Inference

SFT training parameters. We first perform supervised fine-tuning to initialize the model with basic tool-use and spatial reasoning abilities. Table 4 summarizes the main SFT hyperparameters, including the training schedule, optimization settings, and cutoff length.

Parameters	SFT Training
epochs	1.0
global batch size	32
learning rate	1.0×10^{-5}
scheduler	cosine
warmup ratio	0.1
weight decay	0.01
cutoff length	65,536

Table 4: **SFT Parameters.** We report the main optimization and cutoff-length settings used during supervised fine-tuning.

RL training parameters. After SFT, we further optimize the model with reinforcement learning to improve reliable tool use and multi-step spatial reasoning. Table 5 reports the shared RL training and rollout settings, together with the algorithm-specific parameters used by OR-GIGPO and DAPO.

Inference parameters. Table 6 summarizes the inference configuration used to evaluate PERIA.

Inference prompts. We use the inference prompt below to evaluate PERIA. As shown in Figure 5, the prompt defines the available tools, enforces one tool call per action, and requires the model to reason before and after each tool execution. The model is instructed to place intermediate reasoning inside `<think>` and `</think>` tags, issue tool calls using the prescribed `<action>` format, and provide the final answer inside `<answer>` and `</answer>` tags. This protocol encourages the agent to verify tool observations before producing the final response.

Interaction cap. The maximum setting of $T = 11$ consists of at most ten tool-interaction rounds followed by one final-answer round. Profiling the SFT-only checkpoint on a 10% sample of RL-training prompts found that 1.16% of trajectories reached the tenth interaction and 0.81% attempted to continue beyond it and were truncated; 99.19% were not truncated.

B.2 Evaluation Protocol

Evaluation protocol. We adopt a two-stage evaluation protocol for answer-based spatial reasoning tasks. We first apply rule-based matching to compare model predictions with ground-truth answers, which handles exact matches, normalized formatting, and simple equivalent forms. If the rule-based matcher does not mark the prediction as correct, we further use an LLM judge (`GPT-5-Mini`)

Parameters	RL Training
rollout samples per prompt, N	4
train batch size	64
PPO mini-batch size	256
PPO micro-batch size / GPU	4
log-prob micro-batch size / GPU	16
actor learning rate	1.0×10^{-6}
learning rate scheduler	cosine
warmup ratio	0.05
total epochs	3
max prompt length	16,384
max response length	32,768
max action length	4,096
max observation length	8,192
max turns, T	11
sampling temperature	1.0
top_p	1.0
top_k	-1
rollout backend	vLLM
standard clipping coefficient, ϵ_{clip}	0.2
OR-GIGPO-specific parameters	
episode-step advantage weight, ω	1.0
discount factor, γ	0.99
advantage normalization constant, ϵ_{adv}	1.0×10^{-6}
observation similarity threshold, δ	0.9
DAPO-specific parameters	
lower clipping coefficient, ϵ_{low}	0.2
upper clipping coefficient, ϵ_{high}	0.28
total training steps	300

Table 5: **RL Parameters.** We report the rollout configuration, optimization settings, interaction limits, and algorithm-specific parameters for OR-GIGPO and DAPO.

Parameters	Inference
max response length	32,768
max turns	11
temperature	0.0
inference backend	vLLM

Table 6: **Inference Parameters.** Parameters used for PERIA evaluation.

to evaluate whether the prediction meaningfully matches the ground truth. The judge focuses on semantic correctness and ignores minor differences in phrasing or formatting, while allowing equivalent expressions such as unit conversions when they are consistent with the ground truth.

For MapTrace, where the output is a coordinate path rather than a textual answer, we directly evaluate the predicted trajectory against the ground-truth trajectory using normalized dynamic time warping (NDTW). Both paths are represented as normalized coordinates in $[0, 1]$, and the cumulative DTW cost is computed with Euclidean distance between matched points. We regard a prediction as correct if its NDTW score is below the threshold of 1.0. Lower NDTW indicates better alignment between the predicted and ground-truth paths.

Evaluation prompts. For predictions that cannot be verified by rule-based matching, we use the evaluation prompt shown in Figure 6. The prompt provides the image-related question, the ground-truth answer, and the model prediction, and asks the judge to output a binary correctness score. The judge is instructed to consider both relevance and accuracy, allowing paraphrases and formatting differences while rejecting predictions that introduce factual errors.

System Prompt:
You are an advanced AI agent capable of complex reasoning and tool usage. You must strictly adhere to the following protocol for every interaction:
1. Call appropriate tools based on the task;
2. Only persue one tool calling per action;
3. Reasoning Before Action: before selecting a tool, you must analyze the user's request and determine the necessary steps. Output your internal monologue and logic inside <think> and </think> tags;
4. Tool Execution: If a tool is required, generate the tool call immediately after your reasoning.
5. Reasoning After Action: Once you receive the output from a tool, you must analyze the results to determine if further actions are needed or if the task is complete. Output this analysis inside <think> and </think> tags and then decide your next step, which could be calling another tool or providing the final answer.;
6. Final Output: When you have formulated your conclusion, you must wrap your final answer in <answer> and </answer> tags.

Available tools: {tool_definitions_text},
Use this format for tool calls:<action>{{"name": "tool_name", "arguments": {{"param1": "value1"}}}}</action>

User Prompt:
Please answer the following {task_type} question:
Question: {Question}
Please provide a complete step-by-step solution to this problem. Your reasoning should:
1. Analyze the problem systematically
2. Check if the tool execution and answer are correct
3. If there are errors, explain what went wrong and provide the correct reasoning
4. Provide the final answer
Use natural expressions like 'let me think' or 'hmm' when helpful, but keep it concise. It's encouraged to use self-reflection or verification especially in the verifying tool output in the reasoning process.
Provide your detailed reasoning between <think> and </think> tags, then give your final answer between <answer> and </answer> tags.
Output format: {output_format}

Figure 5: **The inference prompt enforces structured tool use and answer formatting.** It defines the tool-use protocol, including tool selection, one tool call per action, pre- and post-action reasoning, observation verification, and the required final-answer format.

C Method Details

C.1 Tool Sandbox

We organize the tool sandbox into two categories: perception tools and interaction tools. Perception tools transform images into structured observations, including text, localized text boxes, object detections, segmentation proposals, counts, and presence signals, powered by three lightweight tool agents: PaddleOCR-VL-1.5 [Cui et al., 2026], Grounding DINO [Liu et al., 2024], and SAM3 [Carion et al., 2026]. The OCR agent is served with vLLM using a maximum context length of 8192 and maximum generation length of 4096; the detection agent uses GroundingDINO base with box/text thresholds of 0.25 and NMS threshold of 0.8; and the segmentation agent uses SAM3.1 with score threshold 0.25 and at most 20 returned proposals. Interaction tools perform deterministic image operations, such as cropping, labeling, drawing, and highlighting, to help the agent inspect and verify spatial evidence during multi-step reasoning.

All tools use `image_index` to specify the input image. For tools requiring spatial coordinates, we use normalized coordinates in a 1000×1000 image space. Bounding boxes follow the format `\boxed{x1,y1,x2,y2}`, where $(x1, y1)$ and $(x2, y2)$ denote the top-left and bottom-right corners.

Evaluation System Prompt:
You are an intelligent chatbot designed for evaluating the correctness of generative outputs for question-answer pairs. Your task is to compare the predicted answer with the correct answer and determine if they match meaningfully.

Instructions: Focus on the meaningful match between the predicted answer and the correct answer, ignoring minor differences in phrasing or formatting. Consider synonyms or paraphrases as valid matches. Evaluate the correctness of the prediction compared to the answer.

Evaluation User Prompt:
I will give you a question related to an image and the following text as inputs:

1. Question Related to the Image: {question}
2. Ground Truth Answer: {ground_truth}
3. Model Predicted Answer: {prediction}

Your task is to evaluate the model's predicted answer against the ground truth answer, based on the context provided by the question related to the image. Consider the following criteria for evaluation: Relevance, whether the predicted answer directly addresses the question posed; and Accuracy, whether the prediction matches the ground truth. If the ground truth answer is open-ended, consider whether the prediction reflects the information given in the ground truth without introducing factual inaccuracies. If the ground truth answer is definitive, strictly compare the model's prediction to the actual answer, while allowing consistent unit conversions such as length and angle.

Examples of correct predictions include: (pred=23.8 billion euros, gt=23.8 billion euros), (pred=60%, gt=60), and (pred="Don't know" (6%)., gt=Don't Know).

Output Format: Your response should include an integer score indicating the correctness of the prediction: 1 for correct and 0 for incorrect. The format should be "Score: 0 or 1".

Figure 6: **The evaluation prompt checks semantic correctness beyond exact matching.** It asks the LLM judge to compare the model prediction with the ground-truth answer for an image-related question and output a binary correctness score. The prompt emphasizes semantic matching, relevance, accuracy, and tolerance to minor formatting or phrasing differences.

Table 7 summarizes the full tool list, including each tool's execution backend, output type, and main parameters.

Perception tools. The perception tools extract structured visual evidence from the image.

- `text_ocr` performs general OCR and extracts visible text from images, such as document text, signs, labels, and natural scene text.
- `text_spotting` detects and recognizes text with localization, returning both text content and bounding boxes. It is useful when the spatial position of text matters, such as station names, road labels, or map annotations.
- `map_text_ocr` extracts map-related text, including place names, road names, station names, and landmarks. It filters common map OCR noise such as pure numbers, scale markers, and isolated symbols.
- `formula_ocr` recognizes mathematical expressions and equations from images and converts them into textual or LaTeX-style outputs.
- `table_ocr` recognizes table structures and extracts rows, columns, and cell contents from structured images.

Tool	Driven by	Output	Main parameters
Visual perception tools			
`text_ocr`	PaddleOCR-VL-1.5 0.9B	Text	`image_index`
`text_spotting`	PaddleOCR-VL-1.5 0.9B	Text	`image_index`
`map_text_ocr`	PaddleOCR-VL-1.5 0.9B	Text	`image_index`
`formula_ocr`	PaddleOCR-VL-1.5 0.9B	Text	`image_index`
`table_ocr`	PaddleOCR-VL-1.5 0.9B	Text	`image_index`
`grounding_dino`	GroundingDINO base	Text	`image_index`, `question`
`auto_segment`	SAM3.1	Text	`image_index`
`bbox_segment`	SAM3.1	Text	`image_index`, `bounding_box`
`text_segment`	SAM3.1	Text	`image_index`, `text_prompt`
`exemplar_segment`	SAM3.1	Text	`image_index`, `bounding_box`
`concept_count`	SAM3.1	Text	`image_index`, `text_prompt`
`presence_check`	SAM3.1	Text	`image_index`, `text_prompt`
Visual interaction tools			
`image_crop`	Function tool	Image	`image_index`, `bounding_box`
`image_label`	Function tool	Image	`image_index`, `text`, `position`
`draw_line`	Function tool	Image	`image_index`, `coordinates`
`draw_path`	Function tool	Image	`image_index`, `points`
`bounding_box`	Function tool	Image	`image_index`, `bounding_box`
`image_highlight`	Function tool	Image	`image_index`, `bounding_box`

Table 7: **Complete tool list in the PERIA Tool Sandbox.** We list all perception and interaction tools, together with their execution backends, output types, and main input parameters.

- `grounding_dino` performs open-vocabulary object detection from text descriptions. It returns detected objects with bounding boxes, confidence scores, and labels, which helps locate landmarks, icons, objects, or semantic regions.
- `auto_segment` automatically segments objects in the full image and returns proposals with bounding boxes, confidence scores, areas, and centroids.
- `bbox_segment` performs region-constrained segmentation inside a specified bounding box, enabling focused analysis or refinement of a target area.
- `text_segment` segments objects described by a natural-language prompt, supporting open-vocabulary object localization and isolation.
- `exemplar_segment` uses a bounding box as a visual exemplar and finds visually similar objects in the image, which is useful for repeated patterns or "find more like this" reasoning.
- `concept_count` counts objects matching a text prompt and returns both the count and instance locations.
- `presence_check` checks whether a described concept is present in the image and returns a presence flag, confidence score, and count.

Interaction tools. The interaction tools return modified images and support active visual inspection and verification.

- `image_crop` crops a specified region for fine-grained inspection, such as verifying small text, landmarks, or local route details.
- `image_label` adds a text label at a specified position, helping annotate landmarks, objects, or intermediate reasoning results.
- `draw_line` draws a single line between two points, which is useful for marking road segments, boundaries, crossings, or direct spatial relations.
- `draw_path` draws a connected path through multiple waypoints, supporting route tracing, maze solving, and multi-step path verification.
- `bounding_box` draws a bounding box around a target region, providing explicit localization of objects, landmarks, or regions of interest.

- `image_highlight` highlights a specified region with a translucent overlay, helping focus attention on relevant areas while preserving surrounding context.

C.2 Trajectory Synthesis

TOOL_SELECTION_PROMPT (For $round = 1$):
You are an AI agent that selects tools for visual analysis tasks.

Based on the question and observations, select ONE tool and explain why.

STRICT OUTPUT FORMAT (you MUST follow this exactly):
<think>
Tool: [tool_name]
Reason: [1 paragraph explaining why this tool is appropriate]
</think>

Rules:
1. You MUST select a tool - NEVER answer the question directly
2. Only output the tool NAME - do NOT include parameters
3. Select only ONE tool per turn
4. Do NOT output anything outside the <think></think> tags

TOOL_CALL_ONLY_PROMPT:
You are an advanced AI agent. In this phase, you must execute the tool call based on previous reasoning.

Instructions:
1. Based on the reasoning provided, execute the tool call immediately
2. Make sure the tool call is consistent with the decision in the reasoning phase.
2. Do NOT provide any final answer in this phase
3. Do NOT provide any text explanation, just call the tool with the parameters specified in the reasoning
4. Never require a tool to directly solve the problem, but rather to analyze the problem and provide more information for you to solve the problem.

Figure 7: **Round-1 prompts force exploration before answering.** At round 1, `TOOL_SELECTION_PROMPT` forces the model to select exactly one tool and explain the reason inside `<think>` tags, while `TOOL_CALL_ONLY_PROMPT` executes the selected tool with concrete parameters and forbids final answers.

Data synthesis procedure. We synthesize tool-use trajectories with an iterative explore-and-exploit procedure. For each example, we initialize the task with the image, question, ground-truth answer, dataset-specific metadata, and the available tool sandbox. We set the number of sampled trajectories per round to $n_{\text{sample}} = 3$ and the maximum number of rounds to $T = 11$, including at most 10 tool-use rounds and one final-answer round. Rather than asking the model to answer directly, we decompose trajectory generation into tool selection, tool execution, observation gathering, and final answering.

At round 1, the model is forced to explore by selecting one tool for visual analysis and then executing the corresponding tool call. For rounds 2 to $T - 1$, the model exploits accumulated observations to decide whether the current evidence is sufficient for answering or whether another tool call is needed. If the model produces an answer before the final round, we validate it before committing it to the trajectory. Correct answers are saved as valid trajectories, while rejected answers are removed from the conversation history and trigger a reflection step that forces the model to gather additional evidence with a new or more informative tool. At round T, if no valid trajectory has been found, the model is prompted to reflect on all observations and provide a final answer.

The notation $y_t^{(i)} = y^*$ in the main text denotes correctness under the task-specific verifier. We validate synthesized trajectories using the same task-specific criteria as our evaluation protocol in Appendix B.2, including rule-based and LLM-judge checks for answer-based tasks and NDTW-based verification for MapTrace. This process yields executable trajectories with explicit tool choices, grounded observations, self-reflection after rejected answers, and verified final responses.

Synthesis prompts. Figures 7 and 8 show the prompts used for iterative trajectory synthesis. At round 1, we use `TOOL_SELECTION_PROMPT` and `TOOL_CALL_ONLY_PROMPT` to separate tool selection from tool execution: the model first selects exactly one tool in the reasoning phase, and then executes the selected tool with concrete parameters without producing a final answer. For rounds > 1, we use `TOOL_SELECTION_OR_ANSWER_PROMPT` when no previous answer in the current trajectory has been rejected, allowing the model to either call another tool or provide a final answer based on accumulated observations. If an attempted answer is rejected by the verifier, we use `EXTEND_TOOL_SELECTION_PROMPT` in the following retry to force self-reflection and additional tool use. At the maximum round, `FINAL_ANSWER_PROMPT` asks the model to reflect on all observations and produce the final answer.

Diversification procedure and prompts. After collecting valid trajectories, we diversify only the intermediate reasoning text while keeping the tool calls, tool parameters, observations, and final

TOOL_SELECTION_OR_ANSWER_PROMPT (For $round > 1$ and have not provided wrong answer in the $round$):
You are an AI agent selecting tools for visual analysis.

Review ALL observations gathered so far. Start your reasoning with a brief
reflection on what you have learned, using connective phrases such as:
- "Ok, I have [what you just did]. Now I need to..."
- "Wait, the result shows [observation]. Let me check..."
- "Good, this confirms [finding]. Next I should..."
- "Hmm, this is not what I expected. I need to..."

Then decide your next action:

If you need more information, select a tool:
<think>
[Your reflection with connective phrase]
Tool: [tool_name]
Reason: [what NEW information this tool will provide]
</think>

If you have gathered CONCRETE evidence and are confident in your answer:
<think>[Your reflection and final reasoning]</think>
<answer>[your answer]</answer>

CRITICAL RULES:
- NEVER respond with "I cannot determine", "I'm unable to verify", or ask the user for clarification. You must work with the image and tools available.
- If previous tool results were unclear or insufficient, select a DIFFERENT tool or use the same tool with different parameters to gather more information.
- Only provide an answer when you have specific, concrete findings from tool results — not when you are uncertain or giving up.
- PREFER using tools to verify conclusions before answering.

EXTEND_TOOL_SELECTION_PROMPT (For $round > 1$ and have provided wrong answer in the $round$):
You are an AI agent selecting tools for visual analysis.

IMPORTANT: Your previous analysis led to an INCORRECT answer. You need to gather MORE information by calling additional tools.

Before selecting a tool, use a self-reflection phrase such as:
- "Wait, I think my previous analysis might be incomplete..."
- "Is it true that [previous assumption]? Let me check..."
- "Current information is not sufficient to determine..."
- "Hmm, this seems inconsistent with... I need to investigate..."
- "Perhaps I should verify [specific aspect]..."
- Or similar expressions that show critical thinking

Your task:
1. Reflect on what might have gone wrong
2. Identify what additional information could help
3. You have used: {used_tools}. Explore new tools for a DIFFERENT perspective

OUTPUT FORMAT:
<think>
[Your self-reflection - choose or generate a phrase similar to the examples above]
Tool: [tool_name]
Reason: [what NEW information this tool will provide to correct the analysis]
</think>

CRITICAL: You MUST call a tool - do NOT provide an answer yet.

FINAL_ANSWER_PROMPT (For $round = max_round$):
You are an advanced AI agent. In this phase, you must analyze all observations and tool results to provide the final answer.

Instructions:
1. Based on all the observations and tool results gathered, analyze them carefully and provide the final answer
2. Wrap your analysis in <think> and </think> tags.
3. Wrap your final answer in <answer> and </answer> tags
4. Be concise and direct in your answer

Figure 8: **Later-round prompts balance further tool use with final answering.** For rounds > 1, `TOOL_SELECTION_OR_ANSWER_PROMPT` lets the model choose between further tool use and final answering; after a rejected answer, `EXTEND_TOOL_SELECTION_PROMPT` forces self-reflection and additional tool use; at the maximum round, `FINAL_ANSWER_PROMPT` requires a final answer.

answers unchanged. As shown in Figure 9, `DIVERSIFY_SYSTEM_PROMPT` defines the rephrasing task and asks the model to preserve the core meaning and technical information. We then select one of three user prompts according to the role of the `<think>` block: `DIVERSIFY_PROMPT_FOR_FIRST_TOOL` rewrites the first tool-selection reasoning, `DIVERSIFY_PROMPT_FOR_SUBSEQUENT_TOOL` rewrites later tool-selection reasoning with conversation context and self-reflection over previous tool results, and `DIVERSIFY_PROMPT_FOR_FINAL_REASONING` rewrites the final reasoning and answer block by reviewing all tool calls and preserving all factual data and conclusions. The rewritten text is inserted back into the original `<think>` tags, while outputs containing forbidden tags or missing the required tool name are rejected.

Diversification cost. The synthesis set contains 7,661 verified trajectories and 11,767 intermediate `<think>` blocks. Rewriting these blocks with Qwen3-VL-235B-A22B-Thinking using TP=8 on one 8×H200 node processes approximately 30M tokens, averages about 150 output tokens per rewrite, and completes in well under eight hours. Diversification changes only intermediate reasoning text while preserving tool calls, observations, and final answers.

DIVERSIFY_SYSTEM_PROMPT:
You are a text rephrasing assistant. Rephrase the given reasoning text while preserving its core meaning and key technical information. Wrap your final rephrased text in <answer>...</answer> tags.

DIVERSIFY_PROMPT_FOR_FINAL_REASONING:
Below is the full conversation history of a geometric analysis model solving \
a problem, followed by its final reasoning block that synthesizes all results.

Full conversation:
{context}

Final reasoning block to rewrite:
{text}

Requirements:
- Review each tool call and its result: was it necessary? Was the result \
correct and reliable? Did it contribute useful information?
- If any tool result appears wrong or unhelpful, explicitly reason about \
how to correct for it and derive the right answer despite the error
- Use self-reflection to verify the logical chain from observations to conclusion
- Preserve ALL factual data, numbers, measurements, and the final conclusion EXACTLY
- Use natural self-reflection phrases (e.g. "Let me verify this makes sense...", \
"Looking back at the results...", "I need to reconsider whether...")
- Change sentence structure and word choice compared to the original
- Do NOT include <think>, </think>, or <action> tags inside <answer>
- Wrap your final rewritten text in <answer>...</answer> tags

DIVERSIFY_PROMPT_FOR_FIRST_TOOL:
Rephrase the following reasoning text where a model decides to use the tool \
"{tool_name}" as its first analysis step.

Requirements:
- Keep the tool name "{tool_name}" exactly as-is in the output
- Vary how and why this tool is chosen as the first step
- Change sentence structure, word choice, and perspective
- Do NOT include <think>, </think>, or <action> tags inside <answer>
- Wrap your final rephrased text in <answer>...</answer> tags

Original text:
{text}

DIVERSIFY_ PROMPT_FOR_SUBSEQUENT_TOOL:
Below is the conversation history of a geometric analysis model solving a problem, \
followed by one of its intermediate reasoning blocks where it decides to call \
the tool "{tool_name}" after analyzing previous results.

Conversation so far:
{context}

Reasoning block to rewrite:
{text}

Requirements:
- Start with a critical analysis of the previous tool call's result: \
is it correct, reasonable, and useful for solving the problem?
- If the result seems questionable, note what might be off and how to account for it
- Then explain why the current information is insufficient and why \
calling "{tool_name}" next is the right decision
- Keep the tool name "{tool_name}" exactly as-is in the output
- Use natural self-reflection phrases (e.g. "Wait, let me verify...", \
"Hmm, this result suggests...", "I should double-check...")
- Change sentence structure and word choice compared to the original
- Do NOT include <think>, </think>, or <action> tags inside <answer>
- Wrap your final rewritten text in <answer>...</answer> tags

Figure 9: **Diversification rewrites reasoning without changing tool behavior.** The prompts rewrite reasoning traces for first-tool selection, subsequent tool selection, and final reasoning while keeping tool calls, observations, and answers fixed.

C.3 OR-GIGPO

Observation-relaxed grouping. OR-GIGPO keeps the two-level credit assignment structure described in the main text, but groups actions by their resulting tool observations within the same task rather than by exact pre-action anchor states. Semantically equivalent tool calls may produce slightly different outputs, such as reordered OCR text, small detection variations, or different natural-language descriptions from tool agents.

Implementation details. For each tool-use turn, we collect the hash and optional text of the observation returned after executing its action, together with the generated token boundary and discounted return. We first group turns by the pair of prompt index and observation hash, ensuring that observations are only compared among rollouts from the same multimodal input. This exact-hash grouping is applied to all tool observations. For image-returning tools, exact matching is usually sufficient because the returned image is deterministic when the tool command and arguments are the same. For textual observations returned by tool agents, however, exact matching can be too strict because the same tool behavior may produce minor differences in wording, ordering, or localization descriptions. Within each prompt, we further merge exact-hash groups whose representative texts have `SequenceMatcher` similarity above $\delta = 0.9$. `SequenceMatcher` computes a normalized sequence-matching ratio based on the total length of matched subsequences; for two observation texts a and b, we use $\mathrm{sim}(a, b) = 2M/(|a| + |b|)$, where M is the total number of matched elements. Step-level advantages are computed only for groups with more than one entry, assigned back to the corresponding token spans, and combined with the episode-level advantage using $\omega = 1.0$ and $\gamma = 0.99$.

C.4 Rewards

For each rollout τ, we assign the final reward to the last valid response token. The trajectory-level reward is defined as:

$$R(\tau) = R_{\mathrm{rep}}(\tau) + R_{\mathrm{format}}(\tau) + R_{\mathrm{correct}}(\tau).$$

Here R_{rep} penalizes degenerate repetition, R_{format} encourages valid tool-use trajectories, and R_{correct} measures final-answer correctness.

Repetition penalty. We compute the repetition penalty on the model-generated text after removing user turns. The penalty is zero for normal responses and becomes negative when repeated patterns are detected:

- -3.0 if a character is repeated at least 50 times;
- -3.0 if a word is repeated at least 20 times;
- -2.0 if a text span of at least 4 words is repeated at least 10 times;
- $-1.5 \cdot \frac{2}{T}$ if a sentence is repeated at least 10 times;
- $-1.0 \cdot \frac{2}{T}$ if a sentence is repeated at least 7 times;
- -1.5 if a word is repeated at least 10 times;
- 0.0 otherwise.

where $T = \max(\text{number of turns}, 2)$. This turn-normalized design avoids over-penalizing longer multi-turn trajectories while discouraging repeated reasoning or tool-use loops.

Format reward. The format reward checks whether the trajectory follows the required tool-use protocol:

- $+1.0$ if the response contains valid reasoning and a final answer;
- 0.0 if the trajectory contains tool actions but is truncated before the final answer;
- -1.0 otherwise.

A valid trajectory should contain reasoning indicated by `</think>` and a final answer wrapped by `<answer>...</answer>`. If the response contains `<action>...</action>` blocks, each action must be preceded by a reasoning segment.

Correctness reward. For general spatial QA tasks, R_{correct} follows the evaluation protocol in Appendix B.2. We first apply rule-based answer matching and then use the LLM-judge fallback described in the evaluation protocol when rule-based matching marks a non-MapTrace prediction as incorrect. For non-MapTrace tasks, the final correctness reward is binary:

$$R_{\text{correct}} = 1.0 \quad \text{if the prediction is judged correct, and } 0.0 \text{ otherwise.}$$

MapTrace reward. For MapTrace, the R_{correct} uses a continuous NDTW-based score instead of the binary success criterion used in evaluation. Given the NDTW distance d_{NDTW}, we compute:

$$R_{\text{correct}}^{\text{MapTrace}} = \begin{cases} 1.0, & d_{\text{NDTW}} \le d_{\text{low}}, \\ \dfrac{d_{\text{high}} - d_{\text{NDTW}}}{d_{\text{high}} - d_{\text{low}}}, & d_{\text{low}} < d_{\text{NDTW}} < d_{\text{high}}, \\ 0.0, & d_{\text{NDTW}} \ge d_{\text{high}}. \end{cases}$$

We set $d_{\text{low}} = 0.3$ and $d_{\text{high}} = 0.8$ by default. If the predicted path cannot be parsed or the verifier marks it as invalid, we assign $R_{\text{correct}}^{\text{MapTrace}} = 0.0$.

D Dataset Details

D.1 Training Data

Table 8 summarizes the data composition used in the SFT and RL stages. The main evaluation uses the complete test set of each benchmark; sample-based diagnostics are explicitly identified in their respective sections.

Data source	SFT samples	RL samples
MapTrace	3,698	2,000
ReasonMap-Plus	1,979	1,580
MM-MapQA	1,716	2,000
ReasonMap	268	572
Mini-o3-Coldstart-Dataset	3,000	–
DeepEyes	–	1,500
Visual Probing	–	1,000
Total	**10,661**	**8,652**

Table 8: **Training combines map reasoning, visual probing, and image-thinking data.** We report the data sources and number of samples used in each training stage.

D.2 Evaluation Domains

We group the evaluation datasets by task domain to clarify the types of spatial reasoning in our experiments.

Map-based reasoning. This group evaluates spatial reasoning over map-like images, where models must connect text labels, landmarks, routes, and global topology. **MapTrace** focuses on route tracing and requires the model to output coordinate paths on maps [Panagopoulou et al., 2025]. **ReasonMap** evaluates subway-map reasoning over station connectivity, transfer relations, and route topology [Feng et al., 2025c]. **ReasonMap-Plus** further expands map reasoning with diverse question types such as counting, true/false verification, and route-related questions [Feng et al., 2025b]. **MapEval** evaluates geospatial map understanding across map-based visual questions [Dihan et al., 2025].

Visual probing and visual search. This group evaluates whether models can actively inspect visual evidence and locate task-relevant regions. **Visual Probing** contains visual search and probing tasks that require locating relevant evidence through regional inspection [Lai et al., 2025]. **V*** focuses on visual search and fine-grained visual reasoning, where the answer often depends on identifying small or localized visual details [Wu and Xie, 2023a].

Physical and geometric spatial reasoning. This group evaluates spatial transformation, motion, and 3D reasoning beyond map-like domains. **Ball Tracking** tests reasoning over object motion and trajectory changes [Wu et al., 2026a]. **Paper Folding** requires geometric transformation reasoning over folded or unfolded 2D patterns [Wu et al., 2026a]. **Cube Three-View Reasoning** evaluates 3D structure understanding from multiple 2D views [Wu et al., 2026a]. **Real-world Spatial Reasoning** covers spatial relations in natural scenes, requiring models to reason about layouts, object positions, and relative directions [Wu et al., 2026a].

Basic visual primitives. **BabyVision** evaluates low-level visual reasoning abilities inspired by early human visual development, including visual tracking, spatial perception, fine-grained discrimination, and pattern understanding [Chen et al., 2026]. It tests whether models possess robust visual primitives rather than relying only on high-level semantic or language priors.

D.3 MapTrace Evaluation Criterion Ablation

Methods (↓)	**MapTrace Thr. 1.0** (↑)	**MapTrace Thr. 0.5** (↑)	**MapTrace Thr. 0.3** (↑)	**Avg. NDTW** (↓)
Commercial Models				
GPT-5	**82.2**	**61.4**	41.8	**0.720**
Gemini-2.5-Flash	66.4	40.6	21.1	29.9
Gemini-2.5-Pro	<u>80.4</u>	<u>60.2</u>	<u>42.5</u>	3.51
Open-Source VLMs and Tool/Reason-Augmented Baselines				
InternVL3.5-8B	70.8	48.7	31.0	5.92
InternVL3.5-14B	76.2	55.8	37.1	3.27
Qwen3-30B-A3B	75.0	55.0	39.1	0.796
Qwen3-32B	75.5	53.7	35.5	0.811
Qwen3-235B	72.9	53.5	37.5	0.827
VTool-R1-3B	40.6	22.8	13.2	2.14
VTool-R1-7B	45.2	29.8	19.4	28.3
R1-OneVision-7B	41.9	22.9	13.3	1012
Mini-o3	30.4	12.8	6.2	3.33
Our Method				
Qwen3-2B	66.2	45.8	30.7	3.56
PERIA-2B	72.2$_{+6.0}$	56.2$_{+10.4}$	38.4$_{+7.7}$	0.812$_{-2.74}$
Qwen3-4B	71.6	49.9	34.0	0.902
PERIA-4B	74.5$_{+2.9}$	57.9$_{+8.0}$	40.7$_{+6.7}$	0.818$_{-0.084}$
Qwen3-8B	74.1	57.5	38.4	0.822
PERIA-8B (ours)	77.7$_{+3.6}$	59.2$_{+1.7}$	**44.1**$_{+5.7}$	<u>0.778</u>$_{-0.044}$

Table 9: **MapTrace evaluation under different NDTW thresholds.** We report threshold-based accuracy under three NDTW thresholds and the average NDTW distance. Higher is better for threshold-based accuracy, while lower is better for Avg. NDTW. Accuracy gains are percentage points; NDTW changes are distance differences. PERIA-8B improves over the Qwen3-8B backbone across all threshold settings and achieves the best accuracy among all models under the strictest threshold. Bold and underlined values indicate the best and second-best results across all listed models, respectively.

We use the NDTW path distance from MapTrace [Panagopoulou et al., 2025], but summarize it as threshold-based accuracy rather than Avg. NDTW: for a threshold τ, we report the fraction of test examples with distance below τ. The main results use $\tau = 1.0$, while Table 9 also reports $\tau = 0.5$ and 0.3. As shown in Table 9, Avg. NDTW is sensitive to extreme values and may not reliably reflect the capability of different models. We therefore also report model performance under stricter thresholds. PERIA-8B achieves the highest threshold-based accuracy among non-commercial models at all three

thresholds, with a larger gain over its backbone under the strictest criterion. At the threshold of 0.3, PERIA-8B achieves the best result among all models.

E Additional Experiments

This appendix reports additional experiments on the training recipe of PERIA. Appendix E.1 examines how the quality of the trajectory teacher affects SFT, and Appendix E.2 studies the rollout group size of OR-GIGPO.

E.1 Teacher Quality

We replaced the GPT-5 trajectory teacher with the open-weight Qwen3.5-397B-A17B teacher while keeping the diversification model, SFT budget, and evaluation protocol fixed. Each SFT-only checkpoint was evaluated once on the same stratified 10% sample from all 13 benchmarks. Table 10 shows that the open-weight teacher already improves over the no-SFT checkpoint, while GPT-5 provides a further gain; this is a diagnostic of teacher quality rather than a replacement for the full-test-set main comparison.

SFT setting	**Average accuracy**
Qwen3-VL-8B, no SFT	34.1
Qwen3.5-397B-A17B teacher	37.6
GPT-5 teacher	**39.8**

Table 10: Teacher-quality diagnostic on the same stratified 10% sample from all 13 benchmarks. The open-weight teacher improves over the no-SFT checkpoint, while GPT-5 provides a further gain in this diagnostic.

E.2 Rollout Group Size

For the rollout-group-size diagnostic, we evaluated Qwen3-VL-2B-Thinking with a fixed 10% sample from each of the six in-distribution benchmarks, shared by all settings in this comparison. As shown in Table 11, increasing the group size from $N = 4$ to $N = 8$ yields a further improvement within the tested range.

Setting	**Average accuracy**	**Gain vs. backbone (pp)**
Qwen3-VL-2B backbone	22.6	–
OR-GIGPO ($N = 4$)	36.9	+14.3
OR-GIGPO ($N = 8$)	**38.2**	**+15.6**

Table 11: Completed rollout-group-size diagnostic on a fixed 10% sample from each of the six in-distribution benchmarks. Within this tested range, $N = 8$ adds 1.3 percentage points over $N = 4$.

F Fine-grained Tool-use Analysis

This appendix analyzes how PERIA uses its tools. Appendix F.1 reports the tool-call distribution under different tool conditions, Appendix F.2 evaluates whether prior agents benefit from the PERIA sandbox without retraining, Appendix F.3 examines the role of map-oriented tools and compares map and non-map benchmarks, and Appendix F.4 analyzes tool effectiveness on Cube Three-View.

F.1 Tool-use Distribution

We further analyze how PERIA-8B uses different tools under the inference-time tool conditions in Table 3. Figure 10 reports the normalized tool-call composition for the full sandbox, the previous-tool setting, the setting without perception tools, and the setting without interaction tools. With the full 18-tool sandbox, PERIA-8B primarily uses `draw_path` (51.0%), `text_spotting` (19.4%), `map_text_ocr` (17.2%), and `image_crop` (9.7%), while hallucinated tool calls remain rare (0.2%).

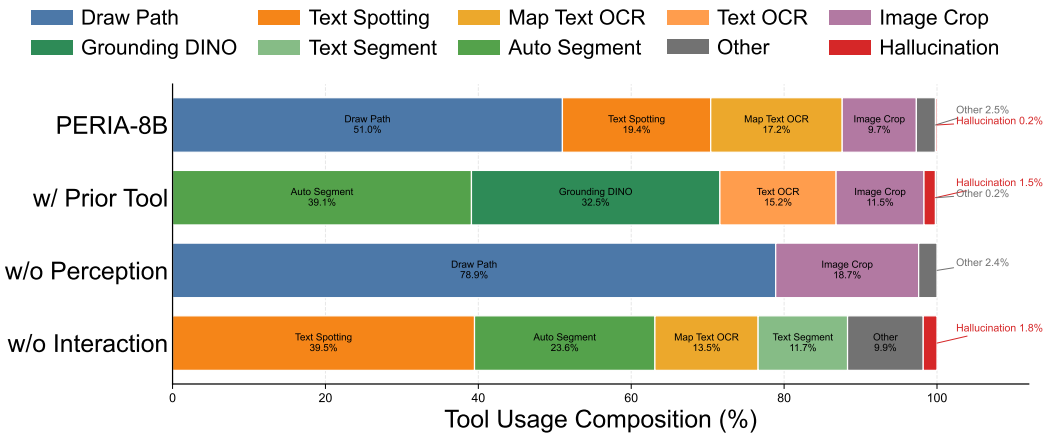

Figure 10: **Tool-use distribution under different inference tool conditions.** PERIA-8B adapts to different available tool sets by using diverse tools. Compared with the previous-tool setting, the full sandbox shows a substantially different usage pattern, suggesting that newly designed tools are actively used for reasoning.

This distribution shows that the trained agent actively combines interaction tools for path-level reasoning with perception tools for extracting textual and spatial evidence.

Under the previous-tool setting, where only generic tools from prior work are available, the usage pattern changes substantially: the model relies mainly on `auto_segment` (39.1%), `grounding_dino` (32.5%), `text_ocr` (15.2%), and `image_crop` (11.5%). This shift suggests that the full PERIA sandbox does not merely increase the number of available tools, but introduces spatially targeted tools that are preferred by the trained agent and lead to stronger performance. In particular, the frequent use of `draw_path`, `text_spotting`, and `map_text_ocr` in the full setting aligns with the needs of route tracing, map reasoning, and fine-grained spatial verification.

The constrained settings further reveal the complementary roles of the two tool families. When perception tools are removed, the model reallocates calls to interaction tools, especially `draw_path` (78.9%) and `image_crop` (18.7%). When interaction tools are removed, it instead relies on perception tools such as `text_spotting` (39.5%), `auto_segment` (23.6%), `map_text_ocr` (13.5%), and `text_segment` (11.7%). These reallocations show that PERIA-8B can adapt to different tool availability, but the corresponding performance drops in Table 3 indicate that either tool family alone is insufficient. We also observe slightly higher hallucinated tool-call rates under constrained settings, especially when interaction tools are removed, suggesting that restricting the sandbox makes tool selection harder. Overall, the analysis supports the effectiveness of our tool design and the necessity of coordinating perception and interaction tools for robust spatial reasoning.

F.2 Zero-shot Interface Transfer

To test whether access to the PERIA sandbox alone explains the performance gap, we evaluated released checkpoints with the PERIA inference prompt and sandbox without retraining. All models were evaluated on the same stratified 10% sample from all 13 benchmarks. The native column uses each checkpoint's original interface; the PERIA-sandbox column uses the transferred interface. Table 12 shows that sandbox access alone does not reproduce the gains of PERIA, indicating that tool competence must be learned.

F.3 Map-oriented Tools and Map/Non-map Balance

We evaluated the trained checkpoint with both `map_text_ocr` and `draw_path` removed at inference time on the complete test sets of ReasonMap-Plus, ReasonMap, and MapTrace. This is a full-test-set diagnostic for these three map benchmarks only. As shown in Table 13, the restricted model remains above the backbone on all three tasks.

Model	Native	PERIA sandbox
VTool-R1-7B	34.5	35.0
R1-OneVision-7B	27.4	29.4
Mini-o3	30.7	13.1
PERIA-8B	–	**51.7**

Table 12: Zero-shot interface transfer on a common stratified 10% sample from all 13 benchmarks. Sandbox access alone gives small changes for the prior agents and substantially reduces Mini-o3 performance; the transferred values do not constitute retrained baselines.

Setting	ReasonMap-Plus	ReasonMap	MapTrace	Avg.
Qwen3-VL-8B	65.7	12.3	74.1	50.7
PERIA-8B, all tools	72.6	**27.1**	**77.7**	**59.1**
PERIA-8B, w/o map tools	**73.4**	18.3	74.7	55.5

Table 13: Map-oriented-tool removal on complete test sets of three map benchmarks. The restricted model remains above the backbone on all three tasks, while losing 3.7 percentage points before rounding relative to the full-tool setting.

Across the four map-related benchmarks, the reported average increases from 51.5% for the backbone to 58.2% for PERIA, a gain of 6.7 percentage points. Across the nine non-map benchmarks, the corresponding average increases from 31.2% to 38.3%, a gain of 7.1 percentage points. These aggregate comparisons use the main Table 2 values; they summarize benchmark groups and do not introduce a new sampling protocol.

F.4 Tool Effectiveness on Cube Three-View

On a 10% sample of the held-out Cube Three-View test set, PERIA-8B obtains 35.4% with tools enabled and 22.9% with direct chain-of-thought without tools. At least one tool is invoked on 75% of examples, most often `text_spotting`. The tool-enabled/direct-CoT comparison is sample-based. Separately, excluding Cube Three-View from the main Table 2 leaves a 2.6-percentage-point average improvement over the backbone across the remaining six OOD benchmarks.

G Qualitative Examples

Example: From view identification to spatial reasoning. Figure 11 shows an example that requires reasoning over multiple cube views. The question asks how many red cubes can possibly be seen from the right view. The model first uses text spotting to identify the three view labels: "front_left", "left", and "front". These labels help ground the diagrams, but they do not directly answer the visibility question.

To obtain the needed visual evidence, the model crops the front-left region and inspects the 3D cube structure. Based on the relative positions of the red and gray cubes, it infers that only one red cube remains visible from the right view after occlusion. The final answer is `A. 1`. This example shows that view labels are useful for grounding, while the final answer still requires localized spatial reasoning.

Example: Recovering from incomplete OCR evidence. Figure 12 shows an example where the initial OCR output is correct but incomplete for the question. The question asks for the phone number at the bottom of a white billboard. The model first applies OCR to the full image, which returns visible storefront text such as "PEINTRE", "LE DANAY", and "Visuel". However, the returned text does not include the queried phone number.

The model therefore checks the sufficiency of the OCR evidence and crops the right side of the image where the billboard appears. After this localized inspection, the phone number becomes visible, leading to the answer `(+237) 650 58 49 04`. This example shows that a tool output can be useful but non-exhaustive, and that additional localization can recover missing task-relevant evidence.

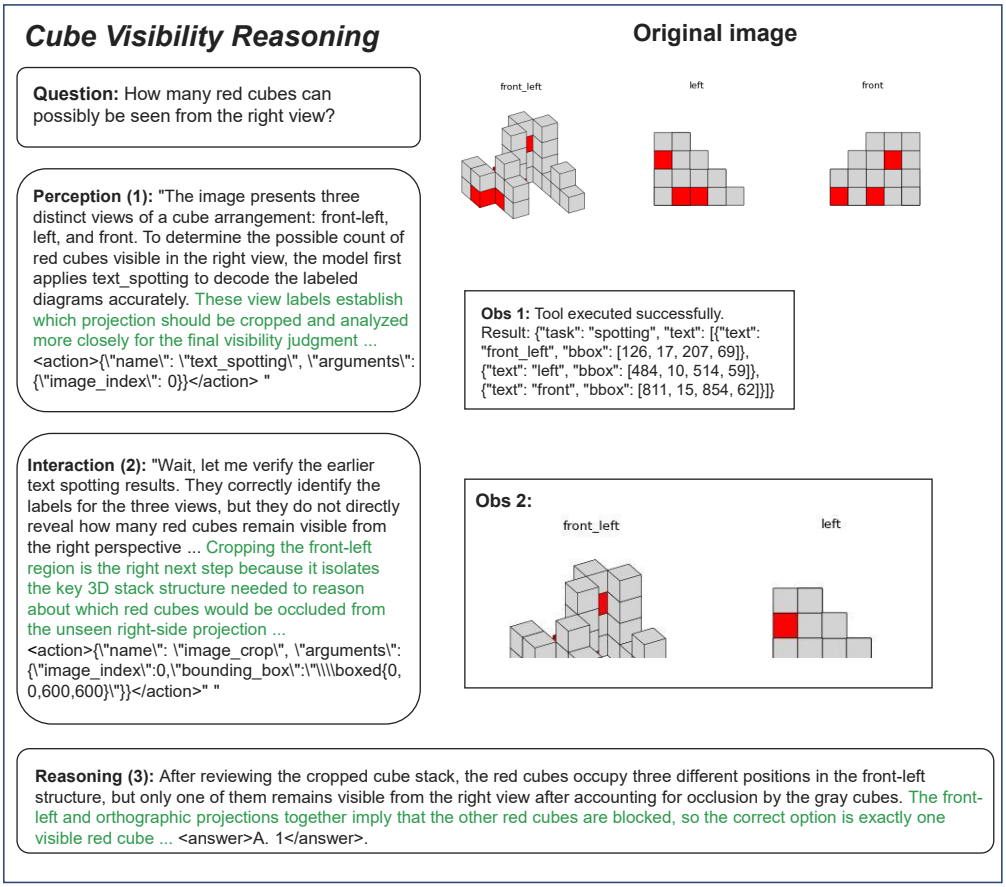

Figure 11: Qualitative example of cube visibility reasoning. Text spotting identifies the view labels, while the final answer requires cropping the cube arrangement and reasoning about 3D occlusion.

Together, these examples show that intermediate tool outputs should be checked for sufficiency, and that perception and interaction tools play complementary roles. Perception tools provide global cues such as text labels and OCR evidence, while interaction tools enable localized inspection through cropping or other visual operations. When the returned evidence is incomplete or only partially relevant, combining these two tool families helps the agent obtain the information needed for the final answer.

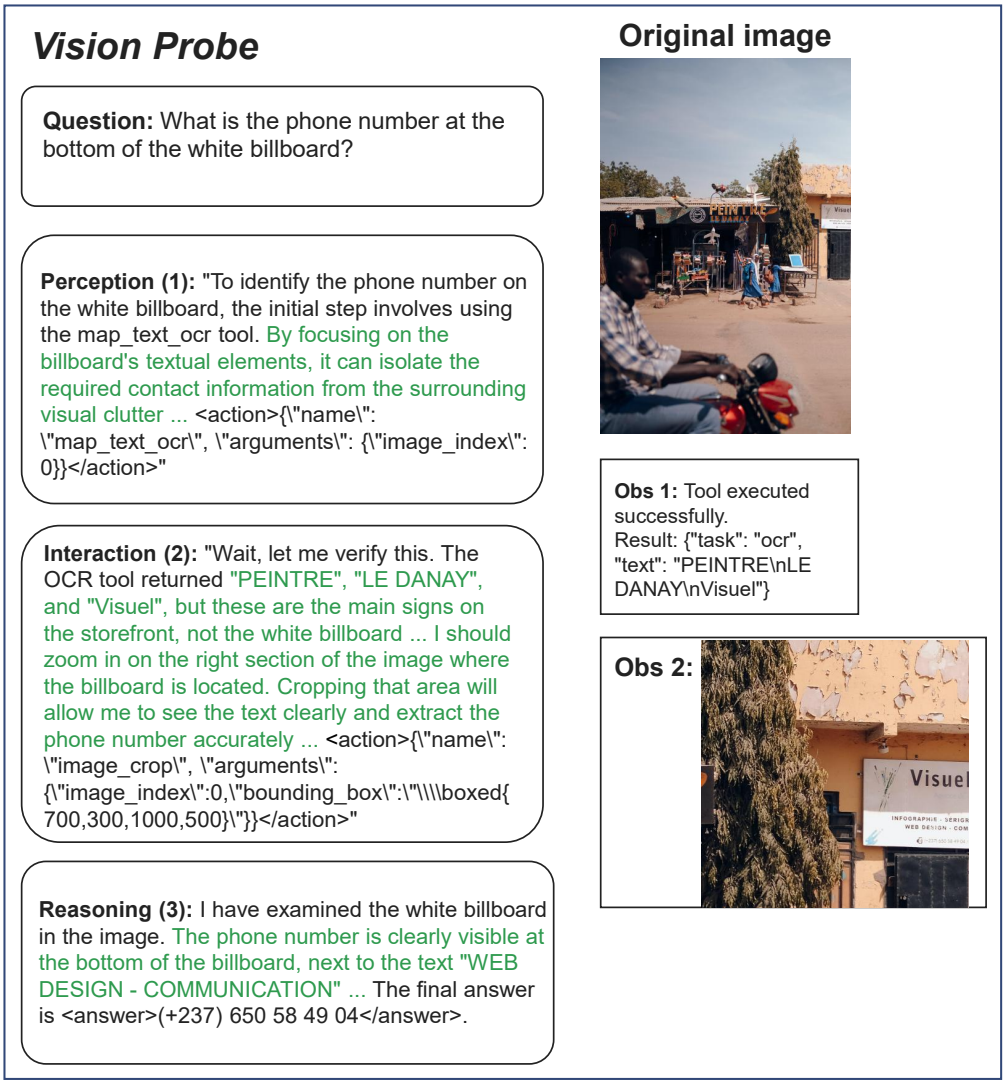

Figure 12: Qualitative example of iterative vision probing. The OCR output contains visible storefront text but omits the phone number required by the question. Cropping the relevant billboard region provides the missing evidence.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction focus on the PERIA framework, perception–interaction tool sandbox, trajectory synthesis, OR-GIGPO training, and empirical gains across spatial reasoning benchmarks, all of which are directly supported by the methodology, ablations, and experimental results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses limitations and future work in Appendix A, including training scale, model scale, inference-turn limits, and extensions to broader spatial reasoning domains and modalities.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not present theoretical results requiring formal assumptions or proofs; it focuses on method design, training algorithms, and empirical evaluation.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper provides the model backbones, evaluation datasets and metrics, baseline settings, training recipe, SFT and RL hyperparameters, tool sandbox details, reward design, OR-GIGPO implementation details, prompt templates, and inference settings in the main text and appendices, providing a reasonable path to reproduce or verify the main experimental results.

Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: The public code and data links are provided on the first page. The code repository includes the tool sandbox, training and evaluation scripts, prompt templates, and instructions for data preparation, training, and inference.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper describes the evaluation datasets, in-distribution and out-of-distribution splits, metrics, baselines, model backbones, SFT and RL training settings, and inference setup in the main text, with additional hyperparameters, prompts, tool settings, and implementation details provided in the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: The paper reports deterministic benchmark accuracy results and extensive ablations, but does not include error bars, confidence intervals, or statistical significance tests. We acknowledge this as a limitation and plan to include repeated-run statistics in future versions.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: The appendix reports hardware and runtime information for reasoning diversification, but per-experiment training and inference runtimes, total compute, and compute used for preliminary experiments are not fully reported.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research is conducted on visual reasoning benchmarks and tool-use training/evaluation without involving human subjects, private personal data, or deceptive data collection, and we have reviewed the NeurIPS Code of Ethics to ensure compliance.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses broader impacts in the Conclusion, including the potential benefits of spatial reasoning agents for visual assistance, mapping, and planning, as well as possible risks from incorrect tool-grounded reasoning, over-reliance on agent outputs, and misuse in surveillance or navigation-sensitive settings.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, scraped sensitive datasets, or data intended for surveillance or identity-related applications. The planned release focuses on tool-use training code, tool sandbox components, prompts, and spatial reasoning trajectories for benchmark-style tasks.

    Guidelines:

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

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All external models, datasets, and tools used in the paper are properly cited with references to their original publications, and their licenses or terms of use are acknowledged where applicable. This includes VLM backbones, prior tool sets, and benchmark datasets.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces new assets including the PERIA tool sandbox, synthesized tool-use trajectories, prompts, and training/evaluation code. These assets are documented in the main text and appendices, including tool definitions, trajectory synthesis procedures, reward details, prompts, implementation settings, and limitations, and will be released with accompanying documentation.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval or equivalent review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: The paper explicitly describes the use of LLMs/VLMs in the core methodology, including GPT-5 for tool-use trajectory synthesis, Qwen3-VL-235B-A22B-Thinking for reasoning-style diversification, and an LLM-judge fallback for correctness verification in reward computation.

    Guidelines:

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