AGENTIC-MME: WHAT AGENTIC CAPABILITY REALLY BRINGS TO MULTIMODAL INTELLIGENCE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) are rapidly evolving from passive observers into active agents. These agents increasingly solve problems through *Visual Expansion*, actively invoking visual tools to transform images. Furthermore, as real-world tasks often require information beyond visual content, modern systems combine these visual operations with *Knowledge Expansion* via open-web search. However, existing evaluations fall short in three critical aspects. (i) Lack flexibility and comprehensiveness in tool integration that supports heterogeneous tool interfaces. (ii) Test image tool use and web search separately, leaving their synergy unexplored. (iii) Evaluate primarily by the correctness of the final answer. Consequently, they cannot verify whether the tools were *actually* invoked, applied *correctly*, or used *efficiently*. To answer what agentic capability truly brings to multimodal intelligence, we introduce Agentic-MME, a process-verified benchmark for Multimodal Agentic Capabilities. Agentic-MME contains 418 real-world tasks across 6 domains and 3 difficulty levels designed to evaluate capability synergy, featuring over 2,000 stepwise checkpoints that average more than 10 person-hours of manual annotation per task. Each task is paired with (i) a unified evaluation framework that supports both sandboxed code execution and structured tool APIs, and (ii) a human reference trajectory annotated with stepwise checkpoints along dual-axis: *S-axis* and *V-axis*. To enable true process-level verification, we move beyond final answers by auditing fine-grained intermediate states. We also quantify efficiency via an *overthinking* metric relative to human trajectories. Experimental results show that the best model Gemini3-pro achieves an overall accuracy of 56.3% and this score falls significantly to **23.0%** on Level-3 tasks, underscoring the difficulty of real-world multimodal agentic problem solving.

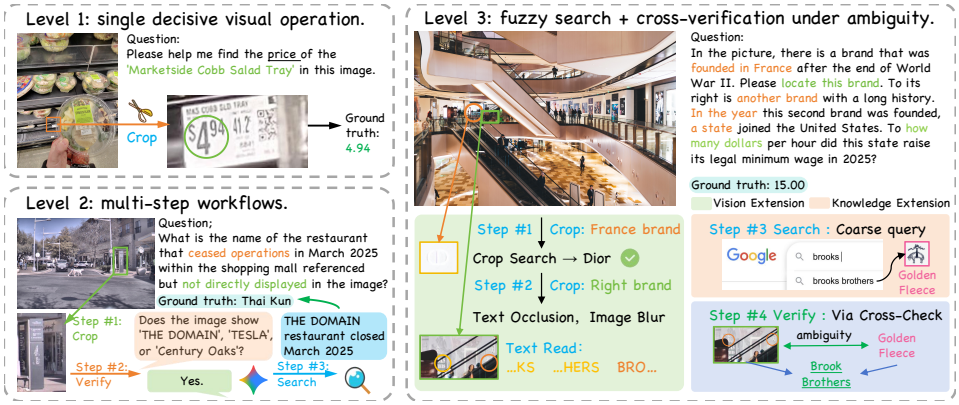

Figure 1: Case Studies in Agentic-MME across three difficulty levels. The examples highlight the benchmark's escalating complexity, evolving from isolated visual operations in Level 1 to deeply synergistic, multi-round visual and knowledge workflows in Level 3.

1 INTRODUCTION

Multimodal Large Language Models (MLLMs) are rapidly evolving from passive observers to active investigators (Shen et al., 2023; Deng et al., 2023; Shi et al., 2025b; Zhang et al., 2025b; Hong et al., 2025). Rather than answering from a static snapshot, modern systems increasingly solve problems by

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

Table 1: Comparison with representative benchmarks. **ImgTools**: dedicated visual manipulation tools; **SearchCore**: open-web search is the core task; **ProcessVerified**: explicit process verification; **Unified**: protocol supports both code & function-calling for visual tools; **Efficiency**: cost/efficiency metric; **Levels**: explicit difficulty levels splits. (●: partly supported.)

Benchmark	Capabilities		Evaluation Protocol			Levels
	Img Tools	Search Core	Process Verified	Unified Code+Tool	Efficiency Metric	
GTA (Wang et al., 2024)	✓	✗	✗	✗	✗	✗
m&m's (Ma et al., 2024)	✓	✗	✗	✗	✗	✗
TIR-Bench (Li et al., 2025a)	✓	✗	✗	✗	✗	✗
VisToolBench (Guo et al., 2025)	✓	✗	✗	✗	✗	✗
MMSearch (Jiang et al., 2024)	✗	✓	✗	✗	✗	✗
MMSearch-Plus (Tao et al., 2025)	●	✓	✗	✗	✗	✓
MM-BrowseComp (Li et al., 2025b)	✗	✓	✓	✗	✗	✓
GAIA2 (Froger et al., 2026)	✗	●	✓	✗	✓	✗
AdaptMMBench (Zhang et al., 2026)	✓	✗	✓	✗	●	✓
Agentic-MME	✓	✓	✓	✓	✓	✓

interacting: they manipulate images to surface fine-grained evidence and consult external resources to verify facts that are not visually present. We refer to this shift as *multimodal agentic capability*, which can be decomposed into two core dimensions: **(1) Visual Expansion**, which enables models to *think with images* by actively transforming and analyzing inputs (e.g., cropping, rotating, enhancing) to uncover latent cues (Wei et al., 2026; Su et al., 2025; Lai et al., 2025; Zhang et al., 2025a; Zheng et al., 2025; Wang et al., 2025); and **(2) Knowledge Expansion**, which enables models to go beyond parametric memory via open-ended web search to validate real-world facts and resolve ambiguity (Chen et al., 2025; Zeng et al., 2026; Narayan et al., 2025). While this active paradigm promises to solve complex real-world problems, current evaluation for *multimodal agentic capabilities* remains fragmented and insufficient. Most existing benchmarks capture specific aspects of tool use but fall short in three critical dimensions: **First, a lack of flexibility and comprehensiveness in tool integration.** Current evaluations often decouple visual tool use from open-web search, treating them as independent modules. Most benchmarks do not provide a unified framework that allows an agent to fluidly select and switch between arbitrary visual and search tools (Li et al., 2025a; Guo et al., 2025; Li et al., 2025b; Tao et al., 2025; Froger et al., 2026; Zhang et al., 2026), which severely restricts the upper bound of agentic behavior. **Second, the synergy between visual and knowledge expansion remains largely unexplored.** There are few realistic scenarios designed to test whether these two capabilities can work in concert. A true multimodal agent should excel in "intertwined" tasks that cannot be solved by simple visual expansion or isolatedknowledge expansiion alone—it is in these deeply coupled workflows that the unique value of an agencitc MLLM is realized. **Finally, a lack of rigorous process verification.** Existing evaluations focus primarily on final-answer correctness, offering limited insight into whether tools were actually invoked, applied correctly, or used efficiently (Hou et al., 2025). When evaluation only inspects the final response, it becomes impossible to diagnose whether a failure stems from limited perception, skipped tool calls, incorrect execution (e.g., cropping the wrong region), or redundant trial-and-error. Without stepwise grounding, these failure modes are conflated, obscuring the true bottlenecks of multimodal intelligence.

To bridge these gaps, we introduce Agentic-MME, a process-verified benchmark for *multimodal agentic capabilities* through a unified and synergistic lens. Agentic-MME targets a realistic agent workflow: an agent must actively manipulate images to extract localized cues and, when necessary, coordinate these cues with open-web retrieval to obtain missing knowledge. To systematically evaluate these capabilities, we stratify tasks into three progressive difficulty levels based on *interaction complexity* (Section 2.2):

- **Level 1 (Visual Expansion Focus):** Tasks isolated to a single visual operation (e.g., one crop), testing the fundamental perception-action loop.
- **Level 2 (Level 1+Knowledge Expansion):** Tasks requiring a simple combination of visual and knowledge expansion. Agents typically extract a visual cue and perform a web search to verify the missing fact. These tasks are solvable through simple tool-chaining, usually within three rounds.
- **Level 3 (Synergistic Coupling):** Challenging, realistic scenarios that demand iterative, interleaved execution of both visual and search tools. In these cases, agents extract visual hints, use them to query the web for validation, and retrieve knowledge to further guide refined visual operations.

The design of Level 3 cases also represents our preliminary exploration into the synergy between visual and knowledge expansion. For instance, to identify an obscure logo (as shown in Figure 1), a model first perform a crop to surface blurry visual hints, then execute a multi-hop search to match potential brands for cross-validating. Neither using only visual tools nor a blind web search can solve such tasks in isolation. In this way, Agentic-MME provides a rigorous diagnostic tool to determine whether an agent fails at individual expansion capabilities or at the complex coordination required for high-order multimodal reasoning.

To address the outcome-based evaluation gap and enable true process verification, Agentic-MME moves beyond final-answer grading by fully human-annotated checkpoints and reference trajectories. This design requires substantial **high-quality human effort**: Agentic-MME features over 2000 stepwise checkpoints, averaging 10+ person-hours of careful manual annotation and verification per task, to make intermediate tool use and evidence explicitly verifiable. We evaluate intermediate behavior along two complementary axes. The **S-axis** audits *Knowledge Expansion* by evaluating search-related strategies; for every search hop, we associate checkpoints with human-verified keywords, reference URLs, and expected intermediate answers. The **V-axis** audits *Visual Expansion* by checking both the proactive intent to invoke tools and whether intermediate artifacts genuinely contain the required visual cues. Concretely, each V-axis checkpoint is paired with a targeted question and ground-truth intermediate processed images.

To ensure fair and comparable evaluation across heterogeneous implementations, Agentic-MME provides a standardized execution harness that unifies *sandboxed code execution* and *function-calling tool APIs*. This unification is non-trivial in practice: different models exhibit drastically different code styles, and their sandboxes vary in libraries, I/O conventions, and artifact formats, which can introduce spurious advantages if left uncontrolled. We perform **model-specific, fine-grained environment engineering** and **normalization** so that outputs are produced, logged, and scored under a consistent protoco. We additionally conduct **human calibration and correction** of execution flows and artifact parsing to eliminate unfairness. For code-writing agents, we implement an AST-based tracer that extracts canonical visual operations from executed code traces, enabling consistent scoring across heterogeneous coding patterns and interaction modes. Finally, we measure efficiency by comparing agent behavior against human reference trajectories and penalizing redundant tool calls via an *overthinking* metric. In summary, our contributions are as follows:

❶ ***Agentic-MME Benchmark.*** We introduce Agentic-MME, a benchmark that evaluates comprehensive **multimodal agentic capabilities**, which integrates *visual tool use* with open-ended *web search* in one workflow, with 418 real-world tasks across 3 difficulty levels, and 6 domains.

❷ ***Process-Verified Stepwise Evaluation.*** We provide human reference trajectories with **2,000** stepwise checkpoints for grounding intermediate behavior: (i) the **V-axis** audits *Visual Expansion* by verifying both the correctness of proactive intent to invoke tools and the faithfulness of intermediate visual artifacts; (ii) the **S-axis** audits *Knowledge Expansion* by verifying search keywords and the correctness of retrieved intermediate answers. The same stepwise signals also capture redundant tool usage relative to human references.

❸ ***Unified Evaluation Framework Across Code and Tool Calls.*** We develop a standardized execution and evaluation harness supporting both sandboxed Python code and structured function-calling APIs. We introduce an **AST-based tracer** that audits executed visual operations in code mode for consistent scoring across interaction styles.

❹ **Empirical study and insights.** We evaluate leading proprietary and open-source models. Our findings expose a fundamental gap: while frontier models possess vast knowledge, they still struggle severely with reliable multi-step planning and precise tool-execution in real-world workflows.

2 AGENTIC-MME

2.1 OVERVIEW

We introduce Agentic-MME, which is designed to evaluate multimodal agentic capabilities. Unlike existing works that isolate visual operations, Agentic-MME targets a realistic scenario where agents actively utilize visual tools to transform and perceive image content, and, contingent on task requirements, coordinate with open-web search to retrieve essential external knowledge.

2.2 TASK SETUP, DIFFICULTY, AND METRICS

Task setup. Each instance provides one or more images and a question. Agents solve the task by actively manipulating the image within a unified tool-augmented interface, which equips with

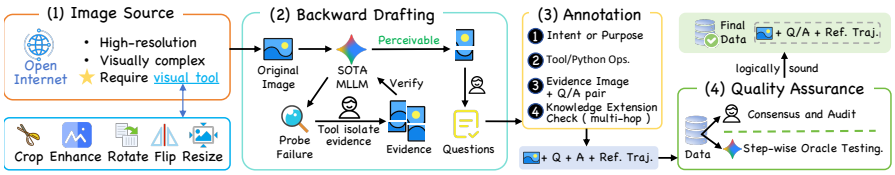

Figure 2: Data Collection and Annotation pipeline, including image sourcing, backward drafting, granular step-wise annotation, and quality assurance.

13 distinct visual operations for *Visual Expansion* and 4 open-web retrieval tools for *Knowledge Expansion*. Implementation details are in Appendix B and Sec. 2.5.

Task Scenarios and Difficulty levels. Agentic-MME encompasses two core scenarios: extracting hidden evidence through active image manipulation, and utilizing visual cues to retrieve or verify facts from the open web. We systematically stratify these tasks into three difficulty levels based on the *interaction complexity* along a reasonable solution path:

- ◇ Level 1 requires a single visual operation (e.g., one *crop*/ *rotate*).
- ◇ Level 2 requires multi-step workflows where agents actively manipulate the image to obtain key visual cues and may leverage open-web to retrieve external factual knowledge beyond visual content. (e.g., *crop*→ *rotate* → *web-search*).
- ◇ Level 3 represents advanced synergistic workflows where the answer cannot be reached through simple sequential tool chaining. Instead, it demands an intertwined, multi-round interaction between visual manipulation and web search. This level encompasses (i) integrating *multiple clues* scattered across different regions or separate images, (ii) performing *advanced CV analysis* (e.g., frequency-domain transformations for recognition), and (iii) resolving severe visual ambiguity (e.g., Figure 1). For the latter, agents must execute an explicit *hypothesis–verification loop*. For example, to identify a highly blurry logo, an agent must first crop the image to extract a vague visual hint, perform multi-hop searches to retrieve candidate entities, and then synergistically cross-validate these retrieved external facts against the processed image to confirm the uncertain information.

This continuous cross-referencing between the visual input and the open web represents the true synergy required for real-world agentic problem-solving. More case studies are in Appendix D.

Metrics. We report final answer accuracy (**Acc**) together with process-aware and efficiency metrics derived from stepwise checkpoints. Concretely, **S** and **V** are computed as the fraction of passed checkpoints on the S-axis and the V-axis. Because simply invoking a visual tool does not guarantee that the correct visual evidence is successfully extracted, we decouple the V-axis for finer diagnosis. We split V-axis into: V_{tool} : whether the agent initiates the required visual tool use at the right step; and V_{true} : whether the generated true intermediate visual artifact actually contain the require evidence. To evaluate efficiency, we introduce an **Overthink** metric to penalize redundant behaviors relative to the human minimal trajectory, defined as Overthink $= \max(0, C_{\text{agent}} - C_{\text{human}})/(C_{\text{human}} + 1)$, where C counts the interactions for tasks. Exact scoring rules (checkpoint matching, evidence verification) are specified in Appendix. A.

2.3 Data Collection and Annotation

Process-level supervision is only meaningful when annotations are rigorous, reproducible, and verifiable. We formulate data construction as a strictly controlled protocol (as shown in Fig 2).

Image Sourcing. We source high-resolution, visually complex images across 6 heterogeneous domains from the open web. To prevent the benchmark from degrading into a simple cropping task, annotators intentionally curate real-world scenarios that require a diverse visual toolkit. Specifically, the dataset includes oversized images requiring localized *cropping*, low-light environments or folded documents needing photometric *enhancement*, misaligned documents requiring *rotation*, mirrored visuals demanding *flipping* and steganographic visuals requiring *resizing* . By embedding decisive evidence within these challenging contexts, the dataset strictly mandates diverse visual manipulation.

Model-in-the-Loop Backward Drafting. Contributors must pass qualification tasks to align on difficulty and granularity. To rigorously guarantee that visual interaction is necessary, tasks are designed via an empirical *model-in-the-loop backward-drafting* mechanism. Annotators first prompt a state-of-the-art model (e.g., Gemini 3 Pro) to comprehensively describe the raw image. By reviewing this initial response, annotators specifically target visual details that the model overlooks

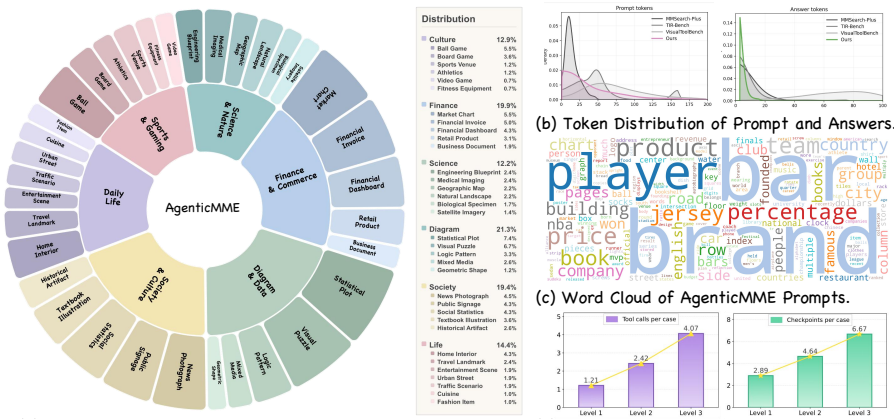

Figure 3: Overview of Agentic-MME Dataset Statistics. The benchmark exhibits broad domain and semantic diversity, with increasing tool calls and checkpoints reflecting the escalating demand for long-horizon reasoning across difficulty levels.

or hallucinates during passive inspection. The annotator then applies visual manipulation tools (e.g., cropping) to isolate the evidence, and verifies the *same model* can successfully perceive the processed image. Crucially, the model's updated response is cross-checked against a human ground truth to ensure it is genuinely correct rather than a confident hallucination. Once validated, the annotator drafts the user question targeting this hidden evidence, making visual manipulation a prerequisite.

Granular Step-wise Annotation. Beyond final answers, we annotate complete reference trajectories. Each checkpoint records the action intent, required tool operations, and verifiable intermediate artifacts. For search steps, annotators additionally provide verified keywords, URLs, and expected intermediate answers.

Answer Standardization. To eliminate evaluation ambiguity and avoid reliance on LLM-as-a-judge for final accuracy, we proactively constrain the output space during task design. Annotators specifically draft questions that inherently yield concise, verifiable answers. Furthermore, each instance includes explicit prompt instructions dictating the required final format (e.g., specific units, categorical options). Supported by an accepted-variant list for minor deviations, this prompt-level control ensures that final answer extraction and scoring remain deterministic and regex-friendly.

2.4 Quality Control and Assurance

Human Verification: Consensus and Audit. To guarantee human solvability and annotation quality, each drafted instance is evaluated by the author and at least two independent verifiers. Verifiers first attempt the task end-to-end without accessing the reference path. Given the visual complexity, they may use manual tools (e.g., web browsers, image viewers) and tool-less vision-language models strictly for perceptual assistance. While the vast majority of instances are successfully solved on the first attempt, any divergent answer triggers a joint review. The team carefully refines ambiguous questions, permanently discarding any instances with unresolved disagreements to maintain the rigor of the dataset. Once strict consensus is reached on the final answer, verifiers systematically audit step-wise annotations, ensuring that all intermediate tool actions, visual artifacts, and retrieved URLs are accurate and logically sound.

Model Verification: Step-wise Oracle Testing. To ensure that current models are fundamentally capable of solving the tasks when following the correct trajectory, we probe all instances using state-of-the-art models (e.g., Gemini 3 Pro). Specifically, we perform *step-wise oracle guided verification*: we walk the model through the human reference path, providing the exact ground truth intermediate observations (e.g., the properly cropped image region or the correct web text) at key steps. A task is retained *if and only if* the model can successfully arrive at the correct final answer. This check confirms that the underlying visual and textual evidence is perceptible to the model. Consequently, an agent's failure during evaluation directly reflects shortcomings in tool execution or multi-step planning, rather than a inability to process the evidence.

Dataset Statistics. As illustrated in Figure 3(a), Agentic-MME features a highly diverse, hierarchical distribution across 6 major domains and 35 sub-categories with prompt diversity further highlighted by the word cloud in Figure 3 (c). Table 2 summarizes the distribution of the tasks across Levels 1–3 ,

Table 2: **Dataset Statistics of Agentic-MME.** textbfDataset Statistics of Agentic-MME. Specifically, **Total Images / Tools** denotes the unique visual inputs and tool invocations collected from human trajectories, while **PerDescri. Len** indicates the average token of the intent description per checkpoint.

Task Difficulty	**Share**	**Ckpts / Task**	**Tools / Task**
Level 1 (Easy)	48.6%	2.89	1.21
Level 2 (Mid)	32.1%	4.64	2.42
Level 3 (Hard)	**19.4%**	**6.67**	**4.07**
Key Property	**Value**	**Key Property**	**Value**
Total Images / Tools	430 / 899	Small Cues ($< 10\%$)	226 (43.1%)
Domains / Sub	6 / 35	Avg. Cue Area	35.8%
Avg. Resolution	1952×1747	Avg. L_{prompt}/L_{ans}	31.9 / 1.5 token
PerDescri. Len	16.38 tokens	External Search	29.4%

alongside the average number of checkpoints and tool invocations in the human reference trajectories. The spatial footprint of visual evidence, showing that over 40% of instances require recovering highly localized information (occupying $< 10\%$ of the image area), which justifies the need for active tool manipulation over passive perception. Figure 3(b) reports the token length distribution of questions and answers. Crucially, the benchmark is grounded by over 2,000 fine-grained stepwise checkpoints. As shown in Figure 3(d) the checkpoint and tool calls density increases significantly from Level 1 to Level 3, consistent with the longer horizons in complex tasks. Notably, every checkpoint is accompanied by a detailed natural language description (averaging 16.38 tokens), ensuring precise interpretability of intent and failure modes for both human and automated evaluation.

2.5 Unified Tool Interface and Execution Harness

Agentic-MME evaluates agents using fully logged, replayable traces, including intermediate actions, returned observations, and generated artifacts (e.g., processed images and retrieved webpages). Our harness supports heterogeneous agents through two interaction modalities: atomic tool calls via function interfaces and sandboxed code execution for free-form visual manipulation. Since agents differ substantially in operation expression, file handling, and I/O conventions, we enforce a unified artifact protocol that normalizes both modalities into an auditable event stream. For code-based agents, we execute programs in an instrumented sandbox with normalized image I/O (e.g., save-path redirection) and apply an AST-based tracer to extract canonical visual operations for consistent checkpoint matching across coding styles. For retrieval, we provide API-based search (text, image, and webpage fetching) and cache queries and payloads to reduce non-determinism and enable deterministic replay (Appendix B).

3 Experiments

3.1 Experimental Setup

Models and baselines. We evaluate diverse open- and closed-source multimodal models on Agentic-MME. The human baseline averages scores from three independent solvers, who may use search engines and perception models as auxiliary tools. We also report No-tool baselines, where models answer directly from raw images without tool invocation.For tool-augmented agents, we evaluate representative open-source models including Thyme-rl(Zhang et al., 2025a), Deepeyesv2(Hong et al., 2025), Qwen3-VL-235B, Qwen3-VL-8B, and Qwen 3 VL-32B(Bai et al., 2025), together with closed-source models including Gemini3 family(Team et al., 2023), Kimi-k2.5(Team et al., 2026), GPT-5.2(OpenAI, 2025), and Qwen3.5-plus(Team, 2026). Results are reported in Table 3.

Two tool interfaces: Code and Atomic. A central goal of Agentic-MME is to evaluate agentic capability across heterogeneous implementations. Using our unified harness (Sec. 2.5), we evaluate each tool-using model under two interfaces: Code mode (**Gen**), where models write sandboxed Python for visual transforms; and Atomic mode (**Atm**), where models interact through structured function calls following the provided schemas (Appx. B). This controlled comparison tests whether tool competence generalizes across interfaces rather than relying on a specific interaction format.

Evaluation protocol. All evaluations are performed on fully logged and replayable traces. Following CodeV (Hou et al., 2025), we use an MLLM-as-a-Judge approach to verify intermediate visual artifacts (V_{true}). Search checkpoints are similarly evaluated by an LLM judge against human-annotated targets, including retrieved URLs, keywords, and intermediate reasoning processes. We use gpt-4o-mini(Hurst et al., 2024) as the primary judge model.

Table 3: Comprehensive evaluation results on the Agentic-MME benchmark. We evaluate tool-augmented models under two distinct interaction interfaces: **Gen** (Code Generation mode, where models write sandboxed Python for visual transforms) and **Atm** (Atomic mode, interacting via structured function-calling APIs). Within each category, models are sorted in descending order based on their highest (Acc) across all modes. The top model in each category is highlighted with a small icon.

Model	Mode	Overall			Level 1				Level 2				Level 3			
		Acc	S	V	Acc	S	V_{to}	V_{tr}	Acc	S	V_{to}	V_{tr}	Acc	S	V_{to}	V_{tr}
Human	–	**93.8**	–	–	**99.0**	–	–	–	**92.6**	–	–	–	**82.3**	–	–	–
No Tooluse Models																
Gemini 3 pro-preview	–	30.2	–	–	42.9	–	–	–	24.6	–	–	–	7.5	–	–	–
Open-Source Models																
Qwen 3 Vl-235B	Gen	28.5	20.8	17.3	43.9	28.6	17.8	18.2	23.3	33.0	20.0	19.3	2.5	10.1	12.1	15.2
	Atm	**34.9**	20.5	35.7	50.0	57.1	27.7	30.2	26.9	30.1	40.7	40.5	10.1	8.1	48.8	48.4
Qwen3 Vl 32B-think	Gen	27.4	8.7	36.5	42.2	21.4	66.4	9.7	19.5	15.7	37.9	18.7	4.6	4.8	30.4	12.6
	Atm	29.9	9.8	12.7	46.3	21.4	17.1	9.4	22.5	17.0	15.5	17.5	4.6	6.1	14.3	14.2
Qwen3 Vl 8B-think	Gen	26.3	10.3	22.2	40.4	21.4	29.5	8.7	17.3	16.4	31.4	18.2	9.1	6.7	19.6	13.9
	Atm	26.6	10.2	12.3	45.6	25.0	10.6	9.2	16.5	20.8	13.4	18.8	3.3	5.9	8.1	13.2
Deepeyesv2	Gen	22.5	4.5	12.9	34.3	0.0	24.5	6.6	13.5	7.0	17.2	0.6	7.5	2.9	7.0	0.8
	Atm	25.2	5.1	26.5	38.9	0.0	24.4	14.4	17.0	9.3	44.8	11.5	3.8	2.0	42.1	9.4
Thyme-rl	Gen	18.0	0.4	36.9	27.7	0.0	63.3	13.0	10.5	1.0	38.8	4.2	6.2	0.0	35.6	4.7
	Atm	23.0	1.8	25.1	32.4	0.0	23.8	13.7	20.7	2.9	44.0	13.3	2.5	1.1	54.7	10.9
Closed-Source Models																
Gemini 3 pro-preview	Gen	49.5	29.7	45.8	64.8	42.9	53.0	38.1	41.8	41.8	48.6	29.5	23.0	26.1	54.1	23.0
	Atm	**56.3**	29.0	64.1	70.6	50.0	77.9	43.2	47.4	37.8	75.9	36.1	33.3	18.2	87.5	28.8
Gemini 3-flash	Gen	47.7	27.3	51.0	62.8	50.0	72.8	28.1	40.0	39.4	62.1	30.6	21.6	26.1	63.1	20.9
	Atm	52.2	26.4	59.4	67.6	57.1	67.0	35.2	43.8	46.3	70.8	31.5	27.2	18.8	67.5	24.8
Qwen3.5-plus	Gen	44.5	27.3	45.0	62.1	42.9	69.6	35.8	39.8	40.3	56.1	31.5	18.3	25.5	35.9	19.3
	Atm	47.1	29.0	58.8	61.8	35.7	71.9	37.4	39.6	41.7	70.7	37.2	22.8	17.5	75.5	26.2
Kimi-k2.5	Gen	35.0	28.0	16.8	50.8	64.3	21.7	27.3	29.0	38.2	9.1	16.9	8.8	22.5	5.1	13.4
	Atm	44.0	26.2	47.3	61.7	42.9	60.9	34.5	39.0	40.3	51.9	27.7	20.9	26.2	60.4	17.6
GPT 5-mini	Gen	33.5	22.3	56.1	47.8	42.9	82.7	35.3	29.2	29.4	70.8	27.3	8.2	18.7	72.0	24.2
	Atm	37.7	20.8	53.1	50.5	35.7	58.3	38.1	31.9	24.7	78.4	30.7	14.3	15.8	79.4	28.9
GPT 5.2	Gen	31.2	20.3	10.4	45.3	21.4	7.6	21.1	24.6	33.0	6.7	15.8	13.5	15.4	6.9	11.6
	Atm	35.0	17.4	58.1	49.0	35.7	79.0	39.1	26.1	22.6	71.0	24.2	16.7	11.4	72.6	12.5

3.2 Main Results on Agentic-MME

Finding 1: All models fall far below human performance, with a sharp accuracy drop on Level-3. Human solvers reach 93.8% overall and remain strong even on the hardest split (L3: 82.3%). In contrast, every model shows a significant decline as task complexity grows. The best-performing system, Gemini 3 Pro (Atm), achieves 56.3% overall but only 33.3% on Level-3. For comparison, Gemini 3 Pro *without any tools* scores 42.9% on Level-1 through passive perception alone, but drops to 7.5% on Level-3; with full tool access, L3 accuracy rises to 33.3%, a 4.4× improvement that demonstrates tools are essential for hard tasks. Nevertheless, the large remaining gap to human performance (33.3% vs. 82.3% on L3) shows that current agents still lack the multi-step planning and reliable tool execution that complex real-world workflows demand.

Finding 2: Open-source models lag behind closed-source models, primarily in search and planning. Closed-source models consistently outperform open-source alternatives across all levels. This gap is most visible on Level-3, where Qwen3 VL-235B drops to 10.1% and Thyme-rl collapses to 2.5%. The S-axis reveals the mechanism behind this gap: Thyme-rl and Deepeyesv2 achieve S scores below 5%, indicating near-total failure in formulating search queries and extracting useful intermediate answers. Qwen3 VL-235B reaches S≈20%, still well below closed-source models. In short, current open-source models can learn to *invoke* tools but have not yet acquired the retrieval planning needed for reliable multi-step problem solving.

Finding 3: Structured tool APIs outperform code generation, but code remains flexible. Across models, Atm matches or exceeds Gen performance. Process metrics reveal that Gen often suffers from insufficient tool initiation: GPT 5.2 achieves $V_{to} \leq 7.6\%$ across levels, with overall V=10.4,

Table 4: **Tool availability analysis.** We reprot Acc in different level tasks

Setting	Gemini 3 Flash				Qwen 3 VL-235B			
	Overall	L1	L2	L3	Overall	L1	L2	L3
Perception-only	39.82	55.84	30.10	15.24	31.94	48.44	23.13	7.41
Image-only	47.61	64.58	38.90	18.79	36.93	57.67	25.58	6.25
Search-only	46.68	65.62	34.33	22.22	37.10	52.08	31.34	11.11
Full (Img+Search)	**52.24**	**67.60**	**43.77**	**27.19**	**42.85**	**58.82**	**34.27**	**19.23**

while Atm increases V_{to} above 70% and V to 58.1. Kimi-k2.5 shows a similar trend. Although code execution introduces additional burdens in program composition and error handling, it provides flexibility through arbitrary operation composition and custom transformations. Gemini 3 Flash (Gen) already achieves 47.7% overall, highlighting the potential of reliable code-based tool use.

Finding 4: Models call tools eagerly but often produce wrong results. Some agents invoke tools aggressively but generate incorrect outputs: Thyme-rl (Gen, L1) reaches $V_{tool} = 63.3$ while V_{true} is only 13.0; Qwen3 VL-32B-think (Gen, L1) shows $V_{tool} = 66.4$ vs. $V_{true} = 9.7$. These models frequently crop or transform the wrong region, wasting interaction budget and injecting misleading artifacts. In contrast, Qwen3 VL-235B (Atm, L3) shows $V_{tool} = 48.8$ and $V_{true} = 48.4$, nearly identical, indicating that its tool parameterization is reliable. Agentic-MME's process-level checkpoints make it possible to pinpoint where and why model fails.

3.3 Further Analysis

A meaningful benchmark should genuinely require the capabilities it claims to test We conduct three validation experiments to confirm that Agentic-MME (i) cannot be solved with text only, (ii) truly demands active tool use rather than passive perception, and (iii) provides high-quality stepwise annotations that capture actionable problem-solving structure.

Visual grounding is essential Removing all images yields near-zero accuracy (Gemini 3 Flash: 2.63%; GPT-5-mini: 1.44%), confirming that Agentic-MME is free from data leakage and that each task genuinely requires visual evidence.

Tasks genuinely require active tool use. To validate that Agentic-MME rewards agentic behavior rather than strong passive perception, we compare four settings: Perception-only, Image-only, Search-only, and Full (Table 4). For both models, the full system consistently outperforms all restricted settings, confirming that the benchmark is designed to require both Visual and Knowledge Expansion. The Level-3 split offers further validation that our hardest tasks capture meaningful capability synergy. For Qwen3 VL-235B, adding image tools alone actually *hurts* Level-3 (7.41%→6.25%), while search-only helps modestly (7.41%→11.11%). However combining both yields 19.23%, gain that far exceeds the sum of individual effects. This super-additive pattern validates our Level-3 design: these tasks are hard to solve by either capability in isolation. Gemini 3 Flash shows the same trend on Level-3 tasks. The results also suggest that unreliable visual tools can be counterproductive without search-based verification.

Stepwise annotations are high-quality and effective A core contribution of Agentic-MME is the human-annotated reference trajectories with over 2,000 stepwise checkpoints. We validate their quality by feeding them to models as guidance and measuring whether performance improves(Table 5). *+Visual Cues* supplies the ground-truth intermediate visual artifacts from our V-axis annotations (e.g., the correctly cropped region); *+Stepwise Guidance* further provides the per checkpoint description derived from our annotation. To prevent leakage, we mask all answer-related keywords in the guidance. Both forms of annotation produce consistent improvements. Stepwise guidance yields substantially improvements, validating that our human trajectories capture actionable planning structure. Notably, the stronger model benefits more from the annotations. A plausible explanation is that the stronger model has sufficient planning ability to incorporate the extra visual evidence into its reasoning. Crucially, performance does not saturate to near-perfect scores, especially on Level-3, despite our step-wise verification during data collection. This reflects the gap between step-by-step verification and autonomous execution. Even given a perfect blueprint, agents must still write API calls, track long contexts, and avoid compounding errors—proving that continuous execution is far harder than isolated perception.

Models systematically overthink. Beyond accuracy, we quantify how efficiently models solve tasks relative to human expert trajectories. The process-level traces also enable detailed diagnosis of failure modes, including missing tool usage, unfaithful visual execution, and redundant exploration. We

Table 5: **Oracle guidance study.** We report acc in different level tasks.

Setting	Gemini 3 Flash				Qwen 3 VL-235B			
	Overall	L1	L2	L3	Overall	L1	L2	L3
Full (baseline)	52.24	67.60	43.77	27.19	42.85	58.82	34.27	19.23
+ Visual Cues	58.37	76.56	45.52	36.25	49.38	72.11	38.17	13.75
+ Stepwise Guidance	**76.21**	**89.58**	**71.94**	**51.25**	**72.80**	**88.59**	**66.67**	**46.91**

Table 6: Efficiency analysis. Avg. calls per task and Overthink (OT). We report human ref.: 2.15 calls per task.

Model	Gen		Atm	
	Calls	OT	Calls	OT
Open-Source				
DeepeyesV2	1.98	0.00	1.95	0.00
Qwen3-235B	8.66	4.03	3.31	1.54
Closed-Source				
Gemini 3 Pro	9.28	2.26	4.66	0.80
Kimi-k2.5	9.43	2.31	4.48	0.74
GPT 5.2	5.50	1.06	2.98	0.26
GPT-5-mini	**12.13**	5.64	**7.22**	3.36
Qwen3.5-plus	7.72	3.59	5.24	2.44

Table 7: Evaluation robustness on Gemini 3 Pro (Atm). We report overall scores.

Judge	Acc	S	V
GPT-5-mini	56.28	28.44	63.27
Gem-2.5-Flash	56.28	29.56	65.64
GPT-4o-mini	56.28	29.00	64.08
Human Expert	56.28	28.67	64.84

provide a complete failure taxonomy and model-wise analysis in Appendix F. Table 6 reports per-task average tool calls and the Overthink ratio. Two patterns emerge. First, Atm mode is universally more efficient than Gen mode. Second, neither interaction extreme guarantees high accuracy. DeepeyesV2 under-utilizes tools (OT=0) yielding low accuracy (22.5%), while GPT-5-mini severely over-explores (12.13 calls/task) but only reaches 33.5% due to redundant trial-and-error. Gemini 3 Pro achieves the highest accuracy (56.3%) by striking a balance, demonstrating that *focused, reliable* tool execution matters far more than exhaustive exploration.

Evaluation robustness and human consistency We verify consistency by re-evaluating Gemini 3 Pro (Atm) with three judges and human experts (Table 7). S/V scores vary by a very small difference across judges, and human expert scores fall within the range, confirming that our structured checkpoint design aligns automated evaluation with human judgment.

4 Related Work

Tool-augmented visual reasoning. While traditional evaluations target static multimodal inputs (Fu et al., 2023; Yue et al., 2024; Li et al., 2023; Shi et al., 2025c; Yu et al., 2023), recent benchmarks explore active multi-tool execution (Wang et al., 2024; Ma et al., 2024) and visual manipulation (Guo et al., 2025; Li et al., 2025a; Shi et al., 2025a). However, these settings treat open-web retrieval as peripheral (e.g., `google_search` constitutes $< 7\%$ of calls for o3/GPT-5), failing to thoroughly assess the synergy between *Visual* and *Knowledge Expansion* in realistic workflows (Table 1).

Multimodal search and process-aware evaluation. omplementary lines of work focus on open-world information seeking, evaluating how models incorporate vision in the loop of search (Jiang et al., 2024; Tao et al., 2025) and multimodal web browsing (Li et al., 2025b). However, as highlighted by CodeV (Hou et al., 2025), relying solely on the correctness of final answers can easily mask unfaithful tool execution. Consequently, while recent benchmarks advance textual provenance tracking, intermediate visual artifacts often remain unverified, despite a growing consensus on the need for strict, step-wise process verification in agent evaluations (Froger et al., 2026; Zhang et al., 2026). Similarly, recent multimodal deep-research frameworks have made inspiring strides in long-form report synthesis (Huang et al., 2026b;a). However, because their primary objective is knowledge retrieval, their visual operations are largely restricted to pre-processing (e.g., parsing webpage screenshots), lacking a comprehensive suite of active visual manipulation tools.

5 Conclusion

We introduce Agentic-MME, a process-verified benchmark for evaluating multimodal agents that combine active visual manipulation and open-web retrieval. Agentic-MME provides a unified execution framework with over 2,000 human-annotated checkpoints, enabling fine-grained analysis of tool usage, visual evidence acquisition, and interaction efficiency. Our evaluation reveals that current multimodal agents still struggle with long-horizon coordination between visual and knowledge reasoning, highlighting the need for more reliable and effective agentic systems.

AI USE STATEMENT

Generative AI tools were used to assist with code implementation, literature search, experimental discussion, and manuscript polishing. The research problem, methodology, and conclusions were developed by the authors. All AI-assisted code and text were reviewed and verified by the authors. Generative AI was not used to create datasets or determine scientific claims. The authors take responsibility for all final content.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Jiawei Chen, Xintian Shen, Lihao Zheng, Zhenwei Shao, Hongyuan Zhang, Pengfei Yu, Xudong Rao, Ning Mao, Xiaobo Liu, Lian Wen, et al. Mindwatcher: Toward smarter multimodal tool-integrated reasoning. *arXiv preprint arXiv:2512.23412*, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, et al. Beyond seeing: Evaluating multimodal llms on tool-enabled image perception, transformation, and reasoning. *arXiv preprint arXiv:2510.12712*, 2025.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model. *arXiv preprint arXiv:2511.05271*, 2025.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Moyan Li, Jia Liu, Todd C Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization. *arXiv preprint arXiv:2511.19661*, 2025.

Peizhou Huang, Zixuan Zhong, Zhongwei Wan, Donghao Zhou, Samiul Alam, Xin Wang, Zexin Li, Zhihao Dou, Li Zhu, Jing Xiong, et al. Mmdeepresearch-bench: A benchmark for multimodal deep research agents. *arXiv preprint arXiv:2601.12346*, 2026a.

Wenxuan Huang, Yu Zeng, Qiuchen Wang, Zhen Fang, Shaosheng Cao, Zheng Chu, Qingyu Yin, Shuang Chen, Zhenfei Yin, Lin Chen, et al. Vision-deepresearch: Incentivizing deepresearch capability in multimodal large language models. *arXiv preprint arXiv:2601.22060*, 2026b.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, et al. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.

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. *arXiv preprint arXiv:2509.07969*, 2025.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

Ming Li, Jike Zhong, Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Yuxiang Lai, Chen Wei, Konstantinos Psounis, and Kaipeng Zhang. Tir-bench: A comprehensive benchmark for agentic thinking-with-images reasoning. *arXiv preprint arXiv:2511.01833*, 2025a.

Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, et al. Mm-browsecomp: A comprehensive benchmark for multimodal browsing agents. *arXiv preprint arXiv:2508.13186*, 2025b.

Zixian Ma, Weikai Huang, Jieyu Zhang, Tanmay Gupta, and Ranjay Krishna. m & m's: A benchmark to evaluate tool-use for m ulti-step m ulti-modal tasks. In *European Conference on Computer Vision*, pp. 18–34. Springer, 2024.

Kartik Narayan, Yang Xu, Tian Cao, Kavya Nerella, Vishal M Patel, Navid Shiee, Peter Grasch, Chao Jia, Yinfei Yang, and Zhe Gan. Deepmmsearch-r1: Empowering multimodal llms in multimodal web search. *arXiv preprint arXiv:2510.12801*, 2025.

OpenAI. Thinking with images. `https://openai.com/index/thinking-with-images/`, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Yang Shi, Yuhao Dong, Yue Ding, Yuran Wang, Xuanyu Zhu, Sheng Zhou, Wenting Liu, Haochen Tian, Rundong Wang, Huanqian Wang, et al. Realunify: Do unified models truly benefit from unification? a comprehensive benchmark. *arXiv preprint arXiv:2509.24897*, 2025a.

Yang Shi, Jiaheng Liu, Yushuo Guan, Zhenhua Wu, Yuanxing Zhang, Zihao Wang, Weihong Lin, Jingyun Hua, Zekun Wang, Xinlong Chen, et al. Mavors: Multi-granularity video representation for multimodal large language model. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10994–11003, 2025b.

Yang Shi, Huanqian Wang, Wulin Xie, Huanyao Zhang, Lijie Zhao, Yi-Fan Zhang, Xinfeng Li, Chaoyou Fu, Zhuoer Wen, Wenting Liu, et al. Mme-videoocr: Evaluating ocr-based capabilities of multimodal llms in video scenarios. *arXiv preprint arXiv:2505.21333*, 2025c.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, et al. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*, 2025.

Xijia Tao, Yihua Teng, Xinxing Su, Xinyu Fu, Jihao Wu, Chaofan Tao, Ziru Liu, Haoli Bai, Rui Liu, and Lingpeng Kong. Mmsearch-plus: Benchmarking provenance-aware search for multimodal browsing agents. *arXiv preprint arXiv:2508.21475*, 2025.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37: 75749–75790, 2024.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language. *arXiv preprint arXiv:2511.21395*, 2025.

Lai Wei, Liangbo He, Jun Lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, et al. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. *arXiv preprint arXiv:2602.11858*, 2026.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, et al. Vision-deepresearch benchmark: Rethinking visual and textual search for multimodal large language models. *arXiv preprint arXiv:2602.02185*, 2026.

Xintong Zhang, Xiaowen Zhang, Jongrong Wu, Zhi Gao, Shilin Yan, Zhenxin Diao, Kunpeng Gao, Xuanyan Chen, Yuwei Wu, Yunde Jia, et al. Adaptmmbench: Benchmarking adaptive multimodal reasoning for mode selection and reasoning process. *arXiv preprint arXiv:2602.02676*, 2026.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, et al. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025a.

Yifan Zhang, Liang Hu, Haofeng Sun, Peiyu Wang, Yichen Wei, Shukang Yin, Jiangbo Pei, Wei Shen, Peng Xia, Yi Peng, et al. Skywork-r1v4: Toward agentic multimodal intelligence through interleaved thinking with images and deepresearch. *arXiv preprint arXiv:2512.02395*, 2025b.

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. *arXiv preprint arXiv:2505.14362*, 2025.

Supplementary Material

A Process-Aware Evaluation Protocol and Scoring Details

A core contribution of is enabling automated, process-level evaluation without human intervention. We score agents using final-answer correctness and step-wise checkpoint compliance along two orthogonal axes: Strategy (**S-axis**) and Visual Evidence (**V-axis**). Checkpoint matching relies strictly on the logged execution traces and generated artifacts.

A.1 S-axis Evaluation (Strategy & Tool Execution)

S-axis checkpoints automatically verify whether the agent adopted the correct high-level plan and invoked the necessary tools. To robustly handle heterogeneous agent architectures, we apply distinct parsing strategies:

- *Pre-defined Tool Agents:* We directly parse the structured function calls to compare the tool selection and arguments against the reference strategy.
- *Code-Writing Agents:* For models that write free-form code, we perform Abstract Syntax Tree (AST) parsing on the executed Python snippets to extract canonical visual operations, ensuring the semantic intent matches the ground truth regardless of coding style.
- *Retrieval Actions:* Web search results are inherently non-deterministic, making exact string matching insufficient. Instead, we deploy an external LLM-as-a-judge. We provide the judge with the annotated step *description*, *search keywords*, *expected answer*, and the agent's *returned observation* (API payload). The S-axis checkpoint passes if the judge determines the returned payload contains sufficient information to satisfy the expected intermediate answer.

A.2 V-axis Evaluation (Visual Evidence Verification)

Even if an agent uses the correct tool, it may extract the wrong region (e.g., cropping the background instead of the target). Inspired by the visual faithfulness evaluation in CodeV (Hou et al., 2025), the V-axis explicitly verifies whether the intermediate visual artifacts *genuinely contain* the decisive evidence.

Each V-axis checkpoint is paired with a highly specific intermediate question (e.g., "Does this cropped image clearly show the red mailbox?"). During execution, a single code block or tool call may yield multiple sub-images. We submit *all* produced visual artifacts to an external VLM judge (e.g., GPT-4o) alongside the intermediate question.

Crucially, because all intermediate outputs are appended to the agent's multimodal context history for subsequent turns, the agent successfully "sees" the evidence if at least one artifact is correct. Therefore, we apply an **any-pass mechanism**: the V-axis checkpoint is marked as passed if *any* of the processed sub-images passes the VLM judge's evaluation.

A.3 Final Scores and Efficiency (Overthink)

Let $\mathcal{C}_S$ and $\mathcal{C}_V$ denote the strategy and visual checkpoint sets for a task. The process scores are the fractions of passed checkpoints:

$$S = \frac{1}{|\mathcal{C}_S|} \sum_{c \in \mathcal{C}_S} \mathbb{I}[c \text{ passed}], \quad V = \frac{1}{|\mathcal{C}_V|} \sum_{c \in \mathcal{C}_V} \mathbb{I}[c \text{ passed}] \tag{1}$$

Final answer accuracy (**Acc**) is computed via normalized exact match.

To quantify redundant interactions, we compute an **Overthink** penalty relative to the minimal human reference trajectory:

$$\text{Overthink} = \frac{\max(0, C_{\text{agent}} - C_{\text{human}})}{C_{\text{human}} + 1} \tag{2}$$

where C_{agent} and C_{human} count the interactions (tool calls or executed code blocks) that produce new observable artifacts. All evaluations run under a strict interaction budget to prevent degenerate looping.

A.4 JUDGE PROMPT TEMPLATES

To make the evaluation fully reproducible, we release the exact prompt templates used by the automated judges. In our implementation, the two axes are scored with different prompts because they inspect different evidence modalities.

S-axis judge For search-related checkpoints, the judge receives the checkpoint description, the model's actual search queries, and a compact summary of the returned search results. The prompt is intentionally *lenient* on query wording but *strict* on whether the expected information was actually retrieved.

S-axis judge prompt template

```
You are evaluating whether a model's search successfully found the
required information.

**Expected Search Strategy:**
{checkpoint_desc}

**CRITICAL: Expected Answer to Find:**
{expected_answer}   # included only when available

**Model's Actual Search Queries:**
{queries_text}

**Search Results Summary:**
{results_text}

**Evaluation Criteria (Two-Stage Check):**

STAGE 1 - Query Relevance (Lenient Requirement):
- Be LENIENT when evaluating query quality - broad queries are
  acceptable if they're reasonable
- Accept queries that mention the main entity, even if not specific
- Accept synonyms, paraphrases, different word orders, translations
- Reject ONLY IF: wrong entity OR completely unrelated topic

STAGE 2 - Expected Answer (Primary Criterion):
- This is the MAIN criterion for passing
- Do the search results actually contain the answer/information?
- If an expected answer is specified, it appears in the search results

**Decision Logic:**
1. If queries are completely unrelated to the topic -> FAIL
2. If queries are reasonable but
   results don't contain expected answer -> FAIL
3. If queries are reasonable and results contain
   expected answer -> PASS

**Response Format:**
VERDICT: [PASS/FAIL]
REASONING: [brief explanation]
```

V-axis judge (for V_{true}). For V_{true} checkpoints, the judge is not asked to reason about the whole task. Instead, it receives a single intermediate artifact and a highly specific question designed by the annotator (e.g., "What is the road name shown in this crop?"). In the released implementation, the judge runs independently on *each* produced artifact, and the checkpoint passes under an any-pass rule if any artifact contains the required evidence.

V-axis judge prompt template

```
System:
You are a helpful assistant that answers questions about images.
Provide concise, accurate answers based on what you see in the image.

User input:
[intermediate visual artifact]
{visual_question}

Judging rule applied by the harness:
- Run the judge on every generated intermediate image.
- Compare the returned answer against the expected short answer.
- Mark the checkpoint as passed if ANY image returns the answer.
```

Judge decoding setup.

B TOOL API SPECIFICATION AND IMPLEMENTATION

B.1 OVERVIEW

Overview

Goal. This appendix documents the exact tool API used by Agentic-MME, including the *raw JSON schema* consumed by OpenAI-compatible tool calling. All intermediate artifacts are tracked by `image_index`: Image 0 is the original input, and every successful image tool call appends one or several new images and returns `new_image_index`.

B.2 CANONICAL FUNCTION-CALLING INTERFACE (OPENAI-COMPATIBLE)

Example tool call payload (OpenAI-compatible wrapper)

```
A tool call is identified by function
name and a JSON object function.arguments
{
  "type": "function",
  "function": {
    "name": "<tool_name>",
    "arguments": "{... JSON string ...}"
  }
}
```

B.3 ATOMIC IMAGE TOOLS (VISUAL EXPANSION)

OpenAI-compatible schema: Geometric & Spatial Tools

```
[
  {
    "type": "function",
    "function": {
      "name": "crop",
      "description": "Crop/zoom into a region of an image using
      normalized bounding box coordinates (0-1000 scale).",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
```

```
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "bbox_2d": {
            "type": "array",
            "items": {
              "type": "number"
            },
            "minItems": 4,
            "maxItems": 4,
            "description": "Bounding box [x1, y1, x2, y2]
            in 0-1000 normalized coordinates"
          },
          "zoom_scale": {
            "type": "number",
            "description": "Magnification factor for the
            cropped region (default: 1.0). Higher values
            produce larger images.",
            "minimum": 0.5,
            "maximum": 5.0,
            "default": 1.0
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index",
          "bbox_2d"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "rotate",
      "description": "Rotate an image by angle degrees
      (positive = counterclockwise).",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "angle": {
            "type": "number",
            "description": "Rotation angle in degrees"
          },
          "expand": {
            "type": "boolean",
            "description": "If true, expand canvas to fit
```

```
            rotated image. Default: true"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index",
          "angle"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "flip",
      "description": "Flip/mirror an image horizontally,
      vertically, or both directions.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "direction": {
            "type": "string",
            "enum": [
              "horizontal",
              "vertical",
              "both"
            ],
            "description": "Flip direction: 'horizontal'
            (left-right mirror), 'vertical' (top-bottom
             mirror), or 'both'. Default: horizontal"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "resize",
      "description": "Resize an image to (width, height)
```

```
      OR by scale factor.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "width": {
            "type": "integer",
            "description": "Target width in pixels"
          },
          "height": {
            "type": "integer",
            "description": "Target height in pixels"
          },
          "scale": {
            "type": "number",
            "description": "Scale factor
            (e.g., 2.0 = double size, 0.5 = half)"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  }
]
```

OpenAI-compatible schema: Color & Contrast Tools

```
[
  {
    "type": "function",
    "function": {
      "name": "grayscale",
      "description": "Convert an image to grayscale.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
```

```
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "autocontrast",
      "description": "Apply automatic contrast adjustment.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "cutoff": {
            "type": "number",
            "description": "Percentage of lightest/darkest
            pixels to ignore. Default: 0"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "invert",
      "description": "Invert the colors of an image.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
```

```
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "equalize",
      "description": "Equalize histogram for better
                      contrast distribution.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "threshold",
      "description": Apply threshold to convert image to binary
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "value": {
            "type": "integer",
            "description": "Threshold value (0-255)"
          },
          "mode": {
            "type": "string",
            "enum": [
              "binary",
              "binary_inv",
              "trunc",
```

```
              "tozero"
            ],
            "description": "Threshold mode. Default: binary"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  }
]
```

OpenAI-compatible schema: Filtering & Advanced Tools

```
[
  {
    "type": "function",
    "function": {
      "name": "blur",
      "description": "Apply Gaussian blur.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "radius": {
            "type": "integer",
            "description": "Blur radius. Default: 2"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "sharpen",
      "description": "Apply sharpening filter.",
      "parameters": {
        "type": "object",
```

```
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "denoise",
      "description": "Remove noise from an image.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "strength": {
            "type": "integer",
            "description": "Denoising strength (1-30)."
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "edge_detect",
      "description": "Detect edges in an image.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_index": {
```

```
            "type": "integer",
            "description": "Index of the image to operate on
            (0 = original, 1, 2... = processed images)",
            "minimum": 0
          },
          "method": {
            "type": "string",
            "enum": [
              "canny",
              "sobel",
              "simple"
            ],
            "description": "Edge detection method.
             Default: canny"
          },
          "label": {
            "type": "string",
            "description": "Description of what this operation
                            is for"
          }
        },
        "required": [
          "image_index"]}}
          }
        ]
```

B.4 Web Retrieval Tools (Knowledge Expansion)

Beyond the visual operations, Agentic-MME exposes four retrieval tools for *Knowledge Expansion*. The released implementation uses Serper.dev for Google text search and Google Lens, Jina Reader for webpage fetching, and an auxiliary image-downloading tool that allows the agent to bring externally retrieved candidate images back into the local image workspace for further inspection.

- ◇ **google_search**: Google text search via Serper.dev, returning top results.
- ◇ **google_lens_search**: reverse image search via Serper.dev Lens. If the target is a local image file, the harness first uploads the image to a temporary image host and then queries Lens using the resulting public URL.
- ◇ **fetch_webpage**: webpage-to-text extraction via Jina Reader.
- ◇ **download_image**: downloads a result image URL into the sandbox.
- ◇ **Caching and replay**: all search requests can be stored as per-task JSON payloads to support deterministic replay; when replay mode is enabled, the harness reuses cached responses instead of making fresh web requests.

OpenAI-compatible schema: Web Retrieval Tools

```
[
  {
    "type": "function",
    "function": {
      "name": "google_search",
      "description": "Search the web using Google via Serper
      API.Use for facts, current information, specifications,
       prices, or any knowledge queries.",
      "parameters": {
        "type": "object",
        "properties": {
          "query": {
```

```
            "type": "string",
            "description": "The search query."
          },
          "gl": {
            "type": "string",
            "description": "Geo location code (e.g., 'us').
            Default: 'us'"
          },
          "hl": {
            "type": "string",
            "description": "Language code (e.g., 'en', 'zh').
            Default: 'en'"
          }
        },
        "required": ["query"]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "google_lens_search",
      "description": "Reverse image search using Google Lens
       via Serper.dev API. Use to identify objects,brands,
        logos, landmarks, products, or text in images.",
      "parameters": {
        "type": "object",
        "properties": {
          "image_ref": {
            "type": "string",
            "enum": ["current", "original"],
            "description": "Quick reference: 'current' for the
             latest processed image, 'original' for the input"
          },
          "image_path": {
            "type": "string",
            "description": "Filename or full path to a
             specific image.After image processing, the
             agent may use a filename such as
            'transformed_image_1.png' to search that artifact."
          }
        },
        "required": []
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "fetch_webpage",
      "description": "Fetch and read the content of a webpage.
      Returns clean text extracted from the URL via Jina Reader.
      "parameters": {
        "type": "object",
        "properties": {
          "url": {
            "type": "string",
            "description": "The webpage URL to fetch
```

```
              (must be http/https)."
          },
          "max_chars": {
            "type": "integer",
            "description": "Maximum characters to return.
             Default: 12000"
          }
        },
        "required": ["url"]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "download_image",
      "description": "Download an image from a URL into the
       local sandbox. Downloaded images are appended to the
       image list and can be used by the same visual tools as
        ordinary inputs. For safety and cost control,
        the harness caps downloads per task.",
      "parameters": {
        "type": "object",
        "properties": {
          "url": {
            "type": "string",
            "description": "Direct image URL to download."
          }
        },
        "required": ["url"]
      }
    }
  }
]
```

C More Dataset Statistics

D Case Studies

This appendix presents representative cases for the levels defined in Sec. 2.2.

D.1 Level 1: Core Visual Expansion

Level 1 tasks isolate the core perception–action loop of Visual Expansion. As illustrated in Figures 5 and 6, these tasks are dominated by a single decisive visual operation that surfaces otherwise inaccessible evidence. They do not require multi-step search, cross-image bookkeeping, or iterative hypothesis refinement.

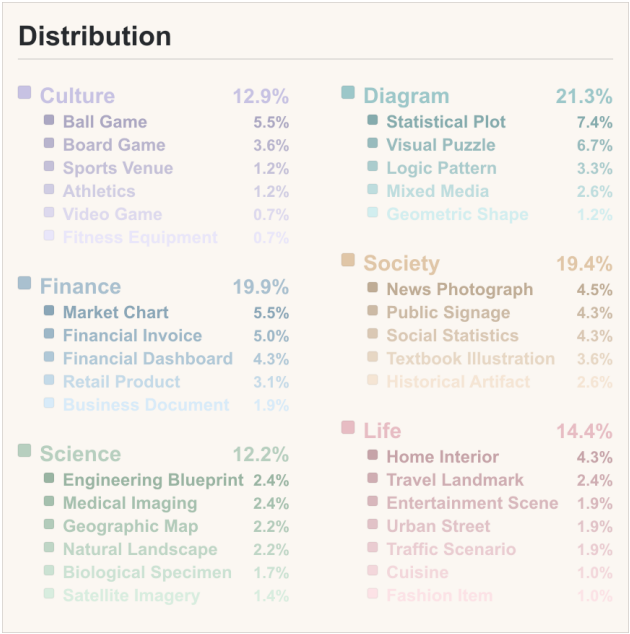

Figure 4: **more fine-grained domain distribution**

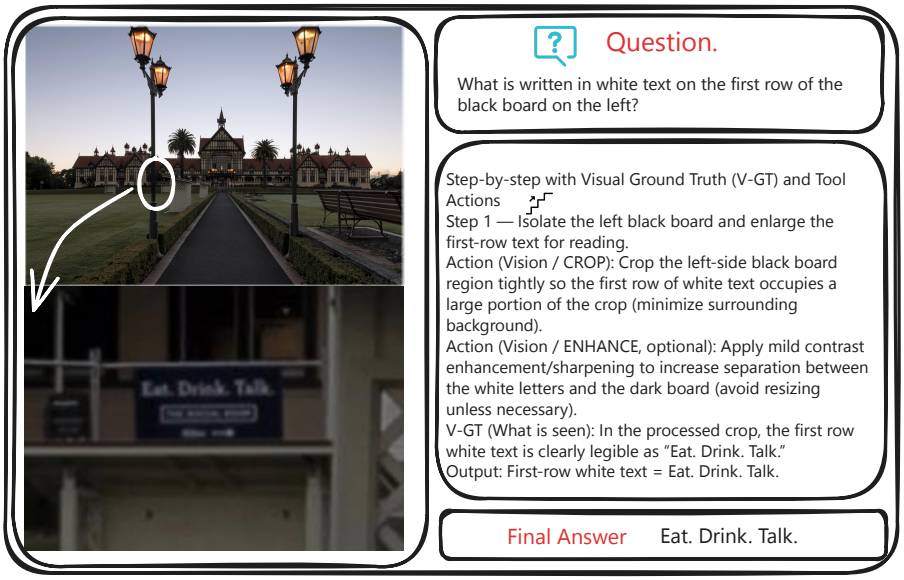

Figure 5: **Level 1 Case: Text Extraction via Localized Enhancement.** The decisive cue is a small blurry background text region that must be isolated and enhanced before it becomes reliably readable.

Figure 5. This case represents a typical Level 1 workflow: the answer-bearing signal is already present in the original image, but passive perception is insufficient because the relevant text is too small or too blurry. A single localized crop, optionally followed by enhancement, is enough to reveal the answer. The task therefore tests whether the model can proactively surface the correct local evidence rather than relying on one-shot visual guessing.

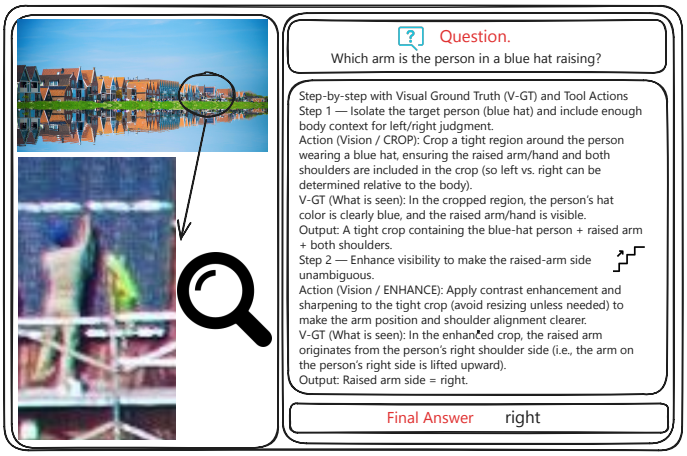

Figure 6: **Level 1 Case: Fine-grained Detail Isolation.** The agent must tightly isolate a distant human figure to resolve a subtle spatial attribute that is ambiguous in the raw image.

Figure 6. This example also falls squarely into Level 1. The core challenge is to isolate a small person region and recover a fine-grained spatial cue, such as the configuration of the arms, that cannot be read reliably from the full image. Unlike higher-level cases, the workflow remains a single-step visual evidence recovery episode and does not require external knowledge or multi-stage planning.

D.2 Level 2: Short Multi-Step Workflows

Level 2 tasks require more than one action, but the solution path is still short and mostly linear. Figures 7 and 8 illustrate two representative Level 2 patterns: short-horizon multi-region composition and structured multi-step visual bookkeeping. Compared with Level 1, the difficulty comes from coordinating multiple intermediate visual observations rather than from a single decisive crop.

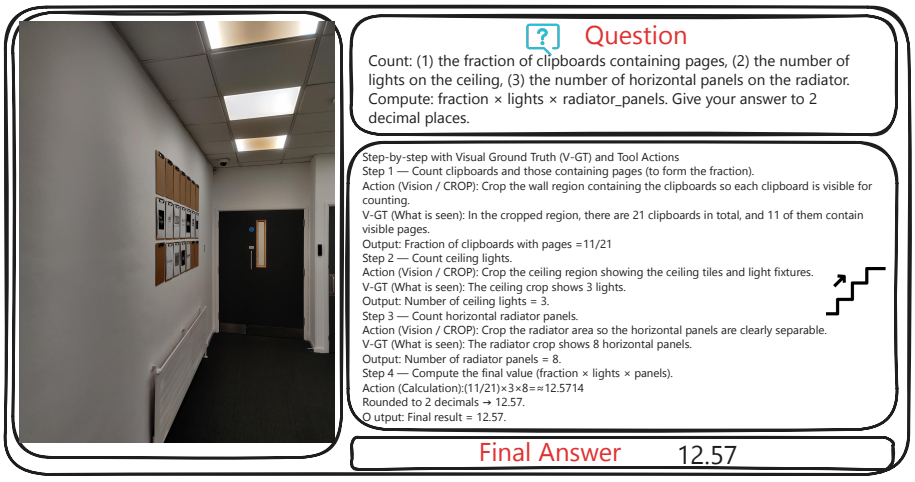

Figure 7: **Level 2 Case: Multi-Region Counting and Arithmetic Composition.**

Figure 7. This case is harder than Level 1 because the answer is not tied to one isolated crop. Instead, the model must perform several localized inspections, preserve multiple intermediate counts, and compose them correctly at the end. However, the workflow is still fundamentally linear: once the relevant regions are identified, the remaining difficulty lies in short-horizon aggregation rather than in iterative search–vision interaction.

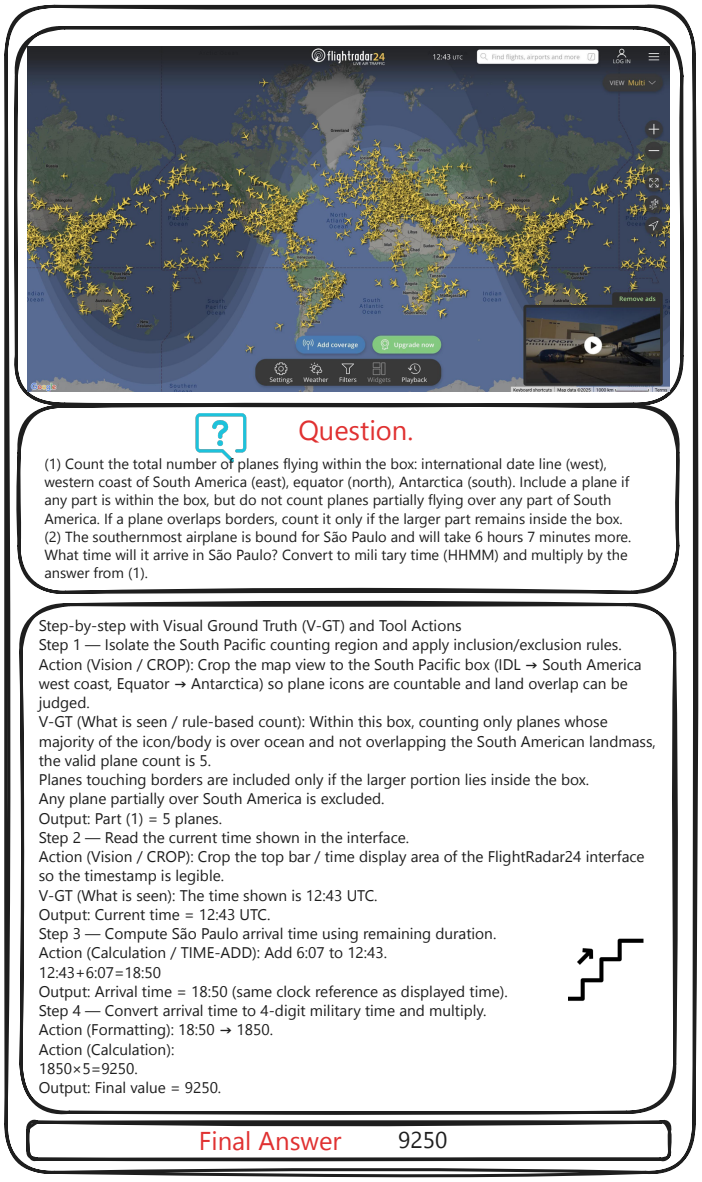

Figure 8: **Level 2 Case: Region-Constrained Counting with Temporal Arithmetic.** The agent isolates the South Pacific region in the flight map, applies inclusion/exclusion counting rules, then reads a separate time cue and combines the two intermediate results through temporal arithmetic.

Figure 8. This example represents another Level 2 pattern in which the model must complete a short sequence of heterogeneous but still straightforward subgoals: region selection, constrained counting, time reading, and final formatting. The task requires explicit intermediate bookkeeping, but it does not yet demand repeated external verification or deep interleaving between visual reasoning and search. It therefore sits naturally in the middle layer between Level 1 single-step perception and Level 3 synergistic workflows.

D.3 Level 3: Advanced Synergistic Problem Solving

Level 3 cases go beyond short sequential chaining. As shown in Figures 9, 10, 11, and 12, these tasks require either advanced global image analysis or a truly intertwined interaction between localized visual evidence and external knowledge. In these settings, the model must refine hypotheses across multiple stages rather than simply execute a fixed linear sequence.

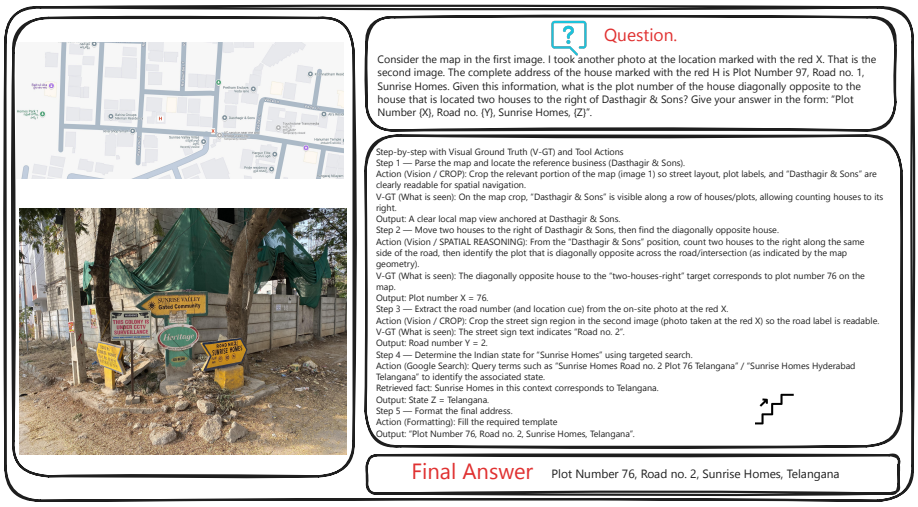

Figure 9: **Level 3 Case: Cross-Image Spatial Grounding with Knowledge Verification.** The agent first crops the map to anchor on the reference business and perform local spatial reasoning, then crops the companion street-view image to recover the road cue, and finally uses open-web retrieval to determine the associated state.

Figure 9. Although this task may appear similar to a short visual-to-search chain at first glance, it is substantially harder in practice and is better categorized as Level 3. The model must first localize the correct region on the map, then reason about a spatial relation relative to a reference entity, then transfer that hypothesis to a different image to extract an auxiliary road cue, and only after that perform external verification. This cross-image grounding plus retrieval dependency makes the workflow more than a simple handoff; We therefore treat it as a Level 3 synergistic case rather than a standard Level 2 example.

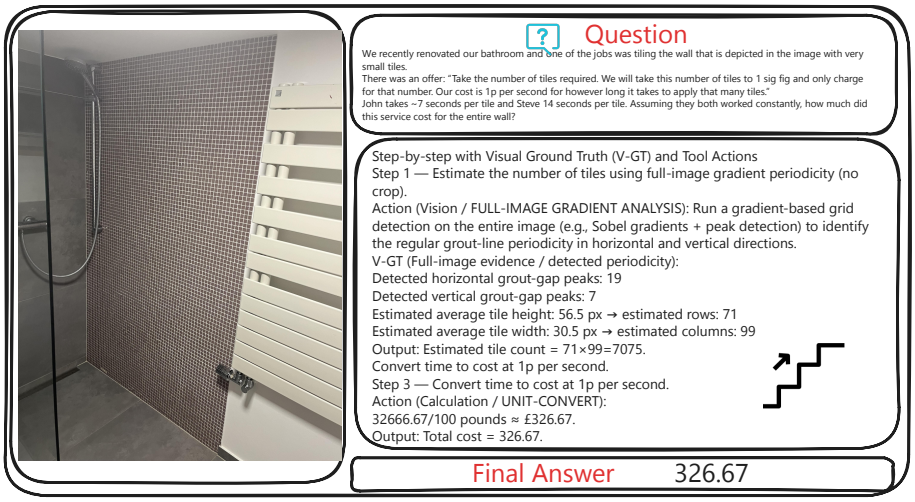

Figure 10: **Level 3 Case: Full-Image Structural Analysis for Dense Tile Estimation.** The agent must infer a latent repetitive grid pattern from the full image, estimate the tile layout, and convert that estimate into a downstream numerical answer.

Figure 10. This case belongs to Level 3 because the decisive signal is distributed globally across the image rather than concentrated in one local patch. The model must reason over large-scale structural regularity, estimate latent periodicity, and then map that estimate to a quantitative answer. Such full-image structural analysis goes beyond the crop-and-read regime of Levels 1–2 and instead reflects the advanced CV-analysis branch of Level 3 described in Sec. 2.2.

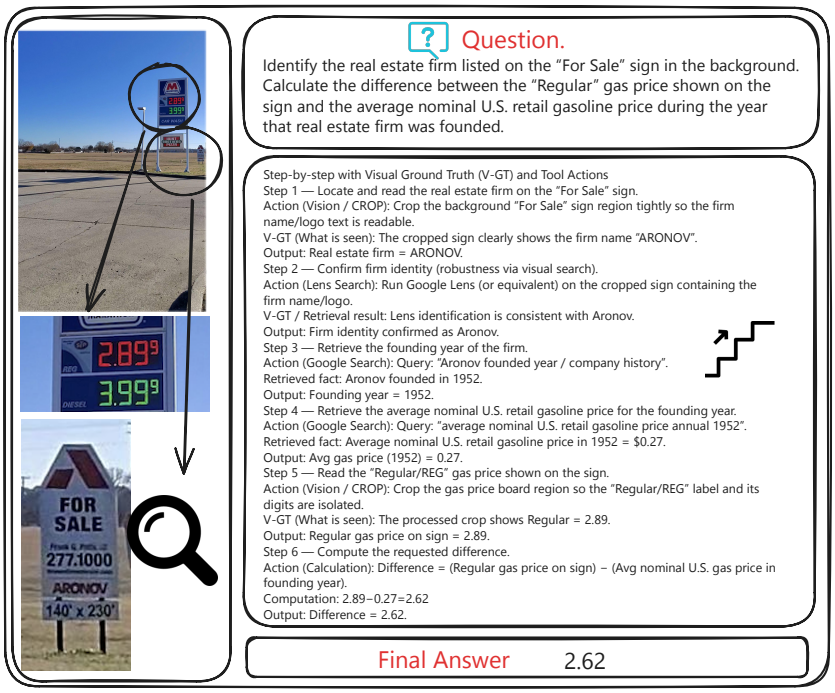

Figure 11: **Level 3 Case: Cross-Domain Multi-Hop Verification.** The agent isolates a weak background logo, uses visual search to identify candidate entities, performs multi-hop web retrieval to obtain the required external fact, and then cross-validates this knowledge against another localized cue before computing the final answer.

Figure 11. This case is a prototypical Level 3 synergistic workflow. The initial visual cue is weak and ambiguous, so a single crop is insufficient. Instead, the model must propose candidate entities, retrieve external evidence through multiple search hops, and repeatedly cross-check the retrieved knowledge against visual observations before it can finalize the answer. Neither isolated visual manipulation nor blind web search can solve this case in isolation.

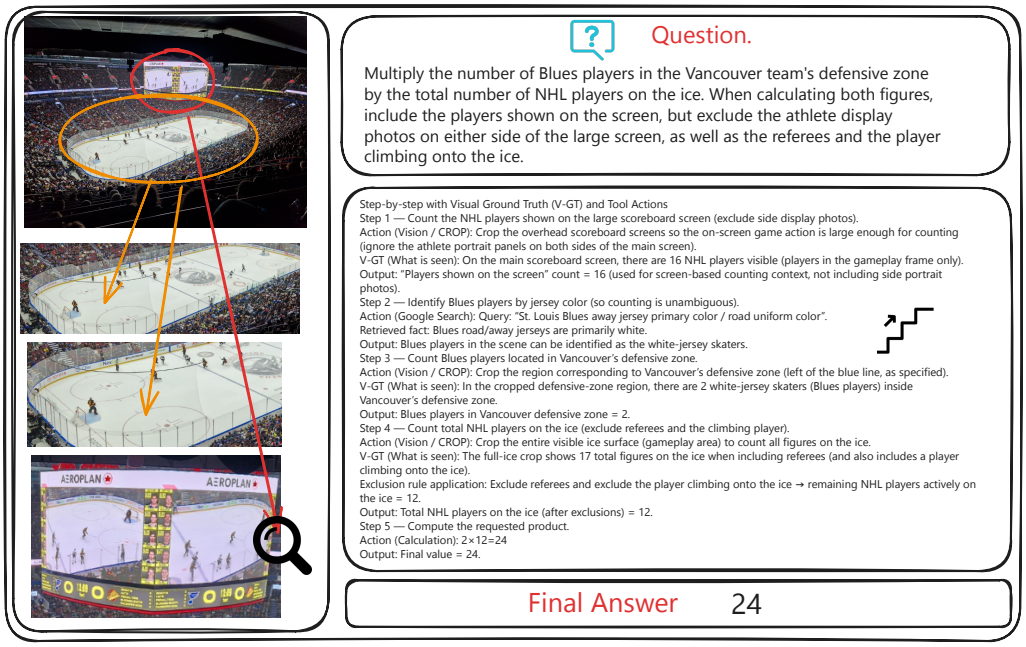

Figure 12: **Level 3 Case: Multi-Step Counting with External Knowledge.** The agent coordinates several targeted crops with an external search to identify team jersey colors, and then applies exclusion logic to reach the final count.

Figure 12. This task is also clearly Level 3 because the answer depends on several mutually dependent subproblems: local evidence extraction from different image regions, external identification of team colors, and a final exclusion process that must stay consistent with all previously recovered cues. The solution is therefore not a simple chain of operations, but an intertwined reasoning process in which visual observations and retrieved knowledge jointly constrain the answer.

E Error Taxonomy

To make the failure analysis in Figure 13 reproducible, we define the seven failure modes used in our manual annotation. Each failed trajectory is assigned one dominant label corresponding to the earliest or most consequential bottleneck that prevents successful task completion.

1. Missing search tools. The task requires explicit open-web retrieval, but the agent never invokes the search tools at the step where external knowledge is necessary. This includes trajectories that rely only on passive visual inspection or local image manipulation when the answer fundamentally depends on non-visual information.

2. Bad search query. The agent does invoke search, but the submitted query is ineffective: it targets the wrong entity, omits the critical visual cue, is too vague to retrieve the intended evidence, or drifts to an unrelated attribute. This category captures failures in transforming localized visual evidence into a useful retrieval query.

3. Unfaithful visual tool use. The agent chooses to use a visual tool, but the produced artifact does not faithfully expose the required evidence. Typical examples include cropping the wrong region, rotating in the wrong direction, over-enhancing into unreadability, or otherwise generating an intermediate image that fails the V-axis verification.

4. Missing visual tool use. The task requires an explicit visual manipulation step, but the agent never performs it. This includes cases where the model answers directly from the raw image, prematurely switches to search, or repeatedly reasons in natural language without taking the necessary visual action.

5. Overthinking Collapse. The agent enters a redundant exploration loop after the necessary evidence could already have been obtained. Typical symptoms include repeated near-duplicate crops, unnecessary additional searches, repeated verification attempts, or excessive trial-and-error that derails the trajectory and wastes interaction budget.

6. Tool-Misexecution. The trajectory fails because of interface-level execution mistakes rather than reasoning alone. Examples include malformed code, invalid tool arguments, runtime errors, missing file saves, or other failures that prevent the intended tool action from being executed correctly.

7. PostVisual-Perception-Deficit. The agent successfully produces a relevant intermediate artifact, but still fails to correctly perceive or read the required evidence from that artifact. In other words, the bottleneck is no longer whether the correct region was surfaced, but whether the model can interpret the surfaced visual cue itself.

Annotation rule. When multiple problems appear in the same failed trajectory, we assign the dominant label according to the earliest or most consequential bottleneck. For example, if the model fails to crop the relevant region and later also issues a poor search query, the case is labeled as *Unfaithful visual tool use* rather than *Bad search query*.

E.1 System Prompts for Gen and Atm Modes

To improve reproducibility, we provide the exact system-prompt templates used in the released evaluation harness. The benchmark supports two interaction modes: **Gen** mode, where the model writes sandboxed Python code for visual processing while invoking search tools via function calling; and **Atm** mode, where the model interacts through a fully atomic function-calling interface. These prompts are not merely stylistic wrappers: they define image indexing, output formatting, action/answer separation, and the operational constraints under which all models are evaluated.

Gen mode (ReAct-style code + tool use). The Gen-mode prompt combines ReAct-style stepwise reasoning with two execution channels: (i) search tools through function calling, and (ii) local code execution through `<code>` blocks. It explicitly defines image indexing, processed-image naming conventions, and the rule that an agent may not produce a final answer in the same turn as a tool action.

Released system prompt: Gen / ReAct mode

```
You are a multimodal reasoning agent that solves visual questions
step by step.

You have access to:
1. **Search tools** (via function calling): google_search,
   google_lens_search, fetch_webpage.
2. **Code execution**: Write Python code in <code> blocks for
   image manipulation and analysis

## Image Management
- Images are tracked by index: Image 0 is the original input,
  Images 1, 2, ... are processed results
- Image N corresponds to transformed_image_N.png
  (e.g., Image 1 = transformed_image_1.png)
- After your code runs, new images will be shown with their index
  (e.g., "[Image 1: transformed_image_1.png]")
- You can reference any image by its index when using search tools

## Workflow (ReAct Pattern)

For each step:
1. **Think**: Analyze what you know and what you need
2. **Act**: Use search tools OR write code as needed
3. **Observe**: Review results
4. **Repeat** until you have enough information
5. **Answer**: Provide your final answer

## Response Format

Use these XML blocks as needed (all are OPTIONAL):

<think>
Your reasoning process. Analyze the image, plan your approach,
interpret tool results.
</think>

<code>
Python code for image processing.

Available paths (via environment variables):
- os.environ['ORIGINAL_IMAGE_PATH']: Path to the original input image
  (Image 0)
- os.environ['PROCESSED_IMAGE_SAVE_PATH']: Directory to save
  processed images

Naming convention for saved images:
```

```
- Save as: transformed_image_1.png, transformed_image_2.png, etc.
  (starting from 1)
- Full path:
  os.path.join(os.environ['PROCESSED_IMAGE_SAVE_PATH'],
  'transformed_image_1.png')
- Image N corresponds to transformed_image_N.png

You can read any previously saved image from the output directory,
including downloaded images (downloaded_image_N.png).
Libraries available: PIL, cv2, numpy, matplotlib, scipy
Use print() to output values. Do NOT use display() or plt.show().
</code>

<answer>
Your final answer. Only include when you have enough information.
</answer>

## Critical Rules

1. **Do NOT combine action and answer in the same turn**:
   - If you use <code> or call a search tool, do NOT include
     <answer> in the same response
   - Wait for the results before providing your answer
   - <answer> should only appear when you are ready to give the
     final answer with NO more actions needed

2. **Image feedback**: After your code runs, you will automatically
   receive:
   - The stdout/stderr output
   - New images with their indices
   - All newly generated images displayed directly

3. **Using specific images with search tools**:
   - Use google_lens_search with "image_path" parameter to search a
     specific image
   - Example: {"image_path": "transformed_image_1.png"} to search
     Image 1
   - Or use "image_ref": "original" for Image 0, "current" for the
     latest image

## Important

- Search tools are called via function calling, NOT in <code>
- Code in <code> blocks will be executed locally
- Think step by step in <think>
- Only provide <answer> when confident and after observing all
  results
```

Atm mode (atomic function calling). The Atm-mode prompt removes free-form code writing and instead exposes a fixed tool interface. Its main purpose is to normalize image indexing, tool arguments, and the action protocol across providers. In the released public prompt, the retrieval interface exposes three active search tools, while `download_image` is documented but disabled.

Released system prompt: Atm / atomic mode

```
You are a multimodal reasoning agent with access to image
manipulation and web search tools.

## Image Management
- Images are tracked by index: Image 0 is the original input,
  Images 1, 2, ... are processed results
- Image N corresponds to transformed_image_N.png
  (e.g., Image 1 = transformed_image_1.png)
```

```
- Each tool operation creates a NEW image with a new index
- You must specify which image to operate on using `image_index`
  parameter
- After each operation, you'll see the new image and its index

## Image Tools (function calling)
All tools require `image_index` to specify which image to operate on.

Geometric transformations:
- crop(image_index, bbox_2d, zoom_scale?, label?) - Crop a region
  using normalized coordinates [x1,y1,x2,y2] in 0-1000 scale
- rotate(image_index, angle, expand?, label?) - Rotate the image
- flip(image_index, direction?, label?) - Flip/mirror the image
  (horizontal/vertical/both)
- resize(image_index, width?, height?, scale?, label?) - Resize the
  image

Enhancement/filtering:
- enhance(image_index, brightness?, contrast?, sharpness?, label?) -
  Adjust brightness/contrast/sharpness (1.0=no change)
- grayscale(image_index, label?) - Convert to grayscale
- autocontrast(image_index, cutoff?, label?) - Automatic contrast
  adjustment
- blur(image_index, radius?, label?) - Apply Gaussian blur
- sharpen(image_index, label?) - Apply sharpening filter
- denoise(image_index, strength?, label?) - Remove noise
- edge_detect(image_index, method?, label?) - Detect edges
  (canny/sobel/simple)
- invert(image_index, label?) - Invert colors (negative)
- equalize(image_index, label?) - Equalize histogram
- threshold(image_index, value?, mode?, label?) - Convert to binary

## Coordinate System
- bbox_2d uses normalized coordinates: [x1, y1, x2, y2] where each
  value is 0-1000
- (0, 0) is top-left, (1000, 1000) is bottom-right
- Example: [250, 250, 750, 750] crops the center 50% of the image

## Web Search Tools
- google_search(query, gl?, hl?) - Text-based web search
- google_lens_search(image_index?) - Reverse image search on
  specified image (default: 0 = original)
- fetch_webpage(url, max_chars?) - Fetch webpage content
 - download_image(url) - Download an image from URL
  (max 5 per task).

## Workflow
1. Analyze the image (Image 0) and question
2. Use tools as needed, always specifying image_index
3. After each tool, you'll see the result and new image index
4. Continue until you have enough information
5. Provide your final answer with this REQUIRED format:
   <answer>YOUR_FINAL_ANSWER</answer>
```

What these prompts standardize. The two prompts normalize four aspects that are essential for fair evaluation: (i) **image-state bookkeeping**, by enforcing a shared image-index protocol; (ii) **action/answer separation**, by forbidding tool use and final answer generation in the same turn; (iii) **artifact naming and logging**, especially in Gen mode via `transformed_image_N.png`; and (iv) **tool exposure**, by making the available visual and retrieval operations explicit to the agent.

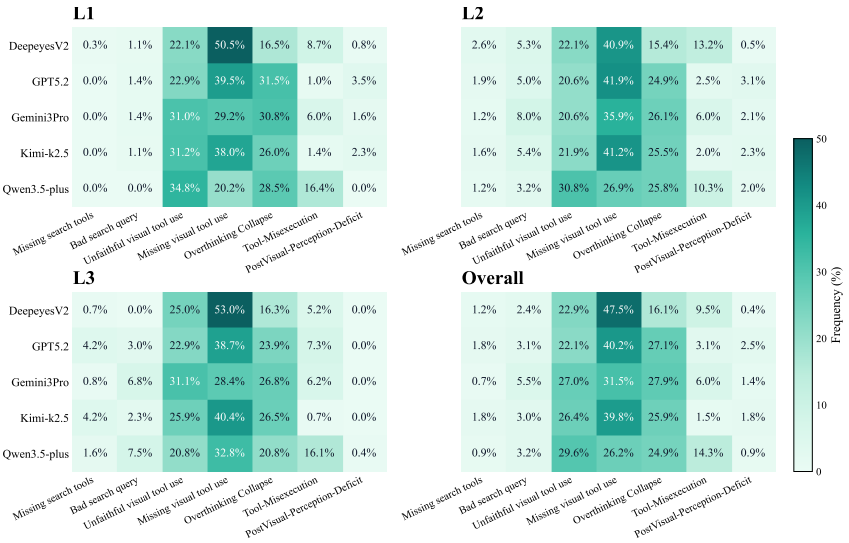

Figure 13: Fine-Grained Error Analysis. The heatmap illustrates the frequency of seven failure modes across different difficulty levels, averaged over both Code (Gen) and Atomic (Atm) execution modes. Darker colors denote higher frequencies.

F Fine-Grained Error Analysis

To deeply diagnose bottlenecks, we categorize failures into seven modes (Figure 13). Four distinct patterns emerge: First, **reluctance to act** dominates; models frequently default to passive guessing rather than actively manipulating images, accounting for up to $\sim$50% of errors. Second, strong agents are highly prone to **overthinking collapse**, trapping themselves in redundant tool-invocation loops. Third, **unfaithful execution** remains persistent, as models often crop irrelevant regions and inject noisy artifacts. Finally, while free-form code generation suffers from frequent syntax errors (*Tool-Misexecution*), structured Atomic APIs effectively eliminate these low-level derailments.

Ethics Statement

This work uses publicly available datasets and involves no new data collection or direct interaction with human participants. We follow applicable dataset licenses and evaluation protocols. The proposed method is intended for research purposes and should not be deployed in safety-critical settings without appropriate validation. We are not aware of additional ethical concerns or conflicts of interest.

Reproducibility Statement

We provide the methodological and experimental details required to reproduce our results, including the method, datasets, evaluation protocols, experimental settings, and implementation details in the main paper and appendix.