VISUAL REASONING AT THE HARNESS LEVEL: A LADDER FOR CONTROLLED EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Advances in foundation models are shifting coding and working agents from predefined tools toward general computational environments. Yet existing visual agents often couple model, harness, and training choices, making it difficult to determine how this transition affects visual reasoning. In this work, we introduce a unified evaluation framework to study how harness design allocates responsibility between the model and its environment. We instantiate this framework as a seven-rung harness ladder organized around expanding computational access and delegating state and evidence management. The ladder connects direct answering, predefined visual tools, generated code, and general workspaces under fixed models and tasks. We further construct VHStar (Visual Harness with Stratified Tasks for Agentic Reasoning), a benchmark of 600 tasks across five visual domains and four calibrated difficulty levels. Experiments across nine models show that code execution with a persistent kernel improves all four full-ladder models over fixed tools, whereas further workspace delegation offers no consistent advantage. Models with similar direct-answer performance benefit differently from computational access, motivating harness designs that expand computational freedom while retaining appropriate support for state and evidence.

1 INTRODUCTION

Visual reasoning aims to solve problems grounded in visual information. In the direct setting, a multimodal large language model answers directly from the given image (Liu et al., 2023; Bai et al., 2025). A growing line of work on *agentic visual reasoning* instead interleaves reasoning with interaction: the model invokes predefined visual operations (Hu et al., 2024; Zheng et al., 2025) or generates programs that construct its own tools (Zhao et al., 2025), and receives new observations between reasoning steps. This paradigm has improved performance on a wide range of tasks, such as mathematical diagrams, charts, and fine-grained perception.

Beyond visual reasoning, advances in foundation models are enabling agents to take greater control over task execution. Coding and working agents illustrate this shift: through general-purpose interfaces such as a shell over a persistent workspace, models compose operations and manage multi-step workflows (Yang et al., 2024; Wang et al., 2024c). Such systems move from tool-centric designs with predefined operations toward environment-centric designs in which models construct task-specific procedures. The *harness* governs this interaction through its action interface, external state, observation return, and control rules (Tang et al., 2026; Li et al., 2026; Lee et al., 2026), defining how responsibility is divided between the environment and the model.

However, existing visual reasoning research has focused more on expanding tool use and generated computation than on the broader transfer of responsibility seen in coding and working agents (Zheng et al., 2025; Lai et al., 2025; Zhao et al., 2025). Existing studies and benchmarks demonstrate progress within individual systems, but provide limited comparative evidence on how model capabilities and harness design jointly shape the benefits of greater autonomy (Hong et al., 2025; Li et al., 2025; Guo et al., 2025a). This motivates us to ask: *how should harnesses balance computational freedom and environmental support to enable effective agentic visual reasoning?*

We study this allocation of responsibility through a seven-rung harness ladder, holding the model and tasks fixed (Section 3). The ladder spans direct answering, predefined visual tools, generated code with a persistent kernel, and general workspaces with greater model control. Each rung combines

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.

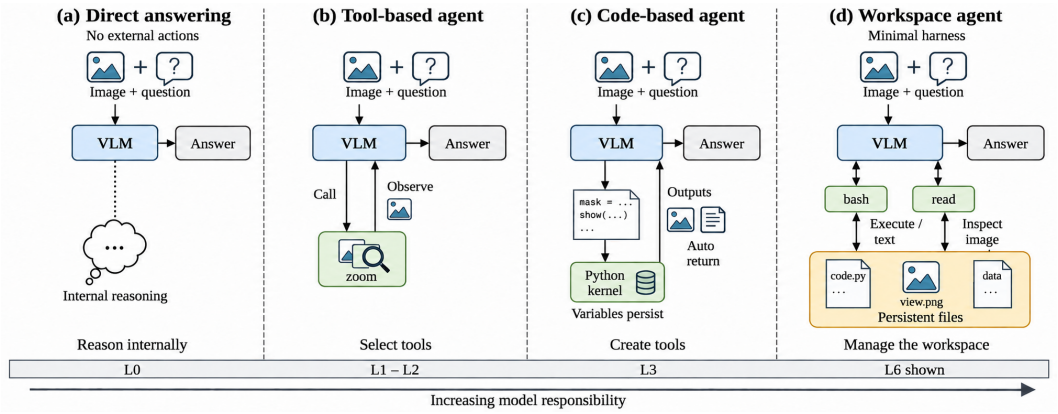


Figure 1: Visual-reasoning interaction paradigms. Direct answering uses internal reasoning; tool-based agents select supplied visual operations, as in DeepEyes (Zheng et al., 2025); code-based agents construct computations with execution feedback, as in PyVision (Zhao et al., 2025); workspace agents additionally manage programs, files, and image inspection. The last panel illustrates L6; L4–L5 bridge the code and workspace regimes. Arrows beneath the panels indicate increasing responsibility, not performance or model chronology. These are interface correspondences, not reproductions of training recipes.

settings of the four harness design axes; comparisons measure the effects of these configurations. All interactive rungs share a common interaction budget, and no configuration requires additional training. This framework connects existing visual reasoning paradigms (Hu et al., 2024; Zhao et al., 2025) and enables evaluation of the shift from fixed tools to model-controlled environments.

Evaluating a harness requires identifying which tasks benefit from it and whether models adapt their interaction to task difficulty. We therefore construct VHStar, a diagnostic benchmark of 600 open-ended tasks from twelve public sources (Section 4). Tasks are grouped into five visual domains to compare harness effectiveness across task types. Four difficulty levels (D1–D4) support analysis of how interaction changes with task difficulty. These levels are calibrated using a fixed model panel answering without tools. Joint analysis of task performance and interaction behavior distinguishes performance gains and regressions from changes in tool use.

We conduct extensive experiments with nine multimodal models, evaluating four across the complete harness ladder (Section 5). Our results show that computational access and delegated control yield distinct returns in visual reasoning. Code execution with a persistent kernel improves all full-ladder models over fixed visual tools, whereas further workspace delegation offers no consistent advantage. Models with similar direct-answering scores obtain sharply different gains from the same harness, and gains on difficult tasks can coexist with lower overall performance. These findings show that harness design must account for both the model's ability to exploit computation and the environmental support needed to use it effectively.

Our contributions are summarized as follows:

1. We establish a unified framework for studying responsibility allocation in visual reasoning, enabling controlled comparisons from predefined tools to model-controlled workspaces.
2. We introduce VHStar, a benchmark of 600 open-ended tasks stratified by visual domain and difficulty to diagnose harness effectiveness and interaction behavior.
3. We identify consistent code-and-kernel gains over fixed tools, uneven returns to workspace delegation, and substantial model differences concealed by direct-answering scores.

2 Related Work

Multimodal large language models. Multimodal large language models connect visual representations with language generation for visual understanding and reasoning (Bai et al., 2025). Early models emphasize visual dialogue and instruction following without external tool feedback (Liu et al., 2023; Zhu et al., 2023). Post-training extends these capabilities to active perception (Zheng

et al., 2025) and graphical user interaction (Qin et al., 2025), while native multimodal models combine joint vision–language pretraining with instruction tuning and reinforcement learning to support broader visual agent capabilities (Kimi Team, 2026). Visual benchmarks cover mathematical diagrams, charts, documents, and fine-grained detail (Lu et al., 2024; Wang et al., 2024d; Roberts et al., 2025). As models become more capable of visual computation, evaluation must also examine how their environments expose actions and return evidence.

Agentic visual reasoning. Early visual programming systems compose specialist modules or generate programs without the language model seeing the image (Gupta & Kembhavi, 2023; Suris et al., 2023). Later approaches bring perception into the reasoning loop: Visual Sketchpad generates intermediate images (Hu et al., 2024); reinforcement learning develops zooming, cropping, and visual search (Zheng et al., 2025; Lai et al., 2025; Hong et al., 2025); and training-free systems such as PyVision let frozen models construct task-specific tools in Python (Zhao et al., 2025). Benchmarks evaluate a fixed tool configuration per system (Li et al., 2025; Guo et al., 2025a), while surveys organize interaction patterns with the harness taken as given (Su et al., 2025). These studies establish the value of tool-mediated reasoning and generated computation. We examine whether these benefits persist as harnesses delegate workspace storage and evidence selection to the model.

Harness engineering. Coding and working agents treat their action interface, external state, observation return, and execution control as a configurable *harness* (Tang et al., 2026; Li et al., 2026). Interface studies show that these choices affect outcomes at fixed weights (Yang et al., 2024; Wang et al., 2024b), while general-purpose runtimes make them reusable (Wang et al., 2024c; Anthropic, 2025). Meta-Harness optimizes the harness end-to-end for a frozen model (Lee et al., 2026), a single generalist harness provides a competitive baseline across agentic task suites (Soni et al., 2025), and recent benchmarks measure whether models can improve the harness itself (Huang et al., 2026; Song et al., 2026). In visual reasoning, SpatialClaw compares code with alternative action interfaces using a persistent kernel and restricted file access (Cho et al., 2026); MUSE combines task-specific perception, deterministic verification, and repair (Lu et al., 2026). These works address action representation and verifier-supported reasoning. We complement them by comparing fixed toolkits, code-and-kernel execution, and file-based workspaces across visual domains with the same models, tasks, and step limit. This distinguishes returns to computational access from those to further state and observation delegation. Computer-use benchmarks focus on operating visual interfaces (Xie et al., 2024; Koh et al., 2024). Our ladder (Section 3) and stratified benchmark (Section 4) support this study.

3 A Harness Ladder for Visual Reasoning

An agentic visual reasoning system combines a foundation model with a supporting *harness* (Tang et al., 2026; Li et al., 2026). The model provides visual understanding, reasoning, and generation; the harness governs its actions, state, observations, and execution. Harness design therefore allocates both computational freedom and responsibility for managing evidence. We make these choices comparable through a shared design space and a seven-rung ladder connecting predefined tools, generated programs, and general workspaces.

Figure 1 shows the shift from direct answering to tool-, code-, and workspace-based interaction.

3.1 Harness Design Space

Formally, a task is a pair $x = (q, I)$ comprising a question and its original images. A model M either answers directly or generates an interaction trajectory τ

$$\tau = (q, I, a_1, o_1, \ldots, a_T, o_T, y), \tag{1}$$

in which the harness executes each proposed action a_t, returns an observation o_t, and the loop terminates in an answer y. Four components specify the harness design:

$$H = (\mathcal{A}, \mathcal{S}, \mathcal{O}, \mathcal{C}). \tag{2}$$

(1) *Action interface* $\mathcal{A}$ specifies the operations available to the model, from none, to a predefined visual toolkit, to open-ended program execution. (2) *External state* $\mathcal{S}$ determines whether and where

Table 1: Harness configurations across the four design axes (Eq. 2). Ops denotes the predefined visual toolkit; operation definitions and interface conventions appear in Appendix B.1.

	Action $\mathcal{A}$	Persistent state $\mathcal{S}$		Observation $\mathcal{O}$	Control $\mathcal{C}$
Rung	Interface	Kernel	Files	Derived-image access	Model-controlled decisions
L0	Direct answer	–	–	–	–
L1	Single operation	–	–	Automatic	Region selection
L2	Multiple operations	–	–	Automatic	Operation selection
L3	Ops + code execution	Yes	Prohibited	Automatic	Operation selection; program construction
L4	Ops + code execution + `read`	Yes	Yes	Explicit `read`	Operation selection; program construction; artifact inspection
L5	Ops + code execution + `read`	No	Yes	Explicit `read`	Operation selection; program construction; artifact inspection
L6	Shell execution + `read`	No	Yes	Explicit `read`	Command construction; artifact inspection

intermediate results persist across actions. A persistent kernel retains variables in memory, while a workspace retains files; conversation history is available independently of either form. (3) *Observation return* $\mathcal{O}$ determines which outputs become available to subsequent reasoning and in what form. Text may return through tool responses, while derived images may be attached automatically or saved for explicit inspection. Executing a computation and observing its result are therefore distinct events. (4) *Execution control* $\mathcal{C}$ specifies the rules under which the model selects actions, inspects results, and stops, including a limit of B interaction steps.

3.2 From Computational Access to Delegated Control

Existing agent paradigms occupy different regions of this design space. Visual reasoning systems let models select visual operations or generate programs while prescribing how observations return to the context (Hu et al., 2024; Zhao et al., 2025; Hong et al., 2025). Coding and working agents delegate more execution and workspace management through general-purpose interfaces (Yang et al., 2024; Anthropic, 2025). This distinction motivates two questions for visual reasoning: what does a model gain from access to general computation, and do workspace configurations that delegate state and evidence management offer further gains?

We organize the ladder around these two transitions. Starting from direct answering, predefined tools introduce visual interaction, and generated programs extend the model's computational repertoire beyond a fixed menu. Workspace configurations then place intermediate results and image inspection under greater model control. Intermediate rungs connect these paradigms by varying how computation, state, and observations are organized. The resulting ladder spans responsibility allocations; its indices identify configurations rather than ranks of capability.

We hold the model and tasks fixed across the ladder, use a common budget for interactive configurations, and require no configuration-specific training. Direct answering anchors the benefit of interaction; comparisons among interactive rungs reveal the returns to different responsibility allocations. Repeating these comparisons across models, domains, and difficulty levels maps where each harness regime is effective. Section 4 introduces the benchmark and evaluation protocol.

3.3 Representative Harness Configurations

Table 1 maps the seven configurations, L0–L6, to the four design axes. Interactive configurations share stopping rules and budgets; the control column lists model-controlled decisions. These configurations represent interface regimes rather than reproductions of specific systems.

Direct answering and fixed tools (L0–L2). L0 answers from the original images in a single response. L1 exposes a single visual operation, *zoom*, for iterative inspection of model-selected re-

gions, a central pattern in *thinking with images* (OpenAI, 2025; Zheng et al., 2025; Lai et al., 2025). L2 expands this interface to eight predefined visual operations (*Ops*), retained in L3–L5. We select a compact repertoire covering region inspection, text extraction, quantitative comparison, and structural analysis, reflecting recurring visual computations in prior agents (Zhao et al., 2025). This design tests selection from a fixed repertoire before introducing model-constructed computation. L1 and L2 return derived images automatically, so their comparison tests menu breadth under the same observation policy. We present operation definitions in Appendix B.1.

Generated code (L3). L3 adds open-ended Python execution and a persistent kernel while retaining the fixed tool menu. The model constructs task-specific computations and retains variables across executions, with text and displayed images returned automatically. Persistent state is confined to the kernel; agent-written files are unavailable. L3 thus delegates computation construction while keeping image return automatic.

Workspace control (L4–L6). L4 retains the kernel and adds persistent files, while replacing automatic image return with explicit `read` actions. The model therefore decides which artifacts to preserve and inspect. L5 removes kernel persistence while retaining files across actions. L6 retains this file-based state and replaces the predefined visual tools and Python interface with `bash` and `read`, through which the model constructs its own visual operations. The ladder compares complete harness configurations rather than isolated components.

4 THE VHSTAR BENCHMARK

To evaluate the harness ladder, we construct VHStar around two complementary dimensions: visual domains capture different reasoning demands, while calibrated difficulty captures accessibility to direct answering. Their combination reveals where interaction extends visual reasoning and where existing capabilities remain sufficient.

4.1 TASK COMPOSITION

VHStar comprises 600 open-ended tasks from twelve public sources. Following prior multimodal benchmark consolidation (Lu et al., 2024), we curate five domains: diagram mathematics (M1), charts, documents, and text (M2), spatial and multi-view reasoning (M3), fine-grained perception (M4), and hard mixed reasoning (M5). Domain assignments follow source task categories.

Selection follows domain–difficulty quotas, retaining both directly accessible tasks and tasks with little observed direct success. M1–M3 each contain 120 tasks distributed equally across difficulty levels; M4 contains 150 tasks for broader perception coverage, while M5 contains 90 tasks concentrated at higher difficulty. This preserves coverage without imposing equal difficulty proportions on every domain. To limit repeated content, selection retains at most one task per group sharing source material, such as a figure, problem, or scene. Source counts and the complete domain–difficulty distribution appear in Appendix A.1.

4.2 DIFFICULTY CALIBRATION AND QUALITY CONTROL

We calibrate difficulty using a fixed four-model panel, with each model answering each candidate three times without tools. Tasks are assigned to D1–D4 by the fraction of usable responses receiving full credit: D1 denotes frequent direct success, D2 and D3 progressively lower success, and D4 no observed success. Calibration responses are separate from evaluation trajectories, and labels remain fixed across harnesses. Difficulty is therefore relative to the calibration panel; D4 does not imply intrinsic unsolvability. These strata provide a common reference for testing whether computational gains extend to harder tasks.

However, low direct-answering success may reflect noisy questions or incorrect reference answers, while successful responses may rely on textual shortcuts rather than visual evidence. We therefore complement difficulty calibration with checks of visual necessity and reference reliability. Following MMStar (Chen et al., 2024), text-only screening removes candidates repeatedly answered correctly without their images. When independent panel responses converge on an alternative answer, a

Table 2: Main results across the seven harness configurations. Entries are mean task scores (%); bold marks the highest observed mean for each model.

	Direct	Fixed tools		Code	Workspace		
Model	L0	L1	L2	L3	L4	L5	L6
DeepSeek-V4.1	31.3	46.9	47.7	**55.1**	49.8	48.7	51.4
GPT-5.6-Luna	34.7	39.4	35.9	**43.9**	41.4	40.1	40.4
GLM-5.3-Flash	39.0	43.3	43.7	53.7	**54.4**	52.0	53.7
Qwen3.8-Flash-Next	40.9	51.9	52.6	58.5	**59.6**	57.5	54.5

model outside the panel solves the task without seeing the reference or panel answers. Challenged or unresolved references are removed and replaced within the same domain–difficulty cell, preserving composition. Calibration and quality-control protocols appear in Appendices A.2 and A.3.

4.3 Evaluation Protocol

The main comparison evaluates each configuration on the same tasks using one trajectory per task, independently of difficulty calibration. We assess final answers against reference answers following the source benchmarks' evaluation criteria. We use rule-based scoring where applicable and an LLM judge for semantic equivalence. We report *mean task score* (%) overall and by domain and difficulty. Each *interaction step* corresponds to one model response.

5 Experiments

We examine the returns to computational freedom and delegated control (RQ1), model differences in realizing these returns (RQ2), and the contribution of returned evidence (RQ3), followed by performance–token trade-offs and evidence verification.

5.1 Experimental Setup

Models and harnesses. We evaluate Qwen3.8-Flash-Next, DeepSeek-V4.1, GLM-5.3-Flash, and GPT-5.6-Luna across all seven configurations (Section 3), adding Qwen3.5-27B, Qwen3.6-27B, Qwen3.8-27B, Gemma-4-31B-IT, and MiniMax-M3 at L0/L3/L6. Difficulty analysis covers all nine models; domain analysis uses the four full-ladder models. Unless a size is specified, Qwen denotes Qwen3.8-Flash-Next (Qwen-FN in figures). Code-enabled configurations share an isolated Python environment. Model, environment, and scoring details appear in Appendices B.2 and B.3.

Evaluation settings. We follow Section 4.3, allowing up to 20 interaction steps with optional tool use. Deployment, reasoning settings, and image-input bounds remain fixed within each model's comparison. Confidence intervals describe paired task variation (Appendix C.1).

5.2 RQ1: Computational Freedom and Delegated Control

We compare complete configurations (Section 3.3) to distinguish computational gains from those of workspace delegation. We present full-ladder, nine-model, and domain results in Tables 2–4, respectively.

First, computational access yields more consistent gains than workspace delegation. Code execution with a persistent kernel improves all four full-ladder models over the fixed toolkit, whereas expanding the operation menu yields smaller or negative changes. Further delegation has mixed returns at L4, and L6 scores below L3 for eight models while approximately matching GLM.

Domain results reveal differences concealed by aggregate scores. GLM's nearly unchanged overall L3/L6 score masks lower mathematics and higher spatial performance. From L3 to L6, mathematics declines across all four models, while spatial changes have mixed signs.

Table 3: Performance and interaction across nine models at L0, L3, and L6. Scores and mean steps are reported by difficulty; All denotes the overall score, and Δ_{L0} is its change from direct answering. Bold marks the best scores within each model at the displayed precision.

		Score (%)					Change (pp)	Mean steps			
Model	Rung	D1	D2	D3	D4	All	Δ_{L0}	D1	D2	D3	D4
MiniMax-M3	L0	58.4	29.1	11.1	0.9	21.3	0.0	1.0	1.0	1.0	1.0
	L3	**68.0**	**39.4**	**22.8**	4.6	**29.6**	+8.3	8.3	12.7	13.1	16.3
	L6	52.8	34.6	21.2	**4.9**	25.1	+3.8	5.2	5.8	8.7	12.0
Gemma-4-31B-IT	L0	**62.4**	**35.4**	13.1	2.1	**24.4**	0.0	1.0	1.0	1.0	1.0
	L3	56.0	29.9	**15.4**	2.4	22.6	−1.8	6.6	7.0	7.0	9.9
	L6	48.8	26.0	12.6	**2.6**	19.6	−4.8	3.0	3.4	4.0	5.2
DeepSeek-V4.1	L0	74.4	45.7	21.3	2.9	31.3	0.0	1.0	1.0	1.0	1.0
	L3	**96.8**	**74.0**	**57.5**	**15.8**	**55.1**	+23.8	9.8	13.8	15.1	17.4
	L6	92.8	70.1	49.5	15.5	51.4	+20.1	9.9	14.5	15.8	17.6
Qwen3.6-27B	L0	79.2	48.0	24.3	0.9	32.9	0.0	1.0	1.0	1.0	1.0
	L3	**81.6**	**52.0**	**30.9**	**11.4**	**39.4**	+6.5	6.2	8.9	10.4	14.4
	L6	76.0	41.7	25.2	5.3	32.6	−0.3	3.7	4.9	5.6	7.9
Qwen3.5-27B	L0	**80.0**	**52.8**	21.5	2.9	**34.1**	0.0	1.0	1.0	1.0	1.0
	L3	60.0	42.5	**26.9**	**4.4**	29.5	−4.5	6.0	7.7	7.8	12.1
	L6	63.2	29.9	18.7	3.6	25.3	−8.8	4.0	4.5	4.5	6.1
GPT-5.6-Luna	L0	81.6	50.4	25.2	2.8	34.7	0.0	1.0	1.0	1.0	1.0
	L3	**84.0**	**64.6**	**42.2**	**7.4**	**43.9**	+9.2	3.4	4.5	5.3	8.3
	L6	83.2	58.3	35.5	6.3	40.4	+5.7	4.3	5.2	6.3	8.8
Qwen3.8-27B	L0	84.0	52.0	24.6	2.3	35.3	0.0	1.0	1.0	1.0	1.0
	L3	91.2	74.0	**58.0**	**18.9**	**55.1**	+19.9	6.2	8.8	10.5	13.6
	L6	**93.6**	**75.6**	47.8	15.7	52.4	+17.1	7.5	9.6	12.3	16.5
GLM-5.3-Flash	L0	83.2	62.2	28.3	4.8	39.0	0.0	1.0	1.0	1.0	1.0
	L3	**91.2**	73.2	54.0	**18.2**	**53.7**	+14.7	6.0	10.1	12.2	14.7
	L6	**91.2**	**74.0**	**57.9**	14.8	**53.7**	+14.7	7.0	10.1	12.3	15.7
Qwen3.8-Flash-Next	L0	91.2	62.2	30.8	3.7	40.9	0.0	1.0	1.0	1.0	1.0
	L3	**92.8**	**85.8**	55.3	**22.3**	**58.5**	+17.6	5.1	7.4	9.2	13.0
	L6	89.6	75.6	**56.5**	18.0	54.5	+13.5	5.3	7.9	9.3	13.7

We test the value of predefined tools by removing the L3–L5 operation menu for Qwen, DeepSeek, and Luna, holding other settings fixed (Appendix Table 13). Removal lowers DeepSeek's L3 score but raises its L5 score; Qwen's largest decrease occurs at L4. Predefined tools thus remain selectively useful, depending on the model and harness.

5.3 RQ2: Model Capabilities Beyond Direct Answering

We compare model gains over direct answering at L3/L6 and present difficulty-conditioned results in Table 3, with Qwen-27B gains and program-use profiles in Figure 2.

Similar direct scores conceal substantial differences in computational benefit. Qwen3.5-27B declines under both L3 and L6, Qwen3.6-27B benefits at L3 but remains near its direct score at L6, and Qwen3.8-27B benefits strongly under both. Their L3 gains range from -4.5 to $+19.9$ points.

Code generation alone does not ensure effective computation. Qwen3.5-27B invokes tools extensively and constructs visual programs at L3, yet recognized operation patterns appear in its L6 submissions on only 6.5% of tasks, compared with 67.7% for Qwen3.8-27B. Qwen3.6-27B likewise shows a marked gap between L3 and L6 use (Appendix C.4). Broader program use accompanies the shift toward positive interaction gains.

Table 4: Performance of L0, L3, and L6 across domains M1–M5 (Section 4.1). Entries are mean task scores (%); bold marks the highest score within each model and column.

		Score (%)					
Model	Rung	M1	M2	M3	M4	M5	All
DeepSeek-V4.1	L0	42.5	29.2	42.5	31.3	4.4	31.3
	L3	**65.0**	57.5	**60.8**	**63.9**	**16.7**	**55.1**
	L6	55.8	**59.2**	55.0	63.7	10.0	51.4
GPT-5.6-Luna	L0	43.3	40.8	32.5	40.3	**8.9**	34.7
	L3	**48.3**	**52.5**	42.5	**56.4**	7.8	**43.9**
	L6	40.0	43.3	**45.0**	55.1	6.7	40.4
GLM-5.3-Flash	L0	40.0	46.7	44.2	46.7	7.8	39.0
	L3	**67.5**	54.2	49.2	**68.3**	16.7	**53.7**
	L6	60.0	**57.5**	**55.8**	65.0	**18.9**	53.7
Qwen3.8-Flash-Next	L0	48.3	45.0	54.2	42.3	5.6	40.9
	L3	**71.7**	**63.3**	63.3	**70.6**	**7.8**	**58.5**
	L6	66.7	58.3	**64.2**	61.8	7.8	54.5

Longer interaction does not ensure higher overall scores. All nine models use more steps on harder tasks at L3/L6 and improve D3/D4 scores at L3 relative to L0. Qwen3.5-27B and Gemma nevertheless regress overall through lower D1/D2 scores. All four full-ladder models retain more initially correct D1 answers at L3 than under repeated direct answering (Appendix Table 12).

5.4 RQ3: From Computation to Visual Evidence

We ablate feedback at L3 and L6: text-only feedback removes derived images, while image-only feedback reduces tool text to completion/error status. Original images and available actions remain fixed. We present the results in Figure 3.

Visual computation supports reasoning through images and text. Removing tool text lowers every model's score at L3 and L6 despite retaining derived images; removing derived images lowers every L3 score despite retaining original images. The channels' relative value depends on the model and interface: at L6, text removal produces larger losses than image removal for Qwen, DeepSeek, and Luna, while GLM shows the opposite ordering. Luna changes little without derived images at L6.

Trajectories show how programs convert visual content into measurements, coordinates, mask statistics, and geometric relations for numerical and symbolic reasoning. Verified L6 traces show Qwen, DeepSeek, and GLM reusing saved numerical results (Appendix C.4). These observations motivate preserving visual and textual evidence, including execution diagnostics, for reuse and validation.

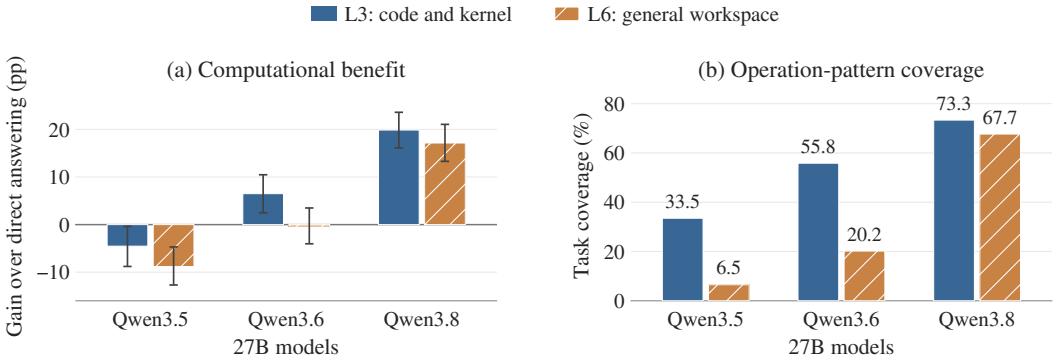


Figure 2: Computational benefit and program use across Qwen-27B models. (a) Gains over direct answering with 95% confidence intervals. (b) Task coverage of operation patterns in submitted code, including failed executions; coverage measures observed use rather than operation correctness.

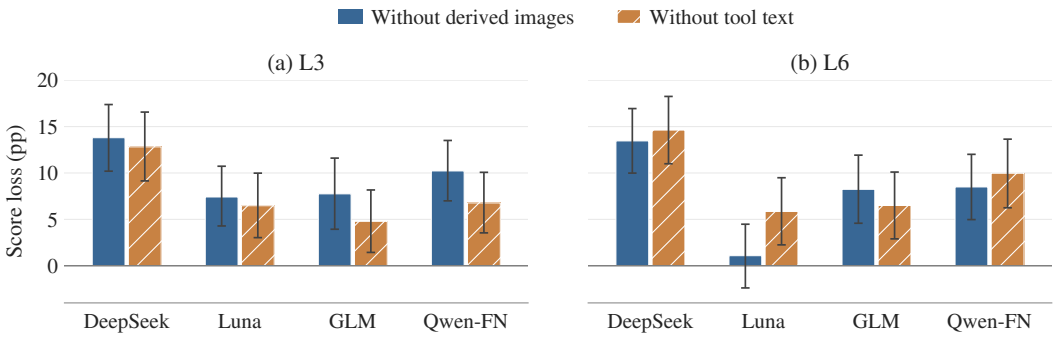


Figure 3: Observation-channel ablations at L3 and L6. Bars show score losses relative to full feedback; error bars indicate 95% confidence intervals. Both panels use the same scale.

Table 5: Score and token consumption across harnesses. Bold indicates the best displayed score for each model.

	L2		L3		L6	
Model	Score (%)	Tokens (k)	Score (%)	Tokens (k)	Score (%)	Tokens (k)
DeepSeek-V4.1	47.7	280.3	**55.1**	224.3	51.4	211.5
GLM-5.3-Flash	43.7	265.0	**53.7**	289.9	**53.7**	275.3
Qwen3.8-Flash-Next	52.6	133.7	**58.5**	124.4	54.5	113.8

5.5 Further Analysis: Interaction and Verification

We compare L2, L3, and L6 across Qwen, DeepSeek, and GLM under the standard interaction budget, reporting scores and mean total tokens per task in Table 5.

Computational freedom can improve performance without increasing token use. L3 outperforms L2 with fewer tokens for Qwen and DeepSeek; GLM uses more tokens for higher scores. L6 reduces token use relative to L3 for all three, but lowers Qwen's and DeepSeek's scores while approximately preserving GLM's. Further delegation thus introduces a model-dependent performance–token trade-off. Expanded comparisons and budget scaling appear in Appendices D.1 and C.6.

Local tests reveal a trade-off between first-use latency and repeated computation: L6 has lower first-use latency on the matched workload, whereas persistent kernels accelerate repeated computation, as confirmed by matched L4/L5 tests (19 versus 266 ms; Appendix D.2).

Reliable execution does not ensure valid visual conclusions. Wrong answers dominate non-full-credit outcomes at L3 and L6, often without recorded execution incidents at L6 (Appendix Figure 4). GLM uses a geometric residual to revise object extraction and solve a multi-view task; DeepSeek repeatedly inspects crops but returns an explicitly unverified, incorrect count (Appendix E.2). These cases motivate consistency checks, alternative measurements, and revision triggered by conflicting evidence, complementing visual critique and verifier-guided repair (Wu et al., 2025; Lu et al., 2026).

6 Conclusion

We present a harness ladder and VHStar to study responsibility allocation in visual agents. Our comparisons distinguish the ability to benefit from visual computation from the ability to retain those gains under broader workspace delegation. Code-and-kernel execution consistently improves over fixed tools across the full-ladder models, while further delegation yields uneven returns. Similar direct-answer performance can conceal substantial differences in these benefits. Feedback and execution analyses establish the value of informative observations and reusable state. These findings favor visual harnesses that expand computational freedom while retaining support suited to model capabilities and task demands, with evidence verification as a route to reliable autonomy.

AI USE STATEMENT

In this work, we used generative AI tools to assist with conceptual and experimental design, benchmark curation and evaluation, implementation, and result analysis and interpretation. We also used these tools for preparing figures and polishing the manuscript. AI-assisted outputs were checked through review, code testing, and consistency checks. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Claude 3.7 Sonnet and Claude Code, February 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report, 2025. URL https://arxiv.org/abs/2502.13923.

John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851. URL https://doi.org/10.1109/TPAMI.1986.4767851.

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

Seokju Cho, Ryo Hachiuma, Abhishek Badki, Hang Su, Byung-Kwan Lee, Chan Hee Song, Sifei Liu, Subhashree Radhakrishnan, Seungryong Kim, Yu-Chiang Frank Wang, and Min-Hung Chen. SpatialClaw: Rethinking action interface for agentic spatial reasoning, 2026. URL https://arxiv.org/abs/2606.13673.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. arXiv preprint arXiv:2606.19348, 2026a. URL https://arxiv.org/abs/2606.19348.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. arXiv preprint arXiv:2609.19969, 2026b. URL https://arxiv.org/abs/2609.19969.

Chengqi Duan, Kaiyue Sun, Rongyao Fang, Manyuan Zhang, Yan Feng, Ying Luo, Yufang Liu, Ke Wang, Peng Pei, Xunliang Cai, Hongsheng Li, Yi Ma, and Xihui Liu. CodePlot-CoT: Mathematical visual reasoning by thinking with code-driven images, 2025. URL https://arxiv.org/abs/2510.11718.

Gemma Team. Gemma 4 technical report. arXiv preprint arXiv:2607.02770, 2026. URL https://arxiv.org/abs/2607.02770.

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

Google DeepMind. Gemini 3.7 Flash, August 2026b. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/. Model card.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, and Rakshith Sharma Srinivasa. Beyond seeing: Evaluating multimodal LLMs on tool-enabled image perception, transformation, and reasoning, 2025a. URL https://arxiv.org/abs/2510.12712.

Zirun Guo, Minjie Hong, Feng Zhang, Kai Jia, and Tao Jin. Thinking with programming vision: Towards a unified view for thinking with images, 2025b. URL https://arxiv.org/abs/2512.03746.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2211.11559.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. DeepEyesV2: Toward agentic multimodal model, 2025. URL https://arxiv.org/abs/2511.05271.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.09403.

Lisheng Huang, Chen Yang, Hao Zhou, Huatong Song, Zongchao Chen, Ran Le, Yang Song, Wayne Xin Zhao, and Tao Zhang. Evo-Bench: Can language models improve agent harness?, 2026. URL https://arxiv.org/abs/2608.09096.

Kimi Team. Kimi K2.5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276, 2026. URL https://arxiv.org/abs/2602.02276.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks, 2024. URL https://arxiv.org/abs/2401.13649.

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

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Yingqiang Ge, Weijie Xu, et al. Agent harness engineering: A survey. https://picrew.github.io/LLM-Harness/main.pdf, 2026. Survey.

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, 2025. URL https://arxiv.org/abs/2511.01833.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Jianglin Lu, Hailing Wang, Xu Ma, Qihua Dong, Mingyuan Zhang, Yizhou Wang, and Yun Fu. MUSE: A unified agentic harness for MLLMs, 2026. URL https://arxiv.org/abs/2606.03005.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.02255.

MiniMax. MiniMax M3: Frontier coding, 1M context, native multimodality—all in one model, 2026. URL https://www.minimax.io/blog/minimax-m3.

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

OpenAI. GPT-5.6 Luna, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Official model documentation.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076. URL https://doi.org/10.1109/TSMC.1979.4310076.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering automated GUI interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026a. URL https://qwen.ai/blog?id=qwen3.8-flash-next.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026b. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026c. URL https://qwen.ai/blog?id=qwen3.6-27b.

Qwen Team. Qwen3.8-27B, 2026d. URL https://huggingface.co/Qwen/Qwen3.8-27B. Official model card.

Patrick Rim, Tom Long, Ekta Prashnani, Ruth Rosenholtz, Ben Boudaoud, Peter Xenopoulos, Alex Wong, Joohwan Kim, and Jae-Hyun Jung. Spatial-IQ: Deconstructing spatial intelligence via hierarchical capability tests, 2026. URL https://arxiv.org/abs/2607.22864.

Jonathan Roberts, Mohammad Reza Taesiri, Ansh Sharma, Akash Gupta, Samuel Roberts, Ioana Croitoru, Simion-Vlad Bogolin, Jialu Tang, Florian Langer, Vyas Raina, Vatsal Raina, Hanyi Xiong, Vishaal Udandarao, Jingyi Lu, Shiyang Chen, Sam Purkis, Tianshuo Yan, Wenye Lin, Gyungin Shin, Qiaochu Yang, Anh Totti Nguyen, David I. Atkinson, Aaditya Baranwal, Alexandru Coca, Mikah Dang, Sebastian Dziadzio, Jakob D. Kunz, Kaiqu Liang, Alexander Lo, Brian Pulfer, Steven Walton, Charig Yang, Kai Han, and Samuel Albanie. ZeroBench: An impossible visual benchmark for contemporary large multimodal models, 2025. URL https://arxiv.org/abs/2502.09696.

Ray Smith. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pp. 629–633, 2007. doi: 10.1109/ICDAR.2007.4376991. URL https://research.google/pubs/an-overview-of-the-tesseract-ocr-engine/.

Huatong Song, Fei Bai, Ming Yang, Renyuan Li, Jia Deng, Jujie He, Zhange Zhang, Daixuan Cheng, Yan Xing, Qi Yun, Xuxing Chen, Danyang Li, Feng Chang, Chuan Hao, Ran Tao, Jian Yang, Bryan Dai, Wayne Xin Zhao, Mingjie Tang, and Ji-Rong Wen. ClawGym II: Exploring black-box RL on agent harness, 2026. URL https://arxiv.org/abs/2608.16798.

Aditya Bharat Soni, Boxuan Li, Xingyao Wang, Valerie Chen, and Graham Neubig. Coding agents with multimodal browsing are generalist problem solvers, 2025. URL https://arxiv.org/abs/2506.03011.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, Linjie Li, Yu Cheng, Heng Ji, Junxian He, and Yi R. Fung. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers, 2025. URL https://arxiv.org/abs/2506.23918.

Didac Suris, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2303.08128.

Xinyu Tang, Han Peng, Guoxin Chen, Yuze Shi, Zitao Su, Peiyu Liu, Wayne Xin Zhao, Yawen Li, and Zhe Xue. Agent systems with harness engineering. https://github.com/RUCAIBox/awesome-agent-harness, 2026. Survey, May 2026.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with MATH-Vision dataset. In *Advances in Neural Information Processing Systems*, 2024a. URL https://papers.nips.cc/paper_files/paper/2024/

hash/ad0edc7d5fa1a783f063646968b7315b-Abstract-Datasets_and_Benchmarks_Track.html.

Peijie Wang, Zhong-Zhi Li, Fei Yin, Xin Yang, Dekang Ran, and Cheng-Lin Liu. MV-MATH: Evaluating multimodal math reasoning in multi-visual contexts, 2025. URL https://arxiv.org/abs/2502.20808.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b. URL https://arxiv.org/abs/2402.01030.

Xingyao Wang et al. OpenHands: An open platform for ai software developers as generalist agents, 2024c. URL https://arxiv.org/abs/2407.16741.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. CharXiv: Charting gaps in realistic chart understanding in multimodal LLMs. In *Advances in Neural Information Processing Systems*, 2024d. URL https://arxiv.org/abs/2406.18521.

Xueqing Wu, Yuheng Ding, Bingxuan Li, Pan Lu, Da Yin, Kai-Wei Chang, and Nanyun Peng. VISCO: Benchmarking fine-grained critique and correction towards self-improvement in visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9527–9537, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wu_VISCO_Benchmarking_Fine-Grained_Critique_and_Correction_Towards_Self-Improvement_in_Visual_CVPR_2025_paper.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Z.ai. GLM-5.3-Flash, 2026. URL https://huggingface.co/zai-org/GLM-5.3-Flash. Official model card.

Hantao Zhang, Jinru Sui, Ed Li, Dirk Bergemann, and Zhuoran Yang. MultiView-Bench: A diagnostic benchmark for world-centric multi-view integration in VLMs, 2026a. URL https://arxiv.org/abs/2607.08970.

Jiarui Zhang, Muzi Tao, Shangshang Wang, Ollie Liu, Xuezhe Ma, and Willie Neiswanger. An exam for active observers, 2026b. URL https://arxiv.org/abs/2607.16165.

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

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing "thinking with images" via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.14362.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models, 2023. URL https://arxiv.org/abs/2304.10592.

Xuanyu Zhu, Yuhao Dong, Rundong Wang, Yang Shi, Zhipeng Wu, Yinlun Peng, YiFan Zhang, Yihang Lou, Yuanxing Zhang, Ziwei Liu, Yan Bai, and Yuan Zhou. VTC-Bench: Evaluating agentic multimodal models via compositional visual tool chaining, 2026. URL https://arxiv.org/abs/2603.15030.

A BENCHMARK CONSTRUCTION

This appendix details the sources, difficulty calibration, and quality-control procedures used to construct VHStar.

A.1 BENCHMARK SOURCES AND COMPOSITION

Tables 6–8 summarize the domain definitions, source datasets, and domain–difficulty distribution. Each source is normalized into a common question, image, answer, and evaluation format. Domain assignments follow source task categories, with VTC-Bench categories mapped individually. The larger M4 allocation covers diverse fine-grained perception tasks, while M5 emphasizes difficult mixed reasoning. Harnesses are compared within each domain's fixed source and difficulty composition.

Table 6: Visual domains, task scope, and sources in VHStar. Source references and counts appear in Table 7.

Domain	Task focus	Source datasets
M1: Diagram mathematics	Mathematical quantities and relations grounded in diagrams.	MathVision, MV-Math, Math-VR, VTC-Bench
M2: Charts, documents, text	Chart values, document structure, and localized or oriented text.	CharXiv-RQ, MVToolBench, VTC-Bench
M3: Spatial and multi-view	Spatial structure, counting, and relations across viewpoints.	Spatial-IQ, MultiView-Bench, VTC-Bench
M4: Fine-grained perception	Visual search, discrimination, counting, and transformation of fine detail.	VisualProbe, TIR-Bench, ActiveVision, VTC-Bench
M5: Hard mixed reasoning	Compositional problems combining multiple visual and reasoning demands.	ZeroBench

Table 7: Source contributions to VHStar by domain. Counts denote tasks; definitions are in Table 6.

Source	M1	M2	M3	M4	M5	Total
MathVision (Wang et al., 2024a)	43	0	0	0	0	43
MV-Math (Wang et al., 2025)	37	0	0	0	0	37
Math-VR (Duan et al., 2025)	34	0	0	0	0	34
VTC-Bench (Zhu et al., 2026)	6	15	1	2	0	24
CharXiv-RQ (Wang et al., 2024d)	0	70	0	0	0	70
MVToolBench (Guo et al., 2025b)	0	35	0	0	0	35
Spatial-IQ (Rim et al., 2026)	0	0	69	0	0	69
MultiView-Bench (Zhang et al., 2026a)	0	0	50	0	0	50
VisualProbe (Lai et al., 2025)	0	0	0	66	0	66
TIR-Bench (Li et al., 2025)	0	0	0	63	0	63
ActiveVision (Zhang et al., 2026b)	0	0	0	19	0	19
ZeroBench (Roberts et al., 2025)	0	0	0	0	90	90
Total	120	120	120	150	90	600

Table 8: Domain–difficulty composition. D1–D4 measure decreasing direct success under the calibration panel (Eq. 4); they are distinct from harness rungs L0–L6.

Domain	D1	D2	D3	D4	Total
M1: Diagram mathematics	30	30	30	30	120
M2: Charts, documents, text	30	30	30	30	120
M3: Spatial and multi-view	30	30	30	30	120
M4: Fine-grained perception	35	35	40	40	150
M5: Hard mixed reasoning	0	2	16	72	90
Total	125	127	146	202	600

A.2 Difficulty Calibration

The calibration panel comprises Qwen3.8-Flash-Next, GPT-5.6-Luna, Gemma-4-31B-IT, and Gemini-3.5-Flash-Lite (Google DeepMind, 2026a). Each model contributes three direct responses per candidate without tools. Let $u_{imr} \in \{0, 1\}$ indicate whether response r of model m on task i is usable and $c_{imr} \in \{0, 1\}$ indicate full credit. We compute

$$U_i = \sum_{m=1}^{4} \sum_{r=1}^{3} u_{imr}, \qquad \widehat{\pi}_i = \frac{\sum_{m=1}^{4} \sum_{r=1}^{3} u_{imr} c_{imr}}{U_i}, \quad U_i \geq 11. \tag{3}$$

A candidate may have at most one unusable calibration response. Full credit requires a score of at least 0.999 for calibration; evaluation retains fractional scores. The difficulty strata are

$$d_i = \begin{cases} \text{D1}, & \widehat{\pi}_i \geq 0.75, \\ \text{D2}, & 0.35 \leq \widehat{\pi}_i < 0.75, \\ \text{D3}, & 0 < \widehat{\pi}_i < 0.35, \\ \text{D4}, & \widehat{\pi}_i = 0. \end{cases} \tag{4}$$

Calibration responses are separate from evaluation trajectories, and labels remain fixed across harnesses. Some panel models also appear in evaluation, so the strata measure panel-relative direct solvability; D4 does not imply intrinsic unsolvability.

A.3 Quality Control and Selection

Candidates undergo public-input and answer-contract checks, image-necessity screening, and difficulty calibration. Following MMStar (Chen et al., 2024), the image-necessity screen presents each question without images to Qwen3.8-Flash-Next and GPT-5.6-Luna, four times each. A candidate is excluded if either model answers correctly in at least two trials. This criterion targets reproducible text-only success under the screening models.

Reference validation targets strong cross-model disagreement with the supplied answer. It flags candidates whose difficulty score is at most $1/12$ and for which at least 60% of usable panel responses converge on one non-reference answer, spanning at least three models and two model families. Gemini-3.7-Flash (Google DeepMind, 2026b), held outside the calibration panel, produces three independent solutions without seeing the reference or panel answers. Challenged or unresolved references are removed and replaced within the same domain–difficulty cell.

Selection enforces the quotas in Table 8 and retains at most one task per sampling group. Groups use source-provided figure, problem, scene, task, or parent identifiers where available; MV-MATH, TIR-Bench, ActiveVision, VisualProbe, and VTC-Bench use source-prefixed media hashes. Eligible tasks have at least eleven usable calibration responses and pass all screening rules. Within each cell, selection prioritizes retained tasks from the preceding release, then newly screened candidates, then the remaining earlier pool; ties use a fixed seed and task-identifier hash. This ordering is independent of harness scores. The selected identifiers, labels, and sampling groups remain fixed for all comparisons.

B HARNESS AND EVALUATION PROTOCOLS

We specify the visual interfaces, execution environment, and scoring protocol used in the harness comparisons.

B.1 VISUAL OPERATIONS AND INTERFACE CONVENTIONS

Ops denotes the eight predefined visual operations shared by L2–L5. They cover region inspection, text extraction, quantitative comparison, and structural analysis. This repertoire reflects complementary tools in Visual Sketchpad (Hu et al., 2024) and recurring computations in PyVision (Zhao et al., 2025), providing intermediate evidence while leaving operation selection and answer synthesis to the model.

Table 9 defines each operation and its output. Operations accept an image reference and an optional region in pixel coordinates; omitting the region selects the full image. Region comparison requires a second region, which is optional for measurement. L1 exposes only `zoom`, which can be invoked repeatedly. The single- and multiple-operation labels in Table 1 refer to operation types.

Table 9: The predefined visual operation repertoire (Ops) shared by L2–L5. Image outputs follow each configuration's observation policy; numerical and textual outputs return through tool responses.

Operation	Function and returned evidence
`zoom`	Crops the selected region and returns an image; outputs exceeding 1,536 pixels on the longest side are downscaled.
`ocr`	Extracts text with local Tesseract OCR (Smith, 2007) and returns the recognized text and nonempty lines.
`color_stats`	Returns RGB means and standard deviations, dominant quantized color buckets, and the fraction of near-gray pixels.
`measure`	Returns region dimensions, aspect ratio, and center coordinates, with center-to-center pixel distance when a second region is supplied.
`compare_regions`	Resizes the second region to match the first and returns mean and maximum absolute pixel differences and the fraction of pixels differing beyond a fixed tolerance.
`count_components`	Applies Otsu thresholding (Otsu, 1979) and counts foreground connected components; returns the count, up to ten largest component areas, and the largest component's area fraction.
`edge_map`	Applies Canny edge detection (Canny, 1986) to the grayscale region and returns an edge image.
`threshold`	Returns a binary mask using a supplied grayscale cutoff or Otsu thresholding.

L3–L5 execute Python, whereas L6 uses `bash`. Kernel state retains interpreter variables; file state retains agent-written artifacts. L3 prohibits agent-written files while allowing harness-managed image returns. L4 retains both forms of state, and L5/L6 start a fresh process for each execution while preserving files. Automatic image return attaches derived images to the context; explicit inspection requires a separate `read` action. All configurations receive the original images; interactive configurations also return textual tool outputs.

B.2 MODELS AND EXECUTION ENVIRONMENT

The four full-ladder models use a fixed deployment and reasoning setting across configurations: high for DeepSeek-V4.1 (DeepSeek-AI, 2026b) and medium for Qwen3.8-Flash-Next (Qwen Team, 2026a), GLM-5.3-Flash (Z.ai, 2026), and GPT-5.6-Luna (OpenAI, 2026). The additional models are Qwen3.5-27B (Qwen Team, 2026b), Qwen3.6-27B (Qwen Team, 2026c), Qwen3.8-27B (Qwen Team, 2026d), and Gemma-4-31B-IT (Gemma Team, 2026) at medium reasoning effort, and MiniMax-M3 (MiniMax, 2026) at xhigh. All nine models use temperature 1, a per-response output limit of 65,536 tokens, and a default budget of 20 interaction steps. Original image inputs retain the same transport bounds within each model's comparison, with a maximum side of 1,536 pixels. Cross-model contrasts concern these evaluated deployments, including their serving and reasoning settings.

Code-enabled configurations share a containerized Python 3.11.2 environment running as an unprivileged user. Pillow, OpenCV, and scikit-image support image processing; NumPy, SciPy, scikit-learn, and Shapely support numerical and geometric analysis; Matplotlib supports plotting and SymPy symbolic computation. Tesseract is available through pytesseract for OCR, with PyMuPDF and imageio for document and image I/O. Package versions are pinned in the container build. Network and GUI access are unavailable to task-solving code. L3 enforces its no-file-write policy, while L6 confines artifact access to the task workspace.

B.3 Evaluation and Scoring

Main comparisons select response index zero at L0 and the first interactive pass at each other configuration, without selecting among answers by correctness. Additional direct responses enter only the consistency control in Appendix C.2. An interaction step is one completed model response, which may invoke multiple tools or provide the final answer; each direct attempt counts as one step. There is no additional finalization step beyond the budget. L3 returns the answer in its final response; L6 uses a designated submission file with the final response as fallback. Invalid or incomplete answers, recorded terminal request failures, timeouts, and budget violations receive zero and are not retried. Reaching the step cap alone does not invalidate an answer. Missing trajectories or pending judgments are not scored as failures.

Scoring follows the source benchmarks' answer contracts. Spatial-IQ uses exact integer matching, MultiView-Bench ordered axis-relation matching, and ActiveVision and ZeroBench normalized exact matching. TIR-Bench (Li et al., 2025) scores spot-difference index sets by intersection over union and jigsaw answers by the fraction of correctly positioned pieces; its other task types use discrete rules. MathVision, MV-MATH, and MVToolBench combine deterministic matching with answer-equivalence adjudication; Math-VR, CharXiv-RQ, VisualProbe, and selected free-form VTC-Bench tasks use the latter. Mean task score averages scores in $[0, 1]$ and scales by 100, retaining partial credit and equaling accuracy for binary-scored tasks.

Answer-equivalence judgments use DeepSeek-V4-Flash (DeepSeek-AI, 2026a) at temperature zero with structured output. The judge receives the question, answer type, reference, and extracted final answer, and awards binary credit for complete equivalence, including required units and ordering. The default output cap is 2,048 tokens, extended to 8,192 or 32,768 for selected judgments. The saved evaluation configuration includes the exact prompt and source-to-evaluator mapping. Fixed task identifiers, selected responses, per-task scores, and evaluator provenance support regeneration of all reported results.

C Supplementary Experimental Results

The following comparisons and controls complement the main results on harness configurations, model capabilities, and returned evidence.

C.1 Statistical Comparisons and Aggregation

Harness utility is the change in mean task score relative to a reference configuration, in percentage points. Table 10 reports paired configuration contrasts on the same tasks. L2→L3 introduces code execution and kernel persistence together; L3→L4 combines writable storage with explicit image inspection; L4→L5 removes kernel persistence; and L5→L6 changes the execution interface while retaining files and explicit inspection. These comparisons concern complete configurations. Equal step limits constrain interaction opportunity without equating token use or computation.

Confidence intervals use 10,000 paired task-bootstrap draws within source, preserving benchmark composition. These are unadjusted 95% intervals for task variation conditional on the selected trajectories, not repeated-run variability. Observation and budget comparisons use the same procedure.

Table 11 checks sensitivity to aggregation. Task-micro weights tasks equally; source-macro gives equal weight to the twelve source means. Invalid outcomes and recorded terminal request failures contribute zero under both conventions. For all four full-ladder models, L3 improves on L0 under either weighting and exceeds L6 under source-macro weighting. The overall advantage of L3 therefore persists when sources receive equal weight.

Table 10: Selected configuration contrasts in percentage points, with 95% confidence intervals.

Model	Contrast	Difference	95% interval
DeepSeek-V4.1	L2→L3	+7.5	[+3.8, +11.1]
	L3→L4	−5.3	[−8.7, −1.9]
	L3→L6	−3.7	[−6.8, −0.7]
GPT-5.6-Luna	L2→L3	+8.0	[+4.4, +11.7]
	L3→L4	−2.6	[−5.9, +0.8]
	L3→L6	−3.5	[−6.9, −0.3]
GLM-5.3-Flash	L2→L3	+10.0	[+6.1, +13.9]
	L3→L4	+0.6	[−3.0, +4.1]
	L3→L6	+0.0	[−3.5, +3.5]
Qwen3.8-Flash-Next	L2→L3	+5.9	[+2.3, +9.3]
	L3→L4	+1.2	[−2.0, +4.4]
	L3→L6	−4.0	[−7.3, −0.7]

Table 11: Sensitivity to task and source weighting. Scores are in %; task-micro weights tasks equally, while source-macro weights the twelve source means equally.

Model	Rung	Task-micro (%)	Source-macro (%)
DeepSeek-V4.1	L0	31.3	31.2
	L3	55.1	54.4
	L6	51.4	51.7
GPT-5.6-Luna	L0	34.7	33.3
	L3	43.9	43.9
	L6	40.4	39.2
GLM-5.3-Flash	L0	39.0	37.8
	L3	53.7	53.8
	L6	53.7	52.3
Qwen3.8-Flash-Next	L0	40.9	39.9
	L3	58.5	58.6
	L6	54.5	55.4

C.2 Repeated Direct Answering

Table 12 compares repeated direct answering with L3 and L6. For each model, we fix the D1 tasks receiving full credit in the selected L0 response and measure the fraction receiving less than full credit in each comparison. The direct baseline averages two additional L0 responses per task under the same generation settings. DeepSeek and GLM use three separate requests; Qwen and Luna use three completions in one request. All use temperature 1, without majority voting or response selection. This control measures conditional direct-answer variability, not repeated-interaction variability or a causal effect of tool use.

Table 12: Answer consistency on directly solved D1 tasks. Non-retention is the percentage of initially correct answers losing full credit; Repeat L0 averages two additional direct answers.

	D1 non-retention (%)		
Model	Repeat L0	L3	L6
DeepSeek-V4.1	22.6	1.1	6.5
GPT-5.6-Luna	12.7	11.8	12.7
GLM-5.3-Flash	12.0	6.7	5.8
Qwen3.8-Flash-Next	6.6	5.3	8.8

L3 has lower observed non-retention than repeated direct answering for all four models, while L6's relative ordering varies by model.

C.3 Effect of the Predefined Operation Menu

We remove the eight-operation menu from L3, L4, and L5 for Qwen, DeepSeek, and Luna while holding the execution backend, external state, observation policy, and budget fixed. Table 13 reports the results.

Table 13: Effect of removing the predefined operation menu. Δ is the score without the menu minus the score with the menu, in percentage points.

Model	Rung	With menu (%)	Without menu (%)	Δ (pp)
DeepSeek-V4.1	L3	55.1	51.7	-3.4
	L4	49.8	48.3	-1.5
	L5	48.7	51.7	$+3.0$
GPT-5.6-Luna	L3	43.9	43.8	-0.1
	L4	41.4	41.9	$+0.5$
	L5	40.1	42.4	$+2.3$
Qwen3.8-Flash-Next	L3	58.5	58.8	$+0.3$
	L4	59.6	55.5	-4.1
	L5	57.5	56.3	-1.2

Menu utility depends on the harness. Removal lowers DeepSeek's L3 score by 3.4 points but raises its L5 score by 3.0 points; Qwen's largest decrease occurs at L4. Luna changes little at L3 and improves at L4/L5. These differences support retaining predefined operations selectively alongside model-generated computation.

C.4 Program Use and Execution Evidence

Tables 14 and 15 report L3/L6 operation profiles for the Qwen-27B models and the four full-ladder models, respectively. Coverage measures recognized API patterns in submitted code; categories overlap, and matches can include script creation, comments, or unexecuted branches. Aliases, custom procedures, and separately executed scripts may escape detection, while array matches can concern non-image data. Normal-completion coverage additionally requires a matching command to complete without an observed exception, using the interface's native success signal. The two

Table 14: Operation-pattern coverage (% of tasks) in code submitted by three Qwen-27B deployments. Identical pattern rules are applied at L3 and L6. Categories overlap and describe submitted code rather than successful execution or semantic correctness.

	Qwen3.5-27B		Qwen3.6-27B		Qwen3.8-27B	
Operation	L3	L6	L3	L6	L3	L6
Crop	3.0	2.5	15.7	7.3	19.2	49.2
Two-dimensional slicing	15.2	3.0	34.7	11.8	44.8	33.8
Resize	3.2	2.5	5.8	9.2	21.0	53.0
Rotate / flip	5.8	1.0	8.5	3.7	11.2	6.3
Color / threshold	11.8	3.0	9.7	5.7	3.5	5.8
OpenCV connected components	2.3	0.2	1.7	3.0	2.0	5.5
Other connected components	2.8	0.0	11.8	2.5	27.2	13.8
Contours / edges	8.7	3.0	7.8	5.5	2.0	2.0
Array / mask statistics	10.2	3.0	31.3	11.8	39.8	30.5
NumPy save	0.0	0.2	0.2	0.3	1.7	6.5
NumPy load	0.3	0.0	0.0	0.0	0.0	3.2
Symbolic mathematics	2.2	0.2	1.2	0.8	2.2	0.8
OCR library	0.3	1.3	1.0	1.7	0.5	0.8
Display / plotting	28.5	1.7	44.0	2.5	47.5	0.2
Save then load (same path)	0.0	0.0	0.0	0.0	0.0	3.0
Any operation pattern	33.5	6.5	55.8	20.2	73.3	67.7
Normally completed match	28.8	6.5	53.5	20.0	72.5	67.5

measures describe submitted-code use and command completion, respectively, rather than operation accuracy.

The Qwen3.8–Qwen3.5 difference in direct score is 1.2 points (95% CI $[-2.4, 4.7]$), compared with 24.4 points ($[19.2, 29.5]$) in L3 gains and 26.0 points ($[20.6, 31.2]$) in L6 gains. Intervals jointly resample paired tasks within source. Qwen3.5-27B already produces executable visual analyses, including a contour-extraction trajectory that answers correctly where its selected direct answer is incorrect. Qwen3.8-27B constructs broader analyses involving component measurements and geometric descriptors. The contrast concerns effective computational use in the evaluated deployments rather than the first appearance of code generation.

Table 15: Paired L3/L6 code-operation coverage. Entries count matching tasks among all 600 first-pass traces per model and rung. Categories overlap.

	DeepSeek-V4.1		Luna		GLM-5.3		Qwen3.8	
Operation family	L3	L6	L3	L6	L3	L6	L3	L6
Crop (PIL API)	236	439	134	240	193	369	197	294
Resize / thumbnail	266	479	155	278	227	393	207	328
OpenCV color / threshold	76	77	59	44	24	32	27	31
OpenCV connected components	27	37	34	33	14	21	18	27
Other component labeling	185	120	5	4	176	75	129	72
Array / mask statistics	324	246	68	25	303	200	181	115
NumPy save	29	40	0	0	29	43	11	25
NumPy load	0	19	0	0	1	20	0	10
Save then load, same path	0	17	0	0	1	19	0	9

L3 prohibits agent-written files, so its save/load matches describe attempted operations. The save-then-load row requires the same path in distinct ordered actions. Separate execution checks verify L6 reuse: Qwen reloads node coordinates, DeepSeek reloads points and computes spatial counts, and GLM reloads and transforms a coordinate template. These cases complement the usage profiles by demonstrating cross-call reuse of numerical evidence in executed trajectories.

C.5 Observation-Channel Ablations

Table 16 reports the complete four-model comparison at L3 and L6. Each full, text-only, and image-only condition evaluates the same 600 tasks. Text-only feedback removes derived images; image-only feedback reduces tool text to completion/error status. Original images and available actions remain unchanged, and removal losses are full-feedback score minus ablated score.

Table 16: Observation-channel results and coverage (%, $n = 1$, 600 tasks per completed cell). Original task images remain available in every condition.

		Score (%)			Removal loss (pp)	
Model	Rung	Full	Text-only	Image-only	w/o images	w/o text
DeepSeek-V4.1	L3	55.1	41.3	42.3	13.8	12.8
	L6	51.4	38.0	36.8	13.5	14.6
GPT-5.6-Luna	L3	43.9	36.5	37.5	7.4	6.5
	L6	40.4	39.4	34.6	1.1	5.9
GLM-5.3-Flash	L3	53.7	46.0	49.0	7.7	4.8
	L6	53.7	45.5	47.2	8.2	6.5
Qwen3.8-Flash-Next	L3	58.5	48.3	51.7	10.2	6.8
	L6	54.5	46.0	44.5	8.5	10.0

These contrasts measure each channel's contribution conditional on the other remaining available, including changes in the model's subsequent actions. They do not decompose total harness benefit into additive channel shares or estimate channel interaction. The conditions use the same first-pass anchors as Figure 3.

C.6 Effect of Interaction Budget

We increase the interaction budget from 20 to 40 steps and the time allowance from 1,200 to 2,400 seconds for Qwen, DeepSeek, and Luna at L3 and L6. Task inputs, per-response token limits, and model settings remain fixed. Table 17 reports scores, interaction lengths, and paired uncertainty estimates on the same 600 tasks per condition.

Table 17: Three-model, two-rung interaction-budget study ($n = 1$, 600 tasks per cell).

		Score (%)		Δ (pp)	Mean steps
Model	Rung	B=20	B=40	[95% interval]	20 $\rightarrow$ 40
DeepSeek-V4.1	L3	55.1	59.4	$+4.3$ [$+1.0$, $+7.4$]	14.5 $\rightarrow$ 21.1
	L6	51.4	57.7	$+6.3$ [$+3.3$, $+9.2$]	14.9 $\rightarrow$ 23.2
GPT-5.6-Luna	L3	43.9	43.4	-0.5 [-3.7, $+2.7$]	5.8 $\rightarrow$ 6.2
	L6	40.4	42.7	$+2.3$ [-1.1, $+5.7$]	6.5 $\rightarrow$ 6.9
Qwen3.8-Flash-Next	L3	58.5	63.9	$+5.4$ [$+2.6$, $+8.3$]	9.3 $\rightarrow$ 11.5
	L6	54.5	57.3	$+2.8$ [-0.6, $+6.2$]	9.7 $\rightarrow$ 12.1

Additional interaction opportunity yields selective gains. Qwen and DeepSeek benefit at L3, and DeepSeek obtains the largest L6 gain ($+6.3$ points), with mean interaction length exceeding the original step limit. Luna keeps its mean below seven steps at both rungs and shows no clear improvement. L3 retains a higher mean score than L6 for all three models under both budgets. Thus, extending the available budget can relieve a constraint without uniformly eliminating differences between interfaces. Step counts measure interaction length; Appendix D.1 examines token consumption separately.

Table 18: Expanded score–token comparisons for configurations with complete token accounting. Tokens are averaged over the same tasks as the reported scores.

Model	Rung	Score (%)	Mean total tokens (k)
Gemma-4-31B-IT	L6	19.6	16.6
DeepSeek-V4.1	L0	31.3	15.1
	L1	46.9	239.3
	L2	47.7	280.3
	L3	55.1	224.3
	L4	49.8	208.5
	L5	48.7	210.8
	L6	51.4	211.5
Qwen3.6-27B	L3	39.4	152.2
	L6	32.6	105.2
Qwen3.5-27B	L3	29.5	96.5
	L6	25.3	52.6
Qwen3.8-27B	L3	55.1	156.7
	L6	52.4	195.9
GLM-5.3-Flash	L0	39.0	18.3
	L2	43.7	265.0
	L3	53.7	289.9
	L4	54.4	271.6
	L5	52.0	256.9
	L6	53.7	275.3
Qwen3.8-Flash-Next	L1	51.9	134.2
	L2	52.6	133.7
	L3	58.5	124.4
	L4	59.6	156.8
	L5	57.5	145.7
	L6	54.5	113.8

D Efficiency Analysis

We examine efficiency through token consumption and matched local execution, separating model interaction from execution overhead.

D.1 Token Consumption

Table 18 extends the main-text token comparison to configurations with complete usage records. Total consumption sums input and output tokens across the selected trajectory, including reported reasoning tokens and cached input without double counting. Repeated context contributes at each interaction. L0 requests returning multiple answers with pooled completion usage are excluded, as are configurations with missing usage or inconsistent component totals. Counts follow provider tokenization conventions and do not equate to monetary cost or FLOPs.

The additional Qwen results show that a minimal workspace does not universally reduce token use. L6 uses fewer tokens than L3 for Qwen3.5-27B and Qwen3.6-27B, with lower scores in both cases. For Qwen3.8-27B, L3 instead achieves a higher score with fewer tokens. Efficiency therefore depends on how the model uses the interface.

D.2 Local Execution and State Reuse

We compare matched local computations in the evaluated container image, using a shared 1024-by-1024 RGB input, two CPU cores, and 4 GiB of memory. Fifteen independent sessions per configuration run in randomized order. First use imports NumPy and Pillow, loads the image, and counts thresholded pixels. Each session repeats that computation five times and performs five queries of a preconstructed binary mask. Persistent configurations retain the mask in memory; stateless configurations reload it from a NumPy file. All returned values agree across configurations.

Table 19: Local execution latency (ms) for matched visual computations. First use includes interpreter and library initialization; repeated computation reloads the same image, while state reuse accesses an existing binary mask. Values average 15 independent sessions, with five repeated operations per session. L3 and L4 retain a kernel; L5 and L6 start a fresh interpreter for each program.

Configuration	First use	Repeated computation	State reuse
L3	720.3	25.0	3.7
L4	257.1	18.5	0.6
L5	264.5	265.8	208.1
L6	250.9	267.6	207.4

Table 19 reports first-use latency and within-session mean repeated latency. L3–L5 use their evaluated Python interfaces; L6 executes the same programs through Bash. Timing excludes model inference and the agent dispatcher. L3's initial cost includes filesystem enforcement and figure-capture setup, so L3/L6 is a complete-configuration comparison. L4/L5 holds the Python interface and observation convention fixed, isolating persistence: similar first-use costs and different repeated costs show the benefit of kernel reuse. Image-return timing is excluded because automatic rendering and explicit saving perform different work.

E Execution and Evidence Verification

We distinguish final answer outcomes, intermediate execution incidents, and evidence-verification behavior.

E.1 Outcomes and Execution Incidents

Table 20 separates full credit, partial credit, wrong answers, invalid submissions, and unusable responses at L0/L3/L6. Wrong answers dominate non-full-credit outcomes, and the L3-to-L6 losses primarily reflect answer quality. For example, Qwen and DeepSeek produce more wrong answers at L6 while their delivery-failure counts remain nearly unchanged. These outcome categories describe terminal results rather than their causes.

Table 20: Outcome accounting of the three representative paradigms ($n = 1$, counts out of 600 tasks). Categories are mutually exclusive. Wrong answer: resolved outcome with zero credit. Invalid submission: the trajectory ended in an unusable submission. No usable response: a resolved direct trajectory ended without a parseable answer, including terminal request failures. All listed outcomes are resolved.

Model	Rung	Full credit	Partial	Wrong answer	Invalid sub.	No response
DeepSeek-V4.1	L0	185	8	404	0	3
	L3	330	1	258	11	0
	L6	306	5	279	10	0
GPT-5.6-Luna	L0	206	5	387	0	2
	L3	261	5	329	5	0
	L6	240	5	354	1	0
GLM-5.3-Flash	L0	231	5	300	0	64
	L3	320	3	252	25	0
	L6	321	3	253	23	0
Qwen3.8-Flash-Next	L0	241	8	351	0	0
	L3	348	4	247	1	0
	L6	324	4	272	0	0

We also examine all first-pass L6 trajectories for the four full-ladder models. An execution incident is a terminal tool error, nonzero shell exit, or explicit traceback or exception. Permission rejection records enforcement of harness access policies and is reported separately from program errors. Other categories distinguish syntax, dependencies, file access, runtime/API errors, and timeouts. Incidents may overlap, unlike terminal outcomes. Figure 4 summarizes both, and Table 21 provides incident

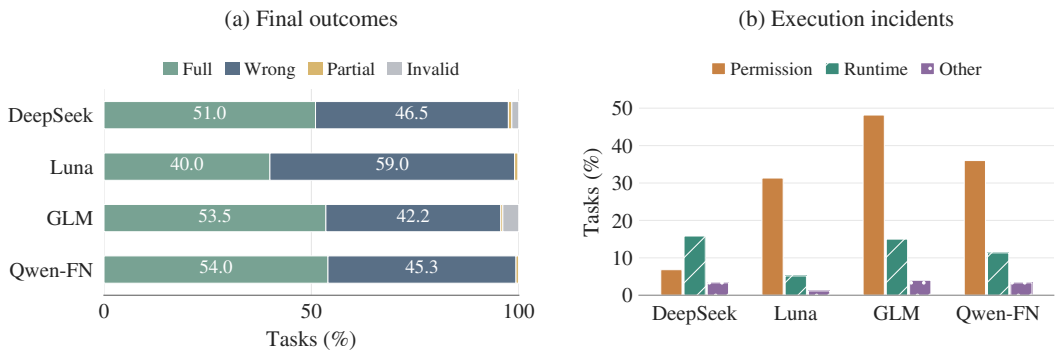


Figure 4: L6 outcomes and execution incidents (% of all tasks). Panel (a) partitions tasks into mutually exclusive final outcomes; labels show full-credit and wrong-answer rates. Panel (b) reports occurrence rates for overlapping execution-incident categories.

counts and their association with final outcomes. Reaching the step limit is recorded separately and is not itself an execution error.

Table 21: L6 execution incidents and final outcomes. Affected tasks (600 per model); rows can overlap.

Diagnostic	DeepSeek-V4.1	Luna	GLM-5.3	Qwen3.8
Permission rejection	41	188	289	216
Syntax / indentation	1	3	6	4
Missing dependency	2	0	0	0
File / path access	11	4	13	10
Runtime / API error	95	31	90	68
Execution timeout	0	0	0	0
Other tool error	6	1	5	6
Any execution incident	142	210	341	276
Full credit despite an incident	73	57	190	136
Wrong answer without an incident	218	205	116	136
Non-full-credit outcome at step limit	224	12	144	96

At L6, 45.8–78.1% of wrong answers have no recorded execution incident, and every model solves some tasks despite earlier incidents. Eventual full credit after an incident does not establish that a particular retry repaired it. These findings distinguish operational reliability from answer validity and motivate the verification analysis below.

E.2 Evidence Verification in L6 Trajectories

We inspect one trajectory for each combination of model, final outcome (full credit or wrong answer), and presence or absence of an execution incident. A fixed hash rule selects sixteen cases independently of their reasoning language. The review distinguishes execution repair, further observation, checks of a candidate interpretation, and changes to a method or answer. Table 22 presents representative cases, which illustrate behaviors rather than estimate their prevalence.

In the GLM multi-view example, an empty-mask error prompts a threshold adjustment. A normally completed fit then produces an anomalous geometric residual, leading to revised object extraction. Six leave-one-view-out fits agree on the final coordinate directions, which match the reference. Later attempts to inspect overlays are blocked, making this a case of useful numerical validation. DeepSeek, by contrast, repeatedly inspects crops but reports that its incorrect object count remains unverified; its digit-reading example attempts barcode decoding without obtaining a successful check. These cases show that additional inspection or attempted verification need not resolve visual uncertainty, motivating checks that can substantively revise intermediate conclusions.

Table 22: Examples of evidence verification at L6. Final correctness follows the benchmark references; the cases illustrate behaviors rather than their prevalence.

Model and task	Observed computation and check	Outcome and implication
GLM-5.3-Flash MultiView-Bench	Revises object extraction after inconsistent geometric fits, then checks coordinate signs across six leave-one-view-out fits.	The signs agree with the reference. Returned numerical evidence supports both diagnosis and validation.
DeepSeek-V4.1 Spatial-IQ	Repeatedly inspects regional crops and contrast-enhanced views, but reports that its final count remains unverified.	The count is incorrect. More observations alone do not resolve the uncertainty.
DeepSeek-V4.1 VTC-Bench	Attempts barcode decoding to check visually read digits; neither the initial decoder nor transformed inputs yield a successful decode.	The submitted digits are incorrect. Checking is attempted, but an effective validation signal is unavailable.

F LIMITATIONS

Our study examines static-image reasoning across representative harness configurations. Extending the framework to video, interactive environments, and longer workflows would broaden the study of visual evidence management. The model comparison concerns evaluated deployments; additional checkpoints would clarify how effective computation develops. The ladder establishes configuration-level differences from one trajectory per task and condition; repeated interactive evaluations and factorial interventions would resolve run-to-run variation and individual component effects. A promising direction is to develop adaptive harnesses that select when to compute, which evidence to inspect, and how to verify intermediate visual conclusions.

G REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the harness configurations and benchmark construction in Sections 3 and 4. Appendices A–E provide the evaluation settings, calibration and quality-control procedures, and supplementary results underlying our comparisons.