VLM-IN-SANDBOX: VISUAL WORKSPACES FOR AGENTIC VISUAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sandboxed computer environments support multi-step reasoning with tools, executable programs, and persistent files, yet their extension from language models to vision-language models (VLMs) introduces a distinct state-management problem. Visual reasoning produces intermediate image-valued evidence—crops, masks, overlays, zoomed regions, and analytic renderings—that must remain addressable without accumulating unboundedly in multimodal context. We introduce VLM-in-Sandbox, a training-free framework for agentic multimodal reasoning in controlled computer environments. Its Visual Workspace registers generated artifacts in an image ledger, maintains a bounded active visual context, and lets the model explicitly promote selected evidence for subsequent inspection. This separates visual evidence generation, performed by sandbox tools, from visual evidence management. Across seven benchmarks and four VLMs, VLM-in-Sandbox achieves higher sample-weighted accuracy than Vanilla VLM and Append-only Sandbox for every model. A compiler-matched 2×2 study on 1,260 examples isolates visibility from retention: the full policy reaches 66.27% accuracy with 18.6% fewer total tokens than the automatic, retain-all control. Paired evaluation of all 6,350 GPT-4.1-mini examples yields 302 rescues and 142 regressions relative to Original Append-only. A local vLLM study with prefix caching also reduces uncached prompt tokens, time to first token, and end-to-end latency, supporting explicit visual evidence management for sandboxed VLM agents.

1 INTRODUCTION

LLM agents shift language models from passive predictors into interactive problem solvers that maintain state, invoke tools, and revise plans across multiple steps (Yao et al., 2023; Zhou et al., 2024; Xie et al., 2024; Koh et al., 2024). Sandboxed computer environments are a natural substrate for this behaviour, exposing a filesystem, shell commands, editing tools, persistent files, and executable programs rather than a single Python interpreter; recent LLM-in-Sandbox work shows they elicit general agentic behaviours that single-turn prompting cannot (Cheng et al., 2026). Appendix D reviews related work. A natural question follows: what additional abstractions are needed to extend this paradigm to vision-language models?

The extension is not a model substitution. In language-centric agents, intermediate evidence is symbolic—code outputs, logs, file contents—all linearisable into the conversation. A VLM agent often produces evidence that remains *visual*: cropped regions, zoomed-in objects, rendered diagrams, analytic overlays, visual edits, or binary masks used to test hypotheses (Gupta & Kembhavi, 2022; Surís et al., 2023; Hu et al., 2024; Fu et al., 2025; Zhao et al., 2025; Zhang et al., 2025b; Hou et al., 2026; Su et al., 2025; Zhao et al., 2026). Generation alone does not make these usable: appended by default, they clutter the multimodal context; left as files, they remain invisible to the model. VLMs do not need more tools; they need a mechanism to retain, index, select, and re-present visual evidence across the trajectory. This need is especially acute as recent visual-reasoning systems increasingly scale inference, post-training, or action policies around intermediate visual operations (Thawakar et al., 2025; Liu et al., 2025; Tan et al., 2025; Huang et al., 2026; Wang et al., 2025; Wu et al., 2026; Min et al., 2026).

Figure 1 illustrates the gap on a high-resolution visual-search instance from HRBench-4K (Wang et al., 2024b). A Vanilla VLM guesses from the downsampled image; Append-only Sandbox generates

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.

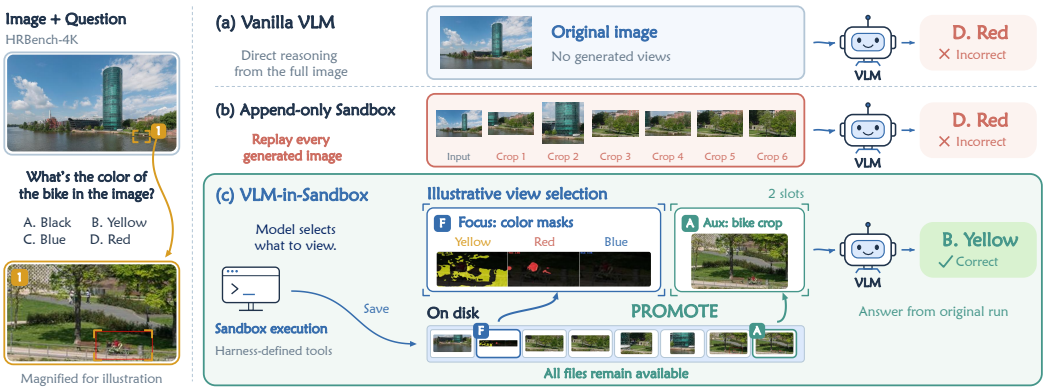

Figure 1: Motivating HRBench-4K example. **Vanilla VLM** reasons directly from the image; **Append-only Sandbox** accumulates generated views in its multimodal history; **VLM-in-Sandbox** keeps visual artifacts on disk and selects evidence for two active slots. The color-mask and bicycle-crop pairing illustrates view selection; the displayed answers are from the original runs, whose trace is discussed in §2.

several crops but buries them in history; VLM-in-Sandbox promotes a curated subset into active visual context and answers correctly. This motivates *visual evidence state*: an addressable record of what evidence exists, where it came from, and which pieces should be exposed to the model at each step.

We propose VLM-IN-SANDBOX, a framework for agentic multimodal reasoning, and VISUAL WORKSPACE, a runtime layer that organises the lifecycle of visual artifacts. An *Image Ledger* registers each generated image with a stable identifier, path, provenance, and parent–child links. A bounded *Active Visual Context*—two named slots, `focus` and `aux`—determines which artifacts are rendered as visual input next. A *Prompt Compiler* deterministically rebuilds each request from task, recent turns, ledger summary, and active context. Crucially, generated artifacts do not enter the active context automatically: the model must issue an explicit PROMOTE operation. This enforces a clean separation between visual evidence *generation* (sandbox) and *management* (workspace).

We evaluate three inference modes—**Vanilla VLM**, **Append-only Sandbox**, and **VLM-in-Sandbox**—across seven multimodal benchmarks spanning three families: high-resolution visual search (HRBench-4K (Wang et al., 2024b), V*Bench (Wu & Xie, 2023)), multimodal mathematical reasoning (WeMath (Qiao et al., 2024), MathVision (Wang et al., 2024a)), and general multimodal reasoning (MMStar (Chen et al., 2024), RealWorldQA (xAI, 2024), MMMU (Yue et al., 2024)). Across four base VLMs, VLM-IN-SANDBOX achieves the best sample-weighted average accuracy among the three modes and reduces token use relative to Append-only Sandbox. Section 3 presents the broad results, compiler-matched controls, and paired reliability analyses. Section 3.4 examines context growth and prefix-cached inference cost; Appendix E presents the qualitative examples. Together, these experiments test not only whether the full system works, but which visual-state choices account for its behavior and efficiency.

Our qualitative analysis in Appendix E shows that the workspace enables diverse computer-aided visual reasoning: models can render overlays, crop regions, compute masks, and use PROMOTE to return selected evidence for later inspection. In summary, we identify visual evidence state as a missing abstraction for sandboxed VLM agents, instantiate it as the VISUAL WORKSPACE, and evaluate it through broad, matched, paired, and systems-level evidence.

2 METHODOLOGY

We introduce VLM-IN-SANDBOX, a framework that equips VLMs with agentic multimodal reasoning in controlled computer environments. Given an image set I and a text query q, the agent interacts with a Docker sandbox through executable tools, produces intermediate visual artifacts, and writes a final answer to a designated file. Our key design principle is the separation of *visual evidence generation*,

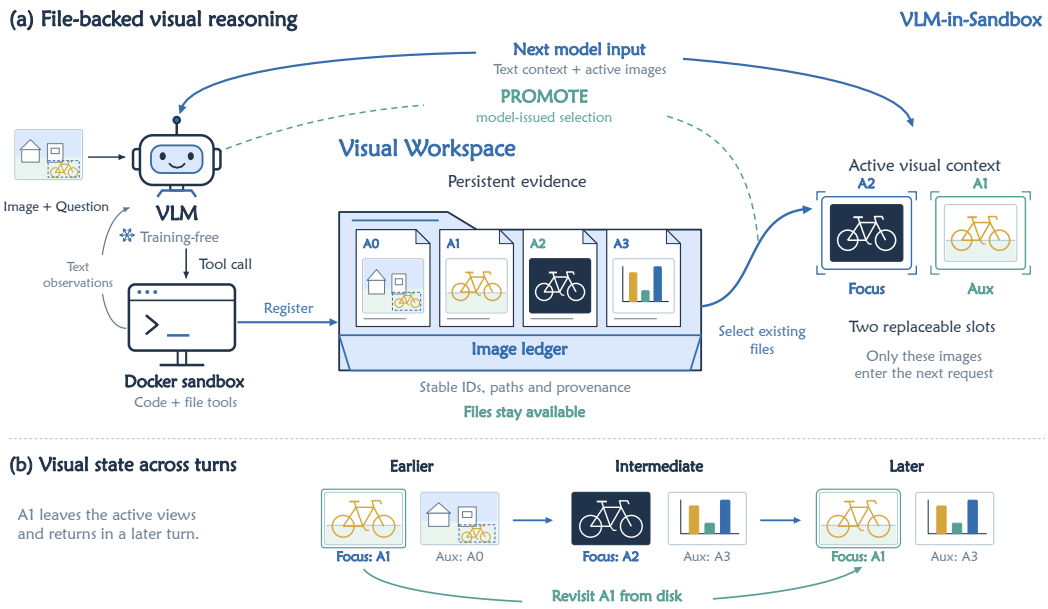

Figure 2: Overview of VLM-IN-SANDBOX. (a) A frozen VLM uses harness-defined tools in a Docker sandbox to generate visual evidence. Visual Workspace preserves files with stable identifiers and provenance; model-issued PROMOTE operations select the `focus` and `aux` images for the next request. (b) Replacing an active view leaves its file available for later selection. The illustrative, nonconsecutive snapshots show an earlier artifact returning to active context without replaying all intermediate images.

performed by sandbox tools, from *visual evidence management*, handled by a runtime layer we call the *Visual Workspace*. Figure 2 shows the architecture, while Algorithm 1 in the appendix gives the full runtime loop; Figure 1 (§1) provides the worked HRBench-4K example we use throughout this section to ground each component in a concrete trace.

2.1 AGENT–ENVIRONMENT LOOP

At each step t, the agent receives a multimodal prompt and either issues a tool call or produces a final answer. Tool calls execute through a sandbox executor exposing three interfaces: `execute_bash` for code execution, `str_replace_editor` for file manipulation and explicit promotion (§2.3), and `submit` to finalize the answer. Our implementation uses Docker to pin dependencies, isolate episodes, and clean up their files, but the executor is replaceable: the method requires the same file-and-tool contract rather than Docker-specific model behavior. The sandbox serves as an external computation layer, supporting operations—cropping, zooming, analytic rendering, plot rendering, annotation, pixel-level statistics, diagnostic mask generation—that are difficult to perform reliably inside a single forward pass.

A defining property of multimodal sandboxing is that tool execution can produce new *visual artifacts* that constitute intermediate evidence for later reasoning. The central challenge is therefore not what tools the model can call, but how the resulting visual evidence is represented, indexed, and routed back into context. In the **Append-only Sandbox** baseline, generated images are appended to the conversation and remain visible in all subsequent turns—a direct multimodal extension of the standard text-only sandbox paradigm. This append-only scheme creates two compounding problems: redundant visual tokens accumulate as the trajectory grows and dilute attention, and the model cannot selectively revisit earlier evidence without it persisting through every intervening turn. The Visual Workspace addresses both by decoupling artifact generation from artifact visibility.

2.2 IMAGE LEDGER: MAKING VISUAL EVIDENCE ADDRESSABLE

The *image ledger* $\mathcal{L}_t$ is a structured registry for all visual artifacts known to the agent. Each input or generated image receives a stable identifier, a file location, and provenance linking it to its source artifacts. Stable identifiers turn anonymous sandbox files into addressable objects, which is essential for tasks requiring iterative localization: a model may first crop a broad region, then a sub-region, then zoom further, and reliable references across these chains are otherwise difficult to maintain.

Payload eviction. Each registered asset has an optional inline payload—the encoded image data used to render it as a visual input. After every step we evict payloads from non-active, non-input assets (Algorithm 1) while keeping their metadata in $\mathcal{L}_t$. The model can still see that an asset exists and reference it by id, but its image data is reloaded from disk only when it is promoted back into active context.

Inactive image payloads can therefore be released without losing the evidence itself. Appendix A summarizes the workspace operations.

2.3 ACTIVE VISUAL CONTEXT AND EXPLICIT PROMOTION

Bounded active context. The *active visual context* $\mathcal{C}_t$ is the subset of registered artifacts rendered as visual input in the next prompt. We implement $\mathcal{C}_t$ as two named slots:

$$\mathcal{C}_t \;=\; \big\{\, \texttt{focus}_t,\; \texttt{aux}_t \,\big\}. \tag{1}$$

The `focus` slot holds the primary evidence the model currently inspects; `aux` holds a paired supporting view such as a parent crop or contextual region (e.g., focus on a zoomed-in object, aux on its surrounding scene). Bounding the active context to two slots is deliberate: a sandboxed VLM can easily generate dozens of images, and re-injecting them all creates redundant visual tokens and expands every subsequent request. The model can still browse the full ledger through textual indices, but only promoted artifacts appear as image inputs.

Explicit promotion. Generated artifacts are *not* automatically visible in subsequent turns. The model must issue an explicit promotion:

$$\textsc{Promote}(a,\, s), \quad a \in \mathcal{L}_t,\; s \in \{\texttt{focus}, \texttt{aux}\}, \tag{2}$$

which moves asset a from the ledger into slot s of the active context. Operationally, this is a subcommand of the editor interface (e.g., `promote asset://asset_0008 slot=focus`); it performs no image processing, only routing. This narrow semantics is important: in Append-only Sandbox, every crop enters history simply because it was produced; in the Visual Workspace, a crop becomes active evidence only when explicitly selected, separating *producing* evidence from *deciding which evidence guides future reasoning*. This also makes negative or exploratory visual work cheap: the agent may create several candidate crops, masks, or overlays during search, while only the artifacts judged useful need to consume active visual slots. Thus PROMOTE is not an additional visual tool, but a routing decision over already-created evidence.

For example, the HRBench-4K agent generates several candidate color masks and registers them as visual artifacts. It promotes their combined visualization to `focus`, compares the evidence, and answers yellow. The remaining artifacts stay on disk without occupying active visual slots. A promotion changes visibility, not the underlying files:

```
promote asset://masks_combined slot=focus
```

The identifier above is illustrative; Figure 1 shows the selection mechanism and the answer from the original run.

2.4 PROMPT COMPILATION

Rather than replaying an unbounded multimodal history, the *prompt compiler* deterministically reconstructs each model request from the workspace state:

$$x_t \;=\; \textsc{Compile}\big(q,\; \mathcal{T}_{[-k:]},\; \textsc{Recap}(\mathcal{T}_{<-k}),\; \mathcal{L}_{t-1},\; \mathcal{C}_{t-1},\; g_t\big), \tag{3}$$

Table 1: **Main results across closed-source and open-source MLLMs.** We compare three inference settings (Vanilla VLM, + Sandbox Only, and + Ours) across seven multimodal benchmarks grouped into three task families: high-resolution visual search (HRBench-4K, V*Bench), mathematical reasoning (WeMath, MathVision), and general multimodal reasoning (MMStar, RealWorldQA, MMMU). Each cell reports accuracy (%). Within each model block, the best accuracy is shown in **bold**. The **Avg.** column reports a sample-size-weighted average across the seven benchmarks.

Model	Setting	Search		Math		General			Avg.
		HRBench-4K	V*Bench	WeMath	MathVision	MMStar	RealWorldQA	MMMU	
Closed-source Models									
GPT-4.1-mini	Vanilla VLM	78.00	68.06	62.64	41.12	64.40	71.76	66.19	65.81
	+ Sandbox Only	77.63	68.06	62.76	44.08	60.47	70.33	66.86	64.95
	+ Ours	**79.25**	**70.68**	**64.08**	**46.71**	**65.67**	**72.94**	**68.10**	**67.47**
Gemini-2.5-Flash	Vanilla VLM	83.50	79.06	81.38	39.14	76.87	74.77	71.81	76.11
	+ Sandbox Only	83.25	81.15	85.46	61.18	76.67	75.03	72.48	78.41
	+ Ours	**84.75**	**83.25**	**86.49**	**62.17**	**78.00**	**76.08**	**73.81**	**79.65**
Open-source Models									
Qwen3.5-9B	Vanilla VLM	77.25	89.53	76.84	42.11	74.40	**75.03**	65.81	72.99
	+ Sandbox Only	77.75	84.29	86.90	55.92	73.87	70.59	72.67	76.79
	+ Ours	**78.13**	**91.10**	**87.93**	**57.89**	**75.00**	71.90	**74.10**	**78.08**
Ministral-3-8B-Instruct	Vanilla VLM	51.50	49.74	45.23	24.67	49.73	55.69	51.33	48.50
	+ Sandbox Only	52.38	43.98	64.54	28.29	57.40	56.08	57.52	56.79
	+ Ours	**53.50**	**51.83**	**65.80**	**29.61**	**58.33**	**57.25**	**58.57**	**58.11**

where $\mathcal{T}_{[-k:]}$ are the k most recent uncompressed action–observation turns (k=3 by default), $\textsc{Recap}(\cdot)$ is a deterministic textual summary of older turns (action types and observation digests, no inline images), and g_t is step-budget guidance discussed below. The compiled prompt concatenates fixed system and task instructions, the older-turn recap, the recent-turns block, a workspace state summary listing asset metadata and recent-artifact indices, the deadline guidance, and the active visual context rendered as inline images. Only artifacts in $\mathcal{C}_{t-1}$ are rendered visually in request x_t; all other registered assets are referenced only through textual metadata. This prevents the failure mode of naive sandboxing—where every generated crop, rendered diagram, and diagnostic image accumulates in the prompt—and bounds visual context size independently of trajectory length.

The runtime also provides deterministic budget reminders and checks that an answer is available before accepting submission. These policies are shared by all matched configurations; Appendix A.2 gives their settings.

3 Experimental Evaluation

We first evaluate the complete method across models and task families, then use matched controls to separate visual selection from bounded retention. Paired outcomes and repeated runs test the reliability of the gains. Section 3.4 summarizes context growth and inference cost; Appendices B and E give the full diagnostics and individual reasoning trajectories.

3.1 Experimental Setup and Main Results

Benchmarks. We use seven benchmarks in three task families: high-resolution visual search—HRBench-4K (Wang et al., 2024b) and V*Bench (Wu & Xie, 2023); multimodal mathematical reasoning—WeMath (*testmini*) (Qiao et al., 2024) and MathVision-mini (Wang et al., 2024a); and general multimodal reasoning—MMStar (Chen et al., 2024), RealWorldQA (xAI, 2024), and MMMU (*dev/val*) (Yue et al., 2024). The complete collection contains 6,350 examples. We report per-benchmark accuracy and sample-size-weighted pooled accuracy.

Models and broad settings. The broad evaluation uses two API models, GPT-4.1-mini and Gemini-2.5-Flash, and two locally served models, Qwen3.5-9B and Ministral-3-8B-Instruct. **Vanilla VLM** receives only the task image(s) and question. **Original Append-only Sandbox**, denoted **+ Sandbox Only** in the submitted main table, provides the same executable tools as our method but automatically appends every generated image to subsequent multimodal history. **VLM-in-Sandbox** is the complete method, including the Visual Workspace. All settings use a temperature of 1.0; sandboxed settings use at most 20 agent steps.

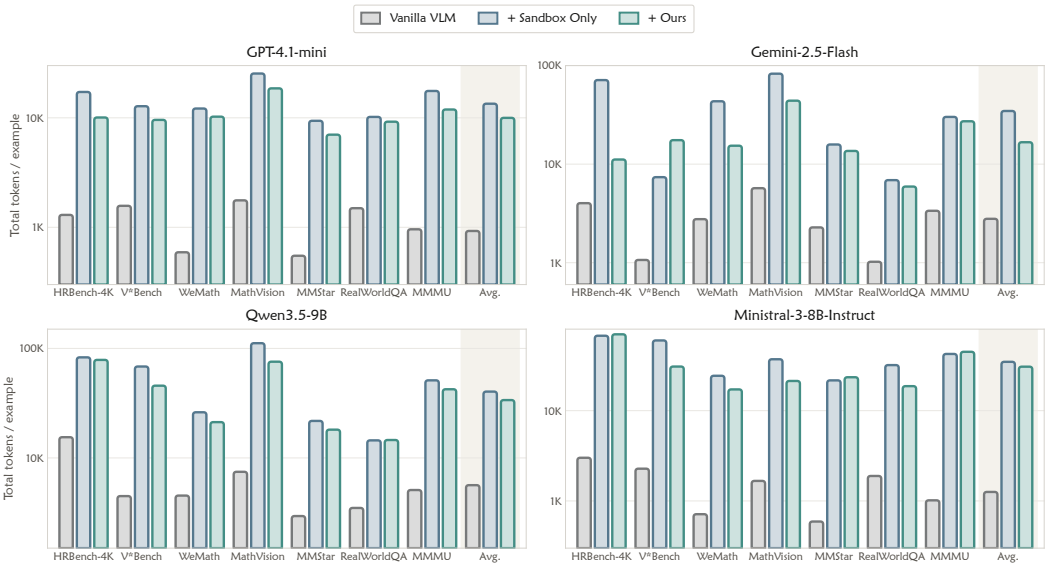

Figure 3: **Average total-token usage across inference settings.** Bars report the average total tokens per instance for each benchmark on a logarithmic scale; the shaded **Avg.** column is the sample-size-weighted average over the seven benchmarks. Sandbox access costs roughly an order of magnitude more total tokens than a single Vanilla VLM call, and the Visual Workspace removes part of that overhead: at the aggregate level, **+ Ours** uses fewer total tokens than **+ Sandbox Only** for every base model while achieving higher accuracy.

Controlled and systems studies. The matched study uses GPT-4.1-mini on complete V*Bench, MathVision-mini, and RealWorldQA splits ($N = 1{,}260$). Its four configurations share the model, prompt compiler, tool environment, inference budgets, and scoring protocol; only visual visibility and retention differ. The local systems study evaluates the same three splits with Qwen3.5-9B served by vLLM on $8\times$H20 with tensor parallelism 8 and request concurrency 1.

Metrics and token scopes. The broad study reports provider- or server-reported *total tokens* summed across calls, matching the submitted analysis in Figure 3; these values are compared only among settings for the same base model and backend. The matched study reports *total tokens* (prompt plus completion). The local vLLM study additionally separates cached and uncached prompt tokens; its token and latency values are compared only within that backend. HRBench uses its official fixed GPT-4o-mini yes/no evaluator on every row; the remaining benchmarks use deterministic or official benchmark-specific extractors and answer checkers. Appendix A.2 documents splits, scoring, and inference settings.

Table 1 shows that VLM-IN-SANDBOX (**+ Ours**) achieves the highest sample-size-weighted accuracy for every base model. Relative to Vanilla VLM, it improves GPT-4.1-mini, Gemini-2.5-Flash, Qwen3.5-9B, and Ministral-3-8B-Instruct by 1.66, 3.54, 5.09, and 9.61 percentage points, respectively. The improvement is broad rather than driven by a single benchmark: **+ Ours** outperforms Vanilla VLM in 27 of the 28 model–benchmark pairs, with Qwen3.5-9B on RealWorldQA as the only exception. The gains span both proprietary and open-source model families, with the largest aggregate improvement observed for Ministral-3-8B-Instruct.

The comparison with **+ Sandbox Only** shows the gain of the complete VLM-IN-SANDBOX design over sandbox access alone. Sandbox access is not uniformly beneficial: for example, it reduces the GPT-4.1-mini average from 65.81 to 64.95, whereas **+ Ours** raises it to 67.47. More generally, **+ Ours** improves over **+ Sandbox Only** in all 28 model–benchmark pairs. The largest gains over Vanilla VLM cluster on WeMath and MathVision, where executable computation is particularly useful. For both open-source models, the largest gain over **+ Sandbox Only** occurs on V*Bench: 6.81 points for Qwen3.5-9B and 7.85 points for Ministral-3-8B-Instruct. The compiler-matched controls in Section 3.2 then isolate visual selection and bounded retention from the surrounding tool harness.

Figure 3 makes the broad inference cost explicit. Both tool-enabled settings generate substantially more tokens than a single Vanilla VLM call, so the relevant efficiency comparison is with the tool-using **+ Sandbox Only** baseline. On the sample-size-weighted average, **+ Ours** reduces token usage by 25.7%, 51.8%, 16.3%, and 11.7% for GPT-4.1-mini, Gemini-2.5-Flash, Qwen3.5-9B, and Ministral-3-8B-Instruct, respectively, while improving accuracy. At the model-level aggregate, VLM-IN-SANDBOX therefore improves both accuracy and total-token efficiency relative to **+ Sandbox Only**. The magnitude of the reduction varies across benchmarks, so we report the aggregate as a summary rather than a uniform per-task effect.

3.2 MATCHED COMPONENT ABLATIONS

The broad comparison evaluates complete inference settings, but does not by itself distinguish selection from retention. We therefore hold the prompt compiler and executable tools fixed and vary these two factors on the complete three-benchmark collection.

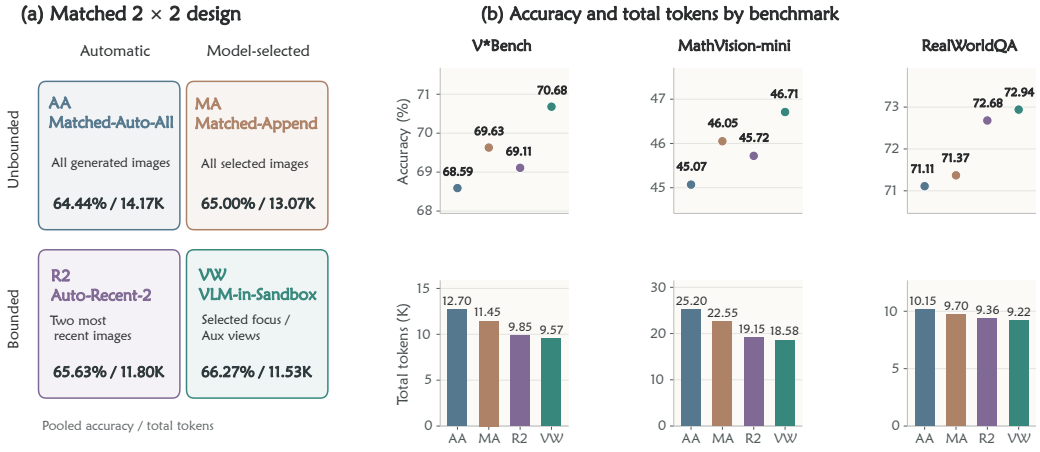

Figure 4: **Matched visual-state controls.** Left: the 2 × 2 design, with sample-weighted accuracy and total tokens per example in each cell. Right: the four settings on complete V*Bench ($N = 191$), MathVision-mini ($N = 304$), and RealWorldQA ($N = 765$), using GPT-4.1-mini. Accuracy panels use separate, labeled scales; token bars start at zero. Original Append-only is not a factorial cell.

Figure 4 presents a 2 × 2 visibility-by-retention study. Matched-Auto-All is the compiler-matched append-only control: every generated image becomes visible and remains in later requests. Matched-Append requires model selection through PROMOTE but retains all selected images. Auto-Recent-2 automatically exposes only the two most recent derived images. VLM-in-Sandbox combines model-directed selection with two replaceable `focus`/`aux` slots.

VLM-in-Sandbox improves over Matched-Auto-All by 1.83 points, with 52 rescues and 29 regressions (paired bootstrap 95% CI [0.40, 3.25]; exact McNemar $p = 0.014$), while using 18.6% fewer total tokens. Bounded retention gives the largest token reduction: 16.7% under automatic visibility and 11.8% after model selection. At two-image capacity, model selection improves over recency by 0.63 points with 2.3% fewer tokens, although the accuracy increment is not statistically conclusive (95% CI $[-0.16, 1.51]$; $p = 0.185$). Thus the complete policy performs best, while the factorial comparison identifies bounded retention as the larger token lever.

The three benchmarks expose different parts of this trade-off. On V*Bench, selected but unbounded evidence is more accurate than automatic recency (69.63% versus 69.11%), but uses more tokens. On RealWorldQA, automatic recency is already close to VLM-in-Sandbox (72.68% versus 72.94%). The complete policy has the highest point estimate on all three, while the importance of selection varies by task. Figure 9 reports all five settings with accuracy and token cost plotted separately, including Original Append-only as a reference outside the factorial; Table 8 reports the paired comparisons.

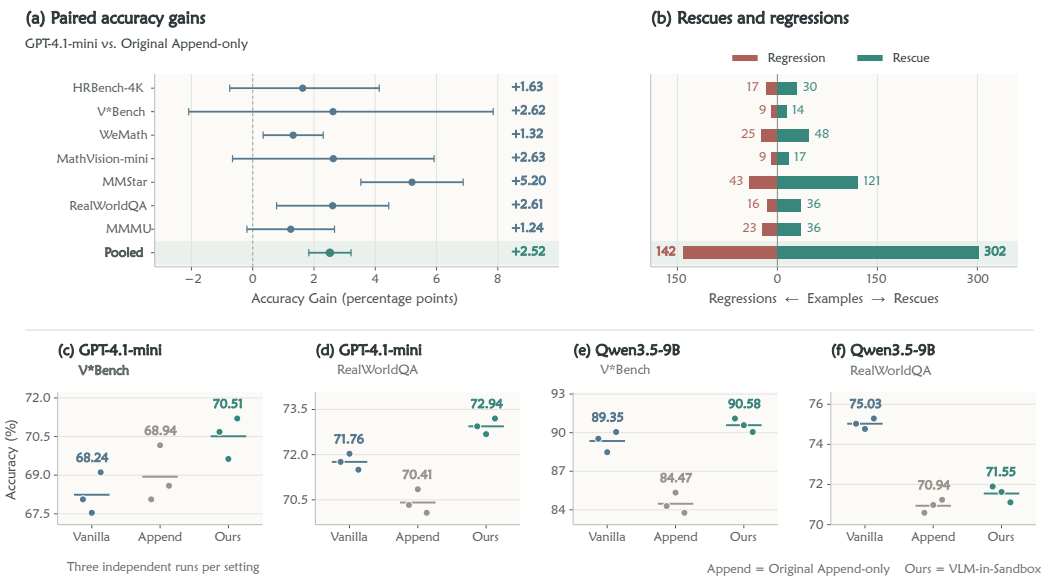

Figure 5: **Reliability across examples and repeated runs.** (a–b) GPT-4.1-mini versus Original Append-only on seven benchmarks ($N = 6{,}350$): paired accuracy gains with 95% bootstrap intervals, and corresponding rescue/regression counts. Pooled resampling is benchmark-stratified with HRBench variants grouped by base question. (c–f) Three runs per setting at temperature 1.0 on complete V*Bench and RealWorldQA splits. Points are individual runs; horizontal bars and labels show means. Accuracy scales differ across panels.

Table 2: **Wrong2Right analysis and promotion use by task family for Qwen3.5-9B.** Wrong2Right and Rate report the number and percentage of vanilla errors corrected by sandbox inference. Avg. Tool Calls reports the mean number of tool invocations per recovered example, and With Promote is the percentage invoking `promote` at least once.

Task	Wrong ✗	Wrong2Right ✗→✓	Rate	Avg. Tool Calls	With Promote
Search	202	70	34.7%	5.64	27.1%
Math	579	331	57.2%	6.15	4.2%
General	934	333	35.7%	5.49	6.3%
Overall	**1,715**	**734**	**42.8%**	**5.80**	**7.4%**

3.3 Reliability and Recovery

Figure 5 summarizes the paired results. Across all seven GPT-4.1-mini benchmarks, VLM-in-Sandbox converts 302 Original Append-only errors into correct answers while regressing on 142 previously correct examples: 2.13 rescues per regression and 160 additional correct answers. The resulting gain is 2.52 percentage points. The paired gain has a 95% bootstrap CI of [1.83, 3.21] and $p < 0.001$. Appendix C specifies the stratified, question-clustered analysis and reports the complete per-benchmark results.

Across all four model–dataset combinations in Figure 5(c–f), VLM-in-Sandbox has a higher three-run mean than Original Append-only. Repeated evaluation also preserves the main trade-off relative to Vanilla VLM. For Qwen3.5-9B, VLM-in-Sandbox has a higher mean on V*Bench (90.58% versus 89.35%), but a lower mean on RealWorldQA (71.55% versus 75.03%). The complete seven-benchmark comparison yields 734 rescues and 411 regressions relative to Vanilla VLM. Thus sandbox reasoning solves many questions missed by the base model, but can also turn a correct direct answer into an error. Table 10 reports all four outcome transitions for both models, with their respective references.

Wrong2Right Analysis and Selective Promotion. Table 2 audits Qwen3.5-9B on all 1,715 examples missed by Vanilla VLM. VLM-IN-SANDBOX corrects 734 of them (42.8%), with substantial variation across task families. Math achieves the highest recovery rate (57.2%) and uses the most tools per recovered example (6.15), yet invokes `promote` in only 4.2% of recoveries. This profile is consistent with computation-heavy trajectories whose symbolic or textual outputs can be consumed without visual re-entry. Search and General offer an informative contrast: their recovery rates (34.7% versus 35.7%) and average tool usage (5.64 versus 5.49 calls) are similar, while Search invokes `promote` 4.3× as often (27.1% versus 6.3%). Given their similar average tool counts, this pattern suggests that promotion frequency tracks the modality of intermediate evidence more closely than tool-call volume.

The recovery rate and promotion rate describe different quantities. The former measures how often VLM-IN-SANDBOX corrects a Vanilla VLM error; the latter measures how often a recovered trajectory invokes this routing operation. Recovered examples use 5.80 tool calls on average, but only 7.4% invoke `promote`. Promotion is therefore selective rather than mandatory: many recoveries use textual tool outputs and persistent workspace state, while high-resolution search more often returns a crop, mask, or overlay to multimodal context. More broadly, the Visual Workspace keeps intermediate artifacts addressable and bounds the active visual context; `promote` provides targeted visual re-entry when needed. These conditional statistics characterize successful trajectories, while component attribution is established by the matched comparisons above.

3.4 Context and Inference Cost

Context scaling. Fixed-trace replays of 100 successful trajectories increase stored artifacts from 4 to 32 without changing the visual information, actions, or outcomes: Matched-Auto-All grows from 8.90K to 42.30K tokens per replay, whereas VLM-in-Sandbox grows from 8.40K to 17.90K while keeping visible derived images near two (Appendix B, Figure 6). This open-loop diagnostic measures request growth rather than closed-loop accuracy; the same appendix reports active-slot and history-policy analyses.

Prefix-cached inference. On the three complete matched-study splits ($N = 1{,}260$), Qwen3.5-9B served by vLLM on 8×H20 uses 16.4% fewer uncached prompt tokens than Matched-Auto-All, with 19.7% lower time to first token and 13.0% lower end-to-end latency, including sandbox lifecycle (Appendix B, Figure 8; Table 12). These measurements describe that local backend and exclude offline scoring.

4 Discussion and Limitations

Visual Workspace reduces repeated visual workload relative to append-only sandboxing, but tool-enabled reasoning still costs more than a single Vanilla call and requires execution resources and permissions; Docker is replaceable by a backend with equivalent tool behavior. Gains depend on the base model's tool and evidence choices, with Qwen's 734 rescues and 411 regressions relative to Vanilla motivating correctness-independent routing that selectively invokes sandbox reasoning. Our evaluation covers static images; extension to video requires temporal retrieval, tracking, and dedicated evaluation. Applications involving sensitive images or high-stakes decisions require appropriate access controls and human oversight.

5 Conclusion

We introduced VLM-IN-SANDBOX, a training-free framework that separates persistent visual evidence from the images exposed to the model. Its Visual Workspace combines addressable artifacts, bounded active slots, and model-directed promotion. Across seven benchmarks and four VLMs, the complete method improves sample-weighted accuracy over Vanilla VLM and Original Append-only Sandbox. Matched controls identify bounded retention as the larger source of token savings, while selected visibility achieves the best observed accuracy. Paired outcomes, scaling replays, and prefix-cached measurements characterize the gains and their trade-offs. These findings support explicit visual evidence management as a core abstraction for sandboxed VLM reasoning.

AI USE STATEMENT

Codex was used to polish the language and assist with LaTeX typesetting of this manuscript.

ETHICS STATEMENT

Our evaluation uses existing visual benchmarks; Appendix A.2 identifies the evaluated splits, public sources, and stated dataset terms. Visual artifacts can expose sensitive image content, so deployments involving private or high-stakes data require appropriate access controls and human oversight. Appendix E.1 illustrates that a correct final answer need not imply faithful interpretation of selected visual evidence.

REPRODUCIBILITY STATEMENT

Appendix A specifies the Visual Workspace operations and runtime loop. Appendix A.2 documents prompt structure, tool interfaces, inference settings, dataset splits, scoring, and software environment; Appendices B and C give the diagnostic protocols and complete quantitative analyses. These descriptions identify the conditions and procedures underlying the reported comparisons.

REFERENCES

Tajamul Ashraf, Amal Saqib, Hanan Ghani, Muhra AlMahri, Yuhao Li, Noor Ahsan, Umair Nawaz, Jean Lahoud, Hisham Cholakkal, Mubarak Shah, Philip Torr, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Salman Khan. Agent-X: Evaluating deep multimodal reasoning in vision-centric agentic tasks, 2025. URL https://arxiv.org/abs/2505.24876.

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

Daixuan Cheng, Shaohan Huang, Yuxian Gu, Huatong Song, Guoxin Chen, Li Dong, Wayne Xin Zhao, Ji-Rong Wen, and Furu Wei. Computer environments elicit general agentic intelligence in LLMs, 2026. URL https://arxiv.org/abs/2601.16206.

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

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training, 2022. URL https://arxiv.org/abs/2211.11559.

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, 2026. URL https://arxiv.org/abs/2511.19661.

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, 2024. URL https://arxiv.org/abs/2406.09403.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-R1: Incentivizing reasoning capability in multimodal large language models, 2026. URL https://arxiv.org/abs/2503.06749.

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.

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.

Xuchen Li, Xuzhao Li, Renjie Pi, Shiyu Hu, Jian Zhao, and Jiahui Gao. Beyond accuracy: Evaluating grounded visual evidence in thinking with images, 2026. URL https://arxiv.org/abs/2601.11633.

Yifan Liu, Fangneng Zhan, Kaichen Zhou, Yilun Du, Paul Pu Liang, and Hanspeter Pfister. Abstract 3D perception for spatial intelligence in vision-language models, 2026. URL https://arxiv.org/abs/2511.10946.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-RFT: Visual reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2503.01785.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models, 2023. URL https://arxiv.org/abs/2304.09842.

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, 2024. URL https://arxiv.org/abs/2310.02255.

Juhong Min, Lazar Valkov, Vitali Petsiuk, Hossein Souri, and Deen Dayal Mohan. Foveated reasoning: Stateful, action-based visual focusing for vision-language models, 2026. URL https://arxiv.org/abs/2604.21079.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong, Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. We-Math: Does your large multimodal model achieve human-like mathematical reasoning?, 2024. URL https://arxiv.org/abs/2407.01284.

Qi Song, Honglin Li, Yingchen Yu, Haoyi Zhou, Lin Yang, Song Bai, Qi She, Zilong Huang, and Yunqing Zhao. CodeDance: A dynamic tool-integrated MLLM for executable visual reasoning, 2026. URL https://arxiv.org/abs/2512.17312.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning, 2025. URL https://arxiv.org/abs/2505.08617.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning, 2023. URL https://arxiv.org/abs/2303.08128.

Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Xiansheng Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-RFT: Reinforcement fine-tuning for visual reasoning of vision language models, 2025. URL https://arxiv.org/abs/2503.20752.

Omkar Thawakar, Dinura Dissanayake, Ketan More, Ritesh Thawkar, Ahmed Heakl, Noor Ahsan, Yuhao Li, Mohammed Zumri, Jean Lahoud, Rao Muhammad Anwer, Hisham Cholakkal, Ivan Laptev, Mubarak Shah, Fahad Shahbaz Khan, and Salman Khan. LlamaV-o1: Rethinking step-by-step visual reasoning in LLMs, 2025. URL https://arxiv.org/abs/2501.06186.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with MATH-Vision dataset, 2024a. URL https://arxiv.org/abs/2402.14804.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models, 2024b. URL https://arxiv.org/abs/2408.15556.

Yikun Wang, Siyin Wang, Qinyuan Cheng, Zhaoye Fei, Liang Ding, Qipeng Guo, Dacheng Tao, and Xipeng Qiu. VisuoThink: Empowering LVLM reasoning with multimodal tree search, 2025. URL https://arxiv.org/abs/2504.09130.

Qianshan Wei, Yishan Yang, Siyi Wang, Jinglin Chen, Binyu Wang, Jiaming Wang, Shuang Chen, Zechen Li, Yang Shi, Yuqi Tang, Weining Wang, Yi Yu, Chaoyou Fu, Qi Li, and Yi-Fan Zhang. Agentic-MME: What agentic capability really brings to multimodal intelligence?, 2026. URL https://arxiv.org/abs/2604.03016.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. VTool-R1: VLMs learn to think with images via reinforcement learning on multimodal tool use, 2026. URL https://arxiv.org/abs/2505.19255.

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

xAI. Grok-1.5 vision preview, 2024. URL https://x.ai/news/grok-1.5v. Introduces the RealWorldQA benchmark.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. LogicVista: Multimodal LLM logical reasoning benchmark in visual contexts, 2024. URL https://arxiv.org/abs/2407.04973.

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.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, Wenhai Wang, Jifeng Dai, and Jinguo Zhu. VisuLogic: A benchmark for evaluating visual reasoning in multi-modal large language models, 2025. URL https://arxiv.org/abs/2504.15279.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. MM-REACT: Prompting ChatGPT for multimodal reasoning and action, 2023. URL https://arxiv.org/abs/2303.11381.

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

Xinlei Yu, Chengming Xu, Guibin Zhang, Zhangquan Chen, Yudong Zhang, Yongbo He, Peng-Tao Jiang, Jiangning Zhang, Xiaobin Hu, and Shuicheng Yan. VisMem: Latent vision memory unlocks potential of vision-language models, 2026. URL https://arxiv.org/abs/2511.11007.

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

Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. MLLMs know where to look: Training-free perception of small visual details with multimodal LLMs, 2025a. URL https://arxiv.org/abs/2502.17422.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. MathVerse: Does your multi-modal LLM truly see the diagrams in visual math problems?, 2024. URL https://arxiv.org/abs/2403.14624.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images, 2025b. URL https://arxiv.org/abs/2508.11630.

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

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.

Shitian Zhao, Shaoheng Lin, Ming Li, Haoquan Zhang, Wenshuo Peng, Kaipeng Zhang, and Chen Wei. PyVision-RL: Forging open agentic vision models via RL, 2026. URL https://arxiv.org/abs/2602.20739.

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

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

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.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. DynaMath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models, 2025. URL https://arxiv.org/abs/2411.00836.

A System and Reproducibility

A.1 Runtime Interface and Algorithm

The Visual Workspace is a typed visual-artifact state layer. Its minimal operations are summarized below; model history, step budgets, deadline guidance, answer handling, and the Docker executor are integration policies surrounding this interface.

Registration records a file's type, provenance, and optional parents and returns a stable asset identifier. Resolve/List retrieves metadata by identifier or enumerates artifacts using metadata filters. Promotion puts an existing asset into an active slot and returns a receipt; ActiveSlots reports the current identifiers and metadata. Payload eviction releases non-active inline image data while preserving the file, identifier, and metadata. These operations are independent of the concrete sandbox executor.

Algorithm 1 expands the runtime loop. The first source image initially seeds `focus` and remains registered in the ledger. All compiler-matched settings use the same source-image initialization; reported derived-image counts exclude this source. Replacing `focus` changes which asset is active but does not delete the source file or its ledger entry.

Algorithm 1 VLM-in-Sandbox with Visual Workspace

Require: Images I, query q, maximum steps $T_{\max}$, raw-turn window k, deadline horizon d

$\mathcal{L}_0 \leftarrow \textsc{Register}(I)$
$\mathcal{C}_0 \leftarrow \{\texttt{focus} : I_1,\ \texttt{aux} : \varnothing\}$; $\mathcal{T} \leftarrow \varnothing$
for $t = 1, \ldots, T_{\max}$ **do**
  $r_t \leftarrow T_{\max} - t + 1$
  $g_t \leftarrow \textsc{DeadlineGuidance}(r_t, d, \textsc{AnswerFileExists}())$
  $x_t \leftarrow \textsc{Compile}(q, \mathcal{T}_{[-k:]}, \textsc{Recap}(\mathcal{T}_{<-k}), \mathcal{L}_{t-1}, \mathcal{C}_{t-1}, g_t)$
  $a_t \leftarrow \mathrm{VLM}(x_t)$
  if $a_t =$ Submit and AnswerFileExists() **then**
    return designated answer-file contents
  else if $a_t =$ Submit **then**
    $(\mathcal{L}_t, \mathcal{C}_t) \leftarrow (\mathcal{L}_{t-1}, \mathcal{C}_{t-1})$
    $o_t \leftarrow$ missing-answer-file rejection
  else if $a_t = \textsc{Promote}(a, s)$ and $a \in \mathcal{L}_{t-1}$ **then**
    $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1}$; $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}[s \mapsto a]$
    $o_t \leftarrow$ promotion receipt
  else
    $(o_t, \Delta\mathcal{L}) \leftarrow \textsc{Execute}(a_t)$
    $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \Delta\mathcal{L}$; $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$
  end if
  $\mathcal{L}_t \leftarrow \textsc{EvictPayloads}(\mathcal{L}_t, \textsc{Assets}(\mathcal{C}_t) \cup I)$
  append (a_t, o_t) to $\mathcal{T}$
end for
return answer-file contents if present; otherwise no answer

Legacy image reads are normalized to `promote` and counted as promotion.

A.2 Prompts, Tools, and Evaluation Settings

The compiler reconstructs every request from canonical state rather than mutating an unbounded chat transcript. The following description specifies the request blocks used by VLM-in-Sandbox and the matched controls.

The request contains six blocks in logical order: fixed system and task instructions, including the question, output location, and tool-use contract; deterministic older-turn summaries of actions, observations, and relevant artifact references; the three most recent uncompressed assistant/action/observation turns; a workspace summary of current active slots and available artifacts; runtime guidance with remaining steps, answer-file presence, and deadline instructions; and multimodal image payloads for

the current `focus` and effective `aux` assets. Active images are rendered as multimodal parts rather than text paths.

System-prompt structure. All settings use common task instructions, file-based execution, and an answer-submission workflow. The excerpt below summarizes these shared instructions; it is not a verbatim full prompt.

```
Solve the task using the supplied images and sandbox tools when useful.
Save scripts and reusable visual artifacts in the workspace.
Base the answer on visual evidence and tool results.
Write only the final answer in the required format, then submit.
```

The visual-policy block distinguishes the settings. Original Append-only attaches generated images to conversation history. Within the matched comparison, Matched-Auto-All retains all generated images; Auto-Recent-2 retains the two most recent; Matched-Append retains all model-selected images; and VLM-in-Sandbox replaces the selected active slots. The matched configurations otherwise share their prompts and execution settings.

The model can execute programs, manipulate files, and submit answers through the interfaces in Table 3. Promotion routes an existing image; it does not perform image processing.

Table 3: **Model-visible tool interfaces.**

Interface	Function
`execute_bash`	Execute code or shell commands and return text observations.
`str_replace_editor`	Read, create, and edit files; its `promote` command selects an existing image for an active slot.
`submit`	Finalize the answer stored in the designated output file.

History and termination. Earlier turns are summarized deterministically from recorded actions and observations, without an additional model call. Budget reminders begin with three calls remaining: finish verification, write the answer, and submit. Submission requires an answer file; at the step limit, an existing non-empty answer is still scored. All matched configurations use these same rules.

Table 4 lists the common inference configuration. The matched controls change only automatic visibility and retention policy as specified in Figure 4; Original Append-only is a submitted reference that uses its original append-only history rather than a cell of the matched factorial.

Table 4: **Core inference settings.**

Setting	Value
Maximum agent steps $T_{\max}$	20
Deadline horizon d	3
Submitted/matched recent-turn window	Three turns plus deterministic recap
Active visual slots	`focus`, `aux`
Temperature	1.0
Maximum token limit	131,072 in submitted and matched runs
Maximum generation request per call	32,768 tokens
Maximum retained response length	8,192 tokens
Request/tool timeout	1,200 s per model request; 120 s per sandbox command
Sandbox backend	Docker executor
Local cache-study serving	vLLM on 8×H20, tensor parallelism 8

Scoring. We follow each benchmark's official evaluation protocol. HRBench-4K uses the same GPT-4o-mini yes/no judge across settings. V*Bench, WeMath, MMStar, RealWorldQA, and MMMU

use option parsing or normalized exact matching; MathVision-mini uses official answer extraction and symbolic/numeric equivalence. Multiple-choice responses contain the option letter; other questions require a short answer. Invalid or missing answers receive zero.

A.3 Datasets and Experimental Configuration

Table 5: **Datasets and scoring protocols.** Public source identifiers and split names specify the evaluated releases. License statements follow the corresponding official dataset cards; mixed-source benchmarks remain subject to their upstream material terms.

Evaluation splits and scoring			
Dataset	**Split/release**	N	**Scorer**
HRBench-4K	`hr_bench_4k.tsv`; 200 questions × four cyclic option orders	800	Official LLM judge
V*Bench	`test`	191	Deterministic
WeMath	`testmini`	1,740	Deterministic
MathVision-mini	`testmini`	304	Official equivalence
MMStar	`val`	1,500	Deterministic
RealWorldQA	`test`	765	Deterministic
MMMU	Merged `dev+validation`, 30 subjects	1,050	Deterministic

Public sources and terms		
Dataset	**Public source**	**License/terms**
HRBench-4K	`DreamMr/HR-Bench`	Dataset-card field: "other"; no standard open license declared
V*Bench	`craigwu/vstar_bench`	No dataset license declared on the official card; upstream terms apply
WeMath	`We-Math/We-Math`	CC BY-NC 4.0
MathVision-mini	`MathLLMs/MathVision`	MIT (dataset card)
MMStar	`Lin-Chen/MMStar`	No standard dataset license declared; upstream sample terms apply
RealWorldQA	`xai-org/RealworldQA`	CC BY-ND 4.0
MMMU	`MMMU/MMMU`	Apache-2.0 (dataset card); source-material disclaimer applies

Table 6: **Overview of experimental settings.** Common inference parameters are listed in Table 4.

Study	**Models**	**Evaluation scope**	**Design**
Main evaluation	Four base VLMs	Seven benchmarks	Vanilla, Append-only, and VLM-in-Sandbox
Matched controls	GPT-4.1-mini	Three benchmarks; 1,260 examples	Visibility × retention
Repeated evaluation	GPT-4.1-mini; Qwen3.5-9B	V*Bench; RealWorldQA	Three evaluations per setting
Local efficiency	Qwen3.5-9B	Three benchmarks; 1,260 examples	Five settings with prefix caching on 8×H20
Context scaling	Recorded trajectories	100 trajectories	Fixed-trace replay with 4–32 artifacts

We retain sample-level predictions, tool trajectories, and visual-state transitions for paired analysis and qualitative inspection.

Software environment. The sandbox uses a Linux Docker image with Python 3.11 and standard libraries for numerical computation, symbolic mathematics, plotting, image processing, and OCR. Local models are served separately through vLLM. The efficiency study uses $8\times$H20 with tensor parallelism 8 and request concurrency 1.

A.3.1 Executor overhead and reliability

We measure the executor with local Qwen3.5-9B under Original Append-only on the complete three-benchmark collection. Each container is limited to 2 vCPUs and 4 GiB RAM, with networking disabled. A prebuilt image is reused; image preparation is excluded from per-example latency. Inputs are read-only, and each episode has an isolated writable workspace.

Table 7: **Executor overhead and reliability** ($N = 1{,}260$). Lifecycle time includes container creation, startup, and removal.

Measurement	Value
Container lifecycle / example	0.82 s
Original Append-only E2E / example	40.1 s
Lifecycle share of E2E	2.0%
Container lifecycle failures	0 / 1,260
Out-of-memory or resource-limit failures	0 / 1,260
Episodes with a failed tool call	10 / 1,260 (0.8%)

These measurements characterize executor overhead and reliability, not a host-versus-container speed difference.

B Workspace and Efficiency Diagnostics

The matched study establishes the accuracy–cost trade-off of visual-state policies. We now examine how a growing artifact collection translates into model-visible context, whether the smaller workload remains beneficial with prefix caching, and how selected evidence supports individual solutions; Appendix E provides the case trajectories.

B.1 Context Growth and Visibility Control

Persistent storage and active visibility serve different purposes: stored files preserve evidence for later access, whereas active slots determine which derived images enter the next request. We test this distinction with fixed-trace replays and then examine the capacity and history settings on actual benchmark runs.

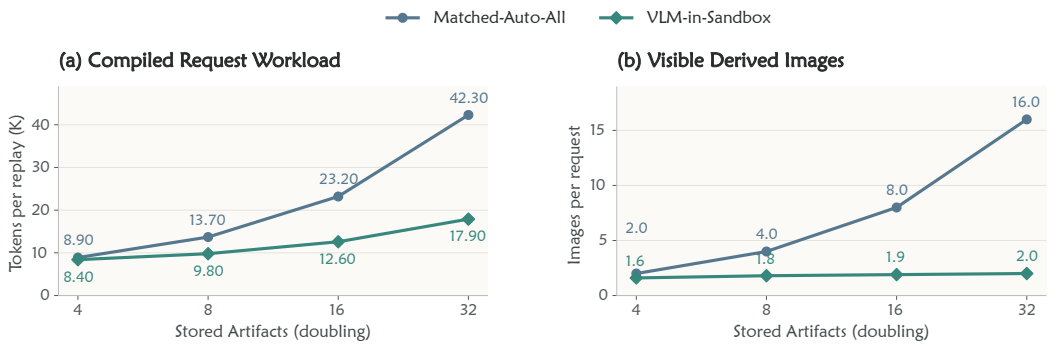

Figure 6: **Stored artifacts versus model-visible context.** Open-loop replays of 100 successful trajectories, with actions, observations, selections, and termination fixed. Pixel-identical copies increase the artifact count without adding visual information. Left: tokens summed over compiled requests per replay. Right: mean visible derived images per request, excluding the source image. No model calls are made.

Figure 6 increases stored artifacts to 4, 8, 16, and 32. Matched-Auto-All grows from 8.90K to 42.30K tokens per replay, whereas VLM-in-Sandbox grows from 8.40K to 17.90K while keeping visible derived images near two. The workspace therefore decouples the size of the stored collection from the number of images repeatedly exposed to the model. Its total token count still grows with textual history and artifact metadata; bounded visual slots do not imply constant total context. This diagnostic measures request scaling, not task accuracy or closed-loop long-horizon performance. Appendix C specifies the replay construction.

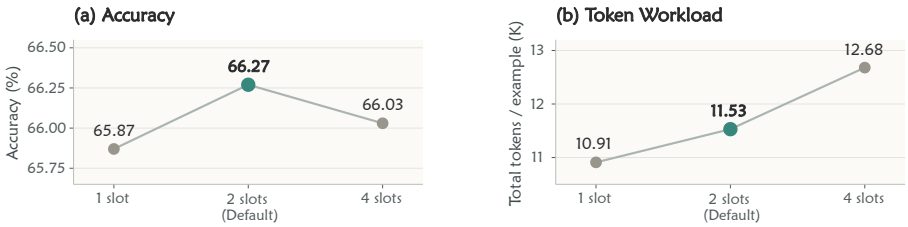

Figure 7: **Active visual capacity.** Sample-weighted GPT-4.1-mini results on the matched-study collection ($N = 1{,}260$). Separate panels show accuracy and mean total tokens per example; the two-slot default is highlighted. Differences are descriptive point estimates.

Figure 7 examines whether a small active context is sufficient. One slot is cheapest, two slots achieve the highest observed accuracy, and four slots increase token use without improving the point estimate. These results support a compact working set, without establishing a statistically unique optimum. The ledger retains files regardless of slot replacement, so a smaller active context need not discard the underlying evidence.

Textual history contributes separately to the request workload. Replacing unlimited raw replay with three recent turns plus deterministic recap reduces total tokens from 12.61K to 11.53K (8.6%) and changes accuracy from 65.87% to 66.27%. This comparison changes both the replay window and older-turn recap; it evaluates the combined history policy rather than recap alone. Table 9 complements the context analysis with step-budget sensitivity.

B.2 Inference Efficiency

A smaller request does not automatically imply a proportional latency reduction: prefix caching can reuse previously processed inputs, and model, vision, and tool execution all contribute to runtime. We therefore compare all five settings under one local serving configuration.

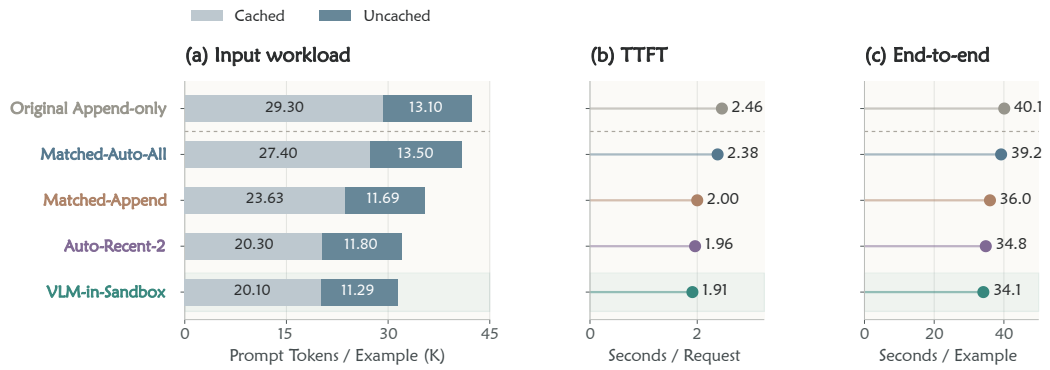

Figure 8: **Inference with prefix caching.** Qwen3.5-9B served by vLLM on 8×H20 (tensor parallelism 8, request concurrency 1), on the same three complete benchmark splits ($N = 1{,}260$). Stacked bars separate cached and uncached prompt tokens per example. TTFT is measured per request; E2E is measured per example. The dashed divider separates Original Append-only from the matched configurations.

Figure 8 shows that prefix reuse does not reverse the efficiency advantage. VLM-in-Sandbox has a lower cache-hit ratio than Matched-Auto-All (64.0% versus 67.0%), but also less uncached input (11.29K versus 13.50K tokens per example). Relative to that control, it reduces total tokens by 23.5%, uncached prompt tokens by 16.4%, TTFT by 19.7%, and E2E by 13.0%. The lower hit ratio reflects a different request workload; it is not, by itself, evidence of a higher inference cost.

Each configuration starts from an empty cache. Timing includes multimodal preprocessing and vision encoding; E2E additionally includes sandbox lifecycle and excludes offline scoring. These measurements characterize this local backend, not API pricing. Table 12 gives the complete prompt, completion, cache, and latency breakdown.

Container creation, startup, and removal take 0.82 seconds per example in the local $N = 1{,}260$ evaluation, or 2.0% of Original Append-only's 40.1-second E2E time. No create/start/remove or resource-limit failures are observed. This is measured lifecycle time, not a host-versus-container difference; Docker remains the execution backend rather than an accuracy-enhancing component. Appendix A.3.1 reports the resource configuration and failure accounting.

C Extended Quantitative Results

C.1 Matched Controls and Component Sensitivity

Figure 9 reports the complete three-dataset matched study. Accuracy and token cost respond to different factors: how much of the accuracy gain, and how much of the token saving, comes from visibility and how much from retention. Original Append-only is shown as the submitted reference but is not a cell of the 2×2 factorial because it uses the original append-only history. The other four configurations share the compiler and tool harness and vary visibility and retention, and the Original Append-only and VLM-in-Sandbox values reuse the corresponding main-evaluation results. Figure 4 summarizes the factorial in the main text; the panels here add the reference setting and the pooled collection.

Reading the two rows separately decomposes the headline gain. Aligning the harness alone, from Original Append-only to Matched-Auto-All, accounts for $+0.79$ pp and a 0.6% token change; the remaining $+1.83$ pp and essentially all of the 18.6% token reduction come from managing visual state. The saving is uneven across splits, from 26.3% on MathVision-mini, the most expensive, to 9.2% on RealWorldQA, the cheapest.

The two factors then divide the work unevenly. Applied separately, selection and bounded retention give $+0.56$ and $+1.19$ pp, close to their joint $+1.83$ pp, whereas their separate token reductions of 7.8% and 16.7% fall well short of adding to the joint 18.6%. Once the active context is capped, selection therefore changes mainly *which* evidence is visible rather than how much of it, and it pays

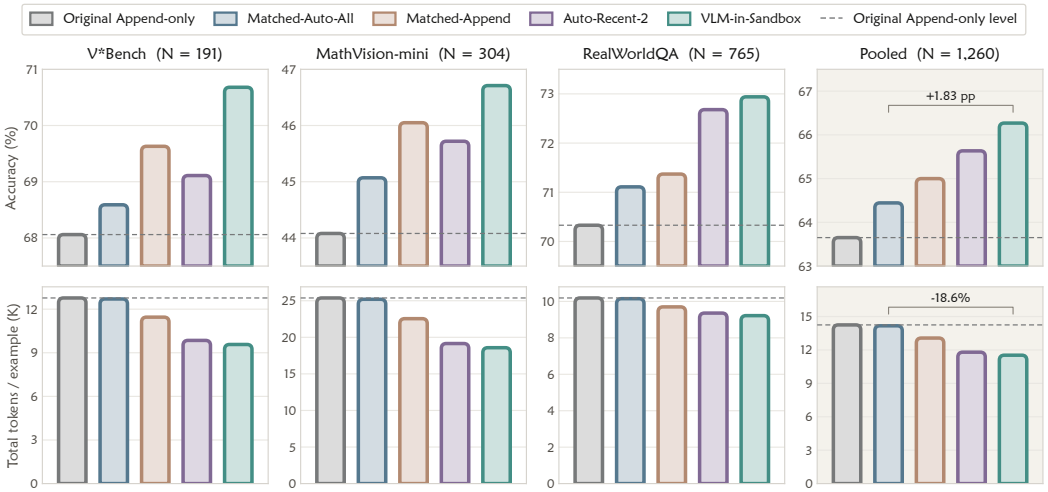

Figure 9: **Full matched-control results.** GPT-4.1-mini on the three complete splits and their sample-size-weighted pooling, with accuracy (top row) and mean total tokens per example (bottom row) on separate axes. The four matched settings cross visibility (automatic or model-directed) with retention (all images or two active slots). Original Append-only uses the original append-only history instead of the shared prompt compiler, so it is a reference rather than a factorial cell; the dashed line marks its level in every panel. Brackets in the pooled column compare the compiler-matched control with VLM-in-Sandbox.

off in accuracy rather than in cost. Table 8 tests these contrasts example by example rather than through aggregate rates: the two comparisons separated from zero at the 5% level both add bounded retention, whereas adding selection at fixed capacity is not conclusive ($p = 0.185$).

Table 8: **Paired comparisons among matched settings.** Confidence intervals use 50,000 fixed-seed paired percentile-bootstrap resamples; p-values use exact McNemar tests. Token change is the right-hand setting relative to the left-hand setting.

Comparison	**ΔAcc.**	**95% CI**	p	**ΔTokens**
Matched-Auto-All → VLM-in-Sandbox	**+1.83 pp**	**[0.40, 3.25]**	**0.014**	−18.6%
Matched-Auto-All → Auto-Recent-2	+1.19 pp	[0.00, 2.38]†	0.072	−16.7%
Matched-Append → VLM-in-Sandbox	+1.27 pp	[0.24, 2.38]	0.026	−11.8%
Auto-Recent-2 → VLM-in-Sandbox	+0.63 pp	[−0.16, 1.51]	0.185	−2.3%

†The endpoint is displayed to two decimals. The bootstrap interval and exact McNemar test are distinct paired procedures and need not cross their decision thresholds identically.

The history-policy comparison in Section B.1 changes both the raw replay window and older-turn recap, rather than isolating recap alone. Figure 7 reports the accuracy and token trade-off for one, two, and four active slots.

Table 9: **Agent-step budget sensitivity** on the matched-study collection ($N = 1{,}260$), GPT-4.1-mini. Only the maximum number of agent steps varies; the default budget is highlighted. *Exhausted* counts episodes that reach the limit before an accepted `submit` call.

Step budget	**Acc. (%)**	**Total tok.**	**Mean steps**	**Exhausted**
10	65.71	10.74K	4.3	26 (2.06%)
20	**66.27**	11.53K	5.0	0
30	**66.27**	11.61K	5.1	0

The return on additional steps falls off sharply. Reducing the default budget to 10 steps saves 6.9% of total tokens (11.53K to 10.74K per example) but costs 0.56 pp of pooled accuracy (66.27% to 65.71%), whereas raising it to 30 steps adds 0.7% tokens and leaves the pooled estimate unchanged at 66.27%. Mean trajectory length rises only from 5.0 to 5.1 steps over that increase, so the additional headroom is almost never used.

Termination is governed by the policy rather than by the cap. At 20 steps no episode reaches the limit, so every trajectory ends in an accepted `submit`; at 10 steps, 26 episodes (2.06%) exhaust the budget, which places the accuracy difference in a small tail of long trajectories rather than in typical behavior. The two runs are not a pure truncation of one another, because the deadline reminder is defined relative to the remaining budget and therefore fires at a different absolute step; the comparison evaluates the step budget as a policy rather than isolating truncation. Removing the reminder from the 20-step configuration leaves 30 episodes (2.38%) without an accepted `submit` call, whereas issuing it with three remaining steps, the submitted default ($d = 3$ in Table 4), reduces that count to zero, and with five remaining steps to three. Episodes stopped at the limit are still scored from a non-empty answer file when one exists, and termination status is recorded separately, so these counts measure protocol compliance rather than discarded examples.

C.2 Paired Reliability and Complete Transitions

Table 10: **Complete outcome transitions to VLM-in-Sandbox.** Each row covers all seven benchmarks ($N = 6{,}350$). Stable correct/wrong means both settings agree on correctness; rescue and regression mean wrong-to-correct and correct-to-wrong, respectively. The reference setting differs by model.

Model	Reference	Stable correct	Rescue	Regression	Stable wrong
GPT-4.1-mini	Original Append-only	3,982	**302**	142	1,924
Qwen3.5-9B	Vanilla VLM	4,224	**734**	411	981

The primary GPT comparison is Original Append-only to VLM-in-Sandbox. We use a paired percentile bootstrap with 50,000 resamples and a fixed analysis seed. Pooled resampling is benchmark-stratified; the four HRBench cyclic variants are grouped by base question and sampled as a cluster. HRBench and the pooled collection use cluster-aware paired randomization tests, while other individual benchmarks use exact McNemar tests.

Table 11: **GPT-4.1-mini paired results by benchmark.** Rescue and regression are defined relative to Original Append-only.

Dataset	N	Rescue	Regression	ΔAcc.	95% CI	p
HRBench-4K	800	30	17	+1.63 pp	$[-0.75, 4.13]$	0.242
V*Bench	191	14	9	+2.62 pp	$[-2.09, 7.85]$	0.405
WeMath	1,740	48	25	+1.32 pp	[0.34, 2.30]	0.0095
MathVision-mini	304	17	9	+2.63 pp	$[-0.66, 5.92]$	0.169
MMStar	1,500	121	43	+5.20 pp	[3.53, 6.87]	< 0.001
RealWorldQA	765	36	16	+2.61 pp	[0.78, 4.44]	0.0078
MMMU	1,050	36	23	+1.24 pp	$[-0.19, 2.67]$	0.117
All samples	6,350	**302**	**142**	**+2.52 pp**	**[1.83, 3.21]**	< 0.001

The pooled analysis is the primary statistical result; the per-benchmark results show how the gains are distributed. Figure 5(c–f) reports all three runs on complete V*Bench and RealWorldQA splits at temperature 1.0. Trial 1 is the submitted Table 1 result; Trials 2–3 are independent repeats. Points are individual runs and horizontal bars are means, not confidence intervals.

The Qwen comparison yields 1.79 rescues per regression and 323 additional correct answers. RealWorldQA is the negative aggregate cell, with 53 rescues and 77 regressions. These outcomes

motivate selective sandbox invocation on questions that a model can answer directly; they do not negate recovery on other examples.

C.3 Scaling and Systems Efficiency

The open-loop diagnostic in Figure 6 freezes trajectories and makes no model calls. It measures how compiled requests scale with stored visual artifacts, not task accuracy or closed-loop behavior. At recorded generation points, we add pixel-identical copies with distinct replay identifiers until the ledger contains 4, 8, 16, or 32 artifacts. Actions, observations, selections, and termination remain fixed; only the requests are recompiled. The copies increase stored evidence without introducing new visual information.

Table 12: **Complete prefix-cache and latency accounting.** Qwen3.5-9B is served with vLLM on 8×H20 (tensor parallelism 8, concurrency 1), $N = 1{,}260$. Token and E2E values are per example; TTFT is per request.

Token workload per example (K tokens)

Setting	Prompt	Cached	Uncached	Completion	Total
Original Append-only	42.40	29.30	13.10	3.67	46.07
Matched-Auto-All	40.90	27.40	13.50	3.55	44.45
Matched-Append	35.32	23.63	11.69	2.91	38.23
Auto-Recent-2	32.10	20.30	11.80	2.75	34.85
VLM-in-Sandbox	**31.39**	20.10	**11.29**	**2.63**	**34.02**

Prefix reuse and latency

Setting	Cache hit (%)	TTFT (s)	E2E (s)
Original Append-only	69.1	2.46	40.1
Matched-Auto-All	67.0	2.38	39.2
Matched-Append	66.9	2.00	36.0
Auto-Recent-2	63.2	1.96	34.8
VLM-in-Sandbox	64.0	**1.91**	**34.1**

Each setting begins from an empty cache. Prompt counts include multimodal placeholders; cached-prefix tokens are a subset of prompt tokens; prefix-KV hit ratio is cached-prefix tokens divided by prompt-input tokens; total is prompt plus completion. TTFT and E2E include multimodal preprocessing and vision encoding, while E2E also includes sandbox lifecycle and excludes offline scoring. Relative comparisons are restricted to this backend.

In the separate GPT-4.1-mini matched study, oversized requests affect 4, 2, 1, 0, and 0 of the 1,260 episodes for Original Append-only, Matched-Auto-All, Matched-Append, Auto-Recent-2, and VLM-in-Sandbox, respectively. This count includes episodes that later recover. Table 7 reports executor lifecycle and failure accounting.

D Related Work

D.1 Sandboxed and Tool-Augmented Visual Reasoning

Recent agent systems place models in interactive web, desktop, and visually grounded environments rather than fixed single-turn traces (Zhou et al., 2024; Xie et al., 2024; Koh et al., 2024). Vision-centric benchmarks further show that multimodal agents must coordinate perception, tool use, and long-horizon decisions (Ashraf et al., 2025; Wei et al., 2026; Zhu et al., 2026). Closest in spirit, LLM-in-Sandbox gives language models a general-purpose computer sandbox with filesystem access, terminal commands, editing tools, and outcome-based reinforcement learning (Cheng et al., 2026). Tool-augmented multimodal systems use visual modules, Python programs, sketches, image edits, expert calls, or learned tool policies to make visual reasoning more compositional (Gupta & Kembhavi,

2022; Surís et al., 2023; Yang et al., 2023; Lu et al., 2023; Hu et al., 2024; Fu et al., 2025; Zhao et al., 2025; Zhang et al., 2025b; Hou et al., 2026; Song et al., 2026; Su et al., 2025; Zhao et al., 2026; Zheng et al., 2026; Wu et al., 2026). These works establish the value of external computation, but most organize reasoning around a particular tool library, interpreter, canvas, or learned action policy. VLM-IN-SANDBOX instead asks how image-valued evidence produced inside a general computer workspace should persist, remain addressable, and return to the model.

D.2 VISUAL EVIDENCE STATE AND EVALUATION

Active visual reasoning raises a state-management question: which visual evidence should remain visible after each step? Prior work studies guided search, visual working memory, small-detail localization, foveated focusing, compact visual abstractions, and latent visual memory (Wu & Xie, 2023; Zhang et al., 2025a; Wang et al., 2025; Min et al., 2026; Liu et al., 2026; Yu et al., 2026). Visual Sketchpad maintains an episode-level visual canvas, while PyVision creates and revisits artifacts through a dynamic Python session (Hu et al., 2024; Zhao et al., 2025); latent-memory methods retain information inside model state rather than as file-backed evidence (Yu et al., 2026). VLM-in-Sandbox is complementary: it supplies a persistent filesystem, stable artifact identity and provenance, a bounded set of active visuals, and model-directed reactivation by identifier without requiring training. Section D.3 gives a compact interface-level comparison.

Benchmarks spanning high-resolution search, mathematical reasoning, real-world perception, and agentic tool use stress the need to coordinate fine-grained evidence with multi-step reasoning (Wang et al., 2024b; Wu & Xie, 2023; Zhang et al., 2025c; Qiao et al., 2024; Wang et al., 2024a; Lu et al., 2024; Zhang et al., 2024; Xiao et al., 2024; Zou et al., 2025; Xu et al., 2025; Chen et al., 2024; xAI, 2024; Yue et al., 2024; Ashraf et al., 2025; Li et al., 2025; Zhu et al., 2026; Wei et al., 2026). Evidence-oriented evaluation further suggests that accuracy alone may hide whether the right evidence was used (Li et al., 2026). We therefore complement aggregate accuracy with matched visual-state controls, complete paired transitions, promotion usage, context-scaling diagnostics, and backend-specific latency measurements.

D.3 COMPARISON WITH RELATED APPROACHES

Table 13 compares how related approaches represent, retain, and revisit intermediate visual evidence.

Table 13: **Interface-level comparison with nearby visual-reasoning families.** "Method-specific" indicates that the property depends on the particular canvas, memory, or tool protocol rather than a general file-backed artifact lifecycle.

Artifact state		
Family	**Persistent artifacts**	**Identity / provenance**
Visual scratchpads (Hu et al., 2024)	Episode canvas/state	Method-specific
PyVision (Zhao et al., 2025)	Session artifacts	Session/tool-specific
Multimodal memory (Yu et al., 2026)	Model or memory state	Not a general file ledger
VLM-in-Sandbox	Sandbox filesystem	Ledger ID + provenance

Model-visible state			
Family	**Active visual budget**	**Reactivation**	**Training**
Visual scratchpads	Method-specific	Update or regenerate	Varies
PyVision	No general slot bound	Tool view or rerun	No
Multimodal memory	Method-specific	Retrieval/update	Usually trained
VLM-in-Sandbox	Two active slots	`promote` by ID	No

Visual scratchpads make intermediate visual reasoning explicit through a canvas; PyVision demonstrates the value of dynamic Python tooling and session artifacts; multimodal-memory methods retain information in learned or method-specific memory. VLM-in-Sandbox contributes a different systems abstraction: generated files receive a persistent lifecycle of registration, provenance, bounded model-visible state, payload eviction, and model-directed reactivation. These approaches are complementary rather than mutually exclusive.

E Qualitative Analysis

Figures 10–12 show three uses of selected visual evidence: comparing complementary views, refining diagnostics, and inspecting rendered alternatives. The cases illustrate reasoning behavior rather than replacing the matched comparisons in Section 3.2.

In the butterfly puzzle, the agent separates a composite into outline, sticker, and candidate panels, then selects the sticker panel as `focus` and the candidates as `aux`. The selected views support direct comparison while the remaining panel stays available as a file.

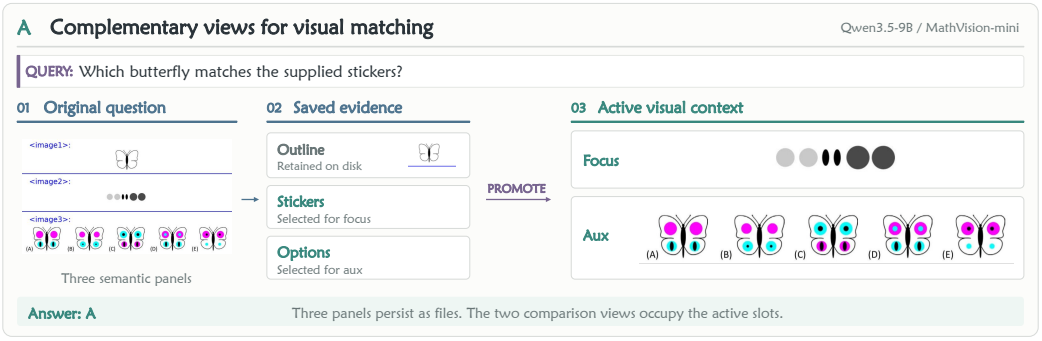

Figure 10: **Complementary views for visual matching** (Qwen3.5-9B, MathVision-mini). The agent retains three panels as files and selects the stickers and candidates for the two active slots. Their comparison supports answer A.

In the road-line example, a broad bright-pixel mask responds to trees and vehicles. The agent narrows its analysis to road crops, edges, and color masks, including a promoted yellow mask, before answering zero. The empty yellow mask is interpreted alongside the other views, not as standalone proof that markings are absent.

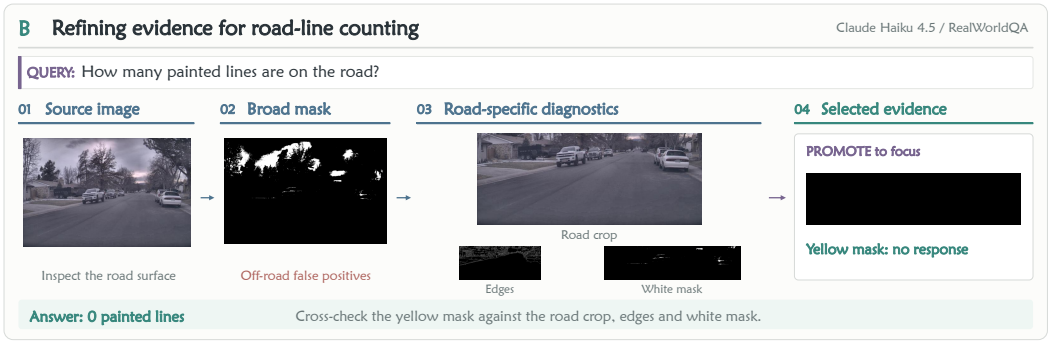

Figure 11: **Refining evidence for road-line counting** (Claude Haiku 4.5, RealWorldQA). Selected stages of a longer trajectory show a broad mask followed by road-specific diagnostics. The promoted yellow mask is considered together with the crop, edges, and white mask before the answer of zero painted lines.

In the symmetry example, the agent renders the consequences of three candidate moves in a single comparison image and promotes it to identify the move that does not preserve a reflection axis. The

butterfly and symmetry examples use Qwen3.5-9B; the road example is an additional Claude Haiku 4.5 trajectory, separate from the four-model main comparison.

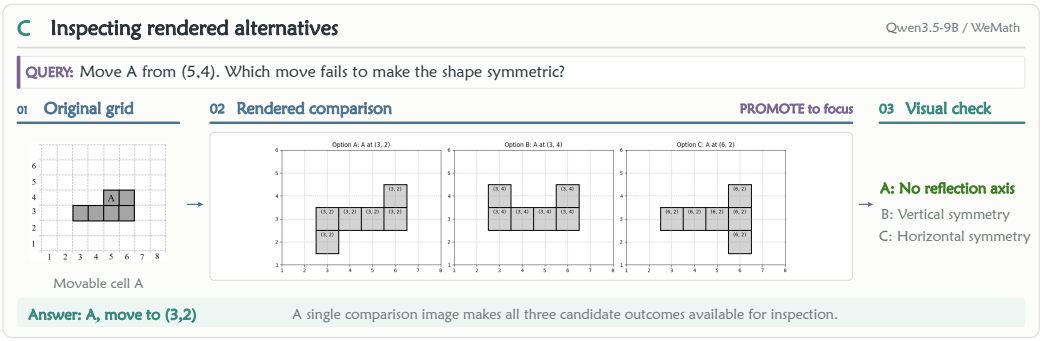

Figure 12: **Inspecting rendered alternatives** (Qwen3.5-9B, WeMath). The agent renders the three candidate moves and promotes the comparison image to `focus`. Moving A to $(3, 2)$ fails to produce reflection symmetry; the other two moves succeed. The original generated rendering is preserved.

These examples share a common pattern: tools produce inspectable evidence, and the model chooses which evidence to bring back into context. Selection alone does not guarantee a correct interpretation. Section E.1 shows an overlay marking vehicle rears that is misread as pavement arrows despite a correct final answer, illustrating why evidence quality and final-answer accuracy must remain distinct.

E.1 A Failure of Intermediate Evidence Interpretation

The parking-aisle example in Figure 13 illustrates a limitation distinct from answer accuracy. The model promotes a detection overlay and interprets its elongated contours as directional pavement arrows, but the boxes visibly lie on vehicle rears. The recorded final answer, yes, is correct for this RealWorldQA example; the stated visual rationale is nevertheless unsupported by the selected artifact. This example motivates checking the spatial grounding of intermediate evidence as well as its visibility. It is not counted as an answer-level regression.

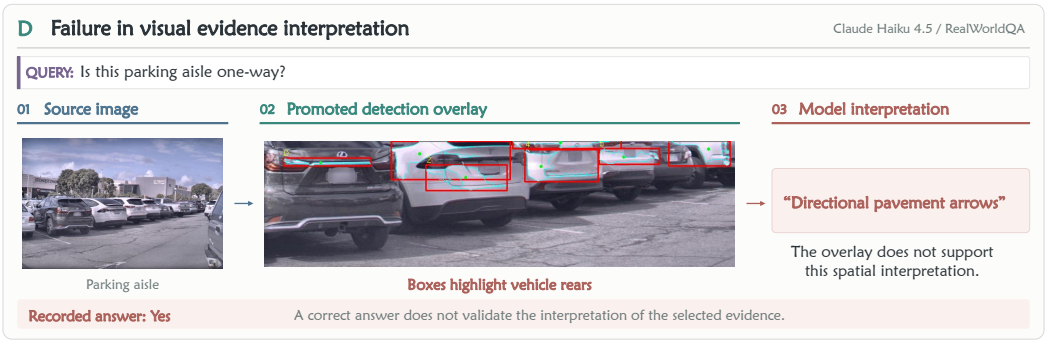

Figure 13: **Correct answer with an unsupported visual rationale** (Claude Haiku 4.5, RealWorldQA). The promoted overlay highlights vehicles, whereas the model describes pavement arrows. The original detection overlay is retained to expose the mismatch between evidence and interpretation.