HLL: CAN AGENTS CROSS HUMANITY'S LAST LINE OF VERIFICATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal agents are increasingly expected to operate graphical interfaces on behalf of users, yet real workflows often contain verification gates designed to constrain automation. Existing GUI-agent benchmarks emphasize navigation and task completion, while such verification steps are omitted. This leaves an important capability under-examined: whether an agent can couple visual understanding with grounded interaction, state tracking, and process-consistent execution under controlled verification constraints. To address this gap, we introduce **HLL**, a controlled benchmark for interactive CAPTCHA-style verification in an authorized local simulator. HLL spans ten task families and factorizes evaluation along three dimensions: intrinsic task difficulty, webpage distraction, and trace-conditioned interaction validation. Agents must produce semantically correct answers through grounded GUI actions, while dynamic variants additionally validate simulator-observable evidence. We evaluate eight multimodal agents in a closed-loop Android/Chrome environment using a common Mobile-Agent-v2 harness. Within-run audits show that correct submissions can still fail the configured dynamic rules. Supplementary controls also identify legal strategies rejected by these rules. A repair study recovers most tested geometric rejections through trace editing. HLL measures simulator-level process compliance, not behavioral authenticity.

1 INTRODUCTION

Multimodal agents are increasingly used to operate graphical interfaces on behalf of users. Recent GUI-agent systems (Qin et al., 2025; Zhang et al., 2025a) can navigate browsers and mobile applications, but completing a workflow also requires handling verification gates that precede actions such as account creation, content access, form submission, or checkout. These gates are important from a defensive perspective because production services use them with server-side signals to limit unwanted automation. This paper isolates the visible interaction component and asks a narrower, measurable question: **how well can multimodal agents integrate perception, grounding, control, and state tracking to complete interactive verification under anti-automation constraints?**

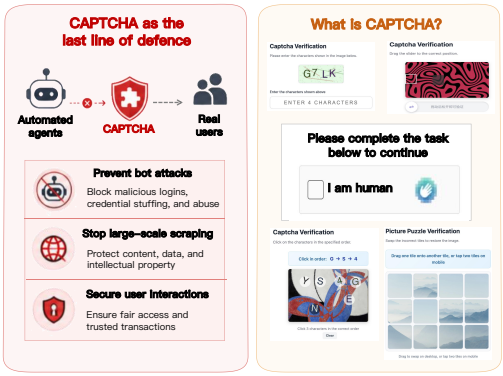


Figure 1: Interactive verification as a controlled test of perception, grounding, and GUI action under anti-automation constraints.

CAPTCHA-style verification provides a compact instance of this problem. Such challenges have evolved from distorted text and image recognition (Algwil, 2023; Searles et al., 2023) toward visual reasoning, clicking, dragging, and interaction-dependent checks (Gao et al., 2021; Ding et al., 2025). For a GUI agent, success therefore involves more than predicting a final answer: the agent must locate the active region, map visual evidence to coordinates, execute valid actions, maintain the evolving page state, and submit a consistent result. We focus on this observable interaction layer rather than the broader production bot-detection stack, which may additionally incorporate device fingerprints, IP reputation, account history, request frequency, or cross-session policies.

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.

Interactive verification remains incompletely characterized in current agent evaluation. Existing web and GUI benchmarks measure navigation and application control (Pan et al., 2024; Deng et al., 2023), browsing and web-game capabilities (Wei et al., 2025; Thomas et al., 2025), or general task completion (Liu et al., 2023b), while verification steps are often filtered out or treated as incidental obstacles. Recent CAPTCHA benchmarks already evaluate multimodal reasoning and interaction (Luo et al., 2025; Wu et al., 2026; Deng et al., 2025). HLL adds explicit process-rule evaluation, legality audits, and trace-editing checks to diagnose gaps between answer correctness, task legality, and configured acceptance. These quantities can disagree even for a correct answer, making both agent execution and validator limitations relevant. Figure 2 illustrates this motivation. A compact challenge can require perception, localization, grounded action, state tracking, and final validation, while identical final answers can arise from very different interaction traces.

To address this gap, we introduce **HLL**, a controlled benchmark for interactive CAPTCHA-style verification by multimodal GUI agents. HLL spans ten task families with heterogeneous interaction requirements and organizes evaluation along three factorized configuration dimensions: intrinsic task difficulty, webpage distraction, and trace-conditioned interaction validation. In dynamic settings, the benchmark records structured *telemetry* through a unified submission payload and applies family-specific rules that test whether a semantically correct answer is supported by sufficient task-relevant interaction evidence. The resulting protocol supports comparisons between clean and distracted pages, static and dynamic verification, and default versus harder variants.

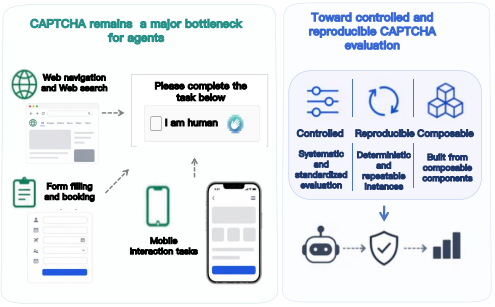


Figure 2: A static answer can conceal failures in localization, grounded action, state tracking, and trace-consistent completion.

HLL is designed as a simulator-level diagnostic benchmark, not as a replica of a commercial bot-detection service. Its dynamic rules provide coarse, configured process constraints under a fixed GUI action interface; they are not learned human-behavior models and do not establish human identity. The benchmark therefore measures how a model performs through the specified harness and environment, while making the failure modes of perception, localization, action calibration, state tracking, and recovery explicit. Any interpretation beyond these controlled conditions requires separate validation on held-out environments and production systems.

In summary, this work makes the following core contributions:

- **Controlled verification benchmark:** We formulate interactive CAPTCHA-style verification as an end-to-end evaluation problem in an authorized local simulator, with explicit separation between semantic correctness and task-relevant interaction evidence.
- **Factorized task suite:** We introduce HLL with ten heterogeneous task families and family-dependent configuration dimensions for intrinsic difficulty, webpage distraction, and trace-conditioned validation.
- **Diagnostic evaluation:** We evaluate eight multimodal agents through a common closed-loop GUI harness and find that semantic correctness alone can overstate successful verification, while trace-conditioned validation exposes additional failures in grounded interaction, state tracking, and process consistency.

2 Related Work

Multimodal GUI agents and their benchmarks. Recent multimodal agents increasingly operate in realistic interfaces, from web environments such as WebArena, VisualWebArena, Mind2Web, and WebCanvas (Zhou et al., 2023; Koh et al., 2024; Deng et al., 2023) to desktop and mobile settings such as OSWorld and AppAgent (Xie et al., 2024; Zhang et al., 2025a), and native GUI agents further improve screen perception and action generation (Hong et al., 2024; Qin et al., 2025; Cheng et al., 2025). Existing benchmarks mainly evaluate general navigation, application control, or task completion (Liu et al., 2023b; Pan et al., 2024; Wei et al., 2025; Thomas et al., 2025), while CAPTCHA-protected pages are often filtered out or treated as incidental obstacles, so they

provide limited evidence about whether agents can handle verification bottlenecks that require fine-grained perception, spatial grounding, action execution, and final submission under anti-automation constraints. Interactive verification also differs from static multimodal understanding benchmarks, where models produce textual answers from fixed images, including VQA and text-aware tasks (Goyal et al., 2017; Zhou et al., 2025) as well as broader multimodal reasoning suites (Lu et al., 2023; Yue et al., 2024; Liu et al., 2024; Yu et al., 2023): in a browser loop, an agent must locate the verification region, ground actions to interface elements, adapt to intermediate state changes, and submit an answer supported by the solving process. A parallel line of work studies GUI-agent robustness under cluttered, adversarial, or dynamically changing environments, showing that agents can be misled by visual distractions and environmental injection (Ma et al., 2024; Chen et al., 2025; Liao et al., 2024), by pop-ups or adversarial web interfaces (Zhang et al., 2025b; Xu et al., 2024), and by broader multimodal or cross-modal attacks (Wu et al., 2024; Bagdasaryan et al., 2023).

CAPTCHA solving and process-level validation. CAPTCHAs were introduced as human verification mechanisms based on problems expected to be easy for humans but difficult for automated programs (von Ahn et al., 2003). Early work studied text-based CAPTCHA usability and security (Bursztein et al., 2011; Algwil, 2023; Searles et al., 2023), while later attacks and defenses extended to image recognition (Chellapilla et al., 2005; Sivakorn et al., 2016; Noury & Rezaei, 2020), visual reasoning (Gao et al., 2021; Wang et al., 2018; Xue et al., 2025), adversarial examples (Shi et al., 2021; Hitaj et al., 2020), and deployed reCAPTCHA-style systems (Plesner et al., 2024). Recent benchmarks already evaluate interaction: Open CaptchaWorld covers 20 CAPTCHA types and annotates cognitive and motor steps with CAPTCHA Reasoning Depth (Luo et al., 2025); MCA-Bench includes clicks, sliding puzzles, and logic questions under a unified robustness evaluation (Wu et al., 2026), and LLM-enhanced solvers highlight the importance of spatial reasoning and multi-step inference (Deng et al., 2025; Teoh et al., 2025). These works have begun to consider reasoning depth, action traces, or grounding annotations (Luo et al., 2025; Deng et al., 2025; Wu et al., 2026), and related visual-spatial reasoning benchmarks expose complementary reasoning gaps (Liu et al., 2023a; Chollet, 2019; Johnson et al., 2017; Ma et al., 2025); HLL complements these directions with inspectable process rules, independent legality checks on sampled traces, and trace-editing controls under family-dependent configurations. These checks expose both execution violations and limitations of the acceptance rules. CAPTCHA systems are also process-oriented in deployment: many verification mechanisms rely on behavioral signals (Acien et al., 2020; 2021), mouse or touch dynamics (Khan et al., 2024; Frank et al., 2012), and continuous authentication signals (Abuhamad et al., 2020; Zaidi et al., 2021) rather than final answers alone. HLL therefore evaluates heterogeneous CAPTCHA families under controlled axes of intrinsic difficulty, webpage distraction, and dynamic interaction validation, and introduces trace-conditioned validation to test whether an agent's answer is supported by task-consistent interaction evidence; it does not reproduce full production-side bot detection.

3 BENCHMARK

In this section, we introduce HLL as a controlled benchmark for measuring verification bottlenecks in multimodal agents. Rather than treating CAPTCHA solving as a narrow recognition problem, HLL formulates it as an end-to-end interaction task in which an agent must perceive the challenge, localize the correct verification region, execute grounded interface actions, and satisfy task-consistent interaction constraints under controlled interface conditions. To make these bottlenecks measurable, HLL organizes evaluation along factorized configuration dimensions and supports fine-grained analysis of where the interaction pipeline breaks down. We first describe the benchmark design goals and formalize the evaluated support set, and then present the task families, configuration dimensions, and evaluation protocol.

3.1 DESIGN GOALS AND BENCHMARK FORMULATION

The design of HLL aims to evaluate end-to-end verification beyond semantic recognition, cover heterogeneous interaction patterns, factorize verification conditions into controllable dimensions, remain reproducible for closed-loop agents, and support interpretable failure analysis beyond a single success flag.

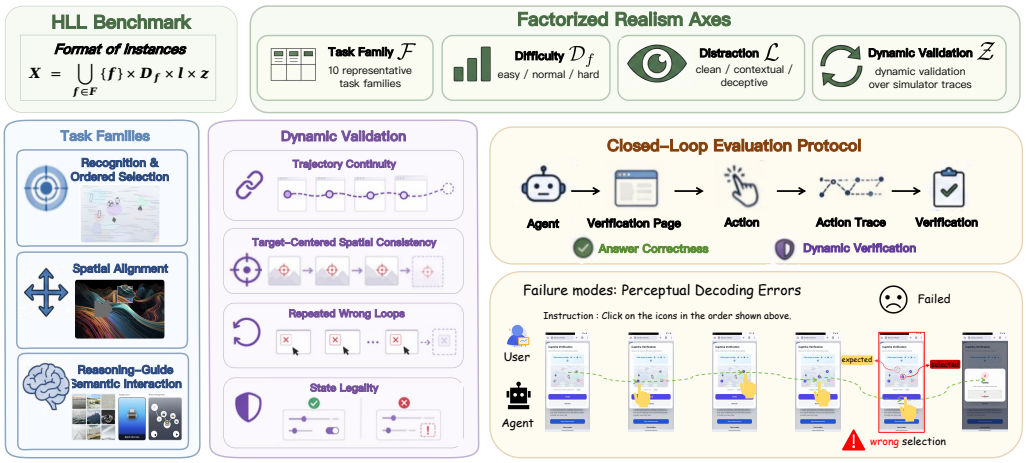


Figure 3: Overview of the HLL benchmark structure. The benchmark combines heterogeneous CAPTCHA task families with controlled realism axes, including intrinsic task difficulty, environmental distraction, and dynamic interaction validation, to expose where multimodal agents fail along the end-to-end verification pipeline.

Following these design goals, we define the benchmark over a family set $\mathcal{F}$, a family-dependent difficulty domain $\mathcal{D}_f$, a distraction-level set $\mathcal{L}$, a dynamic-validation flag $z \in \{0, 1\}$, and a sample index set $\mathcal{S}$. Because not every family supports every configuration, we define a family-dependent set of evaluated cells $\mathcal{C}_f \subseteq \mathcal{D}_f \times \mathcal{L} \times \{0, 1\}$. An evaluated benchmark instance is the tuple.

$$x = (f, d, \ell, z, s) \in \mathcal{X}_{\text{eval}} = \bigcup_{f \in \mathcal{F}} \{(f, d, \ell, z, s) \mid (d, \ell, z) \in \mathcal{C}_f,\ s \in \mathcal{S}\}\,. \tag{1}$$

Here, f specifies the semantic task family, d controls intrinsic hardness when supported, ℓ controls the surrounding webpage context, z indicates whether trace-conditioned validation is enabled, and s indexes generated instances. The family-dependent support set $\mathcal{C}_f$ specifies which combinations of difficulty, distraction, and dynamic validation are actually evaluated for family f.

For coarse-grained analysis, we associate each family with one dominant capability group: recognition, spatial interaction, stateful interaction, or reasoning. This grouping is intended to identify the primary bottleneck exposed by each family.

As summarized in Figure 3, this factorization enables controlled comparisons across clean and cluttered pages, static and dynamic validation, and easy versus hard variants. It shifts evaluation from whether a model solves a CAPTCHA to where and under which realism conditions the interaction pipeline fails. The full distribution of benchmark instances across task families and realism-axis settings is reported in Appendix A.5.

3.2 Task Families and Capability Taxonomy

The semantic backbone of HLL consists of ten *base task families*. These ten families define the benchmark itself; later realism dimensions such as difficulty, distraction, and dynamic validation are layered on top of them and do not constitute new task families.

At a coarse level, the benchmark covers four capability groups. *Recognition and ordered selection* tasks test whether an agent can perceive targets and interact with them in sequence, including text transcription, icon sequence selection, and natural-image sequence selection. *Spatial alignment and local patch reasoning* tasks emphasize geometric precision and low-level action fidelity, including slider alignment, jigsaw alignment, and missing-patch selection. *Stateful puzzle restoration* tasks require the agent to reason over state changes, including board reconfiguration and tile restoration. Finally, *reasoning-guided semantic interaction* tasks stress structured interpretation and coordinated selection, including logic-and-arithmetic interaction and category-guided image selection.

Many task families involve overlapping mixtures of perception, grounding, reasoning, and execution, and HLL does not assume a disjoint decomposition of these capabilities. Instead, each group identifies

Table 1: Static benchmark results on the ten base task families under clean webpages and default task configurations. Family abbreviations follow Appendix A.1. Entries report instance-level success rates (%). **Bold** indicates the best performance, and underline indicates the second best.

Model	Text	SA	ImS	JA	LA	CS	MP	IcS	TR	BR	Avg.
Leading Frontier Models											
GPT-5.4	**100.00**	68.50	43.50	96.25	73.75	96.25	96.75	25.25	36.50	61.50	69.83
Gemini-3.1-Pro	**100.00**	92.25	85.75	76.50	67.50	72.75	73.25	28.25	53.75	**86.50**	73.65
Claude-Sonnet-4.6	**100.00**	48.25	23.75	36.50	17.50	62.75	15.25	22.25	2.00	27.75	35.60
Claude-Opus-4.6	**100.00**	**98.25**	**96.50**	**97.75**	**96.25**	**98.00**	**98.25**	**55.50**	**96.75**	63.00	**90.03**
Grok-4	97.75	34.25	78.50	87.50	76.25	45.75	52.25	32.50	15.25	62.00	58.20
Other Evaluated Models											
GLM-5V	42.25	75.50	1.50	2.50	37.50	3.50	1.25	4.25	1.25	1.00	17.05
MiniMax-M2.7	82.50	11.50	16.25	47.75	24.25	1.75	18.00	1.00	1.75	2.75	20.75
Qwen-Max	55.75	4.25	0.00	0.00	32.50	3.50	1.50	3.25	1.00	0.50	10.23

the *dominant* bottleneck that a family is designed to expose. Detailed descriptions of all ten base families are provided in Appendix A.1.

3.3 Configuration Dimensions and Dynamic Interaction Validation

Beyond task-family coverage, HLL evaluates agents under three controlled configuration dimensions: intrinsic task difficulty, environmental distraction, and trace-conditioned interaction validation. For an instance $x = (f, d, \ell, z, s)$, we denote its configuration by $\rho(x) = (d, \ell, z)$.

At the level of benchmark composition, each family contributes a set of supported evaluation cells $\mathcal{C}_f$, with $m(f) = |\mathcal{C}_f|$ cells from family f and $M = \sum_{f \in \mathcal{F}} m(f) = 33$ in total. In the main study, these 33 cells comprise ten clean-static cells, ten webpage-distraction cells, five hard-variant cells, and eight dynamic-validation cells; they do not form a full factorial design.

Intrinsic Task Difficulty. For task families that naturally admit hardness control, HLL varies the internal complexity of the challenge while preserving its semantic type. This allows us to examine whether failures already appear under the default configuration or become more pronounced as ambiguity and precision requirements increase.

Environmental Distraction. HLL also varies the amount of irrelevant or deceptive webpage content surrounding the true verification region. This axis captures the gap between clean benchmark interfaces and cluttered webpages, where an agent must first localize the relevant interaction region and avoid being diverted by unrelated controls or decoys.

Dynamic Interaction Validation. Finally, HLL distinguishes answer correctness from acceptance under configured interaction rules. Static success depends on the submitted answer; dynamic success additionally requires the recorded interaction evidence to pass the family-specific validator.

Family-specific validators check properties such as minimum drag displacement, path continuity, target-relative click variation, and legal state transitions. All validation rules and thresholds are fixed before evaluation and reported in Appendix A.4.

These validators impose simulator-observable process constraints, including geometric heuristics, retry-pattern checks, and state-transition checks. Due to the simulator, they are not a complete test of task legality, legal strategies can be rejected, and invalid traces can escape detection (Appendix B.6.5).

3.4 Evaluation Protocol

HLL distinguishes answer correctness, trace-conditioned submission success, and episode success. For instance x, policy π acts until success, three failed verifications, the per-attempt action limit, or the episode timeout. A failed verification can therefore be followed by another submission in the same attempt. Its trajectory is

$$\tau_\pi(x) = (o_0, a_0, o_1, a_1, \ldots, o_T, a_T), \qquad a_t \sim \pi(\cdot \mid o_{\le t}, a_{<t}). \tag{2}$$

Table 2: Results under webpage distraction. Entries report instance-level success rates (%). **Bold** indicates the best performance, and underline indicates the second best.

Model	Text	SA	ImS	JA	LA	CS	MP	IcS	TR	BR	Avg.
Leading Frontier Models											
GPT-5.4	**100.00**	**100.00**	64.25	**100.00**	53.75	**100.00**	**100.00**	22.25	37.75	26.00	70.40
Gemini-3.1-Pro	**100.00**	85.75	**92.25**	46.25	67.75	56.50	58.00	13.50	44.50	**93.50**	65.80
Claude-Sonnet-4.6	**100.00**	28.25	23.75	12.50	15.50	46.25	17.75	0.00	2.50	13.50	26.00
Claude-Opus-4.6	**100.00**	83.75	84.25	81.75	**94.50**	85.75	75.50	**38.25**	**100.00**	48.25	**79.20**
Grok-4	**100.00**	32.50	55.50	68.25	64.50	51.50	38.25	14.00	17.75	45.75	48.80
Other Evaluated Models											
GLM-5V	77.75	0.00	2.25	0.00	75.75	4.50	0.00	1.75	0.00	0.00	16.20
MiniMax-M2.7	74.50	33.50	18.25	56.50	21.75	2.25	0.00	0.00	0.00	1.25	20.80
Qwen-Max	73.75	0.00	0.00	2.25	8.50	1.50	0.00	0.00	0.00	0.00	8.60

where o_t is the simulator observation and a_t is the grounded GUI action. The rollout therefore preserves the end-to-end nature of verification under the designated configuration.

For a trajectory prefix τ ending in a verification call, let $\hat{y}(\tau)$ be its submitted answer, y_x the ground truth, and $V_f(\tau) \in \{0, 1\}$ the dynamic validator. Static submission success is:

$$S_{\text{static}}(x, \tau) = \mathbf{1}[\hat{y}(\tau) = y_x]\,. \tag{3}$$

For a configuration with dynamic-validation flag z, attempt-level success additionally requires the recorded interaction trace to satisfy the corresponding family-specific validator:

$$S(x, \tau) = S_{\text{static}}(x, \tau) \cdot \begin{cases} 1, & z = 0, \\ V_f(\tau), & z = 1. \end{cases} \tag{4}$$

Thus, static evaluation measures whether the submitted answer is correct, whereas dynamic evaluation additionally requires simulator-observable interaction evidence consistent with the configured task process.

An episode allows up to $A = 3$ attempts. Let $\tau^{(j,k)}$ end at evaluated submission k of attempt j, with $K_j \leq 3$ such submissions. Then

$$S_{\text{episode}}(x) = \max_{\substack{1 \leq j \leq A \\ 1 \leq k \leq K_j}} S\left(x, \tau^{(j,k)}\right), \qquad A = 3. \tag{5}$$

The maximum is zero if there is no evaluated submission. Each new attempt resets the page, action budget, and failed-verification counter; the episode time budget is not reset. Every scheduled instance remains in the denominator. Budgets are detailed in Appendix B.1.

4 Experiments

This section uses HLL to study how current multimodal agents behave under interactive verification workloads. Our evaluation is designed to answer the following research questions:

RQ1 How well do frontier agents solve the base static CAPTCHA tasks, and which capability gaps persist even without additional realism stress?

RQ2 How does performance change when controlled realism factors like webpage distraction and increased intrinsic difficulty are introduced?

RQ3 How does acceptance under trace-conditioned validation differ from semantic correctness?

RQ4 What failure patterns recur across task families, and how do they relate to the capability dimensions exposed by the benchmark?

4.1 Experimental Setup

Evaluated agents and harnesses. We evaluate eight multimodal agents spanning proprietary and other publicly documented model families: OpenAI's GPT-5.4 (OpenAI, 2026), Google's

Gemini-3.1-Pro (Google DeepMind, 2026), Anthropic's Claude-Sonnet-4.6 (Anthropic, 2026) and Claude-Opus-4.6 (Anthropic, 2025), xAI's Grok-4 (xAI, 2025), GLM-5V (Hong et al., 2026), MiniMax-M2.7 (MiniMax, 2026), and Qwen-Max (Bai et al., 2023). All are run as closed-loop GUI agents through the same Mobile-Agent-v2 (Wang et al., 2024) harness: at each step the model receives the complete rendered page, emits one grounded action, receives an updated observation, and continues until submission, budget exhaustion, or timeout. We additionally ran AppAgent (Zhang et al., 2025a) for three models and UI-TARS-1.5-7B (Qin et al., 2025) under its own native harness on the dynamic-supported families, holding instances, seeds, and simulator fixed while allowing harness-specific prompts and action interfaces to differ. Model IDs, device and API settings, action space, and budgets are given in Appendix B.1.

Protocol and metrics. Each setting cell contains 400 generated instances shared across models, with model-independent seeds derived from (task family, sample index) under base seed 1234; static and dynamic conditions share the generated content but are executed as separate rollouts. Experiments cover three settings: *base static* tasks on clean webpages, *realism stress* from webpage distraction or higher intrinsic difficulty, and *dynamic* variants additionally requiring trace-conditioned validation with condition-specific budgets (Table 6), which separate failures arising from base task understanding, realism stress, and interaction-grounded validation. The primary metric is instance-level success rate as defined in Section 3.4: semantic correctness of the submitted answer in the static and realism-stress settings, and semantic correctness plus all applicable dynamic checks in the dynamic setting. Extended analyses and case studies are provided in Appendix B.

4.2 Validity Checks on the Dynamic Validators

Two control experiments initially calibrate the validators. Three research-team participants completed 240 trials (30 per dynamic-supported family): all answers were correct and 239 passed dynamic validation (99.58%; Wilson 95% CI: 97.68–99.93%). This is a small calibration, not a representative human baseline. In a separate evidence control, A1 used legal GUI scripts and A2 preserved the answer but removed or corrupted required records; all 30 A2 traces per family were rejected (Table 7). Supplementary strategy checks found legal traces rejected in small-step slider dragging (1/20) and center-click icon selection (3/20), while six other strategies had no rejection (20 each). Additional controls detected 40/40 missing-event traces but only 9/10 illegal tile-state transitions (Appendix B.6.5). These rules measure configured process compliance, with observed false rejections and missed violations.

Table 3: Results on hard variants. **Bold**: best, underline: second best.

Model	SA	JA	CS	MP	TR	Avg.
Leading Frontier Models						
GPT-5.4	40.25	20.25	24.75	**79.75**	20.00	37.00
Gemini-3.1-Pro	5.50	20.25	34.75	45.00	20.00	25.10
Claude-Sonnet-4.6	2.50	10.25	9.75	10.00	2.75	7.05
Claude-Opus-4.6	**65.25**	**74.75**	**71.75**	61.00	**43.25**	**63.20**
Grok-4	10.25	14.75	20.00	1.50	4.50	10.20
Other Evaluated Models						
GLM-5V	21.00	1.75	3.00	1.50	1.75	5.80
MiniMax-M2.7	19.00	23.00	2.25	11.75	0.50	11.30
Qwen-Max	4.25	2.00	0.25	0.50	1.00	1.60

4.3 Static Results on Base Tasks

We begin with the base static setting, where agents are evaluated on the ten base task families under clean webpages and default task configurations. Table 1 summarizes the corresponding results. Even in this simplified setting, interactive CAPTCHA solving remains uneven. Claude-Opus-4.6 obtains the highest average success rate but does not saturate the benchmark, retaining errors on ordered icon selection and board reconfiguration, which indicates that clean-page static success is already bottlenecked by perception–action coordination rather than by visual recognition alone. Difficulty is also highly uneven across families: text transcription is largely solved while ordered icon selection and tile restoration remain hard—and the leading agents show complementary rather than uniform strengths, so static CAPTCHA solving behaves as a composition of partially separable skills rather than a single capability. Per-family and per-model breakdowns are analysed in Appendix B.3.

Static failure patterns. The static traces show that failures are not reducible to visual recognition errors. Instead, they arise along the full interaction chain: weaker models misread distorted text or category cues, middle-performing models often identify the correct target but click or drag at imprecise coordinates, and stronger models mainly fail on ordered selection or stateful restoration

Table 4: Dynamic validation results. The **Static Avg.** column (blue) is the arithmetic mean of the eight dynamic-supported task families in Table 1, excluding Text and Logic-Arith., rounded to one decimal place. Entries report instance-level success rates (%). **Bold**: best, underline: second best.

Model	SA	JA	TR	BR	IcS	ImS	CS	MP	Static Avg.	Dyn. Avg.
Leading Frontier Models										
GPT-5.4	0.50	18.50	**50.25**	18.75	11.25	38.50	**51.50**	21.25	65.6	26.3
Gemini-3.1-Pro	**48.25**	12.50	49.75	**90.25**	19.75	**50.50**	38.75	**50.25**	71.1	**45.0**
Claude-Sonnet-4.6	0.00	0.50	0.00	1.00	0.00	0.00	0.75	43.00	29.8	5.7
Claude-Opus-4.6	10.00	**29.75**	0.00	60.25	**20.25**	8.75	21.00	40.00	**88.0**	23.8
Grok-4	0.00	0.75	18.50	48.75	0.00	11.25	11.50	0.00	51.0	11.3
Other Evaluated Models										
GLM-5V	9.00	0.00	0.00	0.50	0.00	0.75	0.00	0.00	11.3	1.3
MiniMax-M2.7	0.00	0.00	0.75	0.00	0.25	0.00	0.00	21.00	12.6	2.8
Qwen-Max	0.00	0.00	0.25	0.00	0.00	17.00	0.00	0.00	1.8	2.2

tasks that require maintaining a structured interface state across multiple actions. Detailed trace-level examples are provided in Appendix B.3.1.

4.4 Robustness under Realism Stress

We next evaluate how agent performance changes when the clean base setting is made more realistic. We consider two complementary stressors introduced in Section 3.3: webpage distraction and increased intrinsic task difficulty.

Webpage distraction changes performance non-uniformly rather than causing a universal drop. Table 2 shows decreases for most models, but GPT-5.4 (+0.6 percentage points) and MiniMax-M2.7 (+0.1 points) are small positive exceptions at the aggregate level. Claude-Opus-4.6 remains the strongest model overall, while weaker models approach zero on many non-text families. The distraction layer adds competing page content without changing intrinsic task difficulty, acceptance tolerances, or the set of behavioural checks, but the observed changes alone do not identify whether recognition or action execution is responsible. Appendix B.4.1 discusses the small positive changes. These comparisons measure the composite webpage condition implemented by HLL, including layout changes and decoys, rather than clutter in isolation.

The bundled hard condition lowers performance for the strongest models. Unlike distraction, which changes page-level context, the hard condition jointly changes family-specific demands such as alignment tolerance, candidate ambiguity, visual contrast, or restoration complexity. Table 3 reports these composite hard variants. Claude-Opus-4.6 remains the most robust, while GPT-5.4 and Gemini-3.1-Pro fall sharply on several alignment and visual-restoration families. These results compare bundled task demands and budgets (Table 6), not the isolated effect of a difficulty parameter. Detailed stress analyses are provided in Appendix B.4 and Appendix B.5.

4.5 Dynamic Interaction Validation

We finally evaluate dynamic variants, where a semantically correct answer must additionally satisfy trace-conditioned validation rules over simulator-observable interaction traces and committed page states. Table 4 reports success rates on the eight task families with dynamic validation support.

Three observations stand out. **First, correct answers need not pass dynamic validation.** In 400 audited episodes, 219 contained a correct submission but only 116 passed jointly. In a focused subset, all 19 first-correct rejections were independently judged task-illegal. Yet legal-strategy controls rejected 1/20 small-step slider traces and 3/20 center-click traces (Appendix B.6.5). In equal-budget runs on 50 paired instances per family, GPT-5.4 succeeded under answer-only versus full validation on 18 versus 1 slider instances and 10 versus 3 tile-restoration instances (Appendix B.6.6).

Second, geometric trace edits often restore acceptance. A repair study in Appendix B.6.7 shows that geometric checks are almost fully satisfiable by post-hoc trace rewriting once their thresholds are known, whereas checks tied to interaction history and committed state are not. The repair study recovered 22/31 geometric cases with default parameters and 30/31 with threshold-aware parameters

fixed before evaluation. Splitting the eight families by governing mechanism in Appendix B.6.7 leaves Gemini-3.1-Pro highest in both groups and widens its margin over GPT-5.4 and Claude-Opus-4.6 on the history/state group, whose repairability was not tested.

Third, static strength does not reliably transfer to configured dynamic acceptance. Claude-Opus-4.6 leads the base static and hard settings but falls below Gemini-3.1-Pro and GPT-5.4 in dynamic average, indicating that stronger answer-generation capability alone does not guarantee successful interaction under additional execution constraints. Gemini-3.1-Pro retains its lead in a leave-one-family-out sensitivity check, whereas the GPT-5.4–Claude-Opus-4.6 ordering changes (Appendix B.6.4). In ten randomly audited Opus dynamic TR episodes in Appendix B.6.5, nine reached a correct answer with an independently judged illegal interaction; one never submitted a correct answer. This audit does not explain all 400 failures. Appendix B.6 provides the full analysis.

Cross-setting failure patterns. Across settings, failures shift with model capability rather than concentrating at a single stage of the interaction pipeline. Failures span perception, grounding, ordered interaction, and state tracking, suggesting that dynamic task execution requires coordinating multiple intermediate decisions rather than only producing a correct final answer. Stronger models reduce some categories of errors but do not eliminate failures caused by inconsistent execution state or accumulated interaction mistakes. As a preliminary intervention, explicit state recording solved 26/50 paired tile-restoration instances versus 19/50 with generic planning under equal budgets, suggesting that maintaining structured execution state can alleviate part of the long-horizon coordination burden. This single-model result is intended as an initial diagnostic rather than a comprehensive method comparison and uses correctness and independently checked task legality as its success criterion, with further details provided in Appendix B.6.6.

Weaker agents frequently fail at perception and target identification, whereas stronger agents more often reach the correct semantic solution but fail in precise grounding, ordered interaction, or stateful execution. This shift suggests that improved recognition does not eliminate downstream execution errors; instead, the dominant bottleneck moves from identifying what should be done toward carrying it out consistently across a sequence of GUI actions. Dynamic validation further exposes cases in which correct answers are not supported by task-consistent interaction histories. These patterns suggest that interactive verification stresses a sequence of partially separable capabilities whose dominant failure modes change with model capability, rather than a single recognition bottleneck.

Supplementary AppAgent and UI-TARS runs. We additionally report end-to-end AppAgent runs for GPT-5.4, Gemini-3.1-Pro, and Claude-Opus-4.6, together with an end-to-end UI-TARS-1.5-7B run under its native harness, using the same instances, seeds, simulator, and Android/Chrome environment as the corresponding Mobile-Agent-v2 runs. Under matched model versions, instances, rules, and budgets, AppAgent achieves 34.34%, 29.44%, and 6.44% for GPT-5.4, Gemini-3.1-Pro, and Claude-Opus-4.6, respectively, reversing the Mobile-Agent-v2 ordering of the first two models. Because harness-specific prompts and action interfaces differ, these rankings characterize complete agent configurations rather than isolated harness effects. UI-TARS-1.5-7B, a GUI-native 7B model rather than a general-purpose multimodal model, reaches 38.00% under its native harness, providing an additional probe of whether GUI-native action training changes the dynamic-validation picture without isolating training, harness, or scale effects. We therefore report these family-level results descriptively in Appendix B.6.2 and do not use them to redefine the benchmark conclusions.

5 Conclusion

This paper introduced HLL, a benchmark for evaluating interactive CAPTCHA-style verification by multimodal GUI agents. By separating semantic correctness from grounded interaction and trace-conditioned validation, HLL exposes failures that final-answer evaluation alone can miss. Across ten task families, current agents show substantial variation in perception, grounding, state tracking, and interaction consistency, while dynamic validation reveals gaps between correct answers and configured acceptance. These gaps can reflect execution errors, validator limitations, or harness effects. Supplementary checks expose both legal-strategy rejections and missed violations. Trace edits recover most tested geometric rejections, but history and state rewrites remain unevaluated. These findings delimit the current rules and motivate evaluating process constraints against independently checked task legality. HLL is intended as a simulator-level diagnostic benchmark and does not estimate commercial CAPTCHA pass rates or model production anti-bot systems.

AI USE STATEMENT.

Large language models served both as systems under evaluation and as writing assistants for language editing. AI assistance was not used for experimental design, benchmark construction, data collection, or result interpretation. The authors retain full responsibility for the accuracy and integrity of this work.

ETHICS STATEMENT

All experiments were conducted in a local, authorized simulator using CAPTCHA instances served by our own infrastructure; no live websites, third-party verification services, or commercial anti-automation systems were accessed. We position HLL as a defensive diagnostic benchmark rather than a tool for bypassing deployed protections, and do not report pass rates on commercial systems. We disclose that some geometric validators can be satisfied by post-hoc trace rewriting once thresholds are known (Appendix B.6.7) to highlight limitations of geometry-only checks. A human calibration study involved three informed team members completing 240 local trials; no identifying, biometric, or device-fingerprint data were collected. Dynamic validation assesses task-consistent interaction evidence rather than operator identity, so passing does not imply human origin and failing does not imply automation.

REPRODUCIBILITY STATEMENT

Task instances are generated deterministically from model-independent seeds with base seed 1234, ensuring identical instances within each evaluation cell. Static and dynamic conditions share generated content but use separate rollouts and condition-specific budgets, while clean and distraction settings follow the same seed schedule. Appendix B.1 documents the environment, interfaces, budgets, model identifiers, and decoding parameters; Appendix A.4 specifies dynamic-validation rules and thresholds. All scheduled instances remain in the denominator, and robustness to family composition is evaluated with a two-stage bootstrap and leave-one-family-out analysis (Appendix B.6.4). Upon acceptance, we will release the task generators, validators, evaluation code, prompts, configurations, per-instance results, and reproduction scripts. Results remain subject to changes in closed-source APIs and characterize models under the specified harness and simulator.

REFERENCES

Mohammed Abuhamad, Ahmed Abusnaina, DaeHun Nyang, and David Mohaisen. Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey. *IEEE Internet of Things Journal*, 8(1):65–84, 2020.

Alejandro Acien, Aythami Morales, Julian Fierrez, Ruben Vera-Rodriguez, and Ivan Bartolome. Becaptcha: Detecting human behavior in smartphone interaction using multiple inbuilt sensors. *arXiv preprint arXiv:2002.00918*, 2020.

Alejandro Acien, Aythami Morales, Julian Fierrez, Ruben Vera-Rodriguez, and Oscar Delgado-Mohatar. Becaptcha: Behavioral bot detection using touchscreen and mobile sensors benchmarked on humidb. *Engineering Applications of Artificial Intelligence*, 98:104058, 2021.

Abdalnaser Muhammad Algwil. A survey on captcha: Origin, applications and classification. *Journal of Basic Sciences*, 36(1):1–37, 2023.

Anthropic. Claude opus 4.6 system card. `https://www-cdn.anthropic.com/14e4fb01875d2a69f646fa5e574dea2b1c0ff7b5.pdf`, 2025.

Anthropic. Claude sonnet 4.6 system card. `https://www-cdn.anthropic.com/bbd8ef16d70b7a1665f14f306ee88b53f686aa75.pdf`, 2026.

Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. Abusing images and sounds for indirect instruction injection in multi-modal llms. *arXiv preprint arXiv:2307.10490*, 2023.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 125–138, 2011.

Kumar Chellapilla, Kevin Larson, Patrice Y Simard, and Mary Czerwinski. Building segmentation based human-friendly human interaction proofs (hips). In *International Workshop on Human Interactive Proofs*, pp. 1–26. Springer, 2005.

Yurun Chen, Xueyu Hu, Keting Yin, Juncheng Li, and Shengyu Zhang. Evaluating the robustness of multimodal agents against active environmental injection attacks. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 11648–11656, 2025.

Pengzhou Cheng, Zheng Wu, Zongru Wu, Tianjie Ju, Aston Zhang, Zhuosheng Zhang, and Gongshen Liu. Os-kairos: Adaptive interaction for mllm-powered gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6701–6725, 2025.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Gelei Deng, Haoran Ou, Yi Liu, Jie Zhang, Tianwei Zhang, and Yang Liu. Oedipus: Llm-enchanced reasoning captcha solver. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 6–20, 2025.

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

Ziqi Ding, Gelei Deng, Yi Liu, Junchen Ding, Jieshan Chen, Yulei Sui, and Yuekang Li. Illusioncaptcha: A captcha based on visual illusion. In *Proceedings of the ACM on Web Conference 2025*, pp. 3683–3691, 2025.

Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security*, 8(1):136–148, 2012.

Yipeng Gao, Haichang Gao, Sainan Luo, Yang Zi, Shudong Zhang, Wenjie Mao, Ping Wang, Yulong Shen, and Jeff Yan. Research on the security of visual reasoning CAPTCHA. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 3291–3308, 2021.

Google DeepMind. Gemini 3.1 pro model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf`, 2026.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Dorjan Hitaj, Briland Hitaj, Sushil Jajodia, and Luigi V Mancini. Capture the bot: Using adversarial examples to improve captcha robustness to bot attacks. *IEEE Intelligent Systems*, 36(5):104–112, 2020.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14281–14290, 2024.

Wenyi Hong, Xiaotao Gu, Ziyang Pan, Zhen Yang, Yuting Wang, Yue Wang, Yuanchang Yue, Yu Wang, Yanling Wang, Yan Wang, et al. Glm-5v-turbo: Toward a native foundation model for multimodal agents. *arXiv preprint arXiv:2604.26752*, 2026.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Simon Khan, Charles Devlen, Michael Manno, and Daqing Hou. Mouse dynamics behavioral biometrics: A survey. *ACM Computing Surveys*, 56(6):1–33, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. Eia: Environmental injection attack on generalist web agents for privacy leakage. *arXiv preprint arXiv:2409.11295*, 2024.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023b.

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

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. *arXiv preprint arXiv:2310.02255*, 2023.

Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. Open captchaworld: A comprehensive web-based platform for testing and benchmarking multimodal llm agents. *arXiv preprint arXiv:2505.24878*, 2025.

Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6924–6934, 2025.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. Caution for the environment: Multimodal llm agents are susceptible to environmental distractions. *arXiv preprint arXiv:2408.02544*, 2024.

MiniMax. Minimax m2.7: Early echoes of self-evolution. `https://www.minimax.io/news/minimax-m27-en`, 2026. Accessed: 2026-04-30.

Zahra Noury and Mahdi Rezaei. Deep-captcha: a deep learning based captcha solver for vulnerability assessment. *arXiv preprint arXiv:2006.08296*, 2020.

OpenAI. Gpt-5.4 thinking system card. `https://openai.com/index/gpt-5-4-thinking-system-card/`, 2026. Accessed: 2026-04-30.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

Andreas Plesner, Tobias Vontobel, and Roger Wattenhofer. Breaking recaptchav2. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1047–1056. IEEE, 2024.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Andrew Searles, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, Gene Tsudik, and Ai Enkoji. An empirical study & evaluation of modern {CAPTCHAs}. In *32nd usenix security symposium (usenix security 23)*, pp. 3081–3097, 2023.

Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang. Adversarial captchas. *IEEE transactions on cybernetics*, 52(7):6095–6108, 2021.

Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. I am robot:(deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 388–403. IEEE, 2016.

Xiwen Teoh, Yun Lin, Siqi Li, Ruofan Liu, Avi Sollomoni, Yaniv Harel, and Jin Song Dong. Are {CAPTCHAs} still bot-hard? generalized visual {CAPTCHA} solving with agentic vision language model. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3747–3766, 2025.

George Thomas, Alex J Chan, Jikun Kang, Wenqi Wu, Filippos Christianos, Fraser Greenlee, Andy Toulis, and Marvin Purtorab. Webgames: Challenging general-purpose web-browsing ai agents. *arXiv preprint arXiv:2502.18356*, 2025.

Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. In *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pp. 294–311. Springer, 2003.

Haipeng Wang, Feng Zheng, Zhuoming Chen, Yi Lu, Jing Gao, and Renjia Wei. A captcha design based on visual reasoning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1967–1971. IEEE, 2018.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*, 2024. URL https://arxiv.org/abs/2406.01014.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Chen Henry Wu, Rishi Shah, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Dissecting adversarial robustness of multimodal lm agents. *arXiv preprint arXiv:2406.12814*, 2024.

Zonglin Wu, Yule Xue, Yaoyao Feng, Xiaolong Wang, and Yiren Song. Mca-bench: A multimodal benchmark for evaluating captcha robustness against vlm-based attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 38039–38047, 2026.

xAI. Grok-4 model card. https://data.x.ai/2025-08-20-grok-4-model-card.pdf, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. Advagent: Controllable blackbox red-teaming on web agents. *arXiv preprint arXiv:2410.17401*, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.

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

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

Ahmad Zairi Zaidi, Chun Yong Chong, Zhe Jin, Rajendran Parthiban, and Ali Safaa Sadiq. Touch-based continuous mobile device authentication: State-of-the-art, challenges and opportunities. *Journal of Network and Computer Applications*, 191:103162, 2021.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–20, 2025a.

Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking vision-language computer agents via pop-ups. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8387–8401, 2025b.

Sheng Zhou, Junbin Xiao, Qingyun Li, Yicong Li, Xun Yang, Dan Guo, Meng Wang, Tat-Seng Chua, and Angela Yao. Egotextvqa: Towards egocentric scene-text aware video question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3363–3373, 2025.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Detailed Benchmark Specification

This appendix provides detailed benchmark information complementary to Section 3. In particular, we document the ten base task families, the construction of distraction levels, the definition of difficulty settings, the scope and mechanism of dynamic interaction validation, and the detailed benchmark statistics used in the experimental study.

A.1 Base Task Families

The semantic backbone of HLL consists of ten base task families. These families define the benchmark itself; realism dimensions such as difficulty, distraction, and dynamic validation are layered on top of them and are not counted as new tasks.

Recognition and Ordered Selection. This group includes *text transcription*, *icon sequence selection*, and *natural-image sequence selection*. These tasks require the agent to first identify the correct targets and then execute the required sequence of actions in order. Their main purpose is to probe visual recognition, instruction grounding, and ordering fidelity.

Spatial Alignment and Local Patch Reasoning. This group includes *slider alignment*, *jigsaw alignment*, and *missing-patch selection*. These tasks stress fine-grained localization, geometric precision, and the ability to transform high-level task understanding into accurate low-level manipulation.

Stateful Puzzle Restoration. This group includes *board reconfiguration* and *tile restoration*. Unlike single-step alignment tasks, these families require the agent to reason over intermediate state changes and progressively move the interface toward a valid target configuration.

Reasoning-Guided Semantic Interaction. This group includes *logic-and-arithmetic interaction* and *category-guided image selection*. These tasks place greater emphasis on structured interpretation, semantic reasoning, and coordinated multi-step interaction before submission.

Per-family Task Details. Result tables abbreviate the families as Text (text transcription), SA (slider alignment), ImS (natural-image sequence selection), JA (jigsaw alignment), LA (logic-and-arithmetic interaction), CS (category-guided image selection), MP (missing-patch selection), IcS (icon sequence selection), TR (tile restoration), and BR (board reconfiguration). For completeness, the ten base families in the benchmark are:

- **Text transcription:** read a short visual code and enter it correctly.
- **Natural-image sequence selection:** click a sequence of targets in the required order over cluttered visual content.
- **Jigsaw alignment:** drag a missing component to its correct geometric position.
- **Slider alignment:** continuously drag a control until a visual gap is correctly aligned.
- **Category-guided image selection:** select all grid images matching a semantic category.
- **Tile restoration:** restore an image by swapping misplaced tiles.
- **Logic-and-arithmetic interaction:** solve arithmetic, symbolic, or interface-mediated reasoning challenges and submit the correct result.
- **Missing-patch selection:** select the correct local patch that completes an image.
- **Board reconfiguration:** manipulate a compact board until a valid five-in-a-row state is formed.
- **Icon sequence selection:** click symbolic targets in the specified sequence.

A.2 Distraction Layer

To simulate more realistic webpage conditions, HLL augments base CAPTCHA tasks with an environment-level distraction layer. This layer is designed to preserve the semantic task while changing the perceptual and interaction context around it.

Level 0: Clean Interface. The CAPTCHA is presented without additional webpage clutter. This setting is used as the default clean benchmark condition.

Level 1: Realistic Webpage Context. The CAPTCHA is embedded in a realistic webpage-style environment with background content and a fixed-size verification dialog. This setting increases the need for localization and contextual grounding while keeping the true interaction region stable.

Level 2: Deceptive Webpage Context. In addition to the background webpage, the environment introduces decoy controls and trap-like interaction elements outside the true CAPTCHA region. This setting is intended to probe whether agents can distinguish relevant verification controls from misleading interface content.

The distraction layer uses four self-constructed HTML themes—shopping, search, social, and email registration—with responsive styling, position/layout changes, and interactive decoys. It is a controlled page-context stressor rather than a complete open-web reproduction: it does not include iframes, cookie banners, ambient popups, third-party advertisements, accessibility overlays, complex scrolling, or production-side bot-detection channels.

A.3 Difficulty Settings

Difficulty is implemented as an intrinsic task-level control and is only enabled for base families whose hardness can be adjusted without changing the semantic nature of the task. In the current benchmark, explicit difficulty control is applied to a subset of families, primarily those involving alignment, candidate ambiguity, or visual confusion.

General Principle. Across families, `easy` reduces ambiguity or precision demand, `hard` increases intrinsic ambiguity or control difficulty, and `normal` serves as the default intermediate setting. However, the concrete implementation differs substantially by family. Rather than treating difficulty as a global scalar, HLL instantiates it through family-specific controls that preserve the original task semantics.

- **Easy.** Easy settings reduce ambiguity or control precision requirements. Typical mechanisms include fewer distractor candidates, simpler layouts, or looser alignment constraints.
- **Normal.** Normal serves as the default benchmark setting and is used whenever no family-specific difficulty control is enabled.
- **Hard.** Hard settings increase intrinsic task complexity while preserving task semantics. Depending on the family, this may involve higher distractor similarity, more candidate options, stricter spatial tolerance, or more ambiguous visual restoration targets.

Families with Explicit Difficulty Control. In the current benchmark, explicit difficulty control is used for *slider alignment*, *jigsaw alignment*, *missing-patch selection*, *tile restoration*, and *category-guided image selection*. Other task families are evaluated only under the default `normal` setting.

Slider Alignment. For *slider alignment*, difficulty is controlled by both geometric precision and manipulation ease. In `easy`, the slider handle is enlarged and the positional tolerance is relaxed, which makes the target easier to align. In `hard`, the slider handle is reduced and the acceptance tolerance is tightened, increasing the precision required for success. In the reported implementation, the hard slider reduces the handle size from 40 to 36 pixels and tightens the positional tolerance from 10 to 8 pixels. These changes are intentionally bundled within the hard condition and are not interpreted as an isolated causal manipulation.

Jigsaw Alignment. For *jigsaw alignment*, difficulty is controlled through a combination of placement ambiguity and tolerance. The benchmark estimates the local visual contrast around candidate gap regions and uses this signal to choose where the missing piece is placed. In `easy`, the target gap is sampled from higher-contrast regions and the effective tolerance is relaxed; in `hard`, it is sampled from lower-contrast regions and the tolerance is tightened. As a result, harder instances place the missing piece in visually less distinctive regions while simultaneously demanding more precise alignment.

Missing-Patch Selection. For *missing-patch selection*, difficulty is controlled by candidate ambiguity. In `easy`, the number of candidate patches is reduced. In `hard`, the number of candidates is increased and distractor candidates are additionally blurred, making local patch discrimination less reliable from coarse texture alone. `Normal` uses the default candidate count without the extra hard-mode distractor treatment.

Tile Restoration. For *tile restoration*, difficulty is controlled by the number of swaps needed to restore the image. In the current static implementation, `hard` increases the scramble from one swapped pair to two swapped pairs, which expands the search space of possible corrections.

Category-Guided Image Selection. For *category-guided image selection*, difficulty is controlled jointly by grid size, the number of correct targets, and the visual specificity of the image query. In `easy`, the task uses a smaller grid and exactly one correct image, and target retrieval favors close-up, visually salient examples. In `hard`, the grid becomes larger, the number of correct targets increases, and image retrieval shifts toward more scene-level or distant examples, which makes the target category visually less canonical. This design increases both perceptual ambiguity and the combinatorial burden of selecting all valid tiles.

A.4 Dynamic Interaction Validation

Dynamic validation is implemented as a task-aware interaction-validation layer on top of semantic correctness. In dynamic settings, an agent must produce the correct answer and provide sufficient task-relevant interaction evidence through a structured submission payload containing the final answer and the recorded telemetry stream. The validator checks only simulator-observable process constraints; it is not a human-identity classifier.

Dynamic Validation Scope. Dynamic validation is enabled only for task families where correctness alone is insufficient to characterize a valid interaction and where the benchmark can define a stable family-level trace check under the available action interface. In the current study, dynamic support

covers eight base families, while text transcription and logic-and-arithmetic interaction are treated as static-only families. This restriction reflects validator observability and task structure, not a claim that other families cannot have dynamic checks in production.

Mechanism Families. Rather than relying on one generic behavior score, HLL organizes trace-conditioned process checking into several mechanism families according to the form of interaction evidence exposed by each task:

- **Trajectory-continuity checks.**
- **Target-centered spatial-consistency checks.**
- **Repeated-wrong-loop checks.**
- **State-legality checks.**

Compositional Use. These mechanism families are not mutually exclusive. A single task may combine, for example, repeated-loop detection with state-legality constraints, or trajectory continuity with other task-specific safeguards. The grouping below therefore reflects recurring validation patterns in the implementation rather than a strict partition of task families.

Trajectory-Continuity Checks. This family is used for drag-based tasks such as *slider alignment* and *jigsaw alignment*. The benchmark records drag events such as `drag_start`, `drag_move`, and `drag_end`, and verifies whether a coherent drag chain occurred. The current implementation checks whether drag interaction is present, whether the drag exhibits sufficient net displacement, and whether the motion avoids teleport-like jumps. These are coarse, reproducible continuity checks under the Mobile-Agent action interface, not a high-frequency model of human motion.

Target-Centered Spatial-Consistency Checks. This family is used mainly for point-selection tasks such as *natural-image sequence selection* and *icon sequence selection*. This rule checks click-offset variation rather than task legality; repeated center clicks can be legal but rejected (Appendix B.6.5). The benchmark records `target_click` events and compares click positions relative to the true target centers. It then checks whether the resulting offset pattern is implausibly consistent across multiple clicks. This is a simulator-level target-relative consistency rule; it is not a learned distribution of human click behavior.

In addition to offset-consistency checks, these tasks may still require a minimally plausible click trace. However, the distinctive signal of this family is spatial regularity relative to actual targets rather than generic event presence alone.

Repeated-Wrong-Loop Checks. This family captures pathological retry behavior in selection- or control-driven tasks. The benchmark records temporally ordered interaction histories and checks whether the same erroneous operation pattern is repeated multiple times. Such repeated cycles are especially informative in agent evaluation because they reveal failure modes that are not visible from final-answer correctness alone.

Repeated wrong loops are used primarily in selection- or enumeration-style tasks. For *category-guided image selection*, the benchmark records `tile_toggle` and `selection_change` events, checks that real selection activity occurred, verifies that enough grid cells were actually touched, and detects whether the same wrong selection pattern is repeated multiple times. For *missing-patch selection*, the benchmark records `candidate_click` and `candidate_select` events and detects repeated wrong candidate patterns.

State-Legality Checks. This family is used in board- and puzzle-like tasks where the intermediate state itself carries meaning. For *tile restoration*, the benchmark records events such as `tile_drag_start`, `tile_drag_enter`, `tile_drop`, `tile_tap`, `swap_commit`, and `initial_order`. It checks whether each committed swap changed only a legal number of positions. For *board reconfiguration*, the benchmark records `cell_click`, `piece_drag_start`, `piece_drop`, and `move_commit`, then checks whether each state change remained within legal single-step bounds and whether repeated move patterns were detected.

Auxiliary Evaluation Safeguards. Some task families additionally include lightweight integrity checks over submitted payloads and simulator-recorded terminal states. We treat these as auxiliary benchmark-side safeguards for agent evaluation and failure analysis, rather than as primary dynamic detection mechanisms, because they help diagnose trace/payload inconsistencies but do not correspond to realistic production-side defenses.

These safeguards appear in several forms. For *missing-patch selection*, the submitted answer can be compared against the last recorded candidate selection. For *tile restoration* and *board reconfiguration*, auxiliary checks may compare the final submission against the last recorded swap or move state. We retain these checks because they are useful for agent-side diagnosis and failure attribution, but we do not count them as part of the core dynamic mechanism taxonomy.

Representative Dynamic Signals. The benchmark records family-specific telemetry rather than relying on a single generic trajectory metric. Across the mechanism families above, representative signals include:

- whether a coherent drag chain with sufficient displacement was recorded,
- whether click offsets are implausibly consistent relative to target centers,
- whether repeated wrong loops appear in selection or control history,
- and whether state changes exceed legal single-step bounds.

Consistency Checks. These interaction signals are converted into a small set of validation checks that are shared conceptually across task families, even though their concrete thresholds and rules are task-specific:

- **Evidence-presence checks:** whether the submission contains the minimum telemetry required for validation.
- **Trajectory-continuity checks:** whether drag-based behavior exhibits real movement and avoids teleport-like jumps.
- **Spatial-consistency checks:** whether target-centered click offsets are implausibly regular.
- **Loop checks:** whether the same wrong interaction pattern is repeated multiple times.
- **State-legality checks:** whether intermediate state transitions remain within legal single-step bounds.

Dynamic Failure Taxonomy. At a high level, dynamic validation can reject an attempt for at least five reasons:

- **Missing interaction evidence:** the submission lacks sufficient telemetry for dynamic verification.
- **Trajectory-continuity violation:** drag-based behavior lacks coherent movement evidence or contains teleport-like jumps.
- **Target-centered spatial anomaly:** click behavior is implausibly regular relative to target-centered offsets.
- **Repeated wrong loop:** the same erroneous interaction pattern is repeated multiple times.
- **Illegal state transition:** board or puzzle updates exceed legal single-step changes.

Supported Families and Primary Mechanisms. For transparency, we group dynamic-support families by their dominant recurring dynamic mechanisms:

- **Trajectory continuity:** slider alignment and jigsaw alignment.
- **Target-centered spatial consistency:** natural-image sequence selection and icon sequence selection.
- **Repeated wrong loops:** category-guided image selection and missing-patch selection.
- **State legality:** tile restoration and board reconfiguration.

Per-Family Dynamic Validation Rules. Beyond the abstract mechanism categories above, each dynamic CAPTCHA variant instantiates a concrete set of validation rules. We summarize the effective validation logic of each family below, focusing on what is checked rather than on page-side implementation details.

Implementation thresholds. The current implementation uses the following numerical checks for the principal dynamic mechanisms. Slider alignment requires at least 35 pixels of recorded displacement and constrains the maximum consecutive jump to $\max(110 \text{ px}, 0.88 \times \text{path length})$. Jigsaw alignment requires at least 32 pixels of displacement and constrains the maximum jump to $\max(125 \text{ px}, 0.90 \times \text{path length})$. For ordered-click tasks with at least three targets, the validator rejects traces whose joint variance of target-relative x/y offsets is below 0.005. Here, path length means the cumulative Euclidean distance between consecutive recorded touch points in chronological order, summed within each gesture; it is distinct from net displacement. These thresholds are fixed simulator parameters and should be understood as reproducible task-consistency proxies rather than universal behavioral criteria.

Mobile-Agent-v2 exposes Tap and endpoint-based Swipe actions rather than a high-frequency finger trajectory with velocity, acceleration, pressure, or pauses. Consequently, a rule-aware agent can satisfy the fixed geometric checks by rewriting the recorded trace without changing the submitted answer. We quantify this directly in Appendix B.6.7: under a white-box threshold assumption, 30 of 31 sampled geometric rejections were recovered by trace editing; rewrites of interaction history and intermediate states were not evaluated. HLL makes this process dimension measurable under the available interface, but does not claim to eliminate such gameability, and the geometric component should be read as a record-level compliance check rather than a behavioral-authenticity test.

- **Slider alignment:** requires a real drag chain with both start and end events, sufficient net displacement, and no teleport-like jump in the recorded trajectory. A trial fails if the slider appears to move too little or if the path is implausibly discontinuous.
- **Jigsaw alignment:** uses the same trajectory-continuity principle as slider alignment. The dynamic check requires a genuine drag trajectory for the puzzle piece, sufficient movement magnitude, and the absence of jump-like discontinuities.
- **Natural-image sequence selection:** requires enough recorded clicks to cover the expected sequence, rejects traces whose click positions collapse to too few distinct locations, and then checks whether click offsets relative to the true target centers are implausibly consistent across multiple selections.
- **Icon sequence selection:** follows the same target-centered spatial-consistency logic as natural-image sequence selection. The dynamic check requires enough icon clicks, sufficient positional diversity, and rejects unnaturally templated relative offsets around icon centers.
- **Category-guided image selection:** requires enough recorded tile toggles to support the submitted selection, checks that the interaction trail touches enough distinct grid cells, and then detects repeated wrong selection loops by testing whether the same erroneous toggle/selection pattern is repeated multiple times.
- **Missing-patch selection:** primarily uses repeated-wrong-selection detection. The benchmark checks whether a real candidate selection was recorded and whether the same wrong candidate pattern is repeatedly retried. In addition, auxiliary benchmark-side safeguards may compare the submitted answer against the final recorded candidate selection.
- **Tile restoration:** requires that a real swap interaction be observed through either a tap-based or drag-based flow, and then checks state legality by verifying that each committed swap changes only a legal number of positions. Auxiliary benchmark-side safeguards may additionally compare the submitted answer against the last recorded swap state.
- **Board reconfiguration:** requires that a real move be observed through click-flow or drag-flow, checks that each committed board update remains within legal single-step bounds, and detects repeated move patterns when the same erroneous sequence recurs. Auxiliary safeguards may additionally compare the submitted answer against the final recorded move state.

Table 5: Benchmark support matrix across task families and realism settings. "Yes" indicates that the corresponding family is evaluated under that setting in the main study.

Task family	Clean static	Webpage distraction	Hard variant	Dynamic validation
Text transcription	Yes	Yes	–	–
Slider alignment	Yes	Yes	Yes	Yes
Natural-image sequence selection	Yes	Yes	–	Yes
Jigsaw alignment	Yes	Yes	Yes	Yes
Logic-and-arithmetic interaction	Yes	Yes	–	–
Category-guided image selection	Yes	Yes	Yes	Yes
Missing-patch selection	Yes	Yes	Yes	Yes
Icon sequence selection	Yes	Yes	–	Yes
Tile restoration	Yes	Yes	Yes	Yes
Board reconfiguration	Yes	Yes	–	Yes
Total supported settings	10	10	5	8

Design Rationale. This organization keeps the dynamic layer analyzable without reducing it to a brittle one-size-fits-all score. It matches the implementation: drag tasks are characterized by trajectory continuity, point-selection tasks by target-centered spatial regularity, some tasks by retry loops, and others by state-legality violations. Auxiliary integrity checks remain available for benchmark-side diagnosis, but the core dynamic layer should be interpreted as task-consistent process checking under a coarse action API. Passing a check does not imply that a trace was generated by a human or that it would be accepted by a production behavioral detector.

A.5 Benchmark Statistics and Support Matrix

Table 5 summarizes the benchmark support matrix used in the main experiments, and Figure 4 summarizes the same benchmark composition across task families and realism-axis settings. The benchmark contains ten base task families. All families support the clean static setting and the webpage-distraction setting. Five families additionally support explicit hard variants, and eight families support dynamic interaction validation. Counting each supported family–setting cell once yields 33 evaluated settings in total: ten clean static settings, ten distraction settings, five hard settings, and eight dynamic settings.

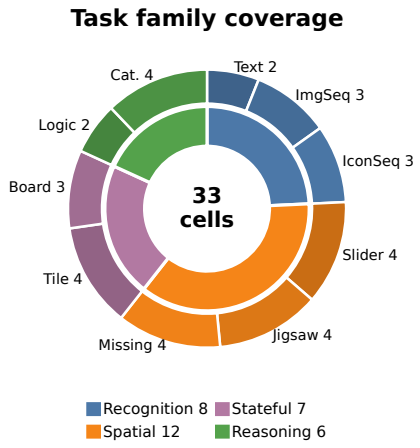


Figure 4: Distribution of benchmark instances across CAPTCHA task families and realism-axis settings.

The support matrix reflects the benchmark design rather than an arbitrary omission of results. Text transcription is retained as a static recognition baseline and is not assigned a dynamic variant in the main definition. Logic-and-arithmetic interaction is evaluated under clean and distracted webpage conditions, but its heterogeneous subtype structure makes a single hard-setting control or family-level dynamic mechanism less directly comparable than the controls used for alignment, selection, patch, restoration, and board-manipulation tasks. Board reconfiguration supports dynamic validation through state-legality checks, but its hard-setting extension is not included in the main study to keep the difficulty-controlled comparison aligned across families with explicit and directly comparable hardness parameters.

B Additional Experimental Details and Full Results

This appendix provides supplementary material for Section 4. It collects experimental configuration details and extended qualitative analyses that clarify the failure modes behind those results. Unless

otherwise specified, all results in this appendix follow the same evaluation protocol defined in Section 4.1.

B.1 Detailed Experimental Setup

This subsection specifies the evaluation configuration used in the main study. Each CAPTCHA instance is generated by a family-specific generator and served through a local HTTP server; no live website or third-party CAPTCHA service is accessed. Text transcription, icon sequence, board, and logic/arithmetic content are primarily generated programmatically. Natural-image sequence, slider, jigsaw, image-grid, missing-patch, and tile-restoration tasks use image pools collected from publicly accessible image databases, after which target sets, perturbations, task states, and ground-truth answers are generated programmatically.

Environment and observation path. We use a Medium Phone Google Play x86_64 Android Virtual Device running Android 16/API 36 at 1080×2400 resolution and 420 dpi with Google Chrome. The local page is opened through Android's VIEW intent. Mobile-Agent-v2 captures the complete screen through ADB; screenshots are neither cropped nor resized, and actions use absolute screenshot-pixel coordinates.

Harness and action interface. All eight models are called through their official provider APIs and operate through the same Mobile-Agent-v2 harness, including the system/task prompt, history format, full-screen observation path, action parser, ADB executor, and local verifier. The model IDs used with the official provider APIs were `gpt-5.4`, `gemini-3.1-pro`, `claude-sonnet-4-6`, `claude-opus-4-6`, `grok-4`, `glm-5v-turbo`, `MiniMax-M2.7`, and `qwen-vl-max`, respectively. The available actions are `Open app`, `Tap`, `Swipe`, `Type`, `Home`, `Enter`, `ScreenShot`, and `Stop`. Tap and Swipe use absolute screenshot pixels, and Swipe duration is fixed to 500 ms. All eight models use the common harness's planning and decision components, with the reflection component disabled. All models use `max_tokens=4096` and temperature 0, except MiniMax-M2.7, which uses temperature 0.01 because its API does not accept 0.

AppAgent supplementary protocol. We additionally evaluate AppAgent (Zhang et al., 2025a) for GPT-5.4, Gemini, and Claude-Opus-4.6. AppAgent and Mobile-Agent-v2 use identical instance IDs, initial pages, seeds, Android AVD, Chrome environment, screen resolution, and local simulator. The API model versions, task and simulator descriptions, validation rules, and action, retry, and time budgets are matched. System prompts and action wrappers are harness-specific. The AppAgent results are descriptive measurements under a second complete agent configuration; they are not intended to isolate a low-level executor or to define a universal harness effect.

UI-TARS supplementary protocol. We further evaluate UI-TARS-1.5-7B (Qin et al., 2025) on the eight dynamic-supported families under its own native harness, prompt format, and point-based action interface, rather than embedding it in the Mobile-Agent-v2 or AppAgent action wrappers, since UI-TARS is trained end-to-end to emit grounded coordinates and actions in its own format. The run uses the same instance IDs, initial pages, seeds, Android AVD, Chrome environment, screen resolution, and local simulator as the main study, and the CAPTCHA task description, simulator description, and dynamic validator are unchanged. UI-TARS is a GUI-native 7B model, in contrast to the general-purpose multimodal models evaluated in the main study; unlike Mobile-Agent-v2 and AppAgent, it does not separate planning and reflection into distinct components, so its results should be read as a single-model, harness-specific data point rather than as isolating scale or training-objective effects from agentic scaffolding. Each family used 400 scheduled instances, covering only the dynamic setting; the static, distraction, and hard-variant settings were not run for UI-TARS. The resulting family-level rates are descriptive measurements and are reported alongside the AppAgent results; they do not redefine the benchmark's main conclusions.

Sampling and condition matching. Each model–family–setting cell contains 400 instances, and all models receive the same instances within a cell. The runner derives model-independent seeds from the task-family and sample-index pair using base seed 1234. Static and dynamic conditions share the generated image, question, answer, and seed but are executed as separate rollouts; dynamic conditions

Table 6: Main evaluation budgets. GUI-action and failed-verification limits apply per attempt; attempt and time limits apply per episode. A new attempt resets the page and the two within-attempt counters, but not elapsed episode time.

Setting	GUI actions per attempt	Attempts per episode	Failed verifications per attempt	Timeout per episode
Clean / distraction	16	3	3	1200 s
Easy / normal / hard	16 / 20 / 24	3	3	1200 / 1600 / 2400 s
Dynamic	24	3	3	2400 s

Table 7: Validator evidence-control experiment. A1 is a normal legal script with a semantically correct answer; A2 preserves the answer but removes or corrupts required trace evidence.

Family	**A0 human**	**A1 legal script**	**A2 corrupted evidence**
Slider alignment	100.00	100.00	0.00
Natural-image sequence	100.00	96.67	0.00
Jigsaw alignment	96.67	96.67	0.00
Category-guided image selection	100.00	96.67	0.00
Missing-patch selection	100.00	93.33	0.00
Icon sequence	100.00	93.33	0.00
Tile restoration	100.00	96.67	0.00
Board reconfiguration	100.00	93.33	0.00

add trace validation. Clean and distraction conditions share the generator and seed schedule. Easy, normal, and hard conditions share the underlying content generation procedure, while hard variants change family-specific item demands.

Scoring and invalid attempts. Every scheduled instance remains in the denominator and permits at most three attempts. A failed verification increments the current attempt's counter; it does not end the attempt until the third failure. An attempt also stops on success, its action limit, or the episode timeout. A new attempt resets the page and action/failure counters while retaining elapsed episode time. Episode success requires at least one successful submission in any attempt. Coordinate execution errors, wrong answers, dynamic rejection, and timeout can therefore cause episode failure. Malformed or unsupported actions, model refusals, and provider API errors are logged as invalid submissions and do not increment the failed-verification counter, but an instance that never succeeds within its action, retry, or time budget is counted as a failure. Static success requires semantic correctness. Dynamic success requires semantic correctness and acceptance by the corresponding family-specific trace-conditioned validator.

Validator evidence-control details. For each of the eight dynamic-supported families, we evaluated 30 paired instances with the same ground-truth answer. The A1 trace used a normal legal GUI script. The A2 trace preserved the answer while deleting the first required `drag_end` event for slider and jigsaw alignment, the first `target_click` for natural-image and icon sequences, the first `tile_toggle` for category-guided selection, or the first `candidate_select` for missing-patch selection. For tile restoration and board reconfiguration, A2 instead changed the first recorded `swap_commit.diff_count` or `move_commit.diff_count` to an illegal value of 3. A1 was run through the GUI and A2 was evaluated by offline replay using the same validator and thresholds. The latter two manipulations test the recorded state-transition fields; they do not claim an independent reconstruction of the complete intermediate board. All A2 paired traces were rejected, while the small number of A1 exceptions are reported as selector failures, missing submissions, or recorded heuristic rejections rather than being removed from the denominator.

Statistical reporting. All success rates use the fixed scheduled-instance denominator. Each instance-level episode is successful if any of its at most three attempts succeeds, so retries are collapsed into one outcome. Aggregates uniformly weight the evaluated families. Fixed-family sampling uncertainty keeps this family set and its weights unchanged; family-resampling analyses instead assess sensitivity to benchmark composition. Appendix B.6.4 reports the latter using a two-stage bootstrap and an exact

Table 8: Family-level human calibration of the dynamic validators. Each family contains 30 trials; the overall row contains 240 trials.

Family	Semantic	Dynamic	Human FRR
Slider alignment	30/30	30/30	0/30
Natural-image sequence	30/30	30/30	0/30
Jigsaw alignment	30/30	29/30	1/30
Category-guided image grid	30/30	30/30	0/30
Missing-patch selection	30/30	30/30	0/30
Icon sequence selection	30/30	30/30	0/30
Tile restoration	30/30	30/30	0/30
Board reconfiguration	30/30	30/30	0/30
Overall	240/240	239/240	1/240 (0.42%)

Table 9: Family-level human completion time and GUI action counts in the calibration study. Time is reported in seconds; action counts follow the recorded GUI action events.

Family	Mean time (s)	Median time (s)	Mean actions	Median actions
Slider alignment	4.1	3.7	1.13	1
Natural-image sequence	7.8	7.1	4.87	5
Jigsaw alignment	9.6	8.7	2.17	2
Category-guided image selection	16.8	15.1	5.27	5.5
Missing-patch selection	10.2	9.2	2.20	2
Icon sequence	9.3	8.4	4.30	4
Tile restoration	18.6	16.9	3.17	3
Board reconfiguration	14.7	13.2	2.97	3

leave-one-family-out check. Difference ranges use an independence approximation; paired inference requires per-instance joint outcomes, and shared instances alone do not make this approximation conservative. Supplementary audits report descriptive counts separately.

Recorded artifacts. For each instance, the pipeline records the final outcome, attempt-level status, simulator observations, action traces, verifier events, and supporting screenshots when available. The released artifact is intended to include prompts, generator parameters, validator thresholds, configuration files, per-instance results, and scripts for reproducing the aggregate tables.

B.2 Human Calibration Protocol

The human calibration used the same Android AVD, local pages, task generators, and dynamic validators as the agent evaluation. Three participants each completed ten instances from each of the eight dynamic-supported families, yielding 30 trials per family and 240 trials overall. All 240 submitted answers were semantically correct, and 239 passed the dynamic validator; the single rejection occurred on a jigsaw-alignment trial. The corresponding semantic and dynamic pass rates were 100% (Wilson 95% CI: 98.42–100%) and 99.58% (97.68–99.93%), respectively. Text transcription and logic-and-arithmetic interaction are static-only and were excluded. This is a small-scale calibration of the current simulator rules, not a representative human baseline, a population estimate, or evidence that the validator identifies human-generated traces.

Human drag-path length and applicability. For trial i containing $G_i \geq 1$ drags, let p_{igk} denote the chronological screenshot-pixel coordinates sampled within drag g. With complete, parseable records of these drags, the trial's cumulative drag-path length is

$$L_i = \sum_{g=1}^{G_i} \sum_{k=1}^{K_{ig}} \|p_{igk} - p_{ig,k-1}\|_2 .$$

Distances are summed within each drag without connecting separate gestures or discrete clicks. This quantity is distinct from start-to-end displacement. It applies to slider and jigsaw alignment, and to

Table 10: Human drag-path statistics from the calibration trials. n_L counts trials with at least one drag and complete parseable records for all drags; click-only families are not applicable.

Family	n_L	Mean (px)	Median (px)
Slider alignment	30	141.3	137
Natural-image sequence	N/A	N/A	N/A
Jigsaw alignment	30	105.7	106
Category-guided image selection	N/A	N/A	N/A
Missing-patch selection	N/A	N/A	N/A
Icon sequence	N/A	N/A	N/A
Tile restoration	30	389.2	408
Board reconfiguration	30	157.3	164

Table 11: Average token usage and elapsed time for the main dynamic evaluation.

Model	Avg. tokens	Avg. time (s)
GPT-5.4	37,369	163.9
Gemini-3.1-Pro	63,973	577.7
Claude-Sonnet-4.6	30,754	202.2
Claude-Opus-4.6	52,721	208.9
Grok-4	76,076	555.8
GLM-5V	42,802	262.8
MiniMax-M2.7	28,285	316.2
Qwen-Max	67,488	621.1

tile restoration or board reconfiguration when participants use drag-based interaction. Click-only trials and point-selection families are marked N/A, rather than assigned zero path length.

For each family f, let $\mathcal{D}_f^{\mathrm{drag}}$ contain trials with at least one drag and complete, parseable records of all their drags. The mean is $\overline{L}_f = |\mathcal{D}_f^{\mathrm{drag}}|^{-1} \sum_{i \in \mathcal{D}_f^{\mathrm{drag}}} L_i$, and the median uses the same set. Its size is reported alongside the summaries, including the interaction-mode breakdown for mixed click/drag families. An empty set yields N/A, rather than a mean of zero. Eligibility does not depend on dynamic acceptance. Missing or incomplete drag logs are reported separately and do not remove trials from other outcome statistics; endpoint-only records support a separate displacement measure and are not pooled with sampled path lengths. No overall path-length average is formed by treating N/A entries as zero.

Dynamic efficiency summaries. The following table reports the available equal-weight averages of family-level token and time measurements for the main Mobile-Agent-v2 dynamic runs. Token counts are provider-reported usage measures and are not treated as directly comparable compute across providers; the time column records the measured seconds under the evaluation runner.

B.3 Full Static Results Analysis

Table 1 reports the complete static results under clean webpages and default task configurations. These results complement the discussion in Section 4.3 by exposing the full cross-model and cross-family structure of performance. Two high-level patterns are immediately visible. First, static performance varies substantially across models: Claude-Opus-4.6 is the strongest model in this setting, while several other agents remain effective only on a subset of families. Second, performance is highly non-uniform across task families, indicating that even without distraction, elevated difficulty, or dynamic validation, CAPTCHA solving still requires a composition of heterogeneous capabilities rather than a single generic perceptual skill.

At the family level, text transcription is clearly the easiest task in the clean static setting, with a mean success rate of 84.8 across evaluated models. Several families involving direct visual localization or single-step recognition are also moderately solvable on average, including jigsaw alignment, slider alignment, and logic-and-arithmetic interaction. In contrast, icon-sequence selection and tile

Table 12: Family-wise summary of static results under clean webpages and default task configurations. "Mean" denotes the average success rate (%) across all evaluated models in the current static sweep. "Best model" reports the strongest model on that family under the same setting.

Family	Mean	Best model
Text transcription	84.8	GPT-5.4 / Gemini-3.1-Pro / Claude-Sonnet-4.6 / Claude-Opus-4.6
Slider alignment	54.1	Claude-Opus-4.6
Image-sequence selection	43.2	Claude-Opus-4.6
Jigsaw alignment	55.6	Claude-Opus-4.6
Logic-arithmetic interaction	53.2	Claude-Opus-4.6
Category-guided image selection	48.0	Claude-Opus-4.6
Missing-patch selection	44.6	Claude-Opus-4.6
Icon-sequence selection	21.5	Claude-Opus-4.6
Tile restoration	26.0	Claude-Opus-4.6
Board reconfiguration	38.1	Gemini-3.1-Pro

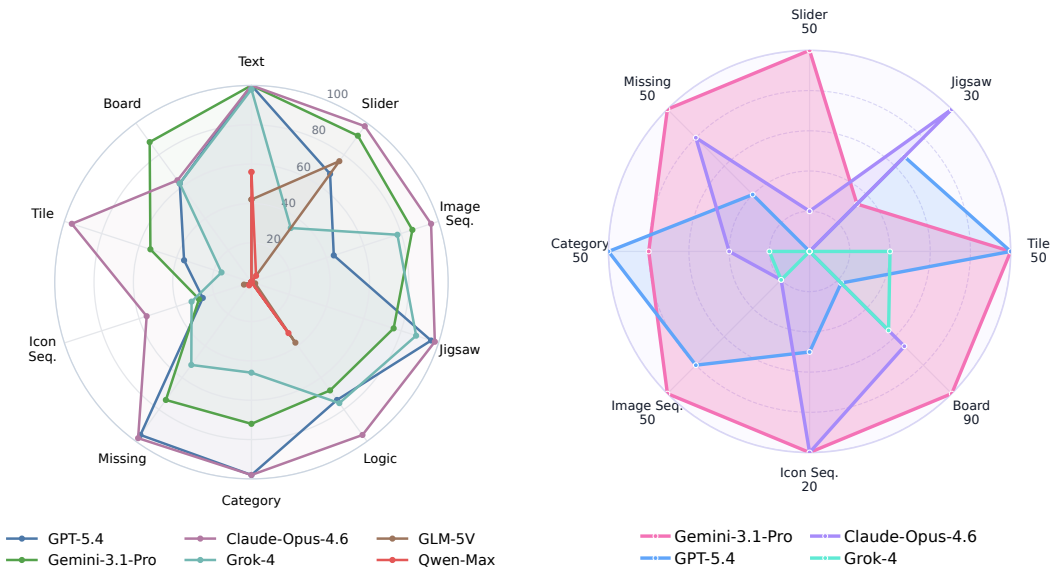
restoration are the least solved families, with mean success rates of 21.5 and 26.0. This ranking is consistent with the intended capability structure of the benchmark: the easier families are those where recognition dominates, whereas the harder families require stronger sequential control, fine-grained spatial grounding, or persistent manipulation of an evolving interface state. Concretely, these harder families demand tighter coupling between recognition and control, since the agent must not only identify the target correctly but also maintain sequential consistency, geometric precision, or state-aware manipulation throughout the interaction; aggregate averages therefore conceal substantial cross-family specialization, with the leading agents showing complementary rather than uniform strengths across recognition, alignment, ordered selection, and stateful tasks.

The model-level profiles are similarly heterogeneous. Claude-Opus-4.6 achieves the strongest overall static performance, reaching near-perfect success on most task families and maintaining a macro-average of 90.0. Its remaining errors are concentrated on ordered icon selection and board reconfiguration, suggesting that even the strongest model is not fully saturated on tasks requiring ordered target interaction or stateful layout reasoning. Gemini-3.1-Pro is the most balanced among the remaining models, with consistently competitive performance across almost all families. GPT-5.4 exhibits a more polarized profile: it is nearly saturated on text transcription, jigsaw alignment, category-guided image selection, and missing-patch selection, but drops sharply on icon-sequence selection and tile restoration. The lower-performing group, including Claude-Sonnet-4.6, GLM-5V, MiniMax-M2.7, and Qwen-Max, further confirms that strong performance on one or two families does not translate into robust end-to-end capability across the benchmark.

On the weak performance of Claude-Sonnet-4.6. Claude-Sonnet-4.6 is an instructive case because it is a frontier-family model that nonetheless scores far below its Opus counterpart (35.6 versus 90.0 on the static macro-average, and 5.7 versus 23.8 on the dynamic one). In the inspected traces, the model generally parses the task correctly and emits well-formed actions, but its predicted click and swipe coordinates deviate substantially from the intended targets. This observation does not isolate model capability from prompt, coordinate-transformation, or executor effects. The failure is therefore concentrated in visual grounding—mapping a correctly identified target to an accurate screen position—rather than in action formatting, instruction following, or API-level errors, which would instead surface as invalid submissions. This also explains why the model retains non-trivial success on missing-patch selection, where the candidate regions are large and coordinate tolerance is correspondingly loose, while collapsing on families that require precise alignment or ordered clicking on smaller targets.

B.3.1 Static Failure Cases

We further inspect representative failed traces from the clean static setting. The taxonomy below organizes the observed errors into eight capability-aligned categories. Each category isolates a distinct bottleneck in the static setting, from low-level perception to higher-level state management and recovery.



(a) Static performance. (b) Dynamic performance.

Figure 5: Family-level performance across the CAPTCHA task families under the static setting (left, complementing Table 1) and the dynamic setting (right, complementing Table 4).

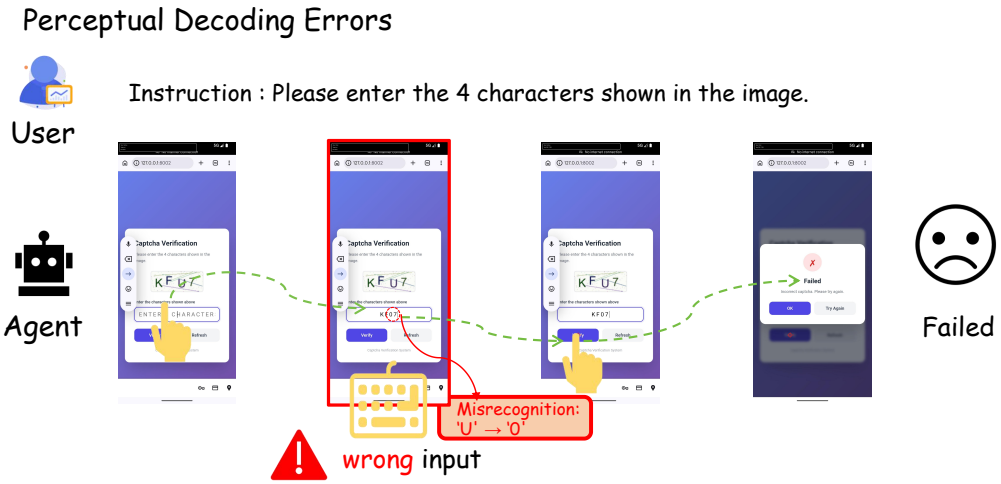


Figure 6: Representative static failure case for perceptual decoding errors.

Perceptual decoding errors. These failures correspond to weak fine-grained visual recognition and unstable OCR. The agent misreads distorted text, confuses visually similar characters, or fails to distinguish subtle local image details before any complex interaction is required.

Target localization and candidate filtering errors. These failures correspond to inaccurate visual localization and insufficient robustness in target screening. The agent may understand the requested target category or sequence but fail to identify the correct candidate, miss valid targets, or over-select visually similar distractors.

Spatial grounding and action mapping errors. These failures correspond to weak visual-to-action spatial mapping. The agent often has the right semantic target in mind, but maps that target to an

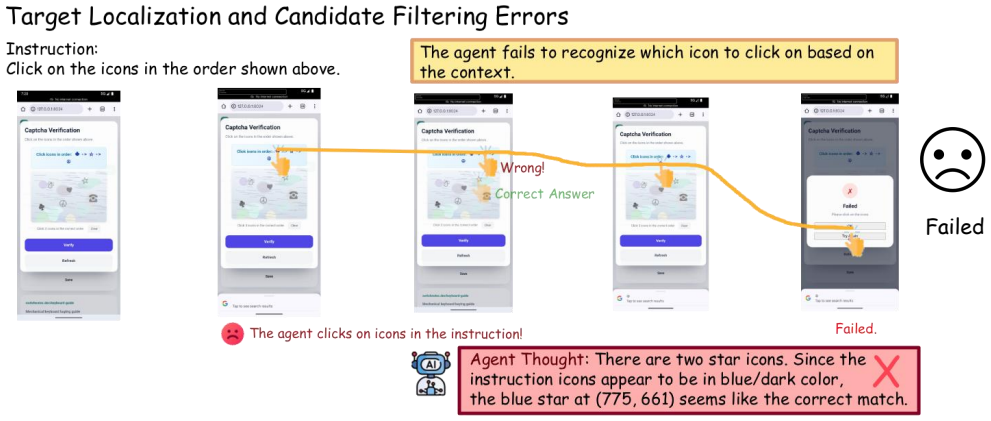


Figure 7: Representative static failure case for target localization and candidate filtering errors.

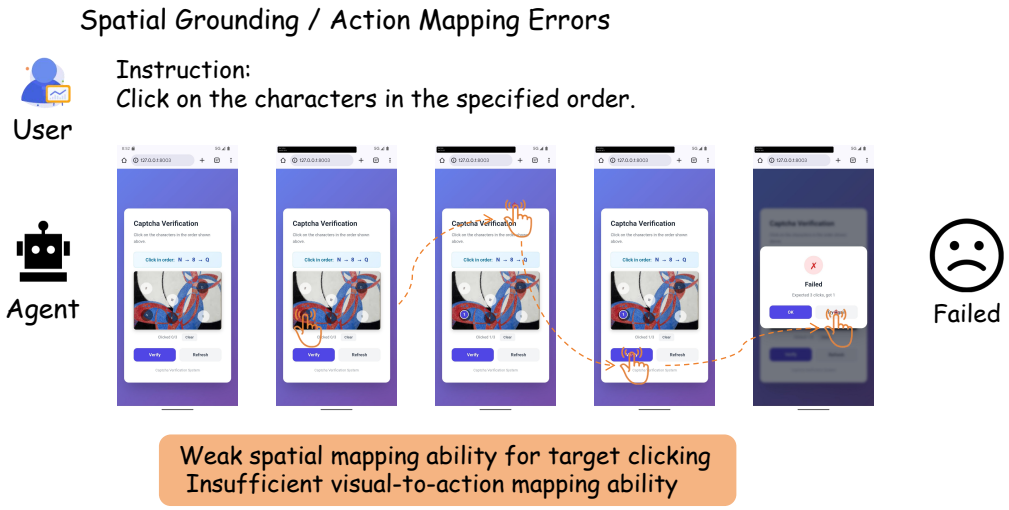


Figure 8: Representative static failure case for spatial grounding and action mapping errors.

incorrect click or drag coordinate, producing the common "recognize correctly but act at the wrong location" pattern.

Geometric calibration failures. These failures correspond to insufficient precision in geometric estimation and fine-grained control. Slider and jigsaw-alignment tasks expose this bottleneck most directly: agents often understand the intended operation, but choose an inaccurate drag distance, stop short of the target, overshoot it, or repeat a poorly calibrated movement.

Visual structure reconstruction failures. These failures correspond to weak visual structure understanding and limited global-consistency reasoning. Tile restoration, board reconfiguration, missing-patch selection, and puzzle-like tasks require the agent to infer an underlying visual or spatial structure; failed traces often show uncertainty about which element is misplaced or how local changes affect the global configuration.

UI affordance and interactive-region misunderstanding. These failures correspond to insufficient UI-structure understanding and weak recognition of interactive regions. The agent may confuse the CAPTCHA region with surrounding controls, misidentify input fields or buttons, or fail to determine which visible elements are actually actionable.

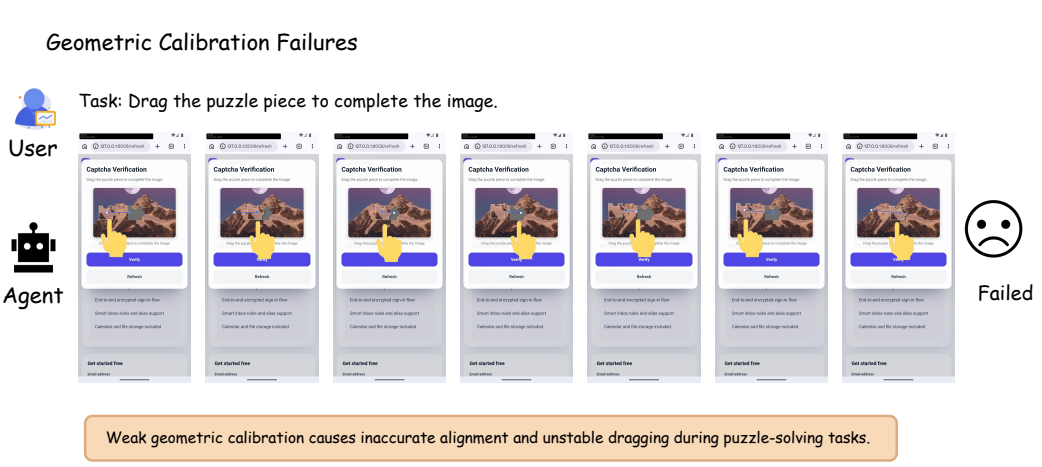


Figure 9: Representative static failure case for geometric calibration failures.

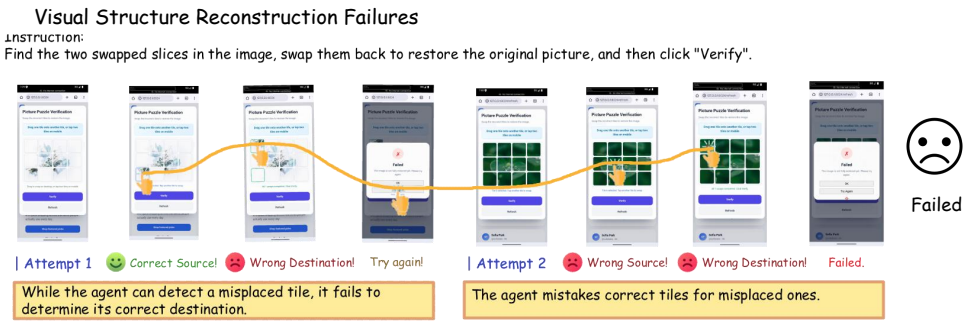


Figure 10: Representative static failure case for visual structure reconstruction failures.

State tracking and completion-judgment failures. These failures correspond to weak state management, context isolation, and task-completion judgment. In multi-step tasks, the agent may lose track of which targets have already been selected, repeat completed steps, submit too early, or continue interacting after the task state has already changed.

Recovery and self-correction failures. These failures correspond to weak correction, abnormal-interface handling, and strategy adaptation after feedback. Across several families, agents recognize that an attempt has failed but enter low-value retry loops, refresh repeatedly, resubmit stale answers, repeat old coordinates, or emit malformed actions instead of adapting their next step.

B.4 Additional Failure Modes under Distraction

B.4.1 Distraction Result Interpretation

The complete distraction results are reported in Table 2. Relative to the clean static setting, webpage distraction does not introduce a new CAPTCHA operation. Instead, it adds page-level context that makes the agent first identify the true verification region and suppress irrelevant surrounding content. We therefore focus only on failure modes that are newly exposed by the distracted webpage setting, rather than repeating the static taxonomy.

Why distraction does not uniformly reduce success. The aggregate effect of distraction is non-monotonic: it lowers success for most models but leaves GPT-5.4 and MiniMax-M2.7 marginally

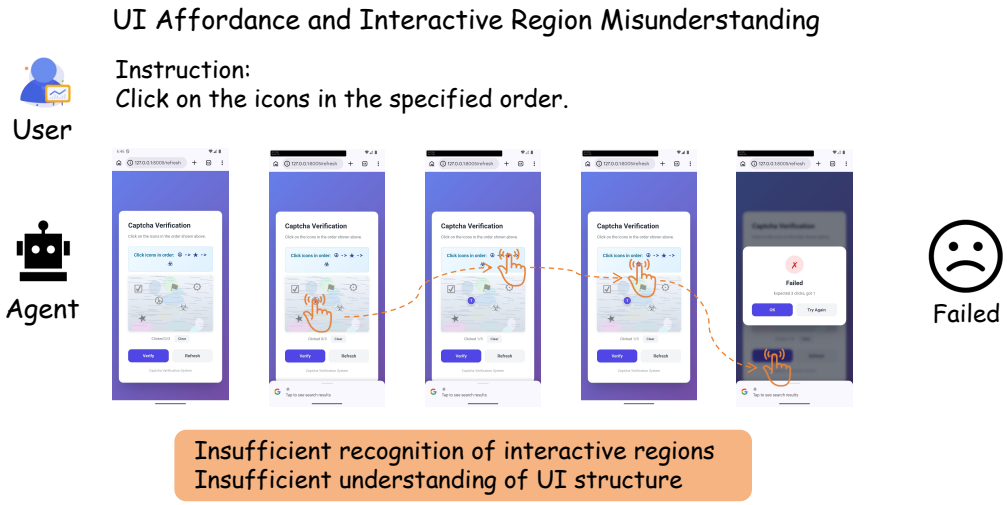


Figure 11: Representative static failure case for UI affordance and interactive-region misunderstanding.

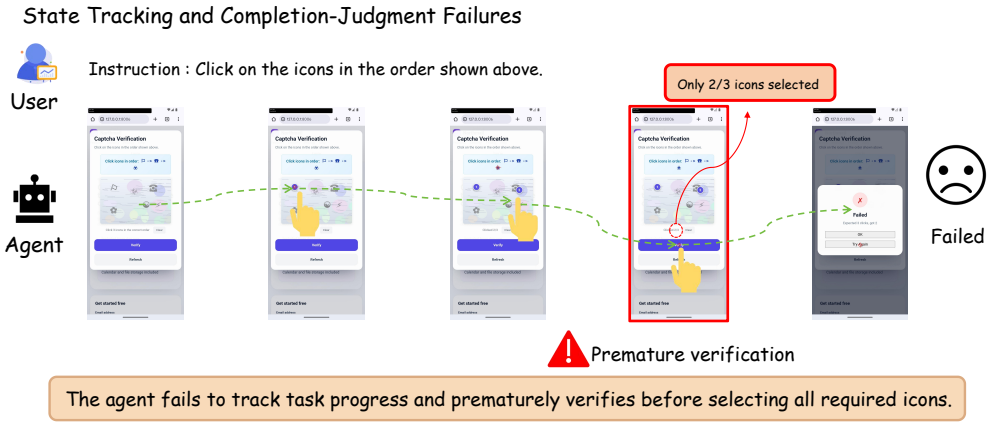


Figure 12: Representative static failure case for state tracking and completion-judgment failures.

higher, and several strong-model cells reach ceiling under distraction that did not under the clean condition. This follows from what the distraction layer does and does not change. It introduces surrounding content that competes for attention during recognition, but it does not alter the intrinsic difficulty of the underlying CAPTCHA, does not change acceptance tolerances, and does not enable any additional behavioural check. These aggregate results do not identify a dominant causal bottleneck, and changes in recognition, localization, and action execution may all contribute. One plausible mechanism for the small positive changes, which we note as a hypothesis rather than a verified cause, is that the surrounding page content gives the model more visual context to register against, yielding slightly better coordinate mapping onto the verification widget. We did not design an experiment to isolate this effect, and the distraction condition should therefore be read as a composite page-context stressor whose aggregate direction is model-dependent.

B.4.2 New Distraction Failure Modes

Task-region localization and distraction suppression failures. Unlike the clean setting, the distracted setting requires the agent to separate the active CAPTCHA card from ordinary webpage content, background text, decorative elements, and decoy controls. Failed traces often show that the

agent either acts on surrounding page elements or spends excessive steps re-reading the page without committing to the true verification region. This failure is distinct from static target localization because the ambiguity occurs at the page level before the CAPTCHA content itself is solved.

Layout-shift spatial generalization failures. Distraction also changes the position, scale, and surrounding context of the verification widget. Some agents appear to rely on coordinate priors learned from clean layouts, repeatedly clicking familiar relative positions even when the target has shifted or the card is embedded differently. This newly exposes weak spatial generalization across page layouts: the agent may still know the CAPTCHA task type, but fail to remap its actions after the interface is placed in a more realistic webpage context.

B.5 Additional Failure Modes under Hard Variants

B.5.1 Difficulty Result Interpretation

The complete hard-setting results are reported in Table 3. Hard variants mostly amplify the static failures described above: perception errors become more frequent, action calibration has less room for error, and state reconstruction becomes harder. The additional analysis below therefore focuses on the failure modes that become newly salient only when the task is made intrinsically harder.

B.5.2 New or Amplified Hard-Setting Failure Modes

High-similarity discrimination failures. Hard image-selection and missing-patch tasks increase candidate similarity or visual ambiguity. This exposes a sharper form of fine-grained discrimination failure: agents can often identify obvious positives, but fail on borderline candidates, visually similar distractors, or local patches that require comparing subtle texture and shape consistency.

Strict-tolerance calibration failures. Hard slider and jigsaw-alignment tasks reduce the acceptable error tolerance. As a result, actions that are semantically correct but slightly imprecise no longer pass. The newly salient issue is not merely that the agent maps a target to the wrong coordinate, but that it cannot reliably produce sufficiently precise movements when the success region becomes narrow.

Complex state-search failures. Hard restoration tasks require more complex search over possible intermediate states, such as identifying multiple misplaced tiles instead of a single obvious swap. This setting newly emphasizes the agent's inability to plan and verify a multi-step correction sequence under a larger state space, even when it can describe the overall puzzle structure.

B.6 Dynamic Validation Analysis

B.6.1 Dynamic Result Interpretation

The dynamic results in Table 4 measure acceptance under the configured process rules. Across the eight supported families, Gemini-3.1-Pro obtains 45.0%, GPT-5.4 26.3%, and Claude-Opus-4.6 23.8%. The Gemini-3.1-Pro lead persists after removing any one family, whereas the latter two models exchange order across these omissions (Appendix B.6.4). This differs from the base static ordering, led by Claude-Opus-4.6. The comparison involves separate rollouts and different condition budgets; it does not isolate a validator effect or attribute all additional failures to the model. Legality audits and validator controls in Appendix B.6.5 provide separate diagnostic evidence.

The strongest degradation appears on task families whose success depends on the relation between the final answer and the action trace. Slider alignment and jigsaw alignment require coherent drag evidence, ordered clicking tasks require target-centered spatial consistency, and restoration-style tasks require legal state transitions. In these families, an answer can be visually or semantically plausible while still failing dynamic validation because the observed trace does not support the submitted result. Ordered icon selection, slider alignment, and jigsaw alignment remain the hardest families under dynamic validation, reflecting the need for target-centered clicks or continuous drag traces. Board reconfiguration is a notable exception for Gemini-3.1-Pro and Claude-Opus-4.6, suggesting that some models can maintain valid state transitions in structured board-like tasks even when they fail the trajectory-continuity families.

Table 13: AppAgent dynamic success rates (%) for the three-model supplementary run. The family order matches Table 4; all entries use 400 scheduled instances per family.

Model	SA	JA	TR	BR	IcS	ImS	CS	MP	Avg.
GPT-5.4	0.00	37.50	62.25	39.75	11.25	53.50	17.75	52.75	34.34
Gemini-3.1-Pro	53.50	10.75	3.00	31.75	18.50	45.25	8.00	64.75	29.44
Claude-Opus-4.6	6.00	7.25	1.75	5.00	5.50	18.00	8.00	0.00	6.44

Table 14: UI-TARS-1.5-7B dynamic success rates (%) under its native harness, for the single-model supplementary run. The family order matches Table 4; all entries use 400 scheduled instances per family.

Model	SA	JA	TR	BR	IcS	ImS	CS	MP	Avg.
UI-TARS-1.5-7B	16.00	39.50	15.25	4.00	52.75	97.75	13.00	65.75	38.00

B.6.2 Supplementary AppAgent and UI-TARS Results

Table 13 gives the family-level AppAgent dynamic success rates, and Table 14 gives the family-level UI-TARS-1.5-7B dynamic success rates. The task order in the underlying runs was `slider_real`, `click_icon`, `click_real`, `puzzle_real`, `pic_puzzle`, `image_grid`, `missing_piece`, and `five_line`; these IDs were mapped to the benchmark families shown in the tables. Each entry uses a denominator of 400 scheduled instances. The AppAgent runs reverse the GPT-5.4–Gemini-3.1-Pro ordering observed with Mobile-Agent-v2, demonstrating that the ranking depends on the complete agent configuration. These comparisons do not separately identify prompt, action-wrapper, executor, or logger effects.

Under its native harness, UI-TARS-1.5-7B reaches a 38.00% average, above GPT-5.4 (26.31%), Claude-Opus-4.6 (23.75%), and Grok-4 (11.34%), and below only Gemini-3.1-Pro (45.00%) in this comparison. Its highest scores occur on natural-image sequence selection (97.75%) and missing-patch selection (65.75%), showing that this model–harness configuration can satisfy the corresponding checks on many instances. Its lowest scores occur on slider alignment (16.00%) and board reconfiguration (4.00%): board reconfiguration requires a multi-step legal-move sequence that is not directly exposed by UI-TARS's native point-and-drag action space, and the slider result suggests that its native drag execution does not always satisfy the trajectory-continuity thresholds tuned against the Mobile-Agent-v2 endpoint-based Swipe action, even though the underlying alignment target is plausibly identified. We treat this pattern as suggestive rather than conclusive: because UI-TARS differs from the main-study models in training objective, parameter count, and agentic scaffolding simultaneously, the comparison cannot isolate which factor drives the gap, and the single-model, single-scale scope does not support a general claim that GUI-native training dominates model scale.

The corresponding eight-family equal-weight efficiency summaries are 80,760.75 tokens and 209.45 seconds for GPT-5.4, 25,308.13 tokens and 243.73 seconds for Gemini-3.1-Pro, and 25,804.25 tokens and 249.23 seconds for Claude-Opus-4.6. These are computed from the reported family-level means; the paper does not infer totals, tail latency, or cross-provider computational equivalence from them.

B.6.3 Aggregate Accounting for Matched Conditions

For the eight dynamic-supported families, each model is evaluated on 3,200 scheduled instances (400 per family). The dynamic aggregate counts are: Gemini-3.1-Pro, 1440/3200 (45.00%); GPT-5.4, 842/3200 (26.31%); Claude-Opus-4.6, 760/3200 (23.75%); Grok-4, 363/3200 (11.34%); Claude-Sonnet-4.6, 181/3200 (5.66%); MiniMax-M2.7, 88/3200 (2.75%); Qwen-Max, 69/3200 (2.16%); and GLM-5V, 41/3200 (1.28%). The matched dynamic-minus-static changes over these eight families are -26.2 percentage points for Gemini-3.1-Pro, -39.3 for GPT-5.4, -64.3 for Claude-Opus-4.6, -39.7 for Grok-4, -24.2 for Claude-Sonnet-4.6, -9.8 for MiniMax-M2.7, -10.1 for GLM-5V, and +0.4 for Qwen-Max.

For clean-to-distraction comparisons over all ten families, the aggregate changes are +0.6 points for GPT-5.4, -7.9 for Gemini-3.1-Pro, -9.6 for Claude-Sonnet-4.6, -10.8 for Claude-Opus-4.6, -9.4 for

Table 15: Dynamic averages and family-composition sensitivity (%). Pooled Wilson 95% intervals treat 3,200 outcomes as exchangeable. Bootstrap ranges are the central 95% of two-stage resamples of families and then instances; they assess composition sensitivity, not fixed-family sampling uncertainty. $\mathrm{SD}_{\mathrm{fam}}$ is the between-family standard deviation.

Model	Macro-avg	Pooled Wilson CI	Bootstrap range	$\mathrm{SD}_{\mathrm{fam}}$
Gemini-3.1-Pro	45.00	43.28–46.73	30.41–60.50	23.5
GPT-5.4	26.31	24.82–27.87	14.72–38.53	18.5
Claude-Opus-4.6	23.75	22.31–25.26	11.91–37.00	19.4
Grok-4	11.34	10.29–12.49	2.59–23.47	16.7
Claude-Sonnet-4.6	5.66	4.91– 6.51	0.09–16.69	15.1
MiniMax-M2.7	2.75	2.24– 3.38	0.00– 8.25	7.4
Qwen-Max	2.16	1.71– 2.72	0.00– 6.69	6.0
GLM-5V	1.28	0.95– 1.73	0.00– 3.69	3.1

Table 16: Leave-one-family-out stability of pairwise dynamic differences. Each entry recomputes the macro-average difference after removing one family; a sign-stable range indicates the ordering is not driven by any single family.

Comparison	All eight	Leave-one-out range	Sign stable
Gemini-3.1-Pro − GPT-5.4	$+18.69$	$[+11.14, +23.18]$	Yes
Gemini-3.1-Pro − Claude-Opus-4.6	$+21.25$	$[+17.18, +26.75]$	Yes
GPT-5.4 − Claude-Opus-4.6	$+2.56$	$[-4.25, +8.86]$	No

Grok-4, -0.9 for GLM-5V, +0.1 for MiniMax-M2.7, and -1.6 for Qwen-Max. These differences are computed from the same 400-instance-per-family sampling schedule and should be interpreted as condition-level effects rather than universal monotonic trends.

B.6.4 Family-Composition Sensitivity and Dynamic Ranking Stability

The estimand for the fixed benchmark is the equally weighted mean over its eight specified families. Its instance-sampling uncertainty should keep those families and weights fixed, resampling within each family and preserving shared-instance pairing for model comparisons. Table 15 addresses a different question: sensitivity to family composition. Its two-stage bootstrap uses 20,000 replicates, resampling families with replacement and then instances within each selected family. The wider ranges include changing family weights; their width does not show that a fixed-family interval understates uncertainty. HLL does not sample task families from a defined population, so these ranges also do not establish generalization to unseen families.

Table 16 reports an exact leave-one-family-out check. Gemini-3.1-Pro retains a positive margin after removing any single family: at least 11.14 points over GPT-5.4 and 17.18 over Claude-Opus-4.6. GPT-5.4 versus Claude-Opus-4.6 changes from -4.25 to $+8.86$ points across these omissions. This is a sensitivity check, not a significance test. The independence-approximated family-resampling difference ranges are $[-0.59, +38.13]$, $[+1.59, +40.81]$, and $[-14.78, +19.75]$, respectively; the first and third include zero. Shared instances do not by themselves make this approximation conservative. The corresponding dynamic-minus-static sensitivity ranges are $[-80.41, -46.31]$ for Claude-Opus-4.6, $[-61.28, -16.81]$ for GPT-5.4, and $[-45.78, -5.03]$ for Gemini-3.1-Pro. These describe separately executed configurations with different budgets, not the isolated effect of enabling validation.

B.6.5 Supplementary Audits of Dynamic Runs and Validators

Within-run semantic and acceptance audit. We randomly sampled 50 episodes without replacement from the 400 main-study Mobile-Agent-v2 dynamic episodes in each of four families: slider alignment (SA), icon sequence selection (IcS), tile restoration (TR), and missing-patch selection (MP). GPT-5.4 and Gemini-3.1-Pro use the same sampled instance IDs (seed 20260925), giving 400 audited episodes. The audit uses the same joint-success definition as the main table. For each episode, C indicates at least one correct submission, J at least one submission jointly correct and dynamically accepted, and R rejection of the first correct submission. Earlier wrong submissions do not affect R,

Table 17: Audit of existing dynamic episodes. C: ever correct; J: joint success; R: first correct submission rejected. R and J can overlap.

Model	Family	Episodes	C	J	R
GPT-5.4	SA	50	31	1	30
	IcS	50	11	6	5
	TR	50	21	20	1
	MP	50	43	12	43
Gemini-3.1-Pro	SA	50	40	23	17
	IcS	50	13	9	4
	TR	50	26	23	3
	MP	50	34	22	12
Total		400	219	116	115

and an episode in R can later succeed. These are episode-level counts within the observed dynamic runs, not single-submission labels (c, L, V) or outcomes of a separate static policy.

Of the 115 episodes whose first correct submission was rejected, 12 subsequently passed; 103 episodes contained a correct submission but never jointly passed. These results distinguish semantic correctness from acceptance under configured rules. They do not establish that every rejected interaction was task-illegal or predict the outcome of a separately executed answer-only policy.

Independent legality audit of rejected correct submissions. We pooled the two models' R episodes within each family in Table 17, randomly selecting up to five per family and retaining all when fewer were available (seed 20260920). This gives 19 episodes: five SA, five IcS, four TR, and five MP. We examined the first correct submission and its preceding actions against task rules and available page/action evidence, without equating validator rejection with illegality. All 19 were judged task-illegal; none was labelled legal or indeterminate. No discrepancy between the recorded and observed actions was identified in these cases. An additional TR case outside the Table 17 pool was inspected and also judged illegal; it is reported separately from the 19-case sample because its selection differed. This focused audit concerns first-correct rejections in four families, not all dynamic failures or a population false-rejection rate. Absence of an observed record discrepancy in these cases does not exclude executor or logger effects elsewhere.

Focused audit of Claude-Opus-4.6 on tile restoration. We randomly selected ten of the 400 main-study Mobile-Agent-v2 dynamic TR episodes without filtering by correctness or record completeness (seed 20260920). For episodes with a correct submission, we assessed the first such submission using task rules and the available page and action evidence, independently of validator acceptance. One episode never submitted a correct answer. The other nine had a correct submission with an interaction judged task-illegal: eight were classified as trajectory-related and one as click-related. None was classified as a legal first correct submission or as indeterminate. All ten episodes failed joint acceptance, consistent with the main-table 0/400. These coarse categories describe only the inspected cases; they neither identify a specific failed rule for every case nor separate model decisions from executor and logging effects, and they cannot explain the full 96.75%-to-0% static–dynamic change.

Legal strategies and constructed violations. We tested two scripted legal strategies on the same 20 instances per family, using distinct assets across instances: 80 instances and 160 traces in total. Correctness and legality were checked against task rules independently of dynamic acceptance. Table 18 reports each strategy separately; the paired strategies are not treated as 160 independent instances. The legal traces rejected by the small-step and center-click strategies show that heuristic process constraints can disagree with task legality.

Separately, we constructed 10 missing-required-event traces per family (SA, IcS, TR, MP) and 10 TR traces with illegal state transitions. All were independently checked as violations and evaluated with the original validator. It rejected 10/10 in each missing-event group and 9/10 in the illegal-state group. These 50 constructed traces are additional controls, separate from the 30-per-family A2 experiment

Table 18: Supplementary legal-strategy checks. The two strategies in each family use the same 20 instances.

Family	Legal strategy	Confirmed legal	Rejected
SA	Direct drag to target	20	0
SA	Small-step drags to target	20	1
IcS	Click natural positions within targets	20	0
IcS	Click exact target centers	20	3
TR	Direct legal solution	20	0
TR	Legal error followed by recovery	20	0
MP	Direct correct selection	20	0
MP	One legal wrong attempt, then recovery	20	0

Table 19: Paired online counts. Each row contains 50 paired instances. For validation rows, A is answer-only and B is full validation; for planning, A is generic planning and B is state recording. Criteria differ between the two experiments, so their TR rates should not be compared directly.

Comparison	Family	Both	A only	B only	Neither	A total	B total
Validation	SA	1	17	0	32	18/50	1/50
Validation	TR	3	7	0	40	10/50	3/50
Planning	TR	16	3	10	21	19/50	26/50

in Table 7. The missed state violation and the legal-strategy rejections delimit the tested rules; neither perfect task-legality coverage nor population error rates follow from these small checks.

B.6.6 Small-Scale Paired Online Comparisons

Both comparisons use GPT-5.4 with Mobile-Agent-v2. The sample size of 50 paired instances per tested family was fixed before viewing formal outcomes. Paired conditions share instance IDs, initial states, API version, action interface, and budgets: 24 GUI actions and at most three failed verifications per attempt, up to three attempts including the first, and 2400 seconds per episode. Each new attempt resets the page and the action/failure counters; elapsed episode time is retained. Thus, at most 72 GUI actions are available across three attempts, subject to the shared time limit. Conditions are interleaved, and development runs are excluded. We report descriptive paired counts without claiming a consistent effect across models or families.

Answer-only versus full validation. This comparison uses 50 SA and 50 TR instances, each run under both conditions (200 formal episodes). Prompts are shared and both conditions return only pass/reject feedback. Condition A stops successfully when an answer is correct; condition B requires correctness and acceptance by the original full validator on the same submission. Success is measured using each condition's stopping rule, so this comparison concerns accepted online completion under two criteria, not a common independent measure of task legality.

Generic planning versus explicit state recording. This comparison uses 50 paired TR instances (100 formal episodes). Condition A uses generic planning; condition B explicitly records the current task state using only screenshots and prior actions, without hidden state or answer access. Planning in either condition is limited to 256 tokens in the same response, with a common total output limit of 4096 tokens. Both conditions are scored by the same criterion: a correct answer with independently confirmed legal interaction. This outcome is distinct from acceptance by the dynamic heuristics and from the stopping-rule outcome in the preceding comparison.

Full-validation completion was 2% versus 36% for SA and 6% versus 20% for TR. State recording achieved 52% versus 38% under the separate legality-based criterion, with 10 pairs succeeding only with state recording and three only with generic planning. The latter is a preliminary example of a diagnosis-informed intervention, not evidence of a general or statistically established benefit.

Table 20: Recorded-trace repair outcomes by triggering mechanism ($n = 43$). Default and threshold-aware parameters were fixed before evaluating these cases. Only geometric edits were tested; history and state rewrites were not evaluated.

Mechanism	Type	Count	Default	Threshold-aware
Maximum-jump violation	Geometric	9 (20.9%)	8/9	8/9
Insufficient net displacement	Geometric	16 (37.2%)	11/16	16/16
Spatial-consistency anomaly	Geometric	6 (14.0%)	3/6	6/6
Repeated wrong loop	Semantic	10 (23.3%)	not tested	
Illegal state transition	Semantic	2 (4.7%)	not tested	
Geometric subtotal		31 (72.1%)	22/31 (71.0%)	30/31 (96.8%)
Recovered / sampled cases		43	22/43 (51.2%)	30/43 (69.8%)

B.6.7 Repairability of Dynamic Rejections

The dynamic validator operates on simulator-observable records rather than physical interaction signals. We sampled 43 dynamic episodes with a semantically correct submission rejected by the validator and labelled each by its triggering mechanism. We evaluated three recorded-trace edits using the same validator and thresholds. All edits preserve the submitted answer and final committed position; click edits remain within the true target region. M1 preserves gesture count and cumulative Euclidean path length, whereas M2 changes gesture segmentation and count. Default and threshold-aware parameter settings were fixed before evaluating these 43 cases; the cases were not used to select or tune parameters. Recovery means acceptance of an edited record, not successful online execution of a repaired policy. The small selected sample describes these cases, not the population frequency of each mechanism.

Repair mechanisms. Three of the five rejection mechanisms admit a trace-only repair. *(M1) Trace densification* addresses maximum-jump violations by linearly interpolating intermediate touch points between recorded points, leaving start point, end point, net displacement, and gesture count unchanged; it reduces the maximum consecutive jump while preserving cumulative Euclidean path length, so the path-length-dependent threshold remains unchanged. *(M2) Collinear gesture merging* addresses insufficient-displacement rejections. Inspecting these episodes shows that they arise when an agent reaches the correct final position through several small successive drags, each individually below the displacement threshold. Merging temporally adjacent, approximately collinear gestures into one recorded gesture changes gesture count and segmentation, while preserving the first start point, last end point, and submitted answer. *(M3) In-target jitter* addresses target-centered spatial-consistency rejections by perturbing each recorded click within its target region, which raises the joint variance of target-relative offsets without changing which target was selected. The remaining two mechanisms are excluded by construction: repairing a repeated-wrong-loop rejection would require deleting recorded retry behaviour, and repairing an illegal-state-transition rejection would require fabricating intermediate board states. Neither is a geometric rewrite of the same interaction, so neither is attempted.

Results. Table 20 reports the outcome. Fixed default parameters recovered 22 of 31 geometric rejections (71.0%); the pre-fixed threshold-aware settings recovered 30 of 31 (96.8%). By mechanism, these counts were 8/9 versus 8/9 for M1, 11/16 versus 16/16 for M2, and 3/6 versus 6/6 for M3. The remaining 12 history- or state-related cases were not subjected to corresponding rewrites, so the study does not establish their resistance to answer-preserving editing.

Interpretation. These results show that many sampled geometric rejections can be reversed by editing the recorded trace under known thresholds. Such acceptance does not establish how the trajectory was produced. History and state checks concern different recorded properties, but their resistance to trace rewriting was not evaluated. The findings motivate checking both geometric and state-based rules against independent task legality rather than treating either group as proof of behavioral authenticity.

Table 21: Dynamic success rates (%) split by validation mechanism. The geometric group contains the four families whose checks the repair study finds largely satisfiable by trace rewriting; the semantic group contains four families governed by history or state checks, whose repairability was not tested.

Model	All eight	Geometric (4)	Semantic (4)
Gemini-3.1-Pro	45.00	32.75	57.25
GPT-5.4	26.31	17.19	35.44
Claude-Opus-4.6	23.75	17.19	30.31
Grok-4	11.34	3.00	19.69
Claude-Sonnet-4.6	5.66	0.12	11.19
MiniMax-M2.7	2.75	0.06	5.44
Qwen-Max	2.16	4.25	0.06
GLM-5V	1.28	2.44	0.12

The available interface limits what these checks can observe. The simulator, in common with current mainstream mobile GUI simulators, exposes discrete Tap and endpoint-based Swipe events and does not expose velocity, acceleration, pressure, or inter-event timing at a resolution that would distinguish a generated trajectory from a physically produced one. Any check definable on this record is therefore a statement about the record, not about its origin. We report this explicitly: HLL measures simulator-observable process consistency, and a substantial gap remains between this and behavioral authenticity verification as deployed in production anti-automation systems. Closing that gap requires simulators that expose higher-fidelity input signals, which we regard as a prerequisite for the next generation of interactive verification benchmarks.

Does the ranking depend on the repairable checks? Given the observed geometric repairs, we examine whether the aggregate ordering also appears among families governed by history or state checks. We therefore split the eight dynamic-supported families by the mechanism that governs them: the *geometric* group (slider alignment and jigsaw alignment, governed by trajectory continuity; icon and natural-image sequence selection, governed by target-centered spatial consistency) and the *semantic* group (category-guided image selection and missing-patch selection, governed by repeated-wrong-loop detection; tile restoration and board reconfiguration, governed by state legality). Table 21 reports macro-averages within each group.

Gemini-3.1-Pro leads both groups, while GPT-5.4 and Claude-Opus-4.6 tie on the geometric group. The Gemini-3.1-Pro margin is larger on the semantic group: $+21.8$ versus $+15.6$ points over GPT-5.4, and $+26.9$ versus $+15.6$ points over Claude-Opus-4.6. Thus, the observed Gemini-3.1-Pro lead under Mobile-Agent-v2 is present in both groups; this split compares different task families rather than disabling rules on the same traces. It does not isolate a rule's causal contribution, test repair resistance, or establish a harness-independent ranking. The split also clarifies the comparison between GPT-5.4 and Claude-Opus-4.6: their geometric-group averages are identical at 17.19, and the entire pooled gap of 2.56 points originates in the semantic group. This is consistent with the leave-one-family-out analysis in Appendix B.6.4, which finds their ordering unstable, and supports reporting them as comparable rather than ranked.

B.6.8 Mechanism-Level Dynamic Failures

Dynamic failures can be organized into three additional process-level categories that are distinct from the static failure taxonomy.

Insufficient interaction evidence. The agent may submit or reach a seemingly correct final answer without producing enough task-relevant evidence in the recorded trace. This occurs when required clicks, selections, drags, or state changes are missing or too sparse to support the final answer. Such cases are especially important because they would be counted as plausible successes under a purely static metric, but fail once the benchmark checks whether the answer is backed by observable interaction.

Trajectory-continuity and process-consistency failures. Drag-based tasks such as slider alignment and jigsaw alignment require a coherent motion trace, not just a final coordinate. Dynamic

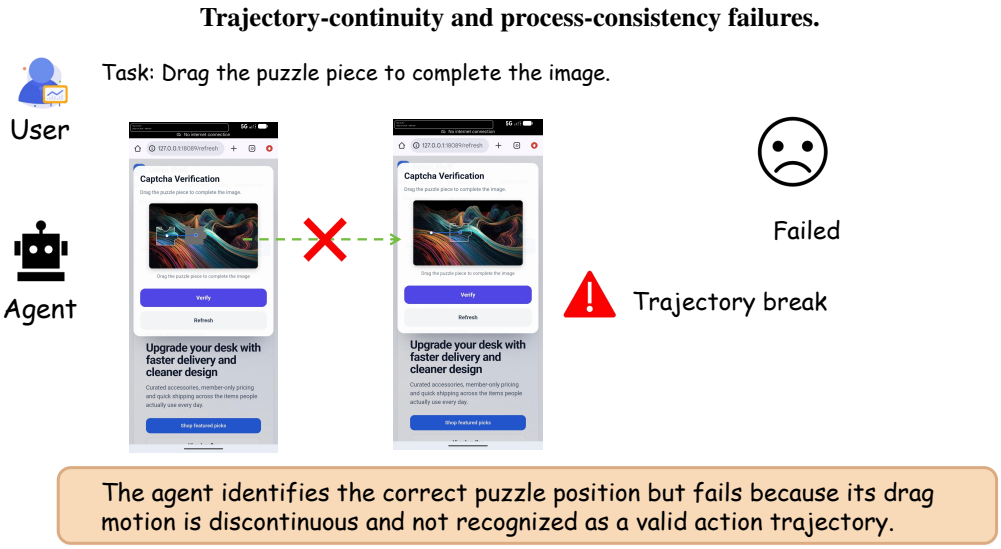


Figure 13: Representative dynamic failure case for trajectory-continuity and process-consistency failures.

validation therefore exposes failures where the recorded action sequence lacks a valid drag start–move–end chain, contains teleport-like jumps, or does not show enough continuous displacement. These rules can reject a correct endpoint, including when legal small-step gestures do not meet a displacement threshold; low acceptance therefore requires interpretation alongside legality checks.

Interaction-history and terminal-state inconsistency. For ordered clicking, board reconfiguration, tile restoration, and selection-based tasks, the submitted answer must be consistent with the sequence of previous actions and the final page state. Dynamic failures arise when the action history contains repeated wrong clicks, illegal transitions, incomplete selections, or a final state that does not match the submitted result. This category captures cases where the model appears to know the answer but cannot maintain a valid process from initial observation to terminal submission.

B.7 Responsible Release and Intended Use

We position HLL as a controlled simulator and evaluation harness for studying interactive verification, not as tooling to attack or bypass arbitrary live third-party verification systems. Release materials are intended to reproduce the benchmark episodes, metrics, and validation rules reported in this paper, together with documentation.