PHOTOARENA: BENCHMARKING PURE-GUI COMPUTER-USE AGENTS IN ADOBE PHOTOSHOP VIA DUAL-TRACK VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Professional creative software offers multiple routes to task completion, including visual GUI interaction and application scripting. When programmatic shortcuts are available, successful task completion alone does not establish an agent's ability to perform the corresponding visual interactions. We therefore focus on a specific question: can computer-use agents complete professional editing workflows using only screenshots, mouse actions, and keyboard inputs? Studying this capability raises two evaluation challenges. First, the interaction protocol needs to distinguish pure GUI performance from task completion through scripts or application APIs. Second, assessing these workflows calls for evidence of both execution and delivery: a final rendered image can conceal missing task-required editable structures, while final artifacts alone do not reveal whether required intermediate operations were performed.

To address these challenges, we introduce PHOTOARENA, a benchmark for professional creative tasks in Adobe Photoshop under a strict **GUI-only interaction protocol**, paired with a **dual-track evaluation framework**. The Outcome Delivery Track checks prompt-required PSD structure and visual completeness before assessing visual consistency, instruction following, and visual quality. The Process Adherence Track aligns execution trajectories with semantic operations using an LLM-based forward anchor and evaluates step-level execution without requiring agent frames to visually match the expert during alignment. We report the two tracks separately to distinguish procedural adherence from final deliverable quality. Evaluated frontier agents execute at least 60% of task-relevant operations, yet end-to-end task success remains at or below 10% under the GUI-only protocol. These findings highlight the difficulty of translating local interaction successes into complete, editable deliverables under the GUI-only protocol.

1 INTRODUCTION

The expansion of foundation models toward direct computer interaction has catalyzed rapid progress in autonomous Computer-Use Agents (CUAs) (Xie et al., 2024; Zhou et al., 2024). Beyond web browsing and basic operating system utilities, deploying agents into high-value professional desktop environments—such as Adobe Photoshop (Adobe Inc., 2024a)—represents an important direction for supporting professional creative workflows. In professional creative software, task completion cannot be reduced to superficial visual generation or isolated discrete commands. Production workflows demand executing extended, multi-step procedures, performing fine-grained visual grounding, coordinating continuous canvas operations, and tracking intermediate application state. Crucially, a qualified deliverable must satisfy both the intended visual outcome and the editable, non-destructive structure explicitly mandated by the task. This introduces a fundamental research question: *Can autonomous agents reliably complete professional editing workflows and deliver compliant creations solely through visual GUI interaction?*

Benchmarking this capability presents two primary challenges. **First, the operating protocol must isolate visual interaction from programmatic automation.** Professional creative suites natively support both visual GUI interaction and programmatic automation (e.g., via ExtendScript or UXP

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

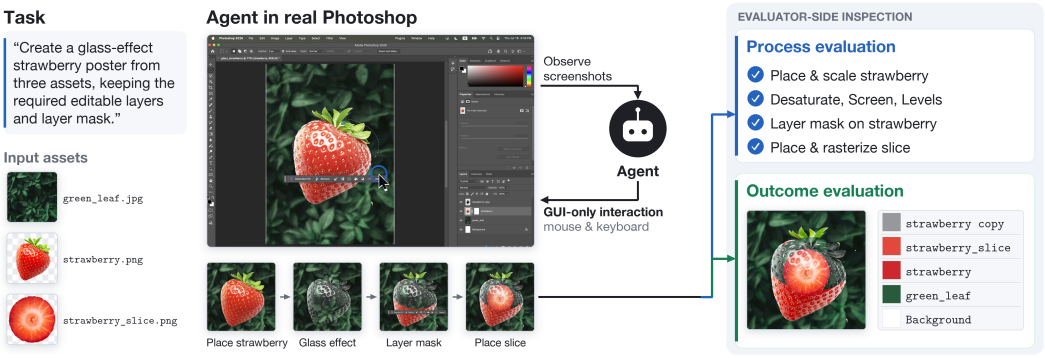


Figure 1: **PhotoArena: GUI-only interaction with dual-track evaluation.** *Left:* a summarized task and input assets for a glass-effect strawberry composition. *Center:* the agent observes Photoshop screenshots and acts through mouse and keyboard inputs, without access to scripts or application APIs; selected editing stages illustrate the workflow. *Right:* evaluator-side inspection separates process evaluation of task-relevant operations from outcome evaluation of the final rendered image and required editable PSD structure. The two tracks assess execution and delivery separately.

scripting APIs) (Adobe Inc., 2024b). However, while scripting assesses code generation, screenshot-based computer use centers on visual perception and manual interface control. To isolate visual GUI capability, the benchmark protocol restricts agent actions to screenshot observations, mouse movements, and keystrokes, with application scripting interfaces unavailable to the agent. This GUI-only protocol supplies no UI accessibility trees or document object models (DOMs) to candidate agents, testing their capacity for visual grounding and continuous spatial control. Crucially, our GUI-only restriction applies strictly to the agent's observation and action interfaces: the agent receives screenshots and acts through mouse and keyboard inputs, whereas the evaluation harness may inspect structured document states and project files to assess execution and delivery without exposing them to the agent.

Second, verification must account for both intermediate execution and terminal delivery. In creative production, inspecting only the final rendered image introduces severe blind spots, overlooking internal editable structures mandated by the task, such as unmerged components or adjustment layers. Conversely, static project file inspection alone cannot verify whether intermediate procedural operations were faithfully executed, which requires evidence from the execution trajectory. Because final deliverable compliance and procedural adherence address distinct failure modes, they demand separate recording and evaluation.

To address these challenges, we introduce **PHOTOARENA**, a benchmark for professional Photoshop tasks coupled with a unified pure-GUI interaction protocol and a dual-track evaluation framework. To reduce ambiguity in creative goals and operational requirements, each task is structured with a three-tier instruction hierarchy (Question $\rightarrow$ Query $\rightarrow$ UI-precise Sub-tasks). Leveraging expert references, execution trajectories, and output project files, PHOTOARENA evaluates performance along two complementary dimensions: an *Outcome Delivery Track (R)* that inspects prompt-mandated project structure and visual completeness before assessing visual editing quality, and a *Process Adherence Track (P)* that aligns execution trajectories with task-relevant operations to evaluate step-by-step procedural execution.

Evaluating multiple frontier models under both native and visual agent harnesses reveals that reliably completing professional GUI workflows remains challenging: while agents demonstrate competence on localized GUI actions (achieving an action execution rate A over 60%), end-to-end delivery success reaches only 7.7% under task-mandated structural and visual verification. These empirical baselines provide a standardized reference point for measuring future progress in GUI-based computer use.

In summary, our key contributions are:

- **A professional GUI-only benchmark.** PHOTOARENA provides professional Photoshop tasks under an explicit screenshot-based, GUI-only interaction protocol, establishing an empirical baseline for frontier models.
- **A dual-track evaluation framework.** A verification paradigm combining procedural operation adherence and final delivery evaluation, covering operational trajectories, visual results, and task-required editable structures.

2 RELATED WORK

Computer-Use Agents: From Structured Environments to Visual GUIs. Multimodal Computer-Use Agents (CUAs) have rapidly progressed across web, mobile, and operating system environments (Zhou et al., 2024; Deng et al., 2023; Koh et al., 2024; Rawles et al., 2025; Xie et al., 2024; Bonatti et al., 2025; Tan et al., 2025; Liu et al., 2024; Mialon et al., 2024), alongside domain-specific benchmarks in software engineering and enterprise applications (Jimenez et al., 2024; Yang et al., 2023; Drouin et al., 2024; Li et al., 2023; Kapoor et al., 2024). However, existing benchmarks frequently provide agents with structured UI accessibility trees or web Document Object Models (DOMs) to expose element coordinates, or evaluate agents through programmatic API and script execution. Creative applications also expose scripting interfaces (e.g., ExtendScript/JSX and UXP) that fall outside the visual GUI interaction modality targeted by this benchmark (Adobe Inc., 2024b). To isolate visual computer-use capabilities, our benchmark protocol deliberately withholds UI accessibility trees, Photoshop document DOMs, and programmatic scripting interfaces from candidate agents, compelling models to perform visual grounding and spatial coordinate reasoning directly from raw screen observations.

Procedural vs. Generative Image Editing. Instruction-guided image editing has largely centered on text-conditioned diffusion models (Brooks et al., 2023; Zhang et al., 2023; Wang et al., 2023; Sheynin et al., 2024; Hui et al., 2025), which synthesize flattened raster images that discard construction history and non-destructive layers. PSBench evaluates GUI-based agents in Photoshop (Zhang et al., 2026), including human checklist analysis of non-destructive editing in its initial study. Its public automated evaluator, as documented in the repository snapshot we cite, makes task-level pass/fail decisions from the task instruction, a task-specific rubric, and initial and result images (PSBench Contributors, 2026). This automated evaluator does not inspect internal project structure. In professional creative workflows, superficial pixel appearance can mask corrupted internal structures, such as collapsed layer hierarchies or unapplied adjustment parameters. Addressing this requires evaluating both procedural execution fidelity and the structural validity of the resulting project deliverables.

Trajectory Verification and Process Evaluation. In sequential decision-making, process-level evaluation complements sparse terminal outcomes by identifying intermediate execution failures (Uesato et al., 2022; Lightman et al., 2024). However, evaluating multi-turn GUI interaction trajectories introduces significant alignment challenges. Naive step-by-step diffing against expert demonstrations imposes rigid temporal synchronization and visual isomorphism (Dwibedi et al., 2019), inadvertently penalizing valid alternate command sequences or exploratory cursor movements (Liu et al., 2024). Our verification framework addresses these challenges by decoupling the inspection of terminal deliverables from trajectory alignment against task-relevant operational milestones.

3 THE PHOTOARENA BENCHMARK

PHOTOARENA evaluates long-horizon creative tasks in Adobe Photoshop 2026 (task-set details in Appendix A.1). Rather than toy edits, each task requires executing 10 to 18 procedural milestones involving multi-layer non-destructive compositions (smart objects, masks, adjustment layers), yielding thousands of coordinated actions across the suite.

3.1 PURE-GUI OPERATING PROTOCOL AND ENVIRONMENT

To evaluate agent interaction with professional creative software, PHOTOARENA deploys agents into a desktop environment running modern Adobe Photoshop 2026. Crucially, our benchmark establishes an explicit separation between evaluator and agent permission domains: our pure-GUI restriction

applies to the agent's observation and action interfaces, withholding UI accessibility trees, Photoshop document DOMs, and application scripting interfaces (ExtendScript/JSX and UXP) (Adobe Inc., 2024b), requiring agents to observe raw screenshots and act through human-analogous mouse and keyboard primitives. Each workflow initializes in an isolated session using the default Essentials workspace with cleared caches, auto-recovery histories, and clipboard buffers.

Within this environment, the evaluation harness retains administrative access to inspect structured document state, verify layer hierarchies and document metadata from saved project files, and manage session lifecycles, while strictly withholding such structural information from candidate agents.

3.2 Three-Tier Instruction Architecture

A central pitfall in agent benchmarking is confounding prompt ambiguity with interface execution failure (Deng et al., 2023; Koh et al., 2024; Liu et al., 2024). To decouple creative intent interpretation from operational GUI competence, every task in PhotoArena is structured into a three-tier hierarchy: (1) **Question (High-Level Goal)**, describing client-facing artistic intent and aesthetic requirements; (2) **Query (Parameterized Summary)**, specifying target dimensions, color profiles, required source assets, and key visual milestones; and (3) **Sub-tasks (UI-Precise Operations)**, detailing step-by-step menu commands, tool parameters, target layers, and masking steps.

In our benchmark evaluation, agents receive the UI-precise sub-tasks. By eliminating linguistic ambiguity, this protocol ensures that observed delivery failures (e.g., the $< 20\%$ delivery rate of frontier models) definitively reflect core limitations in fine-grained visual grounding, active layer tracking, modal dialog coordination, and continuous 2D spatial manipulation.

4 Dual-Track Objective Verification

The creative workflows in PhotoArena require both faithful step-by-step procedural execution and the delivery of editable, non-destructive digital assets. In this context, inspecting only the final rendered canvas risks rewarding destructive shortcuts that flatten layer hierarchies, while a single terminal failure (e.g., during file export) can obscure an otherwise faithful execution. To evaluate whether agents meet both sets of requirements, our framework decouples verification into two complementary tracks:

1. **Outcome Delivery Track** (R, Section 4.1): Assesses the final deliverable by filtering candidate assets through programmatic, fail-closed gates before evaluating multi-facet visual quality (F, I, C).
2. **Process Adherence Track** (P, Section 4.2): Evaluates step-by-step procedural execution along sequential milestones (A, S, D), without re-penalizing errors inherited from earlier steps.

4.1 Outcome Delivery Track: Objective Gates and Multi-Facet Scoring

For the non-destructive deliverables evaluated in PhotoArena, our framework separates functional technical requirements from aesthetic and visual judgments. Within these workflows, maintaining required layers, non-destructive masks, and specified visual entities serves as an essential functional baseline. To reflect these task requirements, the Outcome Track enforces a two-tier verification architecture (Figure 2): candidate deliverables must first clear two binary, fail-closed gates before advancing to multi-facet visual scoring. To eliminate circular evaluation bias and prompt leakage, each evaluation stage operates under strict information isolation (summary in Appendix A.3, Table 2).

Structure Gate (G_{struct}). Because downstream production requires non-destructive editability, a flattened raster image that visually resembles the target remains practically unusable if subsequent adjustments cannot be made. The submission must first contain a readable exported image and a readable `.psd`. The Structure Gate then reads only the layer topology of the `.psd` with a fixed-version PSD parser: layer kinds, layer masks, and clipping relations, without looking at pixels, dimensions, or layer names. The non-destructive steps that the task explicitly requires compile into topology predicates such as "at least one Hue/Saturation adjustment clipped to an image layer" or "at least one masked image layer". Each predicate is first checked on the expert's own `.psd`, so an agent

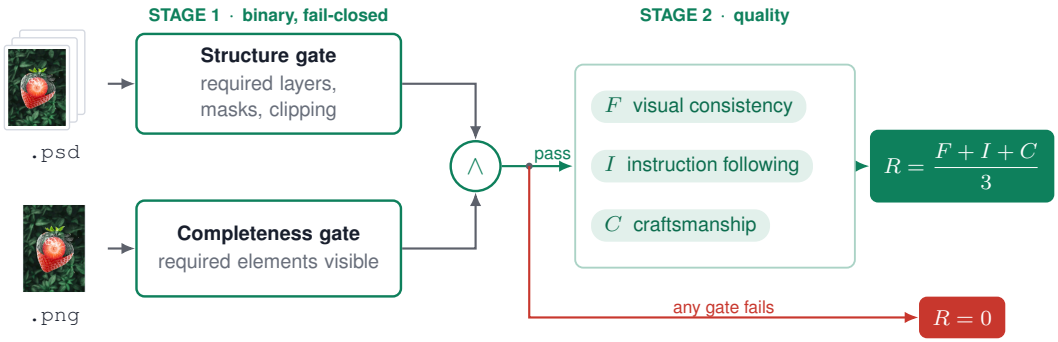


Figure 2: **Outcome Delivery Track Overview.** In Stage 1, candidate deliverables must clear two deterministic, fail-closed gates: the Structure Gate (G_{struct}) inspecting the layer structure of the `.psd` and the Completeness Gate (G_{comp}) confirming required visual entities on canvas. Passing assets advance to Stage 2 for multi-facet visual quality scoring across consistency (F), instruction following (I), and craftsmanship (C).

is never required to produce a structure the reference itself lacks. The asset passes ($G_{\text{struct}} = 1$) only if every admitted predicate holds (details in Appendix A.5).

Completeness Gate (G_{comp}). Inspecting layer structure alone cannot detect "empty shell" layers, where properly typed layers or masks are left blank or fully transparent. The Completeness Gate checks that the final-visible content the task requires (e.g., the target subject, a composited background, or requested text) is present on the exported canvas. Each requirement is written as one criterion and judged separately from the candidate render alone, without the task prompt or the expert reference, so the judge must find the content on the canvas rather than infer it from the instructions. Ambiguous evidence (faint, small, or occluded) is not treated as absence; $G_{\text{comp}} = 0$ only when some required element is clearly missing or contradicted.

Multi-Facet Quality Scoring. Inspired by prior image-editing evaluations (Wang et al., 2023; Sheynin et al., 2024; Zhang et al., 2023), we define three task-specific visual quality facets for deliverables passing both binary gates ($G_{\text{struct}} = 1 \wedge G_{\text{comp}} = 1$). Our protocol assigns each facet a distinct set of inputs:

- **Visual Consistency (F)**: Given the candidate render and the expert render, checks that the task-defining visual configuration is preserved: main subjects, part counts, spatial relations, and defining effects. The expert render is treated as one example solution rather than a pixel template, and the task prompt is supplied only to separate task-defining properties from incidental styling.
- **Instruction Following (I)**: Given the candidate render and the task prompt (without the expert render), checks that every required final-visible outcome is achieved, so alternative creative choices that satisfy the specification are not penalized.
- **Craftsmanship (C)**: Given the candidate and expert renders (without the task prompt), checks workmanship such as mask edges, blending, and text legibility, failing the candidate only for a defect that is visibly worse than the reference.

Rather than using continuous Likert ratings prone to calibration drift and central-tendency bias (Zheng et al., 2023; Wang et al., 2024), each facet (and each Completeness criterion) is a binary pass/fail decided by the majority of three votes ($F, I, C \in \{0, 1\}$). While binary scoring does not imply infallible ground-truth correctness, consensus voting significantly stabilizes evaluation decisions. The final delivery score $R \in [0, 1]$ is defined as:

$$R = G_{\text{struct}} \cdot G_{\text{comp}} \cdot \frac{F + I + C}{3}. \tag{1}$$

Failing either objective gate immediately disqualifies the asset ($R = 0$); only structurally valid and complete assets receive credit proportional to their visual quality.

4.2 PROCESS ADHERENCE TRACK: STEP-BY-STEP TRAJECTORY VERIFICATION

Evaluating agents across extended creative tasks requires assessing procedural fidelity and localizing execution drifts. To avoid fragile pixel comparisons against expert demonstrations, the Process Adherence Track decomposes each workflow into sequential procedural milestones, termed **beats** ($i = 1, \ldots, M$). Beats with no visible effect on the artwork (e.g., opening a file or saving) are not scored. Trajectory evaluation proceeds through three stages: canvas isolation, semantic forward anchoring, and per-beat scoring.

Canvas Isolation. Desktop interaction recordings contain floating parameter panels, dialog boxes, and history palettes that move and overlap during execution. Using raw screen pixels introduces substantial visual noise. To provide clean evidence, our evaluation harness extracts synchronized, headless document exports alongside raw screenshots, isolating the active document canvas for state tracking while reserving UI panel crops for tool verification (Figure 4; details in Appendix A.4).

Semantic Forward Anchor. Locating when a procedural step takes effect on canvas cannot rely on perceptual similarity (e.g., SSIM (Wang et al., 2004) or LPIPS (Zhang et al., 2018)) against expert demonstrations, as valid alternative toolpaths or brush strokes produce intermediate states that deviate from expert pixels. Instead, our **Semantic Forward Anchor** inspects the agent's own clean canvas sequence relative to the sub-task specification, strictly withholding expert frames. For each beat i, the evaluator receives the pre-operation canvas I_{before}, a candidate frame I_{cand}, and the atomic beat description, querying whether the specified visual effect has manifested. Under the assumption that committed creative modifications exhibit state persistence on the active canvas until explicitly altered, we locate the earliest manifestation frame index k_i^* via binary search over an adaptive candidate window of W_i frames, reducing VLM queries to $O(\log W_i)$ probes. If no manifestation is detected, the beat defaults to the candidate-window lower bound, recording an unexecuted step in subsequent evaluation. Complete algorithmic specifications, temporal window derivation, and boundary noise guards are detailed in Appendix A.6.

Per-Beat Scoring (A, S, D). At each anchored frame, a judge scores the beat on three dimensions, each in $\{0, 0.5, 1\}$: **Action Execution (A_i)** asks whether the agent performed the operation at all, using the canvas, the isolated operated layer, and a crop of the Layers/Properties panel (so an adjustment layer or layer style counts even when the flat canvas does not render it); **Step Match (S_i)** asks whether the agent's change on this step is the same kind of operation, at a comparable strength and look, as the expert's change on the same step; and **Goal Contribution (D_i)** asks, independently of the expert, whether the result of this step achieves what the step should contribute toward the task goal. A dimension that cannot be judged from the evidence is marked not applicable and excluded from its average. Beats with no agent frame or no localized anchor are excluded from scoring (prompts in Appendix A.8).

Handling Cascading Errors. In multi-step workflows, early errors compound: an agent that operates on the wrong layer early will leave every later canvas different from the expert's, even if its later operations are executed correctly. Penalizing all of those later steps would hide whether the agent could perform them. S_i therefore compares *changes* rather than states: the expert's before-to-after change is compared with the agent's change starting from the agent's own previous canvas. A difference inherited from an earlier step is charged only once, at the step where it arose, while a new error at any later step is still caught. The first beat that scores below 1 on any dimension is additionally recorded as the **root failure** for diagnosis.

Aggregation. For each trajectory we average each dimension over its scored beats. Averaging these per-trajectory means over the tasks in which the dimension has at least one scored beat gives the benchmark-level scores $\bar{A}$, $\bar{S}$, and $\bar{D}$, and the Process Adherence Score of an agent configuration is

$$P = \frac{\bar{A} + \bar{S} + \bar{D}}{3}. \tag{2}$$

Each dimension thus uses every task in which it can be judged, and P is an aggregate over the benchmark rather than a per-trajectory score. We additionally report $\bar{A}$ on its own as the **Action Rate**, which measures basic interface control regardless of whether the result matches the intent. Beyond

evaluation, these anchored step-level trajectories offer potential utility as step-level supervision for Process Reward Models (PRMs) (Uesato et al., 2022; Lightman et al., 2024) in computer-use agents.

4.3 Comparison with Existing Evaluation Protocols

To contextualize our verification protocol within desktop creative automation, we contrast PhotoArena with PSBench (Zhang et al., 2026), an established benchmark for GUI agents in Photoshop (detailed feature-by-feature comparison in Appendix A.10, Table 3). We distinguish its initial paper protocol from its public evaluator implementation. The initial study describes thresholded pixel- or semantic-similarity evaluation and human checklist analysis of Non-Destructive Editing Consistency (NDEC).[1] The public evaluator snapshot at commit `5a9a911` instead makes a binary pass/fail decision from the task instruction, task-specific rubric, START image, and RESULT image, without inspecting internal document structure (PSBench Contributors, 2026). Our protocol addresses a complementary evaluation setting: PhotoArena targets workflows where deliverables must remain editable project assets and tasks unfold over extended procedural sequences.

These task requirements lead to distinct evaluation priorities in PhotoArena. While inspecting final canvas renders captures visual intent fulfillment, it cannot detect shortcuts (e.g., flattening layers or raster erasing base elements) that violate our benchmark's non-destructive editing requirements. PhotoArena therefore couples programmatic PSD structure inspection of candidate `.psd` project files with visual completeness checks, verifying that non-destructive structural criteria are satisfied before evaluating visual quality. Furthermore, while previous work investigated overall process fidelity through manual checklist inspection, PhotoArena automates step-level trajectory evaluation via semantic forward anchoring, localizing procedural deviations across multi-step rollouts without re-penalizing errors inherited from earlier steps. This expanded diagnostic visibility entails operational overhead—compiling task-specific structural predicates and recording synchronized clean canvas streams—tailored to these multi-step editing workflows.

5 Experiments and Benchmark Results

5.1 Experimental Setup

Evaluated Agent Systems. We evaluate leading frontier models across two major families in production Adobe Photoshop 2026: (1) *The GPT Family*: **Codex (GPT-5.6 Sol)** (OpenAI, 2026b) and **GPT-6 Astra** (OpenAI, 2026a). (2) *The Claude Family*: **Claude Opus 5** (Anthropic, 2026b) and **Claude Fable 5.1** (Anthropic, 2026a). Models interact with the desktop environment primarily through two harness paradigms: (i) **Native-CU**, the native computer use Model Context Protocol (MCP) server (Anthropic, 2024b) provided by Anthropic's Claude Code (Anthropic, 2024a; 2025), dispatching unscaled display screenshots and OS-level event inputs; and (ii) **Peekaboo** (OpenClaw, 2026), a vision-based agent harness providing coordinate grounding and active visual re-verification. All agent rollouts are evaluated under our pure-GUI interaction protocol, in which accessibility trees, Photoshop document DOMs, and application scripting interfaces are withheld from candidate agents (Section 3), capped at an execution budget of 120 minutes per independent trial.

Reporting Conventions. Performance reporting follows the dual-track verification framework (Section 4): pass rates for the fail-closed Structure Gate (G_{struct}) and Completeness Gate (G_{comp}), the mean outcome delivery score R, and strict success rate ($R = 1.0$). All configurations are evaluated across the unified benchmark; delivery failures (e.g., runs that crash or fail to export a valid PSD file) strictly score zero and remain in the denominator under fail-closed verification. Process metrics ($\bar{A}, \bar{S}, \bar{D}, P$; Equation 2) are computed by averaging each dimension per trajectory and then over the tasks in which it can be judged, with P the mean of $\bar{A}$, $\bar{S}$, and $\bar{D}$; beats that cannot be localized or judged are left out rather than scored zero, so the Process Track covers only trajectories with at least one scorable beat.

[1] Initial submission, Sections 3.3.1–3.3.2: `https://openreview.net/pdf?id=DWATchaS4U`.

Table 1: **Main Benchmark Comparison across Frontier Models in PHOTOARENA.** Evaluated across the full task set under default settings (Native-CU, standard reasoning). Runtime is the mean system-reported execution duration per trial over the full task set, including failed and timed-out attempts; timing conventions differ across systems; trials exceeding the 120-minute execution cap or failing to export valid deliverables fail closed and score zero. Process Track columns follow Equation 2, so P is the mean of the three preceding columns; $\bar{A}$ is computed over localized beats with a decisive execution verdict, so it measures control on the steps an agent attempted rather than whole-workflow completion.

Model Family	Model System	Outcome Delivery Track (R)				Process Track (P)				Runtime
		G_{struct}	G_{comp}	Mean R	Strict (R=1)	Action $\bar{A}$	Step $\bar{S}$	Goal $\bar{D}$	Process P	Time (min)
Claude Family (Anthropic)										
	Claude Opus 5	7.7%	7.7%	0.026	0.0%	0.833	0.222	0.833	0.630	37.1
	Claude Fable 5.1	84.6%	82.1%	0.376	7.7%	1.000	0.534	0.917	0.817	10.5
GPT Family (OpenAI)										
	Codex (GPT-5.6 Sol)	43.6%	38.5%	0.094	0.0%	1.000	0.361	0.722	0.694	37.2
	GPT-6 Astra	66.7%	64.1%	0.325	5.1%	1.000	0.626	0.829	0.818	16.0

5.2 MAIN BENCHMARK RESULTS

Table 1 reports the quantitative performance of evaluated frontier agent systems across the PHOTOARENA benchmark suite. These empirical results reveal two central findings:

Delivery Bottleneck under Pure-GUI Interaction. Under our fail-closed objective verification, reliably completing professional creative workflows remains a formidable challenge when agents are evaluated under pure-GUI interaction. While evaluated frontier models demonstrate competence on localized GUI actions, strict end-to-end delivery success remains exceptionally rare when programmatic shortcuts are unavailable. Furthermore, single-gate pass rates demonstrate that agents frequently execute isolated canvas modifications or partially compliant layer structures; however, the joint pass rate clearing both structural and completeness gates simultaneously drops sharply. This reveals that under pure-GUI operation, localized UI actions often fail to produce cohesive, production-ready deliverables.

Action Competence vs. Cascading Delivery Failure. A critical insight enabled by our dual-track evaluation is the pronounced divergence between step-level operational execution and terminal deliverable success under this protocol. When operating through mouse and keyboard inputs alone, agents exhibit substantial competence when executing isolated UI actions (such as selecting tools, adjusting numeric parameters, or toggling dialogs). However, because downstream creative operations depend strictly upon intact active layer bindings, non-destructive layer masks, and precise coordinate selections, an early uncorrected mistake cascades catastrophically, corrupting the final PSD hierarchy. Execution efficiency also differs markedly across models: the two newer models finish a trial several times faster than Opus 5 and GPT-5.6 Sol (Table 1). Trajectory decompositions are provided in Appendix B.

5.3 INTERACTION HARNESS: NATIVE-CU VS. VISUAL AGENT HARNESS

To examine how the interaction interface affects delivery, we hold the model fixed and compare two execution architectures: (1) **Native-CU**, the native computer use MCP provided by Anthropic's Claude Code (Anthropic, 2025), dispatching direct OS-level input events; and (2) **Peekaboo** (OpenClaw, 2026), a vision-based harness that adds coordinate grounding and visual re-verification before and after actions. We report the outcome delivery score R and runtime only: process coverage differs across arms, so comparing P would confound the harness with scoring coverage.

The harness has a large effect for Claude Opus 5 (Figure 3a). Switching from Native-CU to Peekaboo raises its mean R roughly tenfold, and the gain comes mainly from delivery: under Native-CU almost no Opus run produces a deliverable that passes both gates, whereas most do under Peekaboo. Visual re-verification appears to catch UI state mismatches and coordinate errors before later operations build on them, and it does so at no extra runtime (Figure 3b). For Codex (GPT-5.6 Sol), by contrast, Peekaboo leaves delivery essentially unchanged while more than doubling runtime. The benefit of

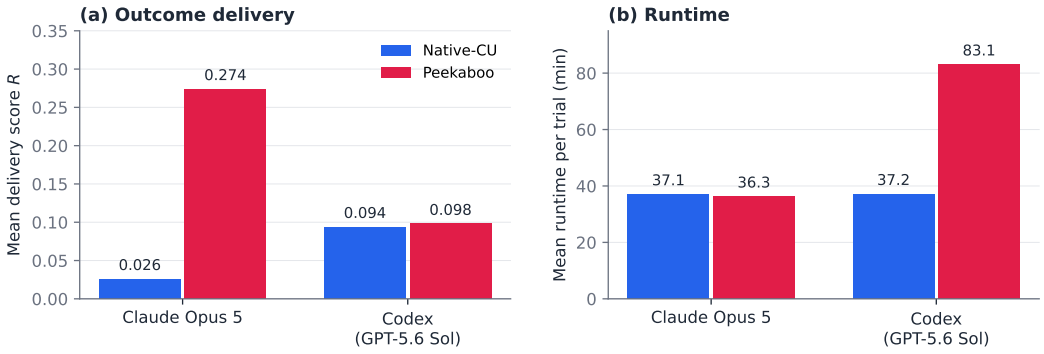


Figure 3: **Interaction harness study: Native-CU vs. Peekaboo on the full task set.** (a) Mean outcome delivery score R, with delivery failures scored zero. (b) Mean system-reported runtime per trial, including failed and timed-out attempts. Peekaboo raises R roughly tenfold for Claude Opus 5 at no extra runtime, while leaving Codex (GPT-5.6 Sol) essentially unchanged at more than twice the runtime.

visual re-verification therefore depends on the model rather than holding across agents. Qualitative examples of long-horizon failures are provided in Appendix B.

6 Conclusions

While our empirical findings provide concrete insights into multimodal agent capabilities, their interpretation is naturally delineated by several operational boundaries. First, the benchmark concentrates specifically on Adobe Photoshop workflows; evaluating agent capabilities across other professional desktop applications warrants further empirical investigation. Second, our conclusions are established under a strict pure-GUI operating protocol driven by screenshot observations and primitive mouse and keyboard actions, and may not generalize directly to automation settings that permit application scripting or accessibility APIs. Finally, although our dual-track verification combines programmatic project inspection with multi-dimensional visual assessments, automated evaluation pipelines remain susceptible to judgment discrepancies and should be regarded as structured indicators rather than infallible measurements.

Within these defined boundaries, PhotoArena establishes an empirical benchmark to assess computer-use agents on professional creative workflows under a pure-GUI protocol that restricts interaction strictly to screenshot observations and mouse and keyboard inputs, with application scripting interfaces, accessibility trees, and document object models withheld from candidate agents. To capture both operational dynamics and final deliverables, our dual-track evaluation framework separately evaluates the execution of task-relevant operations along the trajectory and the functional and visual quality of final deliverables. Empirical evaluations across frontier models indicate that reliably completing multi-step professional creative workflows remains substantially challenging under this protocol, providing a disciplined foundation and standardized baseline for evaluating future progress in visual software agents.

AI use statement

AI models are components of our evaluation framework, where they help align and score agent trajectories and final outputs. The benchmark tasks and expert reference data were not AI-generated. We also used AI tools to polish the writing. The authors take responsibility for the final content.

References

Adobe Inc. Adobe Photoshop. https://www.adobe.com/products/photoshop.html, 2024a.

Adobe Inc. Adobe Photoshop UXP developer tool and scripting guide. https://developer.adobe.com/photoshop/uxp/, 2024b.

Anthropic. Developing a computer use model. https://www.anthropic.com/news/developing-computer-use, 2024a.

Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024b.

Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2025. Accessed September 25, 2026.

Anthropic. Claude Fable 5.1 & Claude Mythos 5.1 system card. Technical report, Anthropic, September 2026a. URL https://www.anthropic.com/claude-fable-5-1-mythos-5-1-system-card.

Anthropic. Claude Opus 5 system card. Technical report, Anthropic, July 2026b. URL https://anthropic.com/claude-opus-5-system-card.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows Agent Arena: Evaluating multi-modal OS agents at scale. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4874–4910. PMLR, 2025.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18381–18390, 2023.

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 *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning (ICML)*, 2024.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17–26, 2019.

Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Cihang Xie, and Yuyin Zhou. HQ-Edit: A high-quality dataset for instruction-based image editing. In *International Conference on Learning Representations (ICLR)*, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. OmniACT: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision (ECCV)*, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. SheetCopilot: Bringing software productivity to the next level through large language models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

OpenAI. GPT-6 Astra system card. Technical report, OpenAI, September 2026a. URL `https://deploymentsafety.openai.com/gpt-6-astra/gpt-6-astra.pdf`.

OpenAI. GPT-5.6 system card. Technical report, OpenAI, July 2026b. URL `https://deploymentsafety.openai.com/gpt-5-6`.

OpenClaw. Peekaboo: macOS desktop automation and model context protocol (MCP) engine for GUI agents. `https://github.com/openclaw/Peekaboo`, 2026. Software documentation; accessed September 25, 2026.

PSBench Contributors. PSBench: Public evaluation implementation. `https://github.com/zyn1216/PSBench/tree/5a9a91119bc1fbbc03a4c4a86498741846f0fad1`, 2026. Repository snapshot, commit 5a9a911 (August 3, 2026); accessed September 25, 2026.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu Edit: Precise image editing via recognition and generation tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In *International Conference on Machine Learning (ICML)*, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Su Wang, Chitwan Saharia, Ceslee Montgomery, Jordi Pont-Tuset, Shai Noy, Stefano Pellegrini, Yasumasa Onoe, Sarah Laszlo, David J. Fleet, Radu Soricut, Jason Baldridge, Mohammad Norouzi, Peter Anderson, and William Chan. Imagen Editor and EditBench: Advancing and evaluating text-guided image inpainting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18359–18369, 2023.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Yinuo Zhang, Zian Cheng, Ziya Zhao, Zongyu Li, Bingshuo Liu, Qingbin Liu, Junxian Cai, Xi Chen, Zhiying Tu, Dianhui Chu, Xiaoyan Yu, and Dianbo Sui. PSBench: Editing image via GUI agents in Photoshop. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=O93cZGxYB1.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A Evaluation Protocol, Evaluator Prompts, and Verification Details

This section provides the complete technical specification, evaluator prompt templates, and execution protocols for the PhotoArena dual-track verification framework. Our goal is to ensure full transparency and reproducibility, covering every detail required to implement, audit, or extend the evaluation pipeline.

A.1 Benchmark Task Set

PhotoArena contains 39 tasks in total. All outcome evaluations reported in the main benchmark comparison and interaction-harness study use this task set; delivery failures remain in the denominator. Process evaluation covers only trajectories with at least one scorable beat; per-dimension task coverage is reported in Table 1.

A.2 Evaluator Models, Execution Settings, and Consensus Protocols

Judge Model. Every vision-language judgement in both tracks is made by OpenAI GPT-4o (OpenAI, 2024).

Outcome Track Judges (G_{comp}, F, I, C). Each outcome judgement is a binary vote returned under a strict JSON schema `{dimension, verdict, reason}`, where `verdict` is `PASS` or `FAIL` (the Completeness Gate additionally allows `UNRESOLVED`). All four judges share the same system message and use temperature $T = 0$ with three fixed seeds; the final label is the majority of the three votes. For the Completeness Gate an `UNRESOLVED` vote is counted as `PASS`, so faint or occluded evidence never fails a submission by itself. Images are sent as PNG with the long edge capped at 1536 px.

Process Track Judges. The forward-anchor probe (Appendix A.6) uses $T = 0.3$ and a 3-vote majority per probe, with frames sent as JPEG (long edge 760 px). The per-beat judge (Appendix A.7) uses $T = 0$ and a single call per beat; agent canvas exports and isolated layers are sent at 520 px, expert full-screen frames at 900 px, and the agent's Layers/Properties panel crop at 1200 px.

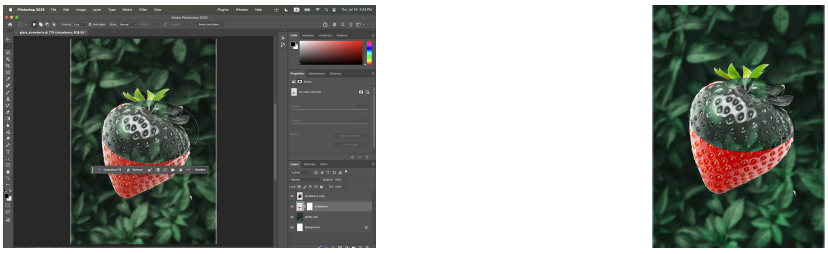
A.3 Evaluator Input Separation and Information Isolation

Beyond the primary boundary between agent and evaluator permissions (Section 3.1), our framework enforces strict *evaluator input separation* across verification components, as summarized in Table 2. In multimodal evaluation, exposing full context (e.g., combining the prompt, intermediate execution history, and expert reference) to every judge introduces severe circular biases: models frequently rationalize defective visual renders when primed with target instructions, or penalize valid creative variations when anchored to reference pixels. Our framework therefore gives each judge only the evidence its facet requires, and states explicitly how any reference evidence it receives may be used.

Table 2: **Evaluator Input Signatures in PhotoArena.** Explicit information isolation ensures orthogonal metrics without circular bias or prompt leakage. Detailed prompt templates and JSON schemas are provided in Section A.8.

Track	Component	Evaluated Aspect	Inputs Provided	Withheld Information
Gate	Structure (G_{struct})	Required layer topology	Candidate `.psd`, expert `.psd`	Rendered pixels
	Completeness (G_{comp})	One required final-visible element	Candidate render, one criterion	Task prompt, expert render
Outcome (R)	Consistency (F)	Task-defining visual configuration	Candidate render, expert render, task prompt	PSD, trajectory
	Instruction (I)	Required final-visible outcomes	Candidate render, task prompt	Expert render
	Craftsmanship (C)	Workmanship relative to the reference	Candidate render, expert render	Task prompt
Process (P)	Forward Anchor	Earliest frame showing the beat	Agent before/candidate frames, beat text	Expert frames, task prompt
	Per-Beat (A, S, D)	Execution, step match, goal contribution	Agent and expert before/after frames, panel, beat, goal	Final deliverables

Each withheld input closes a specific bias. The Structure Gate never sees pixels, so a render that merely looks right cannot mask a flattened or unbound layer stack; it reads the expert `.psd` only to confirm that each required structure is attainable (Appendix A.5). The Completeness Gate sees one criterion at a time and never the task prompt, so the judge must find each element on the canvas rather



(a) Raw OS full-window capture

(b) Synchronized isolated canvas

Figure 4: **Canvas Isolation Harness.** Desktop screenshots (left) contain cluttered UI panels, modal parameter dialogs, and floating toolbars. Our evaluation harness captures synchronized headless document exports alongside raw screenshots to isolate the pure document canvas (right) for faithful verification without visual noise.

than infer it from the instructions. Among the outcome judges, I reads the prompt without the expert render, so valid alternative solutions are not penalized for differing from the reference. F and C do see the expert render, but only as an example solution: F checks that the task-defining configuration (subjects, part counts, spatial relations, defining effects) is preserved, using the prompt only to tell task-defining properties from incidental styling, and C fails a candidate only for a workmanship defect that is visibly worse than the reference, so reference defects are never required and absolute perfection is never demanded. In the Process Track, the forward anchor sees only the agent's own frames, so locating a step never depends on resembling the expert. The per-beat judge does compare against the expert, but only against the expert's change on the same step, starting from each side's own previous state, so differences inherited from earlier steps are not charged again.

A.4 Synchronized Canvas Isolation Harness

Raw desktop frames (Figure 4a) are cluttered by floating panels and modal dialogs that occlude the artwork. Our evaluation harness therefore hooks directly into the host Photoshop engine to capture a synchronized, headless export of the active document canvas at native resolution alongside every raw display frame (Figure 4b). This provides an unoccluded, pristine record of the document canvas across every timestep t. For operations where intermediate modifications are visually masked or occluded by overlying layer groups, clipping masks, or blend modes, the harness additionally extracts the isolated operated layer rendered against an opaque white background (Layer_i). Concurrently, high-resolution panel crops isolating the Layers palette and Properties dock are captured to verify tool selection and parameter inputs independently of canvas changes.

A.5 Structure Gate (G_{struct}): PSD Structure Inspection and Task-Specific Predicates

The Structure Gate reads only the layer topology of the candidate `.psd` using the `psd-tools` parser: for every layer it records the parent group, the layer kind (e.g., pixel, smart object, type, adjustment, or fill layer), whether it carries a layer mask, and the base layer it is clipped to. Pixels, document dimensions, color and effect values, and literal layer names are not inspected, so the gate cannot be satisfied or failed by visual appearance or naming.

For each workflow, the non-destructive steps that the task prompt explicitly requires are written as a small set of topology predicates $\{\phi_1, \ldots, \phi_K\}$. Each predicate requires at least n_k non-group layers that satisfy all of the following constraints it specifies: a required layer kind; an attached layer mask; a clipping relation to an image layer (optionally, a masked image layer); and a position at the top of its layer stack. For example, a workflow that asks for a Hue/Saturation adjustment clipped to a texture and a masked subject compiles into "≥ 1 Hue/Saturation layer clipped to an image layer" and "≥ 1 masked image layer". The candidate passes if and only if all predicates hold:

$$G_{\text{struct}} = \bigwedge_{k=1}^{K} \phi_k. \tag{3}$$

Every predicate is first checked against the expert's own `.psd`. A predicate that the expert file does not itself satisfy (e.g., because the expert saved the file before a later step described in the prompt) is marked unresolved and does not count against the candidate (when no usable expert `.psd` exists, e.g., the expert file has no layer records, the gate is waived for that workflow), so no agent is required to produce a structure that the reference lacks. These predicates are necessary conditions only: they verify that the required non-destructive structure exists, not that each layer plays its intended semantic role. Independently of the predicates, every submission first passes an artifact check: it must contain a readable exported image and a readable `.psd` with at least two layers (one for the few workflows whose expert file is itself single-layer); a flattened or missing project file scores $R = 0$.

A.6 Semantic Forward Anchor: Binary Search Algorithm and Window Bounding

This subsection formalizes the mathematical problem formulation, temporal window derivation, and boundary noise verification mechanics of the LLM Semantic Forward Anchor.

A.6.1 Problem Formulation and Temporal Window Derivation

Let $\mathcal{F} = \langle (I_1, t_1), (I_2, t_2), \ldots, (I_T, t_T) \rangle$ denote the chronological sequence of clean document canvas frames captured by our headless harness during an agent's execution rollout, where each frame I_k corresponds to interaction timestamp t_k. Let $\mathcal{B} = \langle b_1, b_2, \ldots, b_M \rangle$ represent the sequence of procedural workflow milestones (beats), where each beat b_i is specified by an atomic natural language description $\mathcal{T}_i$ (e.g., *"Apply Gaussian Blur to background"*).

For each visual beat b_i, the evaluation objective is to locate the earliest frame index $k_i^* \in \{1, \ldots, T\}$ at which the visual modification mandated by $\mathcal{T}_i$ has physically manifested on the canvas, without using expert reference frames.

Physical Interaction Trigger and Candidate Bounds. To avoid scanning the entire multi-hundred-step trajectory, the search is bounded within an adaptive temporal candidate window $\mathcal{C}_i = \langle I_{lb_i}, \ldots, I_{ub_i} \rangle$:

- **Lower Bound (lb_i)**: Let evt_i denote the frame index of the agent's initial physical interaction triggering beat i (detected via interaction logs or DOM-derived document-state records). To guarantee strict temporal progression across successive beats, the lower bound is constrained by:

$$lb_i = \max(\text{evt}_i, k_{i-1}^* + 1), \quad \text{with } k_0^* = 0. \tag{4}$$

- **Upper Bound (ub_i)**: In desktop creative software, complex operations exhibit variable visual latency: modal parameter dialogs, slider adjustments, and GPU filter rendering mean that the visual result often lags the initial tool click by $+4$ to $+70$ frames. Thus, bounding the search strictly at the subsequent operation's start would prematurely truncate the search before the effect commits. We therefore define the upper bound ub_i as the next structural checkpoint boundary (such as document saving, tab switching, or the terminal frame T):

$$ub_i = \min(\{\text{evt}_j \mid j > i \text{ and } b_j \text{ is a structural boundary}\} \cup \{T\}). \tag{5}$$

  Because the VLM query is strictly operation-specific, subsequent unrelated canvas operations will not cause premature false-positive matches within this window.

- **Pre-Operation Baseline (I_{before})**: The baseline frame representing the document state immediately prior to beat i is taken as the latest clean canvas frame preceding the lower bound:

$$I_{\text{before}}^{(i)} = \begin{cases} I_{lb_i - 1}, & lb_i > 1, \\ \text{unavailable}, & lb_i = 1. \end{cases} \tag{6}$$

For a nonempty candidate window, let $W_i = ub_i - lb_i + 1$ and use zero-based local indexing $\mathcal{C}_i[j] = I_{lb_i + j}$ for $j = 0, \ldots, W_i - 1$. A returned local index j_i^* corresponds to the global frame index $k_i^* = lb_i + j_i^*$ and timestamp $t_{k_i^*}$. An empty window ($lb_i > ub_i$) provides no candidate frame and is excluded from scoring.

A.6.2 Monotonicity Principle and Binary Search

Once committed in Photoshop, canvas raster modifications, vector shapes, typographic layers, and non-destructive adjustments remain persistently visible on the document canvas until explicitly overwritten. Consequently, the Boolean manifestation predicate:

$$f_i(k) = \text{VLM_Judge}(I^{(i)}_{\text{before}}, I_k, \mathcal{T}_i) \in \{0, 1\} \tag{7}$$

behaves as a monotonic step-function over the ordered candidate window $k \in [lb_i, ub_i]$:

$$f_i(k) = \begin{cases} 0, & \text{for } k < k_i^*, \\ 1, & \text{for } k \geq k_i^*. \end{cases} \tag{8}$$

Exploiting this monotonicity, we perform forward binary search (formalized in Algorithm 1) to locate the exact transition point $k_i^* = \min\{k \in \{lb_i, \ldots, ub_i\} \mid f_i(k) = 1\}$ when this set is nonempty.

Algorithm 1: LLM Semantic Forward Anchor via Monotonic Binary Search

Input: Pre-operation canvas I_{before}, candidate canvas frames $\mathcal{C} = \mathcal{C}_i$ of length $W = W_i$, indexed locally by $j = 0, \ldots, W-1$, operation description $\mathcal{T}$, probe budget $B_{\text{max}} = 12$.
Output: Earliest local manifestation index j_i^* (global index $k_i^* = lb_i + j_i^*$), or `None` if unexecuted.

Initialize: $lo \leftarrow 0$, $hi \leftarrow W - 1$, $ans \leftarrow$ `None`, probes $\leftarrow 0$, cache $\leftarrow \{\}$

function QUERYDONE(j):
  if $j \in$ cache **then return** cache[j]
  probes $\leftarrow$ probes $+ 1$
  $d \leftarrow$ VLM_Judge($I_{\text{before}}, \mathcal{C}[j], \mathcal{T}$) *// 3-vote majority, $T = 0.3$*
  cache[j] $\leftarrow d$; **return** d

while $lo \leq hi$ **and** probes $< B_{\text{max}}$ **do**
  $mid \leftarrow \lfloor (lo + hi)/2 \rfloor$
  if QUERYDONE(mid) == `True` **then**
    $ans \leftarrow mid$; $hi \leftarrow mid - 1$ *// Search left half for earlier manifestation*
  else
    $lo \leftarrow mid + 1$ *// Operation not yet present; search right half*

// Boundary Noise Guard: verify against single-probe false-negative at the boundary
if $ans \neq$ `None` **and** $ans > 0$ **and** probes $< B_{\text{max}}$ **then**
  if QUERYDONE($ans - 1$) == `True` **then** $ans \leftarrow ans - 1$

return ans

A.6.3 Boundary Noise Guard and Failure Handling

Boundary Noise Guard. Although 3-vote majority voting ($M_{\text{vote}} = 3$) dramatically stabilizes VLM decisions, minor visual ambiguity near the transition boundary could occasionally yield a false-negative query at $ans - 1$. To ensure absolute temporal accuracy, once the binary search converges on candidate index ans, Algorithm 1 performs a single confirmatory probe on $ans - 1$. If $ans - 1$ is also verified to manifest the operation, the anchor index is corrected to $ans - 1$.

Unexecuted Steps. If the binary search terminates with $ans =$ `None`, the visual effect was never rendered on the document canvas throughout the candidate window. In this case, the harness falls back to anchoring the beat at the candidate-window lower bound lb_i. The per-beat judge then sees no trace of the operation at that frame and scores it accordingly (typically $A_i = 0$).

A.6.4 Query Cost

A linear scan would query the judge once per frame in each beat's candidate window, i.e., $\sum_{i=1}^{M} |\mathcal{C}_i|$ probes per trajectory. Because f_i is monotonic, the binary search needs at most $\lceil \log_2 |\mathcal{C}_i| \rceil + 1$ probes per beat (including the boundary guard), capped at $B_{\text{max}} = 12$; each probe is a 3-vote majority.

A.7 Per-Beat Scoring and Aggregation

Each workflow's beats are first classified once, on the expert side, as *visual* or *procedural*; procedural beats with no visible change to the artwork (e.g., opening a file, creating a blank canvas, saving) are

excluded from scoring. For every visual beat, the per-beat judge receives the expert's frames before and after that beat, the agent's clean canvas before the beat and at its forward anchor, the agent's isolated operated layer when the composite canvas cannot show it (grouped, clipped, adjustment, or smart-object layers), a high-resolution crop of the agent's Layers/Properties panel, the beat description, and the overall task goal. It returns three scores in $\{0, 0.5, 1\}$, or `na` when the effect cannot be judged from the evidence:

- **Action Execution** (A): whether the agent performed the operation at all, using the canvas, the isolated layer, and the panel (an adjustment layer, smart filter, or layer style visible in the panel counts even if the flat canvas does not render it).
- **Step Match** (S): whether the agent's change on this step (agent-before $\rightarrow$ agent-after) is the same kind of operation with a comparable strength and look as the expert's change (expert-before $\rightarrow$ expert-after). Because only the change is compared, a difference that the agent's canvas inherited from an earlier step is not charged again, while a new error at any step is still caught. The first visual beat has no before frame and is compared to the expert's after frame directly.
- **Goal Contribution** (D): independently of the expert, whether the result of this step achieves what the step should contribute toward the task goal, judged on the step's own intent rather than on whether the canvas already looks finished.

Beats with no agent frame or no localized anchor are excluded from scoring rather than assigned zero. Dimensions marked `na` are excluded from their respective task averages. S and D values flagged low-certainty are also excluded from their respective task averages. For each trajectory the scorer reports the mean of A, S, and D over the scored beats. The benchmark-level $\bar{A}$, $\bar{S}$, and $\bar{D}$ average these per-trajectory means over the tasks in which each dimension has at least one scored beat, so the three dimensions may cover different tasks; P is their mean (Equation 2). It also tags the first beat with any score below 1 as the root failure and the scored beats after it as cascade; these tags are diagnostic only and do not change the averages. In the released code, S and D are named `b1` and `b2`.

A.8 Complete Evaluator Prompt Templates and Output Schemas

Below we reproduce the prompts used by the released scorer verbatim; placeholders in braces are filled per task, and images follow the text in the order stated. The four outcome judges share the system message and output schema shown in the first box.

Shared System Message and Output Schema (Outcome Track)

System message (all outcome judges), followed by the JSON schema enforced through structured outputs:

```
You are a strict binary result-only visual judge.  Return only the
requested JSON object.
{"dimension":  "<DIMENSION>", "verdict":  "PASS" | "FAIL" (|
"UNRESOLVED" for Completeness), "reason":  "<string>"}
```

Completeness Gate Evaluator (G_{comp})

User message (one call per criterion; image: candidate render):

```
Judge only whether the specified final visible content is present in
this image.  This is a completeness gate, not a quality, realism or
expert-similarity score.  The criterion already accounts for the
task's final state.  Do not add requirements about source assets,
intermediate layers, named filters, hidden operations or exact numeric
settings.  Blending, transparency, feathering and occlusion may change
how a requested element looks.  PASS when the specified content is
visibly present, even if imperfect.  FAIL only for a clearly absent or
contradicted required element; cite the concrete missing element.
Faint, ambiguous, small or occluded evidence that does not establish
absence is UNRESOLVED, not FAIL. Do not treat a supplied image as
correct merely because it may be an expert reference.  Ignore
application UI.
criterion_id:  {CRITERION_ID}
final_visible_criterion:  {FINAL_VISIBLE_CRITERION}
```

Visual Consistency Evaluator (F)

User message (images: candidate render, then expert render):

```
Compare final images in order CANDIDATE, REFERENCE. Evaluate VISUAL
CONSISTENCY, not workmanship or whether the instructions were followed.
The reference is an example solution, not a pixel template.  First
inspect BOTH images:  identify their main subjects, part counts,
relative positions, containment, overlap/occlusion, orientation, and
defining visual effects.  Then compare those relationships explicitly.
Similar objects alone do not establish consistency.  A change from
embedded/inside to in-front/outside, a missing section, or a different
defining effect is substantive.  Use the prompt only to distinguish
task-defining properties from incidental styling; never let a shared
topic or the prompt's prose override a visible relational mismatch.
Allow different incidental positions, scale, spacing, colors and
decorative choices when they preserve the defining composition and are
not explicitly constrained.  Allow cleaner edges, smoother blends,
better integration and other quality improvements; reproducing
reference defects is never required.  FAIL for an identified
substantive difference in task-defining visual configuration, not for
any pixel/local difference.  PASS when the defining configuration is
preserved.  Cite the compared relationship and explain why any
difference is substantive or incidental.  Identical visible images
have no consistency mismatch.  Ignore hidden process and PSD facts.
TASK CONTEXT (not a separate instruction-following test):
{TASK_PROMPT}
```

Instruction Following Evaluator (I)

User message (image: candidate render):

```
Evaluate only the CANDIDATE's final visible instruction following
using the full task prompt below.  The prompt may include intermediate
Photoshop steps, tools, layer names, PSD structure, blend modes,
filters, masks, exact numeric settings, or implementation details that
are not necessarily visible in the final render.  Treat those as
process context, not visible-result requirements.  Do not infer or
penalize anything that cannot be verified from visible pixels.  First
identify only required visible final outcomes:  subjects, objects,
visible text, composition, spatial relationships, orientation, color,
and visibly apparent effects.  PASS only when every required visible
outcome is clearly achieved.  Any missing, partial, contradictory, or
visibly wrong required outcome requires FAIL. Cite visible evidence
only.
FULL TASK PROMPT:
{TASK_PROMPT}
```

Craftsmanship Evaluator (C; `VISUAL_QUALITY` in code)

User message (images: candidate render, then expert render):

```
Evaluate VISUAL QUALITY using final images in order CANDIDATE,
REFERENCE. The acceptance standard is workmanship at least as good as
the supplied reference, not hypothetical defect-free professional
artwork.  Compare visible edge/mask cleanup, blending/transitions,
compositing integration, texture artifacts and readability.  FAIL only
for a concrete candidate defect that is newly introduced or visibly
worse than the corresponding reference workmanship.  A local defect is
sufficient if clearly worse; do not average it away using an unrelated
improvement.  Name the location and describe the candidate versus
reference evidence.  Defects equally present in both are the baseline,
not grounds to fail.  A cleaner or equally well-crafted result passes.
Do not mistake intentional collage, surreal floating objects, stylized
outlines or graphic effects for defects solely because they are not
photorealistic.  Do not penalize stylistic differences, object
```

```
identity, missing requested content or spatial configuration here:
those belong to consistency, completeness and instruction following.
Do not require matching the reference pixels.  For different layouts
compare analogous surfaces and edges, not identical coordinates.  PASS
if no concrete workmanship regression can be substantiated.  Identical
visible images have equal workmanship.  Use only visible images, never
trajectory, tools, hidden layers or imagined defects.
```

Semantic Forward Anchor Probe

User message (followed by the labelled images `BEFORE`, `CANDIDATE`):

```
You compare two frames from a Photoshop screen-recording of ONE agent
editing a document.
OPERATION TO CHECK: «{BEAT_SEMANTIC_TEXT}»
BEFORE is an earlier frame (the state before/around when this
operation began).  CANDIDATE is a later frame.
Decide ONLY ONE THING: has THIS operation been CARRIED OUT -- i.e.  is
its visible effect PRESENT in CANDIDATE while it was NOT yet present
in BEFORE? Judge the OPERATION described by the text; ignore other
unrelated changes.
Rules:  You are NOT judging quality, correctness, or how well it was
done -- only whether the operation visibly happened.  A placed/pasted
image counts as done once its pixels are visible on the canvas (even
if not yet transformed).  A filter/adjustment counts as done once its
characteristic visual effect appears.  An open dialog/preview with the
effect visibly applied to the canvas counts as done.  A blank canvas,
an unchanged canvas, or only a tool/menu being selected with NO
visible effect yet = NOT done.
Return STRICT JSON {"desc":"<4-10 words:  what CANDIDATE
shows>","done":true|false}.
```

Per-Beat Evaluator (A, S, D; `executed`, `b1`, `b2` in code)

User message (followed by the labelled images `EXPERT-BEFORE`, `EXPERT-AFTER`, `AGENT-BEFORE`, `AGENT-AFTER`, and, when available, `AGENT-AFTER-LAYER` and `AGENT PANEL`; `{GOAL}` is the task prompt and `{SEM}` the beat description):

```
You grade ONE beat of an agent reproducing an expert's Photoshop
workflow toward a task goal.
THE OVERALL TASK GOAL: «{GOAL}»
THIS BEAT is meant to:  {SEM}.
You may get EXPERT-BEFORE, EXPERT-AFTER, AGENT-BEFORE, AGENT-AFTER
canvases, an AGENT PANEL (Layers/Properties crop), and sometimes an
AGENT-AFTER-LAYER. The EXPERT applied this beat's operation
(BEFORE->AFTER).
IMAGE FORMATS -- READ CAREFULLY: the EXPERT-BEFORE/AFTER are
FULL-SCREEN Windows Photoshop screenshots -- the document CANVAS is
the bright rectangle embedded roughly in the center; the rest is
menus/tools/panels.  LOCATE that canvas region and compare IT to the
agent's canvas; also use the expert screenshot's own Layers panel /
menus / open dialog to confirm which tool/filter/adjustment was used.
Look closely -- do NOT rubber-stamp a match just because the embedded
canvas is small.  The AGENT side is given as SEPARATE views:
AGENT-BEFORE/AFTER are clean CANVAS-ONLY exports; AGENT-AFTER-LAYER
(when present) is the isolated operated layer; the AGENT PANEL is a
hi-res crop of the agent's Layers/Properties.  Use ALL of these
together to reason about what each side did.
IMPORTANT about AGENT-AFTER-LAYER: the flat AGENT-AFTER canvas is a
COMPOSITE export that CANNOT render a layer nested in a group /
clipped / an adjustment / a smart object -- so for those steps the
flat canvas shows the UNPROCESSED pixels and looks like nothing
happened.  When an AGENT-AFTER-LAYER image is provided, it is the
agent's OPERATED layer isolated on a white background -- it is the
AUTHORITATIVE view of what THIS step actually did.  Trust it over the
```

```
flat AGENT-AFTER for judging A and B2.  (Do not penalize its white
background or the fact that an isolated adjustment can render
darker/differently than it will in the full stack -- judge whether the
operation is present and sensible.)
Judge THREE things:
A) executed:  did the AGENT actually perform this operation?  Use the
AGENT-AFTER canvas, the AGENT-AFTER-LAYER (if provided) AND the AGENT
PANEL: if the isolated layer shows the change, or the panel shows the
adjustment layer / smart filter / layer style / effect for this
operation was ADDED (e.g.  a 'Hue/Saturation' adjustment layer, a
smart filter in the stack, an 'Outer Glow' fx), the operation WAS
executed -- score 1 -- EVEN IF the flat AGENT canvas doesn't render it.
1 = clearly performed, 0.5 = partial/ambiguous, 0 = no trace anywhere.
B1) low-level correctness (vs EXPERT): comparing the TRANSFORMATIONS
-- did the agent apply the SAME KIND of operation with a comparable
degree/strength as the expert?  Judge the CHANGE introduced, NOT
absolute appearance:  AGENT-BEFORE may already differ from
EXPERT-BEFORE due to earlier steps -- do NOT penalize that inherited
difference, only judge whether THIS operation was applied correctly on
top of whatever the agent had.  1 = correct transformation, 0.5 =
applied but noticeably off, 0 = wrong or not applied.  WHETHER the
operation happened AT ALL is A's job -- do NOT reward mere presence
here.  B1 judges the MATCH: a filter/effect applied at a CLEARLY
heavier or lighter strength, a denser/coarser texture, or a different
color/contrast than the EXPERT-AFTER is 0.5 (noticeably off), NOT 1.
Reserve 1 for a genuinely comparable degree AND look.  If the effect
is NOT visible on the flat canvas (adjustment layer/glow/blend-mode
the export doesn't render -- but the panel confirms it was added),
return b1='na' with b1_certainty='low'.  Do NOT score it 0.
B2) high-level prompt consistency (vs the TASK GOAL): INDEPENDENTLY of
the expert, does the AGENT-AFTER result (use the AGENT-AFTER-LAYER
when provided -- the flat canvas may hide this step's operated layer)
semantically ACHIEVE what THIS step should contribute toward the goal?
Judge the STEP'S OWN intent -- NOT whether the canvas already looks
like the finished product (early steps legitimately don't).  1 = the
step's intended effect is achieved and sensible toward the goal, 0.5 =
attempted but weak/questionable, 0 = the intended effect is ABSENT or
CONTRADICTS the goal (e.g.  the goal wants an ICY BLUE hand but after
this colorize step the hand is still gray).  If this step's
contribution genuinely cannot be seen or inferred (invisible effect,
panel inconclusive), return b2='na' with b2_certainty='low'.
Return STRICT JSON {"executed":1,"b1":1,"b1_certainty":"high","b2":1,
"b2_certainty":"high","note":"one sentence"}.
```

A.9 Trajectory Alignment Visualization

Figure 5 visualizes how the Process Track aligns an agent rollout with the expert demonstration on a multi-step compositing workflow; Figure 7 in Appendix B shows the same alignment on a procedural filter cascade.

A.10 Methodological Comparison with Existing Protocols (PSBench)

To complement the discussion in Section 4.3, Table 3 contrasts PhotoArena with PSBench (Zhang et al., 2026) across core evaluation dimensions, detailing the specific differences in target scope, evidence inputs, and decision mechanisms.

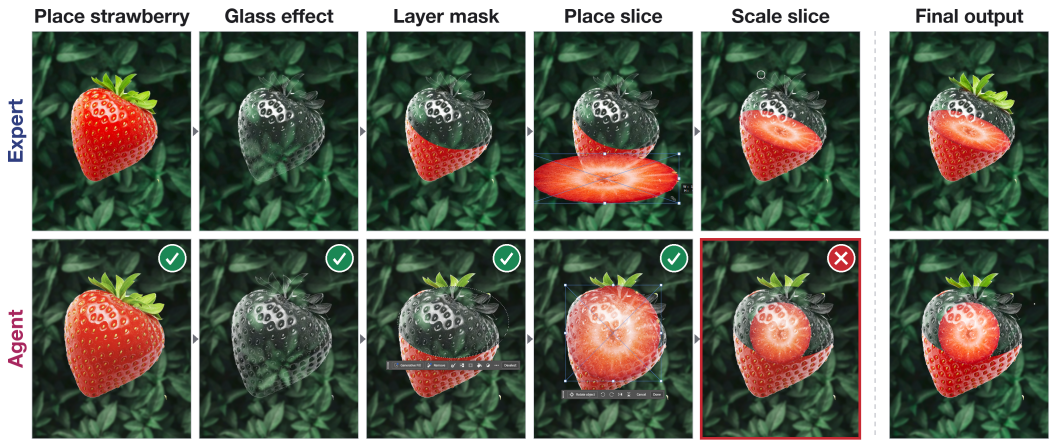


Figure 5: **Step-by-Step Trajectory Alignment in the Process Adherence Track.** Comparing an expert demonstration (top) against an agent execution rollout (bottom) across sequential procedural beats for a multi-step compositing workflow. The agent faithfully executes localized operations across early beats (placing the subject asset, establishing container transparency, applying the non-destructive layer mask, and inserting the fruit slice). At the final scaling step, the agent positions and scales the strawberry slice differently from the expert's placement, resulting in a divergent composition. Our Semantic Forward Anchor identifies the earliest manifestation frame of each beat from the agent's own canvas modifications without requiring perceptual pixel alignment against the expert demonstration, so that per-beat scoring (A, S, D) charges an early deviation once rather than again at every later step.

Table 3: **Comparison of Evaluation Protocols in Desktop Creative Applications.** Contrast across evaluation scope, evidence inputs, and decision mechanisms between PSBENCH (Zhang et al., 2026) and PHOTOARENA. Outcome rows describe the public evaluator at commit `5a9a911` (PSBench Contributors, 2026); process rows describe the initial paper's human NDEC study (Section 4.3). These are distinct sources, not a claim that the repository implements the initial paper's full protocol.

Dimension	**PSBENCH** (Zhang et al., 2026)	**PHOTOARENA (Ours)**
Target Scope	Visual task fulfillment; non-destructive workflow properties in human study.	Production asset integrity; prompt-mandated editable structure and procedural adherence.
Outcome Evidence	START image, RESULT image, task prompt, and task-specific rubric.	Candidate `.psd` file (layer hierarchy and metadata), exported render, prompt, entity list, reference render.
Process Evidence	Full episode screen recording; manual post-hoc checklist inspection (NDEC).	Synchronized headless canvas frames and panel crops; automated forward anchor.
Decision Mechanism	Multimodal LLM judge for outcome pass/fail (repository); manual inspection for process rubrics (initial study).	Programmatic PSD structure inspection (G_{struct}), multi-facet VLM consensus (G_{comp}, F, I, C), stepwise per-beat judge (A, S, D).
Output Granularity	Task-level binary success (outcome); rubric compliance (process).	Fail-closed gates ($G_{\text{struct}}, G_{\text{comp}}$), multi-facet delivery (R), step-level process (P, A).

B Additional Qualitative Deliverables and Trajectory Decompositions

Figure 6 illustrates expert reference deliverables against agent outcomes across six representative creative workflows, showcasing distinct execution regimes ranging from high-fidelity compositions to partial composites and unclosed modal states.

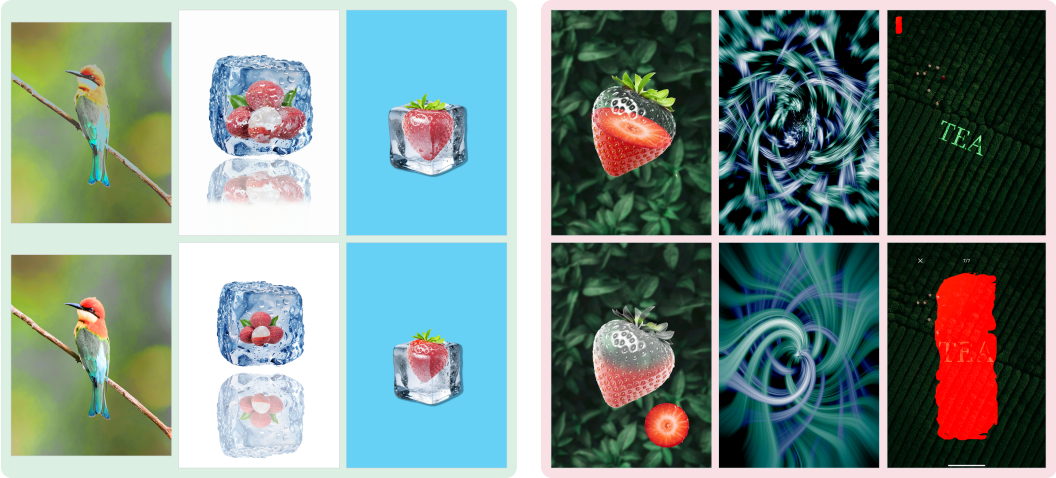


Figure 6: **Expert Reference Deliverables (Top Row) vs. Agent Final Outcomes (Bottom Row) across Diverse Creative Workflows.** Extended gallery illustrating distinct outcome regimes: high-fidelity reproductions in multi-layer adjustment and masking workflows (left three) versus subtle geometric drifts, partial composites, and unclosed modal sub-states (right three).

Figure 7 presents a detailed step-by-step trajectory decomposition for a procedural filter cascade workflow, demonstrating how sequential procedural milestones (Crystallize → Polar Coordinates → Spatter → Gaussian Blur → Hue/Saturation, followed by the title text) are localized and tracked via semantic forward anchoring. The same alignment on a compositing workflow is shown in Figure 5 (Appendix A.9).

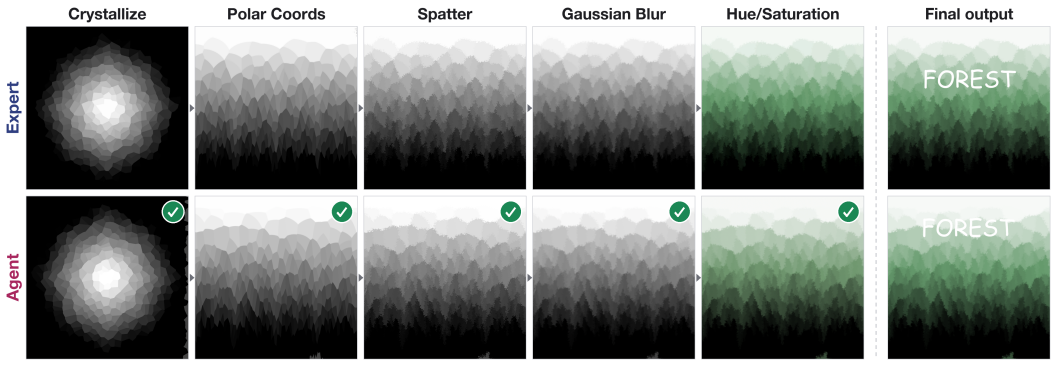


Figure 7: **Trajectory Decomposition in a Procedural Filter Cascade.** The agent faithfully tracks the expert through five distinct procedural milestones before drifting during final typographic placement.