JUSTJUDGE: BETTER EVIDENCE BEATS MORE MACHINERY IN JUDGING GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deciding from a recorded trajectory whether a GUI agent completed its task underpins how these agents are evaluated, how their training data is curated and how they are rewarded. To make that decision reliable, recent work adds machinery, such as critic pipelines and trained reward models. We show that the binding constraint is what the judge is shown. A trajectory is testimony as much as evidence: the agent under judgment writes part of the record, and a judge handed that record inherits the agent's verdict on itself. Prior judges warn the model about the record or discard the agent's text, and added machinery re-reads the same record. JustJudge instead sanitizes the record: it reduces each step to the action issued and a claim-free statement of what it was for, and masks recognized self-reported outcomes. The judge checks this step log against a fixed budget of screenshots anchored at the final state. The judge is a text-only extraction pass and one verification call to a vision-language model, with no training and no environment access. On OSReward, JustJudge with an open 27B model reaches 89.8% accuracy, level with the best proprietary judges the benchmark reports, and leads every judge on the hard split. The gain comes from the evidence, not the model: shown the sanitized record, a 9B model overtakes the reward model trained from it on that split. On the same backbone, published multi-call procedures cost up to 16 times as much and score 8 to 21 points lower. Downstream, the same judge produces stronger agents on OSWorld-Verified, as a filter on fine-tuning data and as the reward for online reinforcement learning. Reliable judging is a question of evidence, not of machinery.

1 INTRODUCTION

Agents capable of operating real graphical user interfaces are advancing rapidly (Wang et al., 2025; Lai et al., 2025; Lv et al., 2026; Awadallah et al., 2025; Lin et al., 2026; Yang et al., 2026). Almost every stage of the agentic pipeline rests on a single operation: deciding whether the agent has accomplished the given task from a recorded trajectory. Benchmarks need that decision to score agents on tasks for which nobody has written a checker, data pipelines require it to choose which rollouts are worth training on, and online reinforcement learning takes it as the reward for every rollout it collects. Hand-written checkers provide the decision for a few hundred curated tasks (Xie et al., 2024; Yuan et al., 2026). However, each is a task-specific program that cannot generalize to the open-ended tasks that agents are expected to handle. Vision-language judges offer a scalable alternative, making reliability, rather than availability, the key bottleneck.

Efforts to improve the reliability of these judges have progressed along two axes. One adds machinery to the judging procedure: critic roles and verification rounds (Li et al., 2026; Wan et al., 2026), repeated sampling with a unanimous vote (Yang et al., 2025), a verdict per screen (Bai et al., 2024), rubrics generated before scoring (Rosset et al., 2026; Quinn et al., 2026), or a reward model trained on annotated trajectories (Sun et al., 2026; Xiao et al., 2025; Chen et al., 2025; Chae et al., 2025; Song et al., 2026). The other acquires evidence beyond the record by letting the judge act in the environment (Cui et al., 2026; Shi et al., 2026a; Dai et al., 2026), the most direct way to resolve ambiguities left by the recording, and one that requires a live, restorable environment. Much of the judging that agent pipelines actually perform happens elsewhere: on corpora recorded weeks earlier or collected by someone else, and in training loops that recycle environments once a rollout ends.

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.

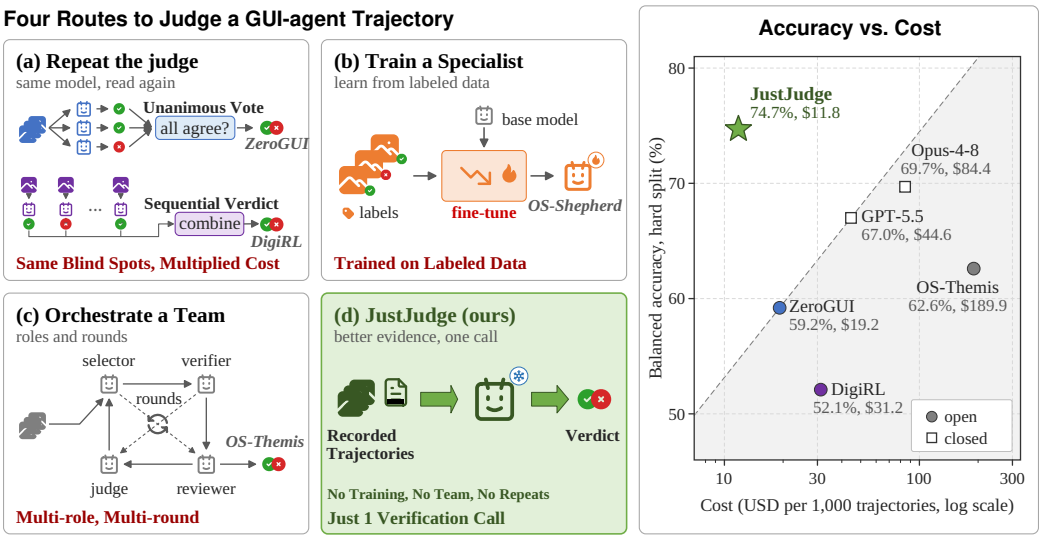


Figure 1: **Four routes to judging a recorded GUI-agent trajectory, and what they buy.** Existing judges (a) run the same model again, as a unanimous vote or a verdict per step, (b) fine-tune a specialist reward model, or (c) orchestrate critic roles over several rounds. JustJudge (d) makes one verification call over the record with the agent's outcome claims masked. Right: balanced accuracy on the OSReward hard split against the cost of judging a thousand trajectories, with open judges filled and the two closed models hollow at the accuracy and cost the benchmark reports for them (§4.1, Table 1). Every judge shown costs more and scores lower than JustJudge, the only point above the dashed cost-accuracy frontier.

The benchmarks that measure the judges are themselves distributed as recordings (Sun et al., 2026; Li et al., 2026; Lin et al., 2025; Lù et al., 2025). There the record is all there is, yet far less attention has been paid to what the judge is shown than to what it is asked to do. Current judges either read the record as logged by the agent pipeline, under a prompt warning them not to trust it, or discard the agent's textual trace and judge from screenshots alone (§2). Both treat the agent's account of what it did and its account of whether it succeeded as one block, to be trusted with a warning or discarded together, and every added procedure re-reads the same block at a multiple of the original cost. Which part of that block is evidence, and which is testimony, is the question this paper asks.

Effectively presenting the record poses three key challenges. First, **self-reported outcomes**: a trajectory is written in part by the agent under judgment, whose step-by-step reasoning and closing declaration of success are statements by an interested party. A judge that reads them inherits the agent's own assessment of its work, an over-acceptance that benchmarks report as the dominant judging error (Sun et al., 2026; Quinn et al., 2026). Second, **partial observability under a frame budget**: trajectories run to dozens of steps at full desktop resolution, more than a judge can afford to read. Whichever screens are dropped become invisible, while the actions taken in them still have to be accounted for (He et al., 2024; Xue et al., 2025; Wang et al., 2026c; Mohammadi, 2026). Third, **reward-grade cost**: a judge that scores every trajectory in a corpus or every rollout in a training loop competes for the same compute as the agent it supervises (Li et al., 2026; Sun et al., 2026). Whatever it is shown has to be cheap to read. Reading a trajectory reliably is therefore a question of evidence: what to remove from the record, what to keep, and where to look.

In this paper, we introduce **JustJudge**, a training-free judge that answers these questions from the record alone and verifies the trajectory in one verification call (Figure 1). Against self-reported outcomes, it sanitizes the record rather than warning the judge about it, replacing the agent's narration with a claim-free step log: each step becomes the action the agent issued, its self-declared outcomes masked and everything else preserved, together with a brief intent that states what the step was for and asserts nothing about what it achieved. Against partial observability, the log covers every step, and the sixteen screenshots shown always include the first and the last five, anchoring the verdict at the final state. The two act as one intervention: the log removes the agent's account of whether it succeeded, and the final frames are where the judge establishes that for itself. Against reward-grade

cost, the evidence goes to that one call, with no training, no task-specific rubric and no environment, so the same judge applies to a live training loop and to a corpus recorded months ago.

Better evidence is worth more than more machinery (Figure 1, right). On OSReward (Sun et al., 2026), JustJudge with an open 27B model reaches 89.8% accuracy, level with the best proprietary judges and 3.7 points above the best open judge it reports, and 74.7% balanced accuracy on the hard split, above every judge reported there. Two controlled comparisons place the gain in the evidence rather than in the model or the instructions. With the backbone fixed and the same warning in the prompt, the claim-free step log is worth 3.1 points on the hard split over the record as logged and removes 12 percent of the judge's false acceptances overall, and the benchmark's own 9B base model, shown its evidence this way, overtakes the reward model trained from it on that split. A flip test isolates the declaration's effect: changing only the agent's declared outcome, with every screenshot and non-terminal action held fixed, moves the judge that reads the record by 4.9 points and by 8.8 on the hard split, while the step log moves by 0.3 on both. Machinery costs more and scores lower: a step-wise judge, a unanimous vote and a critic framework of nineteen calls cost 1.6 to 16 times as much on the same backbone and land 8 to 21 points below it. Downstream, the same judge adds 4.2 points of agent success as a filter on fine-tuning data and 2.0 points as the reward for online reinforcement learning, this time over the environment's own rule-based checker.

Our contributions are three. (i) A diagnosis. The record a judge reads is testimony as much as evidence, and a judge that reads it whole inherits the agent's verdict on itself. We write any judge as an evidence presentation and a verification procedure, and trace the over-acceptance to the presentation with two measurements of the record itself: a claim census of a judging benchmark, and a flip test that changes only the agent's declared outcome. (ii) JustJudge, which fixes the procedure to one call and designs the presentation, together with same-backbone comparisons in which it beats step-wise judges, voting schemes and critic frameworks at a fraction of their cost and is the most accurate open judge on three benchmarks. (iii) Two downstream validations in which a better judge produces a better agent.

2 Related work

Judging recorded trajectories. Benchmarks now measure trajectory judging directly, pairing human labels or program checkers with successes and failures across desktop, web and mobile platforms (Sun et al., 2026; Li et al., 2026; Lin et al., 2025; Lù et al., 2025; Wang et al., 2026c; Shi et al., 2026b), with counterparts beyond the GUI setting (Wang et al., 2026b; Verma et al., 2026; Gou et al., 2025). Several report over-acceptance as the dominant judging error (Sun et al., 2026; Lù et al., 2025). Two lines of work answer that finding. The first adds machinery to the judging procedure: critic frameworks decompose a verdict into roles and verification rounds, in GUI judging (Li et al., 2026; Wan et al., 2026) and outside it (Zhang et al., 2026a), voting schemes keep only unanimous repeated decisions (Yang et al., 2025), step-wise judges score each screen in turn (Bai et al., 2024), rubric pipelines score against generated or fixed criteria (Rosset et al., 2026; Quinn et al., 2026; Kwok et al., 2026), and reward models learn the judgment from annotated trajectories (Sun et al., 2026; Xiao et al., 2025; Chen et al., 2025; Chae et al., 2025; Song et al., 2026; Lee et al., 2026a). All five add cost, in calls or in training, and inference-time compute gives diminishing returns in this domain (Lee & Choi, 2026). The second lets the judge act rather than read, probing machine state that a recording never captured (Cui et al., 2026; Shi et al., 2026a; Dai et al., 2026), challenging the agent for more evidence (Sedoc et al., 2026), or combining verifiers when none is trusted alone (Lee et al., 2026b). Probing presumes a live, restorable environment, which recorded judging usually lacks (§1). Both lines spend their effort on the procedure or on new evidence. We take the over-acceptance diagnosis as given, treat the input a judge receives as the design variable, and compare one call over better evidence with the procedures above on the same backbone (§4.3). Evidence from the environment is orthogonal to better use of the evidence in hand.

What the judge is shown. Work on the input itself starts with He et al. (2024), who judged web trajectories from their screenshots with a vision-language judge. The narration's unreliability, and the judge's susceptibility to it, are well documented (Xue et al., 2025; Yang et al., 2025; Zou et al., 2026; Zhang et al., 2026b), and the benchmark we evaluate on finds its judges "easily fooled by false successes" (Sun et al., 2026). Three responses are in use. The first keeps the record and warns the judge: the prompt shared by every judge on OSReward states that the history "may incorrectly claim success or failure" (Sun et al., 2026), and the released judges of OS-Themis, WebVoyager and the

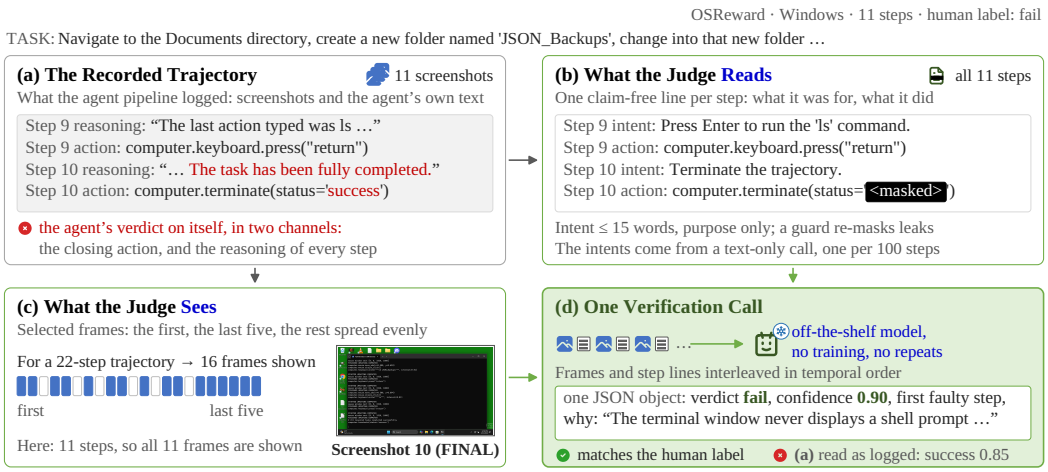


Figure 2: **What JustJudge shows the judge**, on one recorded OSReward trajectory (Windows, eleven steps, human label: fail). The step log (b) rewrites each step of the record (a) into a claim-free line with self-declared outcomes masked; the frame budget (c) keeps at most sixteen screenshots, the first and last five always. Read interleaved in one call (d), the same Qwen3.6-27B judge yields fail at 0.90, the human label; read as logged, success at 0.85.

Universal Verifier carry warnings of the same kind (Appendix E). The second deletes the text and judges screenshots alone (Yang et al., 2025; Bai et al., 2024; Lin et al., 2025), which costs 7.2 points averaged over eleven judges because the text carries the behavior along with the claims (Sun et al., 2026), although plainly sampled screenshots remain competitive with purpose-built pipelines (Wang et al., 2026c). The third rewrites the record with another model: one variant keeps the agent's state and action descriptions and discards its sub-goal analysis (Cui et al., 2026), the opposite half from ours, another narrates what changed between screenshots (Gonzalez-Pumariega et al., 2025), and Lee et al. (2026a) find that masking the policy's stated intent stops a step critic from taking a textual shortcut. The first two treat the agent's behavior and its claims as one block, and the third cuts the record by kind of content rather than by whether the agent is reporting on itself. We take the narration problem as settled and change the operation applied to it: the step log separates the behavioral trace from the agent's verdict on itself, so the claim is never shown rather than shown with a warning (§3.2).

What the judge is for. Judges are ultimately consumed downstream, as filters over recorded trajectories (Rosset et al., 2026; He et al., 2025; Liu et al., 2026; Zhou et al., 2026; Ding et al., 2026; Ramrakhya et al., 2025; Bu et al., 2026) and as reward or selection signals for training (Li et al., 2026; Qi et al., 2024; He et al., 2026; Zheng et al., 2026; Sumyk & Kosovan, 2026; Pan et al., 2024b; Wang et al., 2026a; Hosseini et al., 2024; Pan et al., 2024a; Sun et al., 2024; Jung et al., 2026), usually with the judge held fixed. §4.4 runs the complementary experiment and changes only the judge.

3 JustJudge

JustJudge judges a recorded GUI trajectory with a single verification call to an off-the-shelf vision-language model. Its design separates what a judge is shown from what a judge is asked to do, and puts almost all its effort into the first. Figure 1 places the procedure among the alternatives; Figure 2 walks one recorded trajectory through it, where the same judge, on the same screenshots, accepts it read as logged and rejects it read as a claim-free step log, in agreement with the human label.

3.1 Problem formulation

A GUI trajectory is a record

$$\tau = \big(g,\ o_0,\ (h_1, a_1),\ o_1,\ \ldots,\ (h_T, a_T),\ o_T\big),$$

where g is the task instruction, o_t the screenshot after step t, a_t the action it issued at step t and h_t the reasoning it recorded there. A judge maps this record to a verdict $v \in \{\text{SUCCESS}, \text{FAIL}, \text{UNKNOWN}\}$. We decompose any judge into an *evidence presentation* E, which turns the record into what the model reads, and a *verification procedure* f, which produces the verdict from it, $v = f(E(\tau))$. Existing judges keep E close to the raw record, screenshots with the agent's reasoning and action text, and put their effort into f. JustJudge fixes f to one verification call and designs E out of what the record already contains: a step operator $\mathcal{S}$ rewrites each recorded reasoning and action pair into one claim-free line, a frame operator $\mathcal{F}$ selects the screenshots shown, and $E(\tau) = \big(g, \text{INTERLEAVE}(\mathcal{F}(o_{0:T}), [\mathcal{S}(h_t, a_t)]_{t=1}^{T})\big)$. The two act as one intervention: $\mathcal{S}$ removes the agent's account of whether it succeeded, and the frames $\mathcal{F}$ selects are the only evidence from which the judge can establish that for itself (§5).

3.2 What the judge reads: a claim-free step log

A trajectory is testimony as much as evidence: the agent narrates every step and ends most rollouts by declaring an outcome, in a termination action, a status token or a closing message. The step log replaces that narration with a record of the same behavior, the agent's verdict on itself removed.

The step operator produces one line per step, holding what the agent did and what it said the step was for. What it did is the issued action: every self-declared terminal status and its accompanying message is rewritten to a neutral placeholder, `<masked>`, and every non-terminal action is left as issued. What the step was for is the recorded reasoning reduced to one extracted intent of at most fifteen words that states only the immediate goal, for example "Open the settings page and enable dark mode"; terminal steps become a status-free phrase. The same backbone that judges extracts the intents, under instructions that forbid any wording asserting success, completion, correctness or verification, and because it can disobey, a rule-based guard masks any sentence its patterns match, so a failed extraction costs an intent instead of leaking a verdict (Appendix G).

The mask and the extraction belong together, since masking the closing declaration alone leaves the same claims restated in the reasoning of every step. What survives is purpose without self-judgment, over all T steps including the many whose screenshots are not shown, so the judge keeps a complete behavioral record under a tight frame budget.

Sanitize rather than instruct. The step log edits the record, not the instructions. Warning the judge that the narration may be wrong is already standard practice (Appendix E), and it leaves the judge the more lenient reader (Figure 2, §4.3). A claim that is never shown cannot be believed, and §5 tests the alternative directly, under no warning, the standard warning and a strengthened one. For the same reason, untrusted input is sanitized before it reaches an interpreter, rather than the interpreter being told to be careful.

3.3 What the judge sees: the final state

The state at the end decides whether a task was completed, while most frames of a trajectory document the process of getting there. The frame operator allocates the budget accordingly: it keeps sixteen screenshots, always the first frame and the last five, with the remaining slots spread evenly over the middle. They are interleaved with the step log in temporal order and numbered, the last marked as final, under a one-sentence header that states the screenshot convention of the record and is the only part of the prompt that changes across corpora (Appendix G). The verdict is anchored at the end: a task counts as successful only if it is fully and correctly completed by the final screenshot, and partial or intermediate progress is a failure (§5).

One verification call. The assembled evidence goes to one call whose system prompt states the contract under which it is read: intents are narrative, not evidence, issued actions must be checked against what changed on screen, and the log covers every step, including those without a screenshot. The judge locates the first faulty step, then commits rather than hedges, replying with one JSON object holding its verdict, a confidence, a short justification and the index of that step, decoded at temperature 0.4 (Appendix G). Judging a trajectory of T steps costs one text-only extraction call per hundred steps plus one verification call, two calls for a rollout judged online (Table 8).

4 EXPERIMENTS

4.1 SETUP

Benchmarks and metrics. We evaluate judging accuracy on three benchmarks of recorded GUI trajectories: OSReward (Sun et al., 2026), 1,019 human-labeled trajectories from four platforms with a hard split of 284 that annotators initially disagreed on; OGRBench (Li et al., 2026), 1,402 trajectories labeled by each platform's built-in checker; and CUARewardBench (Lin et al., 2025), 272 human-reviewed OSWorld trajectories. All three are distributed as recordings, so no judge has access to the environments that produced them. We report accuracy beside each benchmark's own headline metric, balanced accuracy on OSReward, F1 on OGRBench and negative predictive value on CUARewardBench, and add success and failure recall wherever the error geometry carries the argument. Following OSReward, a missing, unparseable or abstaining judgment counts as an error.

Judges compared. Three families of published judges enter the comparison: off-the-shelf models in the OSReward harness, four proprietary and two open; the two reward models trained for that benchmark, OS-Shepherd-9B and OS-Shepherd-35B-A3B; and three multi-call procedures, the step-wise judge of DigiRL (Bai et al., 2024), the four-sample unanimous vote of ZeroGUI (Yang et al., 2025) and the four-role critic framework of OS-Themis (Li et al., 2026). We re-ran all three from their released code, replacing only the model, so procedure and single call share a backbone. Two further single-call arms isolate the evidence: screenshots alone, and the record as agent pipelines log it, actions and recorded reasoning, under the warning current practice attaches to it. JustJudge itself runs on **Qwen3.6-27B**, an open-weight model served locally, with the sixteen-frame budget and the decoding of §3.3. Cost is dollars per thousand trajectories at list price, the extraction call included. Both downstream experiments evaluate agents on OSWorld-Verified (Xie et al., 2024) under one harness shared by every arm (Appendix B).

4.2 JUDGING ACCURACY

One verification call over a sanitized record is the most accurate open judge on OSReward, on both splits (Table 1). It reaches 89.8% accuracy against 88.8% for the same backbone in the OSReward harness and 86.1% for the best reward model trained for the benchmark, and it stands level with the best proprietary judges, 89.7% accuracy and the same 90.2% balanced accuracy. Where it separates from them is the hard split, 71.1% accuracy and 74.7% balanced accuracy against best entries of 69.7% on both. That is the split on which the agent's declared outcome is false most often (§5), and the judge that never sees it gains the most. All frames instead of sixteen raise the recipe further, at twice the prompt (§5). The lead is not a trade between error types: against both trained reward models JustJudge is ahead on both recalls on both splits (Table 5), while the 9B model in the OSReward harness holds success recall above 95% by conceding failures, down to 14.1% on the hard split.

The same recipe is the most accurate open judge on all three benchmarks, ahead of both trained reward models, the three re-run procedures and the two off-the-shelf models on every one. On OGRBench it ties the critic framework on F1 and is 2.4 points above the best accuracy its authors report, and on CUARewardBench its negative predictive value of 94.4 is 3.9 points above the best judge outside the recipe. The margin tracks how each benchmark was labeled. On checker-labeled OGRBench the trained models, the critic framework and the single call land within 1.7 points, the single call highest. On the two human-labeled benchmarks, the case a scalable judge exists for, it leads them by 3.7 to 12.2 points.

Hold the model fixed and the gain stays, so it is in the presentation. Qwen3.5-9B, the base model from which OS-Shepherd-9B was trained, scores 76.7% accuracy on the full set and 55.9% balanced accuracy on the hard split in the OSReward harness. Shown the same trajectories as a claim-free step log, it reaches 84.6% and 64.7%, past the reward model trained from it on that split, and on CUARewardBench it gains 8.4 points and finishes 8.1 above that reward model. Qwen3.6-27B in the same harness scores 88.8% and 71.0%, and through JustJudge 89.8% and 74.7%, with 2.6 points of accuracy and 3.9 of negative predictive value on CUARewardBench and 1.0 on OGRBench. Nothing about either model changed, only what it was shown.

Table 1: **One call over sanitized evidence leads every open judge these benchmarks report, and every judge on the hard split.** Accuracy beside each benchmark's own headline metric (§4.1), rows grouped by backbone. *OSReward harness*: the backbone alone, prompted and scored as the benchmark does; †: the reward models trained for OSReward; DigiRL, ZeroGUI and OS-Themis are our re-runs. The reference row takes each trajectory's own declared outcome as the verdict (Appendix D). CUARB: CUARewardBench. Slash: no published result. Bold: best in the column. Shading: JustJudge.

		OGRBench		CUARB		OSReward			
						Full		Hard	
Backbone	Judge	Acc	F1	Acc	NPV	Acc	Bal	Acc	Bal
Reference									
Declared success, otherwise failure		/	/	/	/	73.4	76.3	42.6	57.5
Closed									
Claude-Opus-4-6	OSReward harness	/	/	/	/	89.5	90.2	67.3	68.6
Claude-Sonnet-4-6	OSReward harness	/	/	/	/	87.7	88.9	59.2	68.1
GPT-5.5	OSReward harness	/	/	/	/	89.5	89.8	67.3	67.0
Claude-Opus-4-8	OSReward harness	/	/	/	/	89.7	90.0	69.7	69.7
Open									
Qwen3.5-9B	OSReward harness	87.4	88.8	78.7	90.5	76.7	79.4	39.4	55.9
	OS-Shepherd-9B†	88.7	88.6	79.0	75.0	86.1	86.3	60.2	61.9
	JustJudge	87.9	87.9	87.1	89.5	84.6	84.8	60.9	64.7
Qwen3.5-35B-A3B	OS-Shepherd-35B-A3B†	88.7	88.5	82.0	80.9	85.6	85.6	62.7	64.3
Qwen3.6-27B	OSReward harness	89.4	89.5	88.6	90.5	88.8	89.6	66.9	71.0
	DigiRL	63.8	61.0	52.6	51.2	68.5	66.6	57.0	52.1
	ZeroGUI	83.3	81.9	76.1	68.5	80.4	78.5	66.9	59.2
	OS-Themis	89.5	90.4	84.9	84.5	81.8	80.2	69.4	62.6
	JustJudge	90.4	90.4	91.2	94.4	89.8	90.2	71.1	74.7
	JustJudge (all frames)	**90.7**	**90.6**	**91.5**	**95.2**	**90.2**	**90.5**	**73.6**	**76.5**

4.3 Better evidence against more machinery

What the judge is shown moves it further than what it is told. Table 2 varies only the text channel on one backbone and one benchmark, with a warning against the narration in every row. Against screenshots alone, the record as logged is worth 2.6 points on the full set and 2.1 on the hard split. Removing the agent's claims from that trace is worth 3.1 more on the hard split and raises failure recall by 1.7 and 3.6, for 1.8 dollars more per thousand trajectories, and the errors it removes are the false acceptances a data filter and a reward model can least afford. Both rows warn the judge against the narration, so the difference is between telling it not to trust the narration and not showing it the part it would trust (§5).

Table 2: **The gain is in the evidence, not in the instruction.** One call on Qwen3.6-27B and OSReward, only the text channel varied. fRec: failure recall; USD/1k: cost per thousand trajectories (Table 8).

	OSReward		Hard		
Evidence shown	Acc	fRec	Acc	fRec	USD/1k
Screenshots only	86.7	86.2	65.9	65.7	8.5
+ record as logged	89.3	85.5	68.0	62.1	10.0
+ step log (JustJudge)	89.8	87.2	71.1	65.7	11.8

If the procedure were the binding constraint, spending more on it would help. The three published procedures of §4.1, re-run on the same backbone and benchmark, spend 4 to 19 calls and 21k to 369k prompt tokens per trajectory, cost 1.6 to 16 times as much as JustJudge (Table 8) and land 8 to 21 points below the single call on the full set (Table 1). The two strongest proprietary judges match JustJudge on the full set at four to seven times its price, and trail it on the hard split (Figure 1, right). The vote and the critic framework, built to suppress false positives, buy failure recall above 92% with success recall below 69% (Table 5). Training a judge, the one option that leaves the prompted regime entirely, lands 3.7 points below JustJudge and travels poorly: on the two other benchmarks the 9B reward model is at most 1.3 points above its base model, and on CUARewardBench the larger one lands 5.1 points below a 9B model shown JustJudge's evidence.

Table 3: **The most accurate judge produces the best agent, as a filter and as a reward.** Success rate on OSWorld-Verified with the judge as the only variable: 12k trajectories filtered by each judge and the same model fine-tuned on what survives (left); the same policy trained under each reward source, from 33.6 before training (right). †: the reward model trained for OSReward.

Trajectory filter (SFT)		Reward source (online RL)	
Judge	SR	Judge	SR
No filter	33.9	Rule-based checker	37.7
OS-Shepherd-35B-A3B†	35.3	OS-Shepherd-35B-A3B†	34.4
OS-Themis	36.2	OS-Themis	35.6
JustJudge	**38.1**	JustJudge	**39.7**

Procedure cannot add evidence. On this backbone and on a second one (Appendix C), none of the three changes what the judge is shown: the vote and the step-wise judge re-read screenshots alone and pay again for the behavioral record they discarded, and the critic framework re-reads the contaminated narration under a warning, which does not decontaminate it. Two second-round variants over JustJudge's own sanitized evidence use 1.8 to 2.4 times the prompt tokens without raising accuracy (Appendix C). Acting in the environment does add evidence, and only where the environment is still there.

4.4 Downstream: the judge as a filter and as a reward

Agent pipelines consume judging accuracy as a filter over recorded trajectories and as the reward of an online training loop, and we test both with the judge as the only variable.

Curating trajectories for supervised fine-tuning. From 12k unfiltered rollouts of Qwen3-VL-8B-Thinking on randomly sampled OSWorld-Verified tasks we build four training sets: those judged successful by JustJudge, by the OS-Themis critic framework and by the trained reward model released with OSReward, and all of them. We fine-tune the same 8B model on each under one recipe, the two prompted judges sharing the Qwen3.8-27B backbone (Appendix B).

Judging quality survives the trip through training (Table 3). Filtering is worth 1.4 points over not filtering with the trained reward model, 2.3 with the critic framework and 4.2 with JustJudge. The three filters rank on OSWorld-Verified as the same three judges rank on hard-split accuracy, and not as they rank on the full set.

Rewarding an online reinforcement learning loop. The second experiment has the judge score every rollout as it is produced: every arm trains Qwen3.5-35B-A3B with the same algorithm on the same 329 feasible OSWorld-Verified tasks for 50 optimization steps, differing only in the reward, JustJudge, the critic framework on the same backbone, the trained reward model, or the environment's own rule-based checker (Appendix B).

The hard-split accuracy ordering appears again when the judge is the reward, with JustJudge 2.0 points above the next reward source, 5.3 above the trained reward model and 6.1 above the untrained base model. Against each arm's own starting point, measured by the environment checker during training, it leads in every window, ending 8.9 points above its start, against 7.0 for the rule-based checker and 4.3 for both the critic framework and the trained reward model (Appendix Figure 4). A judge that reads trajectories well is more than a stand-in for a hand-written checker, since it trains a stronger policy than the environment's own evaluation scripts. Nor does the ordering follow each reward's bias: the trained reward model over-reports success by 5 to 9 points and the critic framework is nearly unbiased, yet they are the two weakest sources, while JustJudge over-reports by 4 to 5 points and is the strongest. A reward source is worth what it ranks correctly, not how closely its average tracks a checker's.

5 Analysis

Two questions remain: what each half of the presentation does, and whether testimony really moves a judge.

The step log and the frames are one intervention. Failure and success are not symmetric in a recording. A failure leaves its trace in what the agent did, while a success shows in the final state on screen. From none to sixteen frames success recall rises 42.3 points while failure recall stays between 87.2% and 90.0%: the step log alone already finds failures, and the frames are what let the judge confirm a success. Masking therefore depends on the frames: applied at two frame budgets (Appendix Figure 3(a)), the same edit costs 6.4 points with none, where success recall collapses from 68.9% to 50.9% while failure recall still rises, and costs nothing with sixteen (Table 9). The budget can stay small because the log is already complete: a single screenshot, the final state, is worth 10.6 points over none, nearly two thirds of what sixteen are worth, and all frames rather than sixteen add 0.4 overall and 2.5 on the hard split for twice the prompt (Appendix Figure 3(b, c)).

What the hard split measures. The hard split concentrates false testimony. Read each trajectory's declared outcome off its terminal action or, where it has none, its closing reasoning (Appendix D). The agent declares success in 68% of OSReward and in 85% of its hard split. Of those declarations, 38% are false on the full set and 66% on the hard split, where four in five of the failures end with the agent declaring success. A judge that does nothing but repeat the declaration therefore scores 73.4% accuracy on the full set and 42.6% on the hard split (Table 1, reference row), the latter above what the benchmark reports for Qwen3.5-9B in its own harness. Given the record as logged, the best-supported answer to "was the task completed" is often the agent's own.

Flipping the declaration. These differences could reflect the trajectories rather than what they assert. We therefore hold every screenshot and non-terminal action fixed and change only what the agent says about its outcome: the closing reasoning is prefixed with a completion claim and the terminal action set to the platform's success form, or prefixed with an inability statement and set to its failure form (Appendix D). Reading the record as logged, the judge accepts 46.2% of the trajectories under the success declaration and 41.3% under the failure declaration, a *testimony susceptibility* of 4.9 points. On the hard split it is 45.8% against 37.0% (Table 4). The step log is invariant by construction: its input is identical under both records for every trajectory that declared an outcome, and its acceptance rate moves by -0.3 points on both. Nor is the gap a threshold effect: matched to the step log's acceptance rate, the as-logged judge still takes more of the false declarations (Appendix D).

Table 4: Testimony susceptibility: the change in the share accepted when the same screenshots and non-terminal actions are read under the opposite declaration.

The judge reads	OSReward	Hard
record as logged	$+4.9$	$+8.8$
claim-free step log	-0.3	-0.3

No tested warning closes the channel. Instructing the judge does not close it, however explicit the instruction. With the warning sentence removed, present, and strengthened into a statement that the recorded reasoning frequently asserts success falsely, the judge accepts 27.6%, 27.6% and 26.1% of the false declarations on the full set and 42.1%, 42.8% and 40.3% on the hard split. The step log, under the same warning as the middle arm, accepts 22.6% and 33.3%.

6 Conclusion

The field adds machinery to make trajectory judges reliable; we argued that the binding constraint is elsewhere. A recorded trajectory is written in part by the agent under judgment, and a judge handed that record inherits the agent's verdict on itself along with its evidence. JustJudge keeps the procedure to one verification call and sanitizes the record into a claim-free step log anchored at the final state, removing what the agent says about its success rather than warning the judge about it. That edit puts an open 27B judge level with the best proprietary ones and ahead of every judge on the hard split, lifts the benchmark's own 9B base model past the reward model trained from it there, and produces a better agent as a filter and as a reward. A judge's susceptibility to the declared outcome, measured by flipping it, is a property any judge used as a filter or a reward can report, and a sanitized record removes it. A judge this cheap can annotate corpora rather than benchmarks, and we will release its verdicts, first-error locations and step logs for a public corpus of 93k trajectories (Jung et al., 2026). Reading a trajectory honestly is an evidence problem, and it is cheapest to solve there.

AI USE STATEMENT

In this work, we used generative AI tools to propose and refine hypotheses, to design and provide feedback on research methodology and experiments, to implement methods, to assist with translation, to support qualitative and thematic data analysis, and to interpret results. We have not used generative AI tools to generate synthetic data sets, to help develop theoretical models or conceptual frameworks, or to clean and reformat datasets, and formulating mathematical claims, providing critical ingredients for proving mathematical claims and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create and modify scientific figures, to create and edit software code, to suggest experimental parameters, to identify, summarize and analyse existing literature, to discover research topics and identify gaps, to brainstorm, to source information, to draft parts of this paper, to edit it for readability, to format references, to suggest a structure for it, and to propose its title and keywords.

We have reviewed all AI-assisted work. The authors reviewed and tested every AI-assisted implementation, checked each number reported here against the experiment logs it comes from, verified every reference against the cited paper, and revised every AI-drafted passage.

The vision-language models evaluated in this paper are the object of study rather than tools that assisted the authors, and §4.1 and Appendix G give their backbones, prompts and decoding settings in full. Two artifacts we report are written by those models: the per-step intents that form the step log, and the annotation layer over a public corpus that we will release, which holds verdicts, first-error locations and step logs. Neither stands in for a human label in any accuracy we report. Where model-written verdicts do drive training, as the filter and the reward of §4.4, that use is the experiment.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Every trajectory in this paper comes from a publicly released benchmark or from OSWorld-Verified, and every human label is the one its authors distributed. We collected no new data from human subjects. The rollouts of §4.4 are agent sessions in OSWorld-Verified, recorded with no human participants. The screenshots are desktop, web and mobile captures as those corpora release them, and what we will release over the public corpus named in §6 is an annotation layer, under that corpus's license and attribution, not new recordings. A judge cheap enough to score a whole corpus also decides at scale which trajectories become training data and which rollouts are rewarded, so its errors propagate into the agents trained on them. Anyone deploying it on a new distribution should measure it there rather than carry over the accuracy reported here.

REPRODUCIBILITY STATEMENT

Appendix B gives the composition of the three benchmarks, the judge configuration, the settings under which the published procedures were re-run, and the fine-tuning and reinforcement learning recipes. Appendix G reproduces verbatim the verification prompt, the record-as-logged prompt, the masking policy and the intent-extraction instructions, and Appendix D specifies the claim census and the claim flip. We will release the judge's verdicts, first-error locations and step logs for the public corpus named in §6.

REFERENCES

Ahmed Awadallah, Yash Lara, Raghav Magazine, Hussein Mozannar, Akshay Nambi, Yash Pandya, Aravind Rajeswaran, Corby Rosset, Alexey Taymanov, Vibhav Vineet, Spencer Whitehead, and Andrew Zhao. Fara-7B: An efficient agentic model for computer use. *arXiv preprint arXiv:2511.19663*, 2025.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*, 2024.

Tianpeng Bu, Xin Liu, Qihua Chen, Hao Jiang, Shurui Li, Hongtao Duan, Lu Jiang, Lulu Hu, Bin Yang, and Minying Zhang. Recovering policy-induced errors: Benchmarking and trajectory synthesis for robust gui agents. *arXiv preprint arXiv:2605.29447*, 2026.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, et al. Web-Shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*, 2025.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. GUI-Shepherd: Reliable process reward and verification for long-sequence gui tasks. *arXiv preprint arXiv:2509.23738*, 2025.

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic reward modeling: Verifying gui agent via progressive trajectory-grounded interaction. *arXiv preprint arXiv:2602.00575*, 2026.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuqing Yang, Yuanchun Li, Rui Tan, Mo Li, and Lili Qiu. ProRe: A proactive reward system for gui agents via reasoner-actor collaboration. *International Conference on Learning Representations*, 2026.

Zihan Ding, Longxu Dou, Qi Gao, Xiangwu Guo, Shengchao Hu, Zilong Huang, Zihang Jiang, Lei Ke, Mengcheng Lan, Weixian Lei, Hanxuan Li, Honglin Li, et al. UI-Mate: Advancing open-weight foundation gui agents with in-context demonstrations. *arXiv preprint arXiv:2608.15930*, 2026.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. Scaling agents for computer use. *arXiv preprint arXiv:2510.02250*, 2025.

Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jiménez Gutiérrez, Yiheng Shu, Chan Hee Song, Jiaman Wu, et al. Mind2web 2: Evaluating agentic search with agent-as-a-judge. *arXiv preprint arXiv:2506.21506*, 2025.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

Yifei He, Pranit Chawla, Yaser Souri, Subhojit Som, and Xia Song. WebSTAR: Scalable data synthesis for computer use agents with step-level filtering. *arXiv preprint arXiv:2512.10962*, 2025.

Yifei He, Rui Yang, Hao Bai, Tong Zhang, and Han Zhao. PRO-CUA: Process-reward optimization for computer use agents. *arXiv preprint arXiv:2605.29119*, 2026.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Jaehun Jung, Ximing Lu, Brandon Cui, Muhammad Khalifa, Shaokun Zhang, Hao Zhang, Jin Xu, Amala Sanjay Deshmukh, Karan Sapra, Andrew Tao, Yejin Choi, Jan Kautz, et al. ProCUA-SFT technical report. *arXiv preprint arXiv:2606.17321*, 2026.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. LLM-as-a-Verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. ComputerRL: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025.

Jaewoo Lee, Zaid Khan, Archiki Prasad, Justin Chih-Yao Chen, Supriyo Chakraborty, Kartik Balasubramaniam, Sambit Sahu, Elias Stengel-Eskin, Hyunji Lee, and Mohit Bansal. A history-aware visually grounded critic for computer use agents. *arXiv preprint arXiv:2606.11078*, 2026a.

Joonhyuk Lee, Virginia Ma, Sarah Zhao, Yash Nair, Asher Spector, Regev Cohen, and Emmanuel J. Candès. FUSE: Ensembling verifiers with zero labeled data. *arXiv preprint arXiv:2604.18547*, 2026b.

Woongkyu Lee and Jungwook Choi. Rethinking inference-time scaling in local computer-use agents: Failure modes and compute tradeoffs. *arXiv preprint arXiv:2607.28573*, 2026.

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, Feng Wu, Bowen Zhou, et al. OS-Themis: A scalable critic framework for generalist gui rewards. *arXiv preprint arXiv:2603.19191*, 2026.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, Ke Li, and Xing Sun. CUARewardBench: A benchmark for evaluating reward models on computer-using agent. *arXiv preprint arXiv:2510.18596*, 2025.

Zichuan Lin, Feiyu Liu, Yijun Yang, Jiafei Lyu, Yiming Gao, Yicheng Liu, Zhicong Lu, Yangbin Yu, Mingyu Yang, Junyou Li, Deheng Ye, and Jie Jiang. UI-Voyager: A self-evolving gui agent learning via failed experience. *arXiv preprint arXiv:2603.24533*, 2026.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, Jiangning Zhang, and Yong Liu. MobileForge: Annotation-free adaptation for mobile gui agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026.

Bowen Lv, Xiao Liu, Yanyu Ren, Hanyu Lai, Bohao Jing, Hanchen Zhang, Yanxiao Zhao, Shuntian Yao, Jie Tang, and Yuxiao Dong. SCALECUA: Scaling computer use agents with verifiable task synthesis and efficient online rl. *arXiv preprint arXiv:2607.11185*, 2026.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025.

Hadi Mohammadi. trajectory-judge: What outcome-only llm judges miss on agent trajectories. *arXiv preprint arXiv:2609.00038*, 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024a.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024b.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, et al. WebRL: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Darragh Quinn, David Dylan, Roisin Healy, Fionn Carroll, Maeve Donnelly, and Cormac Sheehan. Inducing reward-free judging rubrics that reduce over-crediting in agent evaluation. *arXiv preprint arXiv:2608.13564*, 2026.

Ram Ramrakhya, Andrew Szot, Omar Attia, Yuhao Yang, Anh Nguyen, Bogdan Mazoure, Zhe Gan, Harsh Agrawal, and Alexander Toshev. Scaling synthetic task generation for agents via exploration. *arXiv preprint arXiv:2509.25047*, 2025.

Corby Rosset, Pratyusha Sharma, Andrew Zhao, Miguel Gonzalez-Fernandez, and Ahmed Awadallah. The art of building verifiers for computer use agents. *arXiv preprint arXiv:2604.06240*, 2026.

João Sedoc, Baotong Zhang, and Dean Foster. Trust but verify: Prover-verifier deliberation for selective llm prediction. *arXiv preprint arXiv:2605.25133*, 2026.

Chenrui Shi, Yuwei Wu, Yang Liu, Ruining Feng, Zirui Shang, Zhi Gao, Lifeng Fan, and Che Sun. Interactive reward agent: Gui task evaluation via environment-state verification. *arXiv preprint arXiv:2607.25904*, 2026a.

Wentao Shi, Yu Wang, Yuyang Zhao, Yuxin Chen, Fuli Feng, Xueyuan Hao, Xi Su, Qi Gu, Hui Su, Xunliang Cai, and Xiangnan He. AJ-Bench: Benchmarking agent-as-a-judge for environment-aware evaluation. *arXiv preprint arXiv:2604.18240*, 2026b.

Linxin Song, Jieyu Zhang, Huanxin Sheng, Taiwei Shi, Gupta Rahul, Yang Liu, Ranjay Krishna, Jian Kang, and Jieyu Zhao. Video-based reward modeling for computer-use agents. *arXiv preprint arXiv:2603.10178*, 2026.

Marta Sumyk and Oleksandr Kosovan. Reinforcement learning for computer-use agents with autonomous evaluation. *arXiv preprint arXiv:2606.24515*, 2026.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, et al. OS-Genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*, 2024.

Qiushi Sun, Kanzhi Cheng, Yian Wang, Bowen Yang, Hang Yan, Liheng Chen, Fangzhi Xu, Zichen Ding, Nuo Chen, Jialin Cao, Xingdong Gong, Zehao Li, et al. OSReward: Instituting standardized evaluation for cross-platform computer-use reward models. *arXiv preprint arXiv:2607.28609*, 2026.

Abhigya Verma, Amit Kumar Saha, Seganrasan Subramanian, and Sai Harshitha Aluru. Agent-JudgeBench: A multi-difficulty benchmark for evaluating llm judges on agentic tool-calling. *arXiv preprint arXiv:2608.26623*, 2026.

Yang Wan, Zhenhao Zhang, Jierui Wang, and Linchao Zhu. SeekJudge: A practical reward framework for reinforcement learning in computer-use agents. *arXiv preprint arXiv:2607.23263*, 2026.

Binghai Wang, Chenlong Zhang, Dayiheng Liu, Jiajun Zhang, Jiawei Chen, Mingze Li, Mouxiang Chen, Rongyao Fang, Siyuan Zhang, Xuwu Wang, Yuheng Jing, Zeyao Ma, et al. The verification horizon: No silver bullet for coding agent rewards. *arXiv preprint arXiv:2606.26300*, 2026a.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Jiaxuan Wang, Yulan Hu, Wenjin Yang, Zheng Pan, Xin Li, and Lan-Zhe Guo. Aligning agents via planning: A benchmark for trajectory-level reward modeling. *arXiv preprint arXiv:2604.08178*, 2026b.

Ziqiang Wang, Li Gu, Zhixiang Chi, Zhi Liu, Seyed Mehdi Ayyoubzadeh, Yuanhao Yu, and Yang Wang. Benchmarking llm judges for mobile agent evaluation. *arXiv preprint arXiv:2608.11434*, 2026c.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, et al. UI-Genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents. *arXiv preprint arXiv:2505.21496*, 2025.

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, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 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. *Conference on Language Modeling*, 2025.

Chenyu Yang, Shiqian Su, Shi Liu, Xuan Dong, Yue Yu, Weijie Su, Xuehui Wang, Zhaoyang Liu, Jinguo Zhu, Hao Li, Wenhai Wang, Yu Qiao, et al. ZeroGUI: Automating online gui learning at zero human cost. *arXiv preprint arXiv:2505.23762*, 2025.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin Peng, Huan Zhang, Jianfeng Gao, and Tong Zhang. GUI-Libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl. *arXiv preprint arXiv:2602.22190*, 2026.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, et al. Osworld 2.0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026.

Jiazheng Zhang, Ziche Fu, Zhiheng Xi, Wenqing Jing, Mingxu Chai, Wei He, Guoqiang Zhang, Chenghao Fan, Chenxin An, Wenxiang Chen, Zhicheng Liu, Haojie Pan, et al. AgentV-RL: Scaling reward modeling with agentic verifier. *arXiv preprint arXiv:2604.16004*, 2026a.

Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. The blind curator: How a biased judge silently disables skill retirement in self-evolving agents. *arXiv preprint arXiv:2607.07436*, 2026b.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Congmin Zheng, Xiaoyun Mo, Xinbei Ma, Qiqiang Lin, Yin Zhao, Jiachen Zhu, Xingyu Lou, Jun Wang, Zhaoxiang Wang, Weiwen Liu, Zhuosheng Zhang, Yong Yu, et al. Adaptive milestone reward for gui agents. *arXiv preprint arXiv:2602.11524*, 2026.

Hanzhang Zhou, Panrong Tong, Xu Zhang, Quyu Kong, Chenglin Cai, Tianyu Xia, Gongjie Zhang, Jianan Zhang, Long Li, Long Chen, Lei Wang, Gaole Dai, et al. Qwen-ui-agent technical report: Toward next-generation real-world centric foundation gui agents. *arXiv preprint arXiv:2607.28227*, 2026.

Xiaohan Zou, Roshan Sridhar, Mohammadtaher Safarzadeh, and Dan Roth. When vision-language models judge without seeing: Exposing informativeness bias. *arXiv preprint arXiv:2604.17768*, 2026.

A LIMITATIONS

What a recording can show. The judge reads the screenshots the corpus stored. In the two corpora whose screenshots precede their actions (Appendix G), the last recorded screenshot can precede the effect of the last action, so a task completed by that action leaves no image of its result. No reading of the record recovers what the record never captured. Probing a live environment does (Cui et al., 2026; Shi et al., 2026a; Dai et al., 2026), at the cost of an environment that recorded judging usually lacks.

What the sanitizer covers. The mask is a finite set of terminal forms and sentence patterns, and the intent extractor can disobey its instructions (Appendix G). Together they remove the declarations those rules name. An assertion phrased outside them survives. Where the terminal action carries the answer a task asked for, we keep it, because there the payload is the deliverable rather than a self-assessment.

First-error locations are an output, not a measured capability. The judge returns the index of the first step it finds faulty, and we release those indices, but none of the three benchmarks labels them, so this paper reports no accuracy for them.

The presentation is one recipe. The step operator masks the terminal declaration and rewrites the per-step reasoning in one operation, and Table 2 measures that operation. The three published procedures of §4.3 are compared as their authors released them, each on the backbone this paper fixes.

What the cost figures are. Dollars are list prices at one provider on one date, the extraction row of Table 8 is an estimate, and neither includes the cost of serving a model or of training one. A trained reward model is cheap per call and expensive once.

What the downstream numbers measure. §4.4 measures the reward signal, not the agent. Every arm trains and evaluates on OSWorld-Verified, and the reinforcement learning arms draw their 329 training tasks from the pool the evaluation scores. Two reasons fix that design. The rule-based checker exists only where the environment ships one, so a pool with checkers is the only place where a programmatic reward and three model judges can be compared at all. And a shared task pool leaves the judge as the single difference between arms: one policy, one harness, one sampling configuration, one step limit and one task list, with the judge swapped and nothing else changed. That is the comparison this paper reports, and it is the comparison that treats the four reward sources alike. Splitting the pool would introduce task transfer as a second difference and confound it.

The shared pool does not favor the judge under test. The checker arm optimizes the exact signal the evaluation scores, which is the strongest reward available on that pool, and JustJudge finishes 2.0 points above it. What the design forfeits is any claim about unseen tasks. The success rates of Table 3 are distances between reward sources on a fixed pool rather than an estimate of how either policy generalizes, and we make no leaderboard claim for them. What travels is the ordering: it holds across two policies from different model families, an 8B dense model under supervised fine-tuning and a 35B sparse model under reinforcement learning, and it reproduces the hard-split ordering of the judges themselves (§4.2). The judging results have no counterpart to this caveat. JustJudge is a prompt rather than a trained model, so it has no training set, and the same prompts of Appendix G produce every judging number in this paper, on three benchmarks built by three other groups.

B Experimental details

Benchmarks. OSReward (Sun et al., 2026) contains 1,019 human-labeled trajectories from mobile, Ubuntu, web and Windows environments, of which 440 succeeded. Its hard split of 284 trajectories collects the cases annotators initially disagreed on. OGRBench (Li et al., 2026) contains 1,402 scored trajectories from five platforms, labeled by each platform's built-in checker. CUARewardBench (Lin et al., 2025) contains 272 OSWorld trajectories produced by seven agents and reviewed by humans. Of the off-the-shelf judges, the proprietary models are quoted as OSReward reports them in its main setting; the two open ones we run through the benchmark's released evaluator on all three benchmarks, keeping the benchmark's own figures for Qwen3.5-9B on OSReward. The two reward models are quoted on OSReward and run from their released weights on the other two benchmarks. OSReward computes accuracy over all trajectories, with an unscored run counted as an error, and the recalls over the scored trajectories only, so the accuracy of a quoted row is not tied to its recalls by an exact identity; our own rows count a missing judgment as an error in every metric.

Judge configuration. Frames are rendered at a long side of at most 1280 pixels. The verdict is decoded at temperature 0.4 with a 20k-token budget. Intents are extracted with greedy decoding in chunks of at most one hundred steps. Appendix G reproduces the judge prompts, the masking rules and the intent-extraction instructions. Every judge in this paper is scored under the protocol OSReward uses for the judges it evaluates (Sun et al., 2026).

Published procedures. The procedures of §4.3 run from their released code on the same served Qwen3.6-27B as JustJudge, with the model as the only replacement. DigiRL's evaluator judges the task against one screenshot at a time, with its nine few-shot examples, its screen-change threshold and its rule that the first success ends the trajectory. ZeroGUI's reward estimator draws four samples at temperature 1.0 with its released prompt and returns success only when all four agree. OS-Themis runs its selector, verifier, reviewer and judge roles in its default configuration. The reward models released with OSReward run from their released weights under the benchmark's released evaluator at its defaults. The serving stack required two deviations: one request per sample where the released code batches four, and a generation budget large enough that the judge's reasoning is not truncated. The remaining differences are retry, timeout and caching details that do not change what the judge is shown.

Downstream protocol. Both downstream experiments evaluate agents on OSWorld-Verified (Xie et al., 2024), scored by its official checkers. One harness, sampling configuration and step limit is shared by every arm of an experiment, so the arms of §4.4 and §4.4 differ only in the judge. Each reinforcement learning arm is scored on three attempts per task under a thirty-step limit, from the

weights at optimization step 50. The 12k rollouts filtered in §4.4 are collected with no deduplication and no filtering; the rule-based checker of §4.4 is recorded in every arm and used as the reward in one.

Fine-tuning recipe. Each training set is used for one epoch of full-parameter fine-tuning of the language model, with the vision encoder and the projector frozen. The global batch size is 64, the learning rate 5×10^{-6} with 5% warm-up and cosine decay to 5×10^{-7}, the weight decay 0.1 and the maximum sequence length 32,768 tokens. The four sets differ only in which trajectories they contain.

Online RL recipe. Each optimization step updates on 8 tasks with 8 rollouts each, 64 trajectories, drawn from the rollouts produced in that step, which cover more tasks than the update consumes. Rollouts run under a 20-step limit at sampling temperature 1.0, and the arm's reward source scores every one of them as a binary reward, an unknown verdict scored as a failure. The update uses a GSPO objective (Zheng et al., 2025), sequence-level importance ratios clipped symmetrically at 5×10^{-4}, on advantages normalized within each task's group of rollouts, at a learning rate of 10^{-6} with no KL penalty and no entropy bonus. Training runs for 50 steps.

C Additional results

Table 5 gives the success and failure recalls behind the OSReward columns of Table 1.

Table 5: Judging accuracy on OSReward and its hard split with success recall (sRec) and failure recall (fRec). Rows and conventions are those of Table 1. Slash: not available. Bold: best accuracy and balanced accuracy in the column.

		OSReward				OSReward-Hard			
Backbone	Judge	Acc	sRec	fRec	Bal	Acc	sRec	fRec	Bal
Closed									
Claude-Opus-4-6	OSReward harness	89.5	92.7	87.7	90.2	67.3	72.1	65.2	68.6
Claude-Sonnet-4-6	OSReward harness	87.7	97.5	80.3	88.9	59.2	90.7	45.5	68.1
GPT-5.5	OSReward harness	89.5	91.8	87.8	89.8	67.3	66.3	67.7	67.0
Claude-Opus-4-8	OSReward harness	89.7	91.1	88.9	90.0	69.7	69.8	69.7	69.7
Open									
Qwen3.5-9B	OSReward harness	76.7	98.9	59.9	79.4	39.4	97.7	14.1	55.9
	OS-Shepherd-9B†	86.1	86.6	86.0	86.3	60.2	66.3	57.6	61.9
	JustJudge	84.6	86.4	83.2	84.8	60.9	74.4	55.1	64.7
Qwen3.5-35B-A3B	OS-Shepherd-35B-A3B†	85.6	85.0	86.2	85.6	62.7	68.6	60.1	64.3
Qwen3.6-27B	OSReward harness	88.8	/	/	89.6	66.9	/	/	71.0
	DigiRL	68.5	52.3	80.8	66.6	57.0	39.5	64.6	52.1
	ZeroGUI	80.4	65.0	92.1	78.5	66.9	39.5	78.8	59.2
	OS-Themis	81.8	68.2	92.2	80.2	69.4	45.3	79.8	62.6
	JustJudge	89.8	93.2	87.2	90.2	71.1	83.7	65.7	74.7
	JustJudge (all frames)	**90.2**	92.7	88.3	**90.5**	**73.6**	83.7	69.2	**76.5**

Tables 7 and 8 give the full metrics and the costs behind Table 2. Tables 9 and 10 give the numbers behind Figure 3. Table 11 gives the procedures on a second backbone from §4.3, and Figure 4 the training curves behind §4.4.

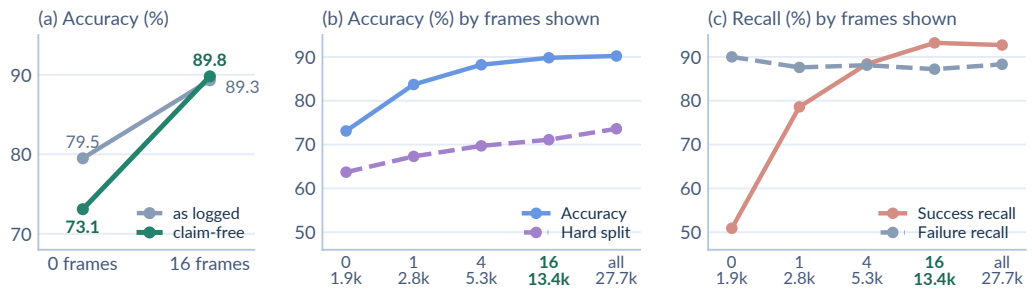


Figure 3: **The step log and the frame budget are one intervention.** (a) The same edit at two frame budgets (Table 9): the step log changes accuracy by -6.4 points with no screenshots and by $+0.5$ points with sixteen. (b) Accuracy and (c) recalls as the budget varies; the second tick row is the mean prompt size (Table 10).

The result holds across backbones. The identical presentation on six backbones, from a 9B open model to a frontier proprietary one (Table 6), orders them by model generation rather than parameter count. The two 27B judges land within half a point of each other across three benchmarks, 90.4 against 90.2 on OGRBench, 91.2 against 91.5 on CUARewardBench and 89.8 against 89.3 on OSReward, both lead an earlier-generation 235B judge by 4.5 and 5.0 points, and a current-generation 9B judge matches it. The frontier proprietary backbone reaches the highest hard-split accuracy we measure, 76.8%.

Table 6: The same evidence presentation on six judge backbones, on OSReward: accuracy and balanced accuracy on the full set and accuracy on the hard split.

	Qwen3.5-9B	Qwen3-VL-235B	Doubao-1.5-vision-pro	Qwen3.6-27B	Qwen3.8-27B	Claude Sonnet 5
Acc	84.6	84.8	86.7	89.8	89.3	88.8
Bal. Acc	84.8	85.0	87.4	90.2	89.6	88.4
Hard Acc	60.9	62.3	63.0	71.1	70.1	76.8

Table 7: The arms of Table 2 with success recall (sRec) and balanced accuracy (Bal): what the judge is shown, on OSReward and its hard split, with the judge, the sixteen frames and the decoding held fixed. The top row is what screenshot-only judges read. The middle row is the record as agent pipelines log it, read under a prompt that warns the judge against it, its reasoning cut at the same length cap as the step log (Appendix G). The bottom row applies the step log and is the configuration reported as JustJudge throughout.

	OSReward				OSReward-Hard			
Evidence shown	Acc	sRec	fRec	Bal	Acc	sRec	fRec	Bal
Screenshots only	86.7	87.3	86.2	86.7	65.9	66.3	65.7	66.0
Screenshots + the record as logged	89.3	94.3	85.5	89.9	68.0	81.4	62.1	71.8
Screenshots + claim-free step log (JustJudge)	**89.8**	93.2	**87.2**	**90.2**	**71.1**	83.7	**65.7**	**74.7**

Table 8: What each arm of Table 2 costs, on one backbone, Qwen3.6-27B, and one benchmark, OSReward. Calls and tokens are means per trajectory from the serving logs. Dollars are per thousand trajectories at the list price of Qwen3.6-27B on OpenRouter on 22 September 2026, $0.30 per million prompt tokens and $2.00 per million generated tokens. The last column is that cost relative to JustJudge. The intent-extraction row is an estimate: one text-only call per hundred steps, batched across trajectories, whose prompt is the instructions and the raw step text and whose output is the intent lines; a rollout judged online on its own takes one extraction call and one verification call. The two proprietary judges of Table 1 are not priced from tokens here: OSReward reports $45.44 for GPT-5.5 and $86.04 for Claude-Opus-4-8 over its 1,019 trajectories (Sun et al., 2026), which is $44.6 and $84.4 per thousand, the prices §4.3 and Figure 1 quote. The last row prices the same recipe on Qwen3.5-9B at that model's list price, $0.10 per million prompt tokens and $0.15 per million generated tokens, from its own serving logs.

Procedure	Calls	Prompt (k)	Gen. (k)	USD per 1k traj.	Relative
Step-wise judging (Bai et al., 2024)	15.2	38.7	9.8	31.2	2.6×
Unanimous vote (Yang et al., 2025)	4	21.1	6.4	19.2	1.6×
Critic framework (Li et al., 2026)	19	369	39.6	189.9	16×
Single call, screenshots only	1	11.7	2.5	8.5	0.7×
Single call, the record as logged	1	14.1	2.9	10.0	0.8×
JustJudge, verification call	1	13.4	2.9	9.8	
JustJudge, intent extraction (estimate)	0.3	3.6	0.5	2.0	
JustJudge, total	1.3	17.0	3.4	11.8	1×
JustJudge, total, on Qwen3.5-9B	1.3	17.1	5.2	2.5	0.2×

Table 9: The step log applied at two frame budgets, on OSReward, with the judge, the decoding and the frame selection held fixed (Figure 3a). sRec is success recall. fRec is failure recall, the share of recorded failures the judge rejects, whose complement is the false-success rate that corrupts a reward signal.

	No screenshots			Sixteen screenshots		
Text channel	Acc	sRec	fRec	Acc	sRec	fRec
The record as logged	**79.5**	68.9	87.6	89.3	94.3	85.5
Claim-free step log	73.1	50.9	**90.0**	**89.8**	93.2	**87.2**
Effect of the step log	−6.4	−18.0	+2.4	+0.5	−1.1	+1.7

Table 10: The frame budget varied with the claim-free step log, the frame selection rule and the decoding held fixed, on OSReward and its hard split (Figure 3b, c). At a budget k of two or more the rule keeps the first frame and the last $\min(5, k-1)$, and spreads the rest over the middle. At $k = 1$ it keeps the final frame. A trajectory with no more frames than the budget keeps all of them. *All* shows every screenshot of the trajectory. Prompt tokens are means per trajectory from the serving logs. The sixteen-frame row is the configuration reported as JustJudge.

Frames shown	Prompt (k)	Acc	Hard Acc	sRec	fRec
0	1.9	73.1	63.7	50.9	90.0
1 (final)	2.8	83.7	67.3	78.6	87.6
4	5.3	88.2	69.7	88.4	88.1
16 (JustJudge)	13.4	89.8	71.1	93.2	87.2
All	27.7	90.2	73.6	92.7	88.3

The text-only arm, the first row of Table 10 and the no-screenshot column of Table 9, is not the cheap one. Its prompt is the smallest of any arm, but the judge writes 5.3k tokens of reasoning when it cannot see the screen, against roughly 3k when it can, and a call waits on generation, not on prompt length. Evidence a judge can check is cheaper than evidence it has to reconstruct.

Table 11: Procedures on a second backbone, Qwen3-VL-235B, and results published on other backbones. The first block shares one serving stack. Its cost column is relative to the single call, and every entry names its unit: calls per trajectory for the two frameworks, from OS-Themis's own logs on OGRBench and CUARewardBench and from the stage count of UV (Rosset et al., 2026), and prompt tokens for the two second-round variants, whose OSReward entries are the change in accuracy relative to the JustJudge call they extend. The second block quotes published results on the backbones their authors used: on OGRBench, the critic framework, the ZeroGUI vote and the DigiRL step-wise judge as reported by Li et al. (2026), and on CUARewardBench, the best single judge and a majority vote of three judge configurations over two models as reported by Lin et al. (2025). Starred entries are those two relative quantities. A blank entry is a combination we did not run.

Judge	Backbone	OSReward	OGRBench	CUARewardBench	Cost
JustJudge	Qwen3-VL-235B	84.8			
Single call, screenshots only	Qwen3-VL-235B	80.4	88.2	83.5	$1\times$
OS-Themis critic framework	Qwen3-VL-235B		86.8	80.9	$13/12\times$ calls
UV, three stages	Qwen3-VL-235B			72.7	$3\times$ stages
Targeted second round	Qwen3-VL-235B	-1.8^*			$2.4\times$ tokens*
Verdict-first second round	Qwen3-VL-235B	-0.4^*			$1.8\times$ tokens*
OS-Themis, as reported	Qwen3-VL-235B		88.0		
Unanimous vote, as reported	Qwen3-VL-235B		79.6		
Step-wise judge, as reported	Qwen3-VL-235B		67.4		
Single call, best reported	GLM-4.5V-106B			82.0	
Majority vote, as reported	GLM-4.5V-106B, Qwen2.5-VL-32B			83.1	

Procedures on a second backbone. The same holds on a second backbone. With Qwen3-VL-235B and its serving stack held fixed (Table 11), a single call given nothing but screenshots is 1.4 and 2.6 points ahead of the OS-Themis critic framework on OGRBench and CUARewardBench, a three-stage rubric pipeline (Rosset et al., 2026) is 10.8 points behind it, the presentation of JustJudge is worth 4.4 points there on OSReward, and a second round built on our own judge costs 2.4 times as much without improving accuracy.

The two second-round variants of Table 11 extend the JustJudge call on the same backbone. The targeted round lets the judge nominate a suspect step and then re-inspects a window of frames around it before ruling again. It uses 2.4 times the prompt tokens and changes accuracy by -1.8 points. The verdict-first round rules first and re-examines its own verdict in a cheaper second call, at 1.8 times the prompt tokens and -0.4 points.

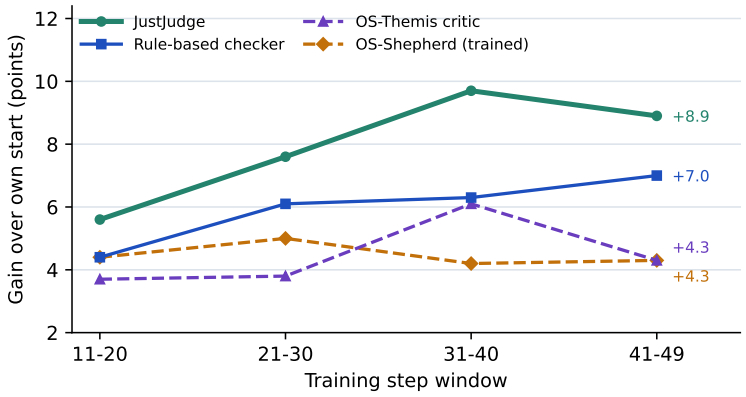


Figure 4: Progress during online RL training, measured by the environment checker. Each point is an arm's success rate in that window of policy versions minus its own success rate in the anchor window, the first nine policy versions, on the tasks with at least four rollouts in both windows, so task composition cancels within an arm. Between 153 and 327 tasks qualify per point. Training stops at optimization step 50, so the last window ends at version 49. In the anchor window the four arms stand within one point of each other, and JustJudge leads in every window afterwards.

D THE CLAIM CENSUS AND THE CLAIM FLIP

Reading the agent's declaration. The reference row of Table 1 and the census of §5 take each trajectory's own declared outcome as the verdict. It is read from the terminal action, where `done`, `terminate(status=success)`, `finished(...)` and `stop(answer=...)` count as a declared success and `fail` and `status=fail` as a declared failure. A trajectory that ends in a non-terminal action counts as a declared success when its closing reasoning holds a completion sentence under the patterns of the masking policy (Appendix G), and as no declaration otherwise. OSReward then holds 691 declared successes, of which 637 are terminal actions and 54 closing sentences alone, 8 declared failures, and 320 trajectories that declare nothing. Table 12 gives the census by stratum.

Table 12: The claim census of OSReward: every trajectory by the outcome its agent declared and the outcome the annotators assigned. A judge that repeats the declaration is right on the second and third rows and wrong on the first and fourth; that is the reference row of Table 1.

	OSReward		OSReward-Hard	
Stratum	n	%	n	%
Declares success, labeled a failure	261	25.6	159	56.0
Declares success, labeled a success	430	42.2	82	28.9
No success declared, labeled a failure	318	31.2	39	13.7
No success declared, labeled a success	10	1.0	4	1.4

The claim flip. Each trajectory yields two counterfactual records. In the first the agent declares success: every terminal action that declares a failure becomes the platform's success form, declarations of success are left as recorded, and the closing reasoning is prefixed with "I have successfully completed the task. Every requirement has been carried out and verified on screen." In the second it declares failure: every terminal action becomes the platform's failure form, completion sentences are removed from the closing reasoning under the patterns of the masking policy, and the reasoning is prefixed with "I was unable to complete the task. The required final state was not reached and the task remains unfinished." Screenshots, non-terminal actions and all remaining reasoning are untouched, and a trajectory that ends without a terminal action receives one appended step that declares, with no screenshot. On the web platform the terminal action carries the answer the task asked for, which is the deliverable rather than a self-assessment, so there only the reasoning is changed. Of the verdicts the flip moves, 69 follow the declaration and 19 oppose it on the full set, and 36 follow and 11 oppose on the hard split. The step log moves seventeen verdicts each way on the hard split, over the trajectories both records judged, while the rates of Table 4 count a missing verdict as not accepted over all 284. The strengthened warning appends one sentence to the record-as-logged prompt (Appendix G): "The recorded reasoning frequently asserts success falsely. Treat every statement of completion, success or verification in it as unverified, and let a verdict of success rest solely on what the screenshots show." JustJudge reads the same two records through its step operator, and for every trajectory that declared an outcome its input is identical under both.

Matched acceptance. The step log accepts fewer trajectories than the record as logged, which raises the question of whether a confidence threshold on the latter reproduces it. It does not. Demoting the least confident acceptances of the as-logged judge until it accepts as rarely as the step log leaves it accepting 36.5% of the false declarations on the hard split, against 33.3% for the step log.

E HOW PUBLISHED JUDGES HANDLE THE AGENT'S NARRATION

Table 13 collects, verbatim, what published judges tell the model about the agent's own account of its work, and what they show it. Where a prompt is published only in a judge's released code, the source is the repository named below the table. The rows fall into the three responses of §2: judges that keep the narration and warn the model about it (the OSReward prompt, OS-Themis, WebVoyager, the Universal Verifier and AJ-Bench), judges that delete it (ZeroGUI and DigiRL, and WebJudge for the final response) and judges that rewrite it (VAGEN). AgentRewardBench shows the agent's thoughts

with no instruction, and UI-TARS-2 learns the judgment from annotated verdicts. JustJudge is the only row that keeps the agent's stated intents and removes its claims.

Table 13: What published judges show the model and what they say about the agent's narration. Quotations are verbatim from the paper or from the released prompt.

Judge	Shown to the model	Instruction about the narration
OSReward judging prompt, shared by its 27 judges (Sun et al., 2026)	per-step thoughts and actions, last five frames	"The action history may incorrectly claim success or failure, and actions listed in the history are not guaranteed to have been successfully executed."
OS-Themis critics (Li et al., 2026)	think, operation and action per step, screenshots	"think: internal reasoning (UNRELIABLE; never accept as proof)"; "Evidence-first: Trust only observed results" (released code)
WebVoyager evaluator (He et al., 2024)	final response, last frames	"there may be discrepancies between the text and the screenshots"; "Result response may contradict the screenshot, then the content of the screenshot prevails" (released code)
Universal Verifier (Rosset et al., 2026)	action history, final output, screenshots	"Trust the Action History over the Agent Output. Penalize hallucinations where the Output claims something not supported by the Action History." (released code)
AJ-Bench judge (Shi et al., 2026b)	trajectory text, live environment	"Do not rely solely on trajectory text."
AgentRewardBench judge (Lù et al., 2025)	actions and "the thought process of the agent", first and last screenshot	none (released code)
UI-TARS-2 reward model (Wang et al., 2025)	"the full text history", last five screenshots	trained on annotated verdicts
ZeroGUI reward estimator (Yang et al., 2025)	screenshots only	"The agent's responses are excluded. They may contain hallucinations of success, even when the task actually fails."
DigiRL evaluator (Bai et al., 2024)	one screenshot at a time	no agent text is shown
WebJudge (Xue et al., 2025)	actions, key screenshots	"we choose not to rely on the final response . . . the final response is prone to contain hallucinations"
VAGEN (Cui et al., 2026)	consolidated memory, environment	"summarize the state observation and action description within the reasoning . . . while discarding the sub-goal analysis"
JustJudge	claim-free step log, sixteen frames	self-declared outcomes masked before the judge sees them; intents extracted without claims

Released code: OS-Themis, `https://github.com/OS-Copilot/OS-Themis` (`PROMPT/BASE_PROMPT.py`); WebVoyager, `https://github.com/MinorJerry/WebVoyager` (`evaluation/auto_eval.py`); the Universal Verifier, in the Fara-7B evaluation package `https://github.com/microsoft/fara` (`webeval/src/webeval/rubric_agent/prompts.py`); AgentRewardBench, `https://github.com/McGill-NLP/agent-reward-bench` (`agent_reward_bench/judge/defaults.py`).

F Additional cases

Two OSReward trajectories, each with the verdict, confidence and justification of the judge reading the record as logged and of JustJudge on the same frames. In the first the record is held fixed and only the declared outcome is flipped (§5), and the judge's account of the final screenshot follows the declaration. The second shows how little text it takes: one sentence of self-assessment is enough to reach maximum confidence in the wrong verdict.

> **The knob that was never turned** (Ubuntu, 4 steps, human label: fail; one record under two declarations). *Task:* in Mixxx, turn the HIGH EQ knob of Deck 1 fully counterclockwise,

move the pitch slider off its center position, and set the first hot cue. *Record declaring success:* success (confidence 0.70), "Final screenshot shows H EQ knob rotated to its minimum (lower-left), pitch slider off center, and Hot Cue '1' lit with a cue marker on the waveform…" *The same record declaring failure:* fail (confidence 0.72), "Two of the three sub-goals are visibly complete in the final screenshot…However, the HIGH (H)…" *JustJudge, identical input under both:* fail (confidence 0.75), "…the HIGH EQ knob was not turned down: Step 1's drag targeted the green kill button below the knob…"

Stopping one step short (Ubuntu, 4 steps, human label: fail). *Task:* right-click the `country_boundaries` layer to open its context menu and access the options that change its display properties. *Agent's closing step:* "The task was successfully completed.", followed by a termination action declaring success. *Record as logged:* success (confidence 1.0), "The agent successfully right-clicked on the `country_boundaries` layer…This menu contains the 'Display…' " *JustJudge:* fail (confidence 0.65), "The right-click in step 3 opened the context menu, but the agent terminated in step 4 without clicking the menu item that opens the display properties."

G PROMPTS

This appendix reproduces the verification prompt, the masking policy and the intent-extraction instructions of JustJudge. The second paragraph quotes the two clauses that the record-as-logged arm of §4.3 changes.

Verification prompt. The system prompt of JustJudge, used unchanged on every benchmark and every backbone in this paper:

```
You judge whether a GUI-agent task SUCCEEDED from the task
instruction, sequential screenshots of the trajectory, and the
agent's per-step log of INTENT and ACTION. Each step lists a brief
intent (the agent's own stated plan for that step) and the action it
issued.  Intents are the agent's narrative, NOT evidence of what
happened — they can be mistaken or self-serving, and any
self-assessment of success has been masked as <masked>; never treat
the agent's wording as evidence.  Analyze EACH screenshot in order:
check what the listed actions should have done and what actually
changed on screen; issued actions are NOT guaranteed to have executed.
The log covers every step, including steps whose screenshots are not
shown.  Identify the first step where the trajectory goes wrong (if
any).  The task counts as success ONLY if it is FULLY and CORRECTLY
completed by the FINAL screenshot — partial or intermediate progress
is failure.  Do not hedge or re-check repeatedly; decide and commit.
Output ONLY JSON (last line):  {"first_error":int|null,
"verdict":"success|fail|unknown","confidence":0..1,"why":str}
```

The user turn is assembled from the record. It opens with a header,

```
TASK: <task instruction>

<k> of <n> sequential screenshots are shown.  The step log below
lists, for each of the <T> steps in order, the agent's brief stated
INTENT ('Step k intent:')  and the action it issued ('Step k
action:'), including steps whose screenshots are omitted.  <frame
semantics>
```

whose last sentence, the frame semantics of the corpus, is the one component that changes across corpora. On OSReward it reads

```
Each screenshot shows the state at the moment the step's action was
ISSUED (before that action took effect); the final screenshot may not
show the last action's effect.  Some steps' screenshots are missing
from the source data; their actions are still listed.
```

On OGRBench it is the same sentence without the closing clause about missing screenshots. On CUARewardBench it reads

```
There is NO initial-state screenshot.  Each screenshot is the state
the agent observed when issuing the NEXT listed action; the state
before the first action is not shown, and the final screenshot shows
the resulting end state.
```

The selected screenshots follow the header, each labeled `Screenshot i` and the last one `Screenshot i (FINAL)`. They are interleaved in temporal order with the `Step k intent:` and `Step k action:` lines of the steps that fall between them, so every step appears in the log whether or not its screenshot is shown.

The record-as-logged prompt. The prompt of the record-as-logged arm of Table 2 is the same text with two edits. The clause describing the log,

```
log of INTENT and ACTION. Each step lists a brief intent (the agent's
own stated plan for that step) and the action it issued.  Intents are
the agent's narrative, NOT evidence of what happened — they can be
mistaken or self-serving
```

becomes

```
log of REASONING and ACTION. Each step lists the agent's recorded
reasoning for that step ('Step k intent:'  lines) and the action it
issued.  The recorded reasoning is the agent's narrative, NOT
evidence of what happened — it can be mistaken or self-serving
```

and the clause describing the mask,

```
and any self-assessment of success has been masked as <masked>; never
treat the agent's wording as evidence.
```

is deleted where nothing is masked. The `Step k intent:` lines then carry the recorded reasoning verbatim, cut at the same length cap as the intents.

Masking policy. The mask is applied to every step before the judge sees it. In the action line, the closing tokens `done` and `fail` of the OSWorld dialect become `terminate(status=<masked>)`. A `terminate(...)` call becomes `terminate(status=<masked>)` with its payload dropped. The AndroidWorld closing message `finished(...)` becomes `finished(content=<masked>)`. The `status` and `goal_status` fields of JSON dialects have their values replaced by `<masked>`. In the intent line, every sentence is tested against two patterns: one for self-verdicts (a subject among *task, it, everything, request, I, we*, a form of *be* or *have*, and one of *complete, finish, success, done, accomplish*) and one for outcome assertions (*successfully*, or any subject with *is, are, was, were, has been, have been* and one of *complete, done, finished, created, saved, applied, verified, accomplished*). A matching sentence is replaced by `<masked>`, consecutive masks are merged, a line with nothing left becomes `<masked>`, and lines are cut at 240 characters. The same guard is applied to every extracted intent when it is read (§3.2).

Intent-extraction instructions. The same backbone that judges the trajectory extracts the intents offline from the agent's recorded reasoning alone. It sees neither screenshots nor labels. Each call receives the recorded reasoning of up to one hundred consecutive steps, one block per step, and returns one intent line per block under the following rules, reproduced verbatim ("src" is the block holding one step's recorded reasoning). A line that violates them is regenerated, and the guard above is applied once more when the line is read.

```
For each src, write the agent's brief stated INTENT for that step:
1.  ≤15 words, one line, present tense, behavioral ("Click the search
icon to open the search field.").  Count your words; 16+ is a hard
violation.
2.  Summarize ONLY what the agent itself stated it was trying to do.
Do not invent intent from outside knowledge; do not add facts the src
does not state.
```

```
3.  STRICTLY FORBIDDEN: any success / completion / correctness /
verification claim —— even if the src asserts it ("Perfect!  I've
successfully..."").  Never output words like:  success, successfully,
complete(d), finished, done, accomplished, correct(ly), verified,
achieved, "task is done", "goal met".  Rephrase to the zero-status
behavioral intent instead (e.g.  src "I've successfully filtered the
list, now I'll export it" → "Export the filtered list.").
4.  Terminal steps (the agent decides to stop/terminate/report):
write a zero-status phrase such as "Terminate the trajectory." or
"End the session and report the findings." —— never the claimed
outcome or status.
5.  If the src is only a situation description with no actionable
plan, summarize the immediate goal it implies from ITS OWN words,
still claim-free.
6.  Src text may be in Chinese or English; the intent line is always
English.
7.  Never leave an intent empty; never output placeholder text like
"N/A".
```