Self-Calibrated GUI Reward Model via Inverse Dynamic Modeling

Zeyi Sun[1,2] **Shengyuan Ding**[3] **Xingpeng Xia**[4] **Jinsong Li**[5]
Xie Chen[1] **Dahua Lin**[2,5] **Jiaqi Wang**[6]

[1]Shanghai Jiao Tong University
[2]Shanghai Artificial Intelligence Laboratory
[3]Fudan University
[4]University of Michigan
[5]The Chinese University of Hong Kong
[6]Shanghai Innovation Institute
szy2023@sjtu.edu.cn, wangjiaqi@pjlab.org.cn

Abstract

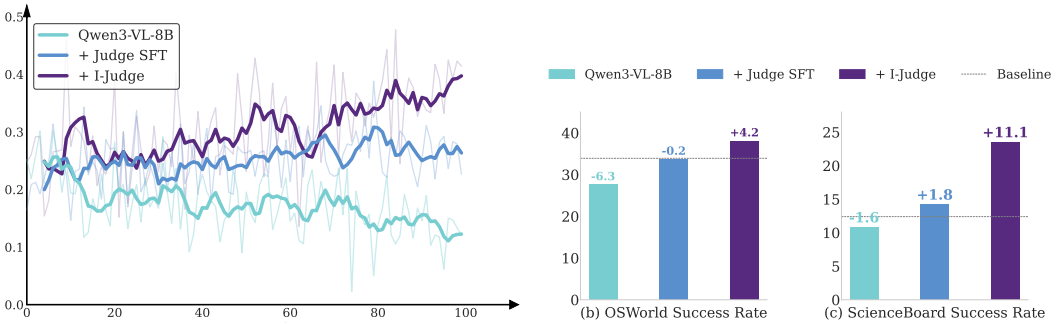
Developing robust long-horizon GUI agents is bottlenecked by reliable reward signals. Ground-truth verification often requires costly human annotation or environment-specific checkers, while scalable VLM judges can assign positive rewards from superficial visual evidence. We propose I-Judge, a self-calibrated GUI reward-modeling framework with two complementary components. First, inverse dynamic modeling (IDM) provides an auxiliary objective: executor-logged actions supervise the reconstruction of each action from consecutive screenshots without additional human annotation, helping the outcome judge learn action–transition regularities. Second, at runtime, the IDM's agreement with the executed action log provides a one-sided audit that down-weights positive rewards unsupported by visible transitions. This consistency signal audits trajectory evidence rather than task correctness; it cannot recover false negatives or detect failures that are visible only in latent environment state. Under controlled reward-model evaluation, I-Judge improves precision while retaining competitive recall, and end-to-end RL experiments on OSWorld, ScienceBoard, and AndroidWorld show consistent gains across two base policies. Additional comparisons on AgentRewardBench and three complete RL systems further separate the effects of auxiliary IDM training and runtime calibration. Code, data, and models will be released.

1 Introduction

The pursuit of generalist agents that can automate complex tasks on graphical user interfaces (GUIs) has seen rapid progress with the advent of Multimodal Large Language Models (MLLMs) [OpenAI, 2023, Yan et al., 2023, Bai et al., 2025a,b, Wang et al., 2024, Yang et al., 2025, Hurst et al., 2024, Zheng et al., 2024, Anthropic, 2024, 2025]. While open-loop agents [Putta et al., 2024, Agashe et al., 2024, 2025, Sun et al., 2025c, Hu et al., 2024, Hong et al., 2023, 2024] have established strong baselines on benchmarks such as OSWorld [Xie et al., 2024] and AndroidWorld [Rawles et al., 2024], they often struggle with long-horizon robustness and self-correction. Reinforcement Learning (RL) emerges as a promising paradigm to bridge this gap, potentially allowing agents to continually improve through interaction with massive parallel environments [Sun et al., 2025b, Wang et al., 2025a, Sun et al., 2025c, Lu et al., 2025].

However, the efficacy of RL hinges critically on the quality of the Reward Model (RM), which faces a dilemma between scalability and quality. Reliable reward signals are notoriously scarce: while negative signals (failures) are abundant, successful trajectories—critical for guiding RL—are rare

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Reward score w.r.t. steps during multi-turn RL training.

(b) Success rate on in-domain and out-of-domain benchmarks.

Figure 1: Our I-Judge System significantly improves the reward score during training and consequently leads to gains on both in-domain and out-of-domain benchmarks.

in complex tasks. Ground-truth verification typically relies on environment-specific programmatic checkers (e.g., assessing file changes), which are brittle and unavailable for *novel software* or open-ended web tasks. Consequently, one is forced to rely on human annotation [Xie et al., 2024, Sun et al., 2025a], which is expensive, slow, and fundamentally unscalable. Learned VLM judges [Xue et al., 2025] offer a scalable alternative, but they suffer from "reward hacking," where agents exploit the hallucinations of an imperfect verifier. Training a better judge circles back to the original problem: one needs massive amounts of successful trajectories grounded by human experts, creating a catch-22 that bottlenecks the development of generalist agents.

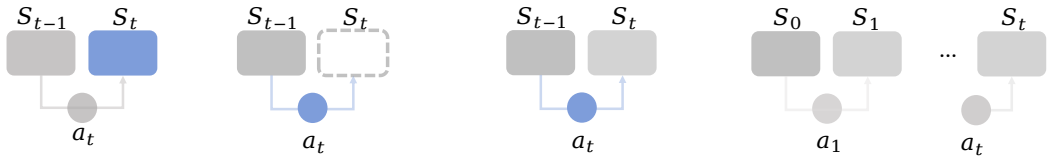
In this work, we propose to break this cycle by leveraging Inverse Dynamic Modeling (IDM) as a scalable, self-supervised pretraining objective (Figure 2). Unlike outcome supervision (Success/Fail), which requires human judgment, the "labels" for IDM—the actions taken between two states—are generated automatically by the agent itself. We can deploy thousands of agents in parallel to interact with the environment [Lu et al., 2025, Wang et al., 2025a], collecting interaction data $\big((s_t, s_{t+1}) \rightarrow a_t\big)$ with executor-logged action targets and no additional human annotation. By training the model to reconstruct the action a_t from the visual transition $s_t \rightarrow s_{t+1}$, we encourage it to learn action–transition regularities of the GUI. These targets record the action issued by the executor; they do not imply that the action was correct for the user's task or uniquely identifiable from pixels.

Building on this insight, we propose a novel framework for Self-Calibrated Reward Modeling. Specifically, we first integrate IDM as an auxiliary task that helps the VLM judge represent fine-grained GUI transitions. We then reuse the learned action predictor at inference time: agreement between reconstructed and executor-logged actions yields a consistency score that down-weights positive outcome rewards lacking visible action–transition support. This audit is deliberately one-sided and does not equate consistency with task correctness. Our technical contributions are threefold:

- **Executor-supervised representation learning.** IDM training uses automatically logged actions to improve an outcome judge without step-level human reward labels; this auxiliary objective provides the larger of our two measured gains.
- **Runtime evidence audit.** We introduce a trajectory-level soft reward that multiplies the base judge's positive prediction by action consistency. It suppresses visually unsupported positives but cannot turn a negative base judgment into a positive one.
- **Controlled empirical evaluation.** Across OSWorld [Xie et al., 2024], ScienceBoard [Sun et al., 2025a], AndroidWorld [Rawles et al., 2024], and AgentRewardBench [Lù et al., 2025], we separately evaluate reward quality, IDM auxiliary training, runtime calibration, and integration with complete RL systems.

2 Related Work

GUI agents and foundation models. The rapid evolution of Multimodal Large Language Models (MLLMs) [OpenAI, 2023, Hong et al., 2023, Cheng et al., 2024, Guo et al., 2025, Bai et al., 2025b,a] has reshaped GUI automation. Specialized foundation models such as UI-TARS [Qin et al., 2025,



(a) World Model (b) GUI Agent (c) Inverse Dynamic Model (d) Judge Model

Figure 2: Comparison of modeling paradigms for GUI agents. (a) World Models predict future states from actions. (b) GUI Agents predict actions from history. (c) **Inverse Dynamic Models (IDM)** (Ours) infer the action a_t that caused the transition $S_{t-1} \rightarrow S_t$. (d) Judge Models evaluate the full trajectory. We leverage the IDM (c) to self-calibrate the noisy signal from the Judge (d).

Wang et al., 2025a], Atlas [Wu et al., 2024], OpenCUA [Wang et al., 2025b], ShowUI [Lin et al., 2024], GUI-Actor [Wu et al., 2025a], and CoAct [Song et al., 2025] push action modeling, coordinate-free grounding, or code-as-action. Holistic frameworks [Yang et al., 2026, Chen et al., 2025] aim to generalize across OS ecosystems, and synthesis pipelines such as OS-Genesis [Sun et al., 2024] and AgentTrek [Xu et al., 2024] address data scarcity.

Reinforcement learning for GUI agents. RL is increasingly used to enforce long-horizon success: DigiRL [Bai et al., 2024] and WebRL [Qi et al., 2024] for web, CODA [Sun et al., 2025b] (dual-brain architecture inspired by neuroscience [Ito, 2000]) for scientific software, SEAgent [Sun et al., 2025c] for self-evolution, and ARPO [Lu et al., 2025] for off-policy multi-turn RL with experience replay. Robust evaluation is provided by OSWorld [Xie et al., 2024], ScienceBoard [Sun et al., 2025a], and AgentRewardBench [Lù et al., 2025]; we focus on the reward model that drives end-to-end multi-turn RL.

Reward modeling and critics. RL success hinges on reward quality, but sparse environmental feedback is brittle and VLM judges hallucinate [Xue et al., 2025, Wu et al., 2025b]. Process reward models work well for math but transfer poorly to GUIs. We differentiate by using *inverse dynamics* as a self-supervised objective for reward *calibration*, distinct from forward world models or intent-predicting policies, aligning with the "Era of Experience" [Silver and Sutton, 2025, Sun et al., 2025c].

3 Method

We present I-Judge, a framework for Self-Calibrated GUI Reward Modeling (overview in Fig. 3, with paradigm comparison in Fig. 2). It has two components: (1) **self-supervised IDM training** that uses unlabeled trajectories and executor-logged actions to learn GUI action–transition regularities (Fig. 3(a)); and (2) **runtime reward calibration** that uses the IDM to audit the visible support for agent actions during policy optimization, softening positive rewards for ambiguous trajectories. The first component supplies the larger empirical gain; the second adds a smaller, separately measurable effect.

3.1 Preliminary: the GUI POMDP

We model GUI automation as a POMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$. The latent state $s_t \in \mathcal{S}$ (DOM tree, memory, etc.) is emitted as a visual observation $o_t \in \mathcal{O}$ via $o_t \sim \Omega(s_t)$, and the agent executes $a_t \in \mathcal{A}$ from history $h_t = (o_0, a_0, \ldots, o_t)$ under instruction $\mathcal{I}$, with transition $\mathcal{T}(s_{t+1} \mid s_t, a_t)$. The objective is a policy $\pi_\theta(a_t \mid h_t, \mathcal{I})$ that maximizes $\mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t]$. Because the ground-truth reward r_t is typically unavailable, we rely on a learned *judge model* $\mathcal{J}_\phi(h_T, \mathcal{I}) \rightarrow [0, 1]$ that approximates $P(\text{Success} \mid \tau)$ for trajectory τ.

3.2 Modeling paradigms for representation learning

To learn robust representations for $\mathcal{J}_\phi$, we contrast three objectives over trajectory data $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$. (i) *Behavioral cloning* (Fig. 2(b)) maximizes $\log \pi_\theta(a_t \mid s_t, \mathcal{I})$, so its representation focuses on *intent* but is blind to the visual *consequence* s_{t+1} that a judge must verify. (ii) *World modeling* (Fig. 2(a)) maximizes $\log P_\phi(s_{t+1} \mid s_t, a_t)$. In GUIs, s_{t+1} contains a large amount of pixel-level noise unrelated to the action (system clock, ads, animations), so the model wastes capacity on

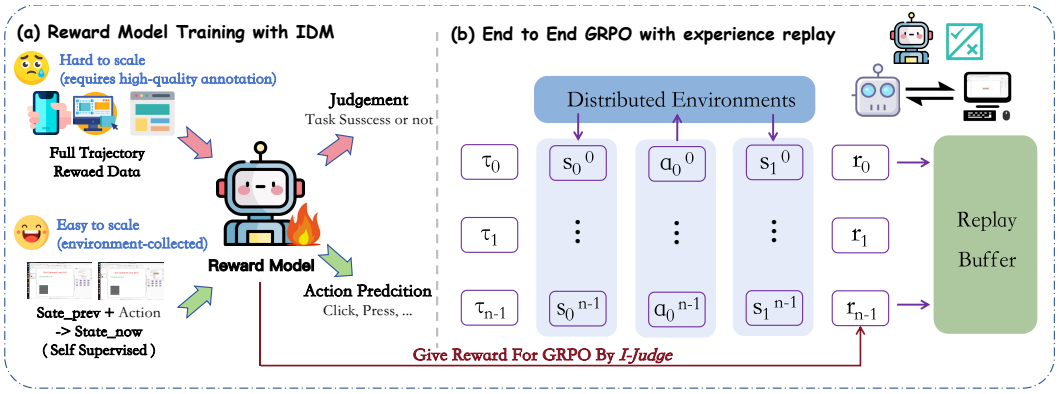


Figure 3: **Overview of the I-Judge framework.** (a) **Reward Model Training with IDM.** We combine scarce human-labeled outcome data with self-supervised trajectories. The IDM auxiliary task reconstructs the executor-logged action a_t from the transition $s_t \to s_{t+1}$, encouraging representations of GUI dynamics. (b) **End-to-end GRPO with experience replay.** We deploy the pretrained I-Judge in a distributed RL pipeline. The IDM consistency score audits whether visible transitions support the action log and softly calibrates the outcome judge's positive rewards.

stochastic nuisance variables. (iii) *Inverse Dynamic Modeling (IDM)* (Fig. 2(c)) is the discriminative dual: given the concatenation $[s_t; s_{t+1}]$, predict the intervening action a_t:

$$\mathcal{L}_{\text{IDM}}(\psi) = -\mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_t \log P_\psi(a_t \mid s_t, s_{t+1})\right]. \tag{1}$$

Optimizing $\mathcal{L}_{\text{IDM}}$ encourages the encoder to extract *difference features* $\Delta(s_t, s_{t+1})$ that predict the intervening action. From an information-bottleneck view, the target A is low-dimensional and semantic, so $I(Z; A)$ is less dominated by high-entropy visual nuisance variables than $I(Z; S_{t+1})$ in forward modeling. Empirically, increasing unlabeled IDM data produces nonlinear gains in downstream judge precision (Table 4).

3.3 Self-calibrated reward mechanism

We introduce a runtime calibration mechanism (Figure 4) that compares the *executor-logged sequence* $A = \{a_1, \ldots, a_T\}$ with the *inferred sequence* $\hat{A} = \{\hat{a}_1, \ldots, \hat{a}_T\}$ predicted by the IDM, where $\hat{a}_t = \arg\max_a P_{\text{IDM}}(a \mid s_t, s_{t+1})$. We define the *Action Consistency Score* $\mathcal{C}$ based on the normalized distance between these two sequences:

$$\mathcal{C}(\tau) = \frac{1}{2T}\sum_{t=1}^{T}\Big(\mathbb{I}\big(\text{type}(\hat{a}_t) = \text{type}(a_t)\big) + \big(1 - \text{dist}(\hat{a}_t, a_t)\big)\Big), \tag{2}$$

where $\mathbb{I}(\cdot)$ checks whether action types match and $\text{dist}(\hat{a}_t, a_t) \in [0, 1]$ is a normalized, type-specific argument distance (Appendix B.6). High consistency means that visible transitions are well explained by the logged actions. It does *not* establish that those actions advance the instruction: a wrong button can be clicked cleanly, and latent state changes may be invisible. We use this score to calibrate the soft reward:

$$R_{\text{final}} = r_{\text{base}} \cdot \mathcal{C}(\tau)^\alpha. \tag{3}$$

Because $r_{\text{base}} \in \{0, 1\}$ and $\mathcal{C} \in [0, 1]$, calibration is one-sided: it can suppress a positive reward but cannot recover a false negative of the base judge. It is therefore an evidence audit, not a correctness oracle or a process reward. Table 3 selects one global $\alpha = 0.8$ on OSWorld and freezes it for the other benchmarks; overly aggressive calibration penalizes valid but unfamiliar actions.

3.4 Policy optimization via self-calibrated GRPO

To optimize the policy π_θ with our trajectory-level calibrated reward, we employ Group Relative Policy Optimization (GRPO) [Shao et al., 2024], which eliminates the need for a separate value

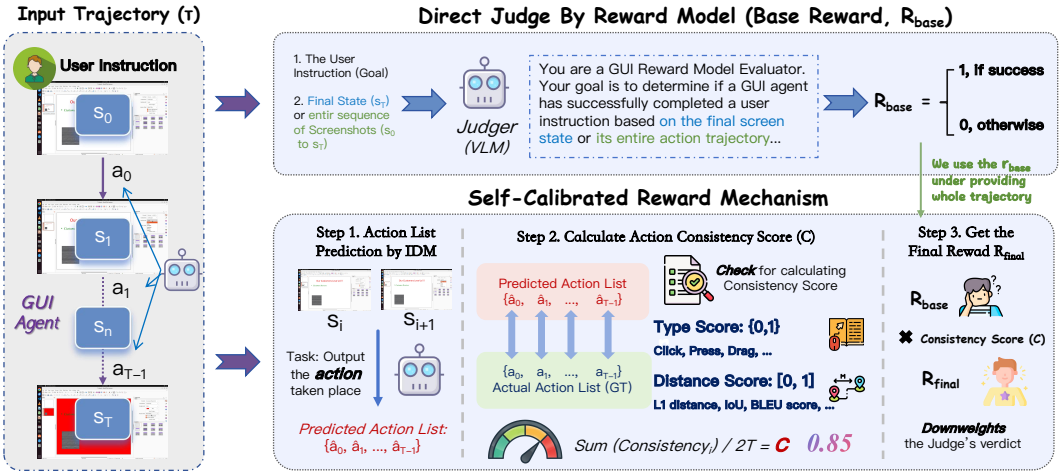


Figure 4: **Inference pipeline of I-Judge.** The evaluation process involves three steps. **(1) Direct base judgment.** The VLM Judge takes the user instruction and the agent's trajectory (visual history) as input and outputs a preliminary binary success label (R_{base}). **(2) Action consistency audit.** Simultaneously, the IDM analyzes step-by-step visual transitions ($S_i \rightarrow S_{i+1}$) to infer actions ($\hat{a}_i$). We compute the *Action Consistency Score* ($\mathcal{C}$) by comparing IDM-predicted actions with executor-logged actions, aggregating action-type matching and argument similarity. **(3) Final reward calibration.** The final reward R_{final} scales R_{base} by $\mathcal{C}$. If the trajectory is visually inexplicable (low consistency), a positive reward is dampened.

network by using group-based advantage estimation. For each user instruction $\mathcal{I}$, we sample a group of G trajectories $\{\tau_1, \ldots, \tau_G\}$ from the current policy $\pi_{\theta_{old}}$. The optimization objective is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\min\left(\rho_{i,t}\,\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\,\hat{A}_i\right)\right] - \beta\,\mathbb{D}_{\text{KL}}. \tag{4}$$

Crucially, the advantage $\hat{A}_i$ is computed using our *self-calibrated reward* R_{final}, bridging IDM-based verification and policy learning:

$$\hat{A}_i = \frac{R_{\text{final}}(\tau_i) - \mu_R}{\sigma_R}, \qquad R_{\text{final}}(\tau_i) = r_{\text{base}}^{(i)} \cdot \mathcal{C}(\tau_i)^{\alpha}, \tag{5}$$

where μ_R and σ_R are the mean and standard deviation of the calibrated rewards within the group. When the IDM detects a discrepancy (low $\mathcal{C}$), R_{final} drops and the group-relative advantage can become negative, suppressing trajectories whose positive verdict lacks visible support. Cleanly executed but semantically wrong behavior and hidden-state failures remain outside this mechanism's scope.

3.5 Autonomous data synthesis and curriculum

A limitation of prior frameworks such as ARPO [Lu et al., 2025] is their reliance on fixed benchmarks with human-defined verifiers. We instead use an autonomous data engine inspired by SEAgent [Sun et al., 2025c]: a VLM proposes diverse instructions conditioned on the current GUI state. For the *RL curriculum only*, we run 16 rollouts from the current actor per task and discard all-success or all-failure groups, which carry little group-relative gradient. The IDM corpus is sampled from the unfiltered transition pool; task-level success difficulty is not used to select IDM examples.

4 Experiments

4.1 Experimental setup

We adopt the standard MDP for GUI agents [Xie et al., 2024]: a trajectory $\tau = \{s_t, a_t\}_{t=0}^{T-1}$ pairs screenshots s_t with keyboard/mouse actions a_t, and the policy $\pi_\theta(a_t \mid s_t, h_{t-1})$ predicts the next action from up to 15 visual frames. The base policies are UI-TARS-1.5 [Qin et al., 2025, Wang

Table 1: **Judge model performance comparison.** Precision (P) and recall (R) on three benchmarks. Runtime calibration consistently trades some recall for higher precision, while IDM auxiliary training improves the underlying judge.

Method	AgentRewardBench		OSWorld-Reward		ScienceBoard-Reward	
	P	R	P	R	P	R
Qwen3-VL-8B (LS)	67.5	85.1	57.2	78.7	44.6	79.3
Qwen3-VL-8B (FS)	70.5	81.1	61.2	72.1	48.2	71.4
Qwen3-VL-8B Direct FT	73.5	88.4	72.3	81.2	56.2	77.5
Qwen3-VL-8B + IDM	77.2	**89.1**	75.5	**83.0**	73.3	**79.9**
I-Judge	**80.1**	86.3	**79.3**	79.4	**79.2**	76.4

et al., 2025a] and Qwen3-VL [Bai et al., 2025a]. End-to-end RL is performed with Group Relative Policy Optimization (GRPO) [Shao et al., 2024] in a distributed rollout framework with 256 parallel environments, following ARPO [Lu et al., 2025] (Fig. 3(b)). Reward signals are produced by the reward model in Fig. 3(a); we compare vanilla reward models trained on human-labeled data with our I-Judge.

I-Judge is trained on two complementary datasets. **(1) Outcome supervision data:** 16,592 human-labeled trajectories from AgentNet [Wang et al., 2025b] with binary success labels. **(2) Unlabeled dynamics data:** an LM-based proposer (Qwen-2.5-32B [Yang et al., 2024]) generates tasks following SEAgent [Sun et al., 2025c], while trajectories are sampled from the current actor in each run (full prompts in Appendix D). The proposer never supplies actions or IDM labels. Executor logs yield 500K action–transition examples (300K OSWorld and 200K ScienceBoard) from the unfiltered task pool; the 16-rollout difficulty filter is used only for the RL curriculum. The replay buffer holds 2,000 trajectories, the trajectory length is capped at 50 steps, and the calibration factor is $\alpha = 0.8$ by default. All training runs use 4×8 NVIDIA H200 GPUs.

4.2 Main results

Judge model performance. Table 1 compares I-Judge against baselines on AgentRewardBench [Lù et al., 2025] and our OSWorld-Reward and ScienceBoard-Reward sets (statistics in Appendix B). IDM auxiliary training improves the underlying judge, while calibration raises precision from 77.2/75.5/73.3 to 80.1/79.3/79.2 and lowers recall from 89.1/83.0/79.9 to 86.3/79.4/76.4. This is the expected signature of a one-sided precision filter, not evidence that calibration repairs false negatives.

End-to-end RL performance. Plugging these reward models into the RL loop, Table 2 reports final SR over 5 random seeds. Holding the IDM-trained judge fixed, runtime calibration adds $+2.8/+4.2/+2.2$ SR for Qwen3-VL-8B and $+2.2/+0.5/+0.9$ for UI-TARS on OSWorld/ScienceBoard/AndroidWorld. The auxiliary-training effect is larger overall, so we treat it as the primary component and calibration as an additional policy- and benchmark-dependent audit. Controlled prior-method comparisons appear in Section 4.4.

4.3 Ablation studies

To further validate the design choices of our self-calibration mechanism, we conduct ablation studies focusing on the calibration factor α and the choice of pretraining objective.

Effect of calibration (α). α controls the IDM penalty in $R_{\text{final}} = r_{\text{base}} \cdot \mathcal{C}^{\alpha}$. Here $\alpha = 0$ is the IDM-trained judge with calibration disabled. We select $\alpha = 0.8$ on OSWorld and freeze it before ScienceBoard evaluation; ScienceBoard instead peaks at 0.5, confirming that we did not tune per test benchmark. Aggressive calibration ($\alpha \geq 1$) can penalize valid but unfamiliar actions.

Benefit of unlabeled self-supervised training. We vary the ScienceBoard portion of IDM pretraining from 0 to 200K action–transition examples (Table 4). IDM action accuracy grows from 53.4% to 81.2%, while downstream judge precision rises from 56.2% to 73.3% (recall 77.5% $\rightarrow$ 79.9%). The gains are nonlinear but we do not claim a formal phase transition.

IDM vs. forward dynamics. A forward world-model baseline [Ha and Schmidhuber, 2018] that predicts next-state pixels must model nuisance variation such as cursors, clocks, and animations.

Table 2: **End-to-end RL performance.** Final success rate (SR, %) reported as mean ± standard deviation over five random seeds (0–4). AndroidWorld is not used to train or tune the reward model.

Base Model	Reward Model	OSWorld	ScienceBoard	AndroidWorld
Qwen3-VL-8B	no opt	$33.4_{\pm 0.7}$	$13.0_{\pm 1.6}$	$49.4_{\pm 1.0}$
	Qwen3-VL-8B (LS)	$27.9_{\pm 0.9}$	$11.4_{\pm 1.4}$	$43.8_{\pm 1.2}$
	Qwen3-VL-8B Direct FT	$34.1_{\pm 0.8}$	$14.7_{\pm 1.9}$	$47.0_{\pm 1.1}$
	Qwen3-VL-8B + IDM	$35.8_{\pm 0.6}$	$19.6_{\pm 1.2}$	$52.1_{\pm 0.8}$
	I-Judge	$\mathbf{38.6_{\pm 0.5}}$	$\mathbf{23.8_{\pm 1.0}}$	$\mathbf{54.3_{\pm 0.7}}$
UI-TARS-1.5-7B	no opt	$27.9_{\pm 0.8}$	$8.6_{\pm 1.5}$	$55.2_{\pm 0.9}$
	Qwen3-VL-8B (LS)	$26.7_{\pm 0.9}$	$9.1_{\pm 1.3}$	$49.8_{\pm 1.2}$
	Qwen3-VL-8B Direct FT	$29.2_{\pm 0.8}$	$11.0_{\pm 1.7}$	$52.6_{\pm 1.0}$
	Qwen3-VL-8B + IDM	$30.7_{\pm 0.7}$	$17.4_{\pm 1.1}$	$56.7_{\pm 0.8}$
	I-Judge	$\mathbf{32.9_{\pm 0.6}}$	$\mathbf{17.9_{\pm 1.0}}$	$\mathbf{57.6_{\pm 0.7}}$

Our IDM instead predicts a lower-dimensional semantic action target; the scaling results support the utility of this objective without implying that action reconstruction verifies task completion.

Transfer to the mobile domain. AndroidWorld [Rawles et al., 2024] is not used to train or tune the reward model. I-Judge raises Qwen3-VL-8B from 49.4% to 54.3% and UI-TARS-1.5 from 55.2% to 57.6%. These results demonstrate transfer to the evaluated mobile setting, without claiming universal GUI generalization.

4.4 Comparison with prior reward modelling approaches

We report complementary comparisons at three levels. A direct AgentRewardBench comparison evaluates recent reward models on the same labeled trajectories and official protocol (Table 6); with the matched Qwen2.5-VL-7B backbone, I-Judge obtains 77.9 precision, 88.1 recall, and 82.7 F1, exceeding WebJudge [Xue et al., 2025] by 2.6/5.6/4.0 points. We separately evaluate ProgRM's native public WikiHow protocol while distinguishing published references from our newly generated trajectories (Appendix K.1). Finally, we integrate I-Judge into DigiRL, WebRL, and SEAgent while preserving each system's RL algorithm and matching initialization, task splits, and interaction budgets (Table 7); all nine system–benchmark pairs improve.

To further isolate reward construction in our desktop setting, we compare with two representative families: (i) **ProgRM** [Zhang et al., 2025], a step-wise progress reward derived via an LCS-based self-annotation algorithm over successful trajectories; (ii) **VPT-IDM** [Baker et al., 2022], the classical inverse-dynamics formulation that learns an action predictor from a small action-labelled set plus a large unlabelled video corpus. For a fair OSWorld-domain comparison, we re-implement ProgRM and retrain it on the same 16,592 AgentNet trajectories used for our supervised component (rather than reusing its public WikiHow-Android checkpoint, which would suffer from a large domain gap). VPT-IDM uses a 2K action-labelled subset of AgentNet plus the 500K unlabelled rollouts collected for our IDM pretraining. All methods share the identical RL setup (Qwen3-VL-8B base policy; ARPO/GRPO; 5 random seeds). Full annotation cost decomposition is reported in Appendix H.

Why I-Judge outperforms ProgRM and VPT-IDM. Both prior baselines improve over the outcome-only Direct FT baseline by $\sim$1–3 SR points (Table 5), showing the value of transition-level supervision. Both remain below I-Judge, particularly on ScienceBoard (16.5 / 17.0 vs. 23.8) and AndroidWorld (50.2 / 51.4 vs. 54.3). We attribute the gap to two design choices: (i) ProgRM's LCS-derived progress relies on *multiple successful trajectories per task*, so in novel software where successful demonstrations are scarce its recipe library degenerates and the progress signal becomes brittle; (ii) VPT-IDM treats inverse dynamics as the primary reward, whereas I-Judge uses it as auxiliary training and as a multiplicative evidence audit of an outcome judge. As detailed in Appendix H, the 500K action–transition targets are read from executor logs and require no additional human labeling.

Calibration without IDM auxiliary training. To isolate the runtime mechanism, we apply consistency calibration to Direct-FT and zero-shot language-supervised (LS) judges that were not trained with the IDM objective, while fixing the UI-TARS policy and all RL settings. Table 8 shows positive gains in all six cells. Because these judges produce malformed joint outputs in a small fraction of cases, we filter only by parseability and continue collection until each group has 16 valid samples;

Table 3: **Calibration factor α.** One global value is selected on OSWorld and transferred to ScienceBoard.

α Value	OSWorld	ScienceBoard
0 (no calib.)	$35.8_{\pm 0.6}$	$19.6_{\pm 1.2}$
0.5	$36.4_{\pm 0.6}$	$\mathbf{25.6_{\pm 1.1}}$
0.8 (Ours)	$\mathbf{38.6_{\pm 0.5}}$	$23.8_{\pm 1.0}$
1	$37.4_{\pm 0.6}$	$20.0_{\pm 1.2}$
2	$35.0_{\pm 0.7}$	$10.8_{\pm 1.5}$
3	$29.7_{\pm 0.9}$	$9.1_{\pm 1.6}$

Table 4: **Scaling unlabeled pretraining.** More action–transition examples improve IDM accuracy and downstream judge precision.

Data Volume	IDM Acc.	Prec.	Rec.
no IDM	53.4	56.2	77.5
10K	58.5	56.9	76.9
50K	67.2	63.0	75.1
100K	74.3	68.1	78.4
200K (Full)	**81.2**	**73.3**	**79.9**

the caption reports the resulting raw-rollout overhead. The Direct-FT rows are the primary controlled comparison, while LS tests whether the runtime signal remains useful without outcome fine-tuning.

Does source-task difficulty matter for IDM? The curriculum filter does not select IDM data, but task difficulty could still affect transition quality indirectly. We therefore train Easy-, Medium-, and Hard-only IDMs with matched trajectory counts, de-duplicated transitions, and five runs per bin. Action accuracy spans only 80.9–81.4, and downstream F1 differs by at most 0.4 points; moreover, the best bin rotates across metrics (full results in Appendix K.4). We conclude that task-level success difficulty is not a useful IDM data-selection axis in this setting.

Cross-domain IDM transferability. To test transfer beyond environment-specific memorisation, we restrict IDM pretraining to one domain at a time (Appendix I). Single-domain models retain gains on held-out environments, although the full mix remains best. A stress test on 155 semantically equivalent action pairs (Appendix J) also reveals a non-negligible mean consistency gap ($\Delta\mathcal{C} = 0.14$), especially for hotkeys and shift-clicks; we treat this as a limitation rather than evidence that all equivalent actions are handled.

Compute overhead and reward-curve stability. The IDM is a 0.6B-parameter Qwen2-VL-style encoder that processes one (s_t, s_{t+1}) pair at a time and runs in parallel with the VLM judge; on our 4×8 H200 setup it adds $\sim$8% wall-clock time per GRPO update (40.2 vs. 37.1 minutes) and 7.2 GB peak memory. Averaged over the six cells in Table 2, cross-seed standard deviation drops from 1.13 for Direct FT to 0.72 for I-Judge. The reward curves increase without the mid-training collapse observed for Direct FT.

Where does I-Judge still fail? To characterise the remaining failure modes, we manually inspect 200 trajectories on which I-Judge assigns a high reward but the ground-truth checker reports failure (false positives) and 200 trajectories on which it assigns a low reward but the checker reports success (false negatives), evenly split across OSWorld and ScienceBoard (full taxonomy in Appendix C). Three error modes dominate the residual gap. (1) *Hidden-state preconditions* (39% of FPs): the agent visibly opens a settings panel and clicks "Apply," so both the judge and the IDM see a coherent action→state transition, yet the underlying file remains unchanged because of an unsaved password prompt outside the visible region. The IDM cannot disambiguate this from a successful save. (2) *Long-tail action types* (28% of FNs): rare actions such as multi-key chord shortcuts (e.g., `Ctrl+Shift+P`) appear $<0.3\%$ of the time in the unlabelled corpus, so the IDM under-predicts them and the consistency score $\mathcal{C}$ collapses, dragging R_{final} below the GRPO advantage threshold. (3) *Animation-induced ambiguity* (33% combined): smooth scroll, fade, and tooltip animations produce intermediate frames where the IDM cannot localise the originating click; this primarily affects ScienceBoard's plotly widgets. The first failure mode is a fundamental limitation of any vision-only reward model and motivates future work on light-weight DOM-aware sensors; the second and third motivate the learnable action-equivalence embedding discussed in Appendix J, which we identify as the most promising next research step.

Why can a one-sided audit still help policy optimization? Under group-relative optimization, a false positive receives a wrongly signed advantage and is actively reinforced, whereas a false negative primarily removes a useful sample. A bounded precision–recall trade can therefore improve the policy update even though calibration does not repair every classification error; Table 1 shows this intended signature.

Table 5: **Comparison with prior reward modelling approaches.** Reward-model precision (P, %) on our two reward evaluation sets, plus end-to-end RL success rate (SR, %; mean ± std over 5 seeds) on the three benchmarks. AndroidWorld is fully out-of-distribution (OOD) for all reward models. I-Judge combines self-supervised IDM with runtime calibration of a VLM judge, achieving the strongest results on every metric.

Method	Reward Type	Reward-Model Prec.		End-to-End RL Success Rate (%)		
		OSW-R	SB-R	OSWorld	ScienceBoard	AndroidWorld
Direct FT (ORM) [Wang et al., 2025b]	outcome only	72.3	56.2	$34.1_{\pm 0.8}$	$14.7_{\pm 1.9}$	$47.0_{\pm 1.1}$
ProgRM [Zhang et al., 2025]	step progress	71.8	60.5	$35.4_{\pm 0.7}$	$16.5_{\pm 1.4}$	$50.2_{\pm 1.0}$
VPT-IDM [Baker et al., 2022]	inv. dyn. (lab.)	70.6	65.0	$35.0_{\pm 0.8}$	$17.0_{\pm 1.5}$	$51.4_{\pm 1.0}$
Qwen3-VL-8B + IDM (no calib.)	inv. dyn. (self)	75.5	73.3	$35.8_{\pm 0.6}$	$19.6_{\pm 1.2}$	$52.1_{\pm 0.8}$
I-Judge (Ours, full)	inv. dyn. + cal.	**79.3**	**79.2**	$\mathbf{38.6_{\pm 0.5}}$	$\mathbf{23.8_{\pm 1.0}}$	$\mathbf{54.3_{\pm 0.7}}$

Table 6: **Direct reward-model comparison on AgentRewardBench.** All methods use the official labeled trajectories and evaluation protocol. Rows marked Qwen2.5-VL-7B provide the controlled same-backbone comparison; the complete set of official references is in Appendix K.2.

Reward model	Backbone	Prec.	Recall	F1
Qwen2.5-VL judge	Qwen2.5-VL	64.5	86.1	73.7
SEAgent [Sun et al., 2025c]	Qwen2.5-VL-7B	71.6	84.5	77.5
DigiRL reward [Bai et al., 2024]	Qwen2.5-VL-7B	65.1	79.4	71.5
WebRL reward [Qi et al., 2024]	Qwen2.5-VL-7B	68.3	80.2	73.8
WebJudge [Xue et al., 2025]	Qwen2.5-VL-7B	75.3	82.5	78.7
I-Judge	Qwen2.5-VL-7B	**77.9**	**88.1**	**82.7**
I-Judge	Qwen3-VL-8B	79.1	88.6	83.6

Table 7: **Compatibility with complete RL systems.** Success rate (%) under matched Qwen3-VL-8B initialization, task splits, and interaction budgets. Adding I-Judge changes only the reward path.

Method	OSWorld	ScienceBoard	AndroidWorld
DigiRL [Bai et al., 2024]	37.2	15.2	52.2
+ I-Judge	38.1 (+0.9)	17.9 (+2.7)	54.1 (+1.9)
WebRL [Qi et al., 2024]	37.7	13.9	51.5
+ I-Judge	39.0 (+1.3)	18.5 (+4.6)	53.4 (+1.9)
SEAgent [Sun et al., 2025c]	39.3	17.2	53.8
+ I-Judge	**40.5** (+1.2)	**23.9** (+6.7)	**56.1** (+2.3)

Table 8: **Runtime calibration without IDM auxiliary training.** UI-TARS-1.5-7B policy; SR is mean±standard deviation over five seeds. Valid joint-output rates are approximately 91% for Direct FT and 93% for LS, corresponding to 9.9% and 7.5% raw-rollout overhead.

Judge	Calibration	OSWorld	ScienceBoard	AndroidWorld
Direct FT	no	$29.2_{\pm 0.8}$	$11.0_{\pm 1.7}$	$52.6_{\pm 1.0}$
Direct FT	yes	$\mathbf{31.1_{\pm 0.4}}$	$\mathbf{13.1_{\pm 1.2}}$	$\mathbf{53.4_{\pm 0.6}}$
LS	no	$26.7_{\pm 0.9}$	$9.1_{\pm 1.3}$	$49.8_{\pm 1.2}$
LS	yes	$\mathbf{28.5_{\pm 0.7}}$	$\mathbf{14.8_{\pm 1.1}}$	$\mathbf{52.8_{\pm 0.9}}$

5 Conclusion and Limitations

We introduced I-Judge, which reuses executor-supervised inverse dynamics in two roles: an auxiliary objective for learning action–transition representations and a runtime audit of whether visible transitions support the action log. The distinction between these roles is important. Auxiliary IDM training changes the judge's representation and can improve both precision and recall, whereas multiplicative runtime calibration is algebraically one-sided: it can only reduce an already-positive outcome reward. Our controlled decomposition confirms the corresponding difference in magnitude. Auxiliary training provides the larger gain, while holding the IDM-trained judge fixed and enabling calibration adds 0.5–4.2 success-rate points depending on policy and benchmark. Calibration also remains beneficial when applied to judges trained without the IDM objective, showing that its effect is not merely inherited from pretraining.

The experiments evaluate this decomposition at reward-model and policy levels. On AgentRewardBench, the matched Qwen2.5-VL-7B version of I-Judge improves over WebJudge by 2.6 precision, 5.6 recall, and 4.0 F1 points. Under end-to-end optimization, I-Judge improves two base policies across OSWorld, ScienceBoard, and AndroidWorld. It also improves DigiRL, WebRL, and SEAgent under matched initialization, splits, and interaction budgets, indicating that action–transition supervision complements different curriculum and policy-update designs. The source-task difficulty study gives a useful negative result: Easy-, Medium-, and Hard-only IDM corpora perform similarly when transition counts are matched, supporting our clarification that difficulty filtering belongs to the RL curriculum rather than IDM data selection.

More broadly, these results suggest a precise role for inverse dynamics in GUI reward learning. The executor supplies inexpensive action targets, allowing a judge to learn transition-sensitive features without step-level human reward labels. At runtime, the same prediction task audits the provenance of visible evidence. This audit is particularly suited to policy optimization because a false-positive reward is actively reinforced, while a false negative primarily removes learning signal. I-Judge therefore does not replace an outcome judge; it adds a complementary, precision-oriented check at the point where unsupported positive rewards are most harmful.

Limitations. Consistency is not correctness: a semantically wrong action can be executed cleanly, and a pixel-only IDM cannot detect hidden-state failures or distinguish actions with overlapping visual effects. Multiplicative calibration cannot recover a false negative of the base judge. Rare actions, hotkeys, no-op transitions, delayed updates, and animations can also depress consistency for valid behavior, so calibration strength must be selected conservatively. Our evaluation covers three environments, with IDM data from two desktop platforms, and assumes access to sandboxed parallel rollouts. Future work should combine visual consistency with artifact inspection, structured environment state, or active validation procedures; learn equivalence-aware action distances; and test broader mobile, web, and deployment conditions. Broader impacts are discussed in Appendix F.

Acknowledgments and Disclosure of Funding

This project is funded in part by Shanghai Artificial Intelligence Laboratory, Shanghai Innovation Institute, and the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK. Dahua Lin is a PI of CPII under InnoHK. The authors declare no competing interests.

References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: An Open Agentic Framework that Uses Computers Like a Human. *arXiv preprint arXiv:2410.08164*, 2024.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A Compositional Generalist-Specialist Framework for Computer Use Agents. *arXiv preprint arXiv:2504.00906*, 2025.

Anthropic. Claude computer use, 2024. URL `https://www.anthropic.com/news/3-5-models-and-computer-use`.

Anthropic. Claude's extended thinking, 2025. URL `https://www.anthropic.com/research/visible-extended-thinking`.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025b.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Xuetian Chen, Yinghao Chen, Xinfeng Yuan, Zhuo Peng, Lu Chen, Yuekeng Li, Zhoujia Zhang, Yingqian Huang, Leyan Huang, Jiaqing Liang, Tianbao Xie, Zhiyong Wu, Qiushi Sun, Biqing Qi, and Bowen Zhou. OS-Map: How Far Can Computer-Using Agents Go in Breadth and Depth? *arXiv preprint arXiv:2507.19132*, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. *arXiv preprint arXiv:2401.10935*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Ha and Jürgen Schmidhuber. World Models. *arXiv preprint arXiv:1803.10122*, 2018.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A Visual Language Model for GUI Agents. *CoRR*, abs/2312.08914, 2023. doi: 10.48550/ARXIV.2312.08914. URL `https://doi.org/10.48550/arXiv.2312.08914`.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. CogAgent: A Visual Language Model for GUI Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, et al. OS Agents: A Survey on MLLM-based Agents for General Computing Devices Use, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*, 2024.

Masao Ito. Mechanisms of motor learning in the cerebellum. *Brain research*, 886(1-2):237–245, 2000.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. ShowUI: One Vision-Language-Action Model for GUI Visual Agent. *arXiv preprint arXiv:2411.17465*, 2024.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. ARPO: End-to-End Policy Optimization for GUI Agents with Experience Replay. *arXiv preprint arXiv:2505.16282*, 2025.

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.

OpenAI. GPT-4 Technical Report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL `https://doi.org/10.48550/arXiv.2303.08774`.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents. *arXiv preprint arXiv:2408.07199*, 2024.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. *arXiv preprint arXiv:2411.02337*, 2024.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. *CoRR*, abs/2501.12326, 2025. doi: 10.48550/ARXIV.2501.12326. URL `https://doi.org/10.48550/arXiv.2501.12326`.

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. *arXiv preprint arXiv:2405.14573*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

David Silver and Richard S Sutton. Welcome to the Era of Experience. *Preprint of a chapter to appear in Designing an Intelligence, edited by George Konidaris, MIT Press (forthcoming)*, 2025.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Silvio Savarese, Zeyuan Chen, Jieyu Zhao, et al. CoAct-1: Computer-using Multi-Agent System with Coding Actions. *arXiv preprint arXiv:2508.03923*, 2025.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. OS-Genesis: Automating GUI Agent Trajectory Construction via Reverse Task Synthesis. *arXiv preprint arXiv:2412.19723*, 2024.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, et al. ScienceBoard: Evaluating Multimodal Autonomous Agents in Realistic Scientific Workflows. *arXiv preprint arXiv:2505.19897*, 2025a.

Zeyi Sun, Yuhang Cao, Jianze Liang, Qiushi Sun, Ziyu Liu, Zhixiong Zhang, Yuhang Zang, Xiaoyi Dong, Kai Chen, Dahua Lin, and Jiaqi Wang. CODA: Coordinating the Cerebrum and Cerebellum for a Dual-Brain Computer Use Agent with Decoupled Reinforcement Learning. *arXiv preprint arXiv:2508.20096*, 2025b.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. SEAgent: Self-Evolving Computer Use Agent with Autonomous Learning from Experience. *arXiv preprint arXiv:2508.04700*, 2025c.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. UI-TARS-2 Technical Report: Advancing GUI Agent with Multi-Turn Reinforcement Learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. OpenCUA: Open Foundations for Computer-Use Agents. *arXiv preprint arXiv:2508.09123*, 2025b.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. GUI-Actor: Coordinate-Free Visual Grounding for GUI Agents. *arXiv preprint arXiv:2506.03143*, 2025a.

Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Qiushi Sun, Zhaoyang Liu, Zhoumianze Liu, Yu Qiao, Xiangyu Yue, Zun Wang, and Zichen Ding. OS-Oracle: A Comprehensive Framework for Cross-Platform GUI Critic Models. *arXiv preprint arXiv:2512.16295*, 2025b.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. OS-Atlas: A Foundation Action Model for Generalist GUI Agents. *arXiv preprint arXiv:2410.23218*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials. *arXiv preprint arXiv:2412.09605*, 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. In *Conference on Language Modeling (CoLM)*, 2025. arXiv:2504.01382.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. GPT-4V in Wonderland: Large Multimodal Models for Zero-Shot Smartphone GUI Navigation. *arXiv preprint arXiv:2311.07562*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Bowen Yang, Kaiming Jin, Zhenyu Wu, Zhaoyang Liu, Qiushi Sun, Zehao Li, Jingjing Xie, Zhoumianze Liu, Fangzhi Xu, Kanzhi Cheng, Qingyun Li, Yian Wang, Yu Qiao, Zun Wang, and Zichen Ding. OS-Symphony: A Holistic Framework for Robust and Generalist Computer-Using Agent. *arXiv preprint arXiv:2601.07779*, 2026.

Danyang Zhang, Situo Zhang, Ziyue Yang, Zichen Zhu, Zihan Zhao, Ruisheng Cao, Lu Chen, and Kai Yu. ProgRM: Build Better GUI Agents with Progress Rewards. *arXiv preprint arXiv:2505.18121*, 2025.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a Generalist Web Agent, if Grounded. *arXiv preprint arXiv:2401.01614*, 2024.

A Overall algorithm

We provide the detailed training procedure of our self-calibrated reward framework, including the integration with GRPO and the robust experience-replay mechanism adapted from ARPO [Lu et al., 2025].

Algorithm 1 End-to-end multi-turn RL with reward from the self-calibrated IDM judge model.

Input: Policy π_θ, IDM M_{IDM}, Judge M_{Judge}, Task Set $\mathcal{T}$
Hyperparameters: Group size $G = 16$, Calibration factor $\alpha = 0.8$, Buffer Size N, Learning Rate η
Initialize Experience Replay Buffer $\mathcal{B} \leftarrow \emptyset$
for epoch $= 1$ to E **do**
  for each training step **do**
    // Step 1: Distributed Rollout
    Sample a batch of tasks $\tau \sim \mathcal{T}$
    for each task T_i in batch **do**
      Collect G trajectories $\{(s^k_{0:T}, a^k_{0:T})\}_{k=1}^{G}$ using π_θ
      for each trajectory k **do**
        // Step 2: Reward Computation
        Get Base Reward: $r^k_{\text{base}} \leftarrow M_{\text{Judge}}(s^k_{0:T}, T_i)$
        Verify Action Consistency:
          $\mathcal{C}^k \leftarrow 0$
        for step t in trajectory **do**
          $\mathcal{C}_t \leftarrow \frac{1}{2}\big(\mathbb{I}[\text{type}(\hat{a}_t) = \text{type}(a_t)] + 1 - \text{dist}(\hat{a}_t, a_t)\big)$
          $\mathcal{C}^k \leftarrow \mathcal{C}^k + \mathcal{C}_t/T$
        end for
        Final Reward: $R^k \leftarrow r^k_{\text{base}} \cdot (\mathcal{C}^k)^\alpha$
        // Step 3: Buffer Management (ARPO)
        if $R^k > \delta_{\text{success}}$ **then**
          Add successful trajectory to $\mathcal{B}$
        end if
      end for
    end for
    // Step 4: Policy Optimization with ARPO
    $D_{\text{train}} \leftarrow$ Current Batch of Groups
    $D_{\text{replay}} \sim \text{Sample}(\mathcal{B})$
    Mix batches: $D_{\text{optim}} \leftarrow D_{\text{train}} \cup D_{\text{replay}}$
    for each group $g \in D_{\text{optim}}$ **do**
      Calculate Advantage $A_k = \frac{R^k - \text{mean}(R^g)}{\text{std}(R^g) + \epsilon}$
      // Critical: ARPO Filtration
      if std$(R^g) \approx 0$ (All Success or All Failure) **then**
        continue {Skip groups with zero variance (no learning signal)}
      end if
      Update $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{GRPO}}(g)$
    end for
  end for
end for

B Implementation details

B.1 Detailed hyperparameters

We list the comprehensive hyperparameters used in our reinforcement learning and inverse-dynamic-modeling experiments in Table 9.

B.2 Benchmark specifications

Table 10 details the composition of the benchmarks used to evaluate I-Judge's generalization capabilities.

B.3 Reward-model evaluation set statistics

To rigorously evaluate the performance of our reward models, we constructed specific evaluation sets for OSWorld and ScienceBoard, adhering to the methodology established by AgentRewardBench [Lù et al., 2025]. This involves collecting trajectories from diverse agents and verifying them against ground-truth signals to ensure a robust assessment of the judge's capability to identify success and failure. The statistics of these datasets are reported in Table 11.

B.4 Environment setup

IDM and judge fine-tuning use 8×NVIDIA H800 GPUs. Each end-to-end RL run uses four 8-GPU NVIDIA H200 workers; 256 parallel environments run on CPU-only worker nodes.

B.5 Action space

Table 12 defines the unified action space used by our agent, mapping human-understandable actions to the executable API.

B.6 Action distance metrics

Table 13 details the specific distance metrics used to compute the calibration score for each action type.

C Qualitative case studies

We provide qualitative examples before and after I-Judge RL training, including transfer to ScienceBoard. We also show an instance where consistency down-weights a positive VLM judgment unsupported by the visible transition.

C.1 OSWorld: failure correction

Figure 5 illustrates a typical correction pattern. In this task, the agent is required to "Open the 'Privacy' settings page in Chrome".

Before training (failure). The SFT agent struggles with navigation. It initially hesitates, scrolling up and down the settings menu repeatedly (Steps 2–6). It misinterprets the "Safety Check" section as "Privacy" and attempts to click a sub-link that leads to a help page instead of the actual settings panel (Step 8). Finally, it gets lost in the help documentation and fails to maximize the correct window (Step 15).
Thought: "I need to find privacy settings. I see 'Safety Check', that sounds relevant. Let me click 'Check now'. Hmm, nothing happened. Maybe I should scroll down? I see 'Help'. Maybe the settings are in Help? Clicking Help..."

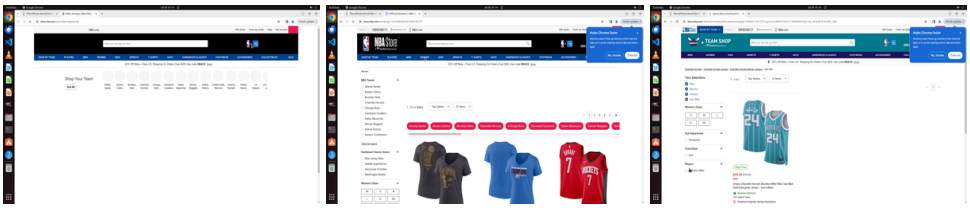
After training (success). The I-Judge-trained agent exhibits much more purposeful behavior. It immediately recognizes the "Settings" menu structure, navigates directly to the left sidebar (Step 2), locates "Privacy and security" (Step 7), and confirms the selection. The trajectory is significantly shorter (13 steps vs. 15+ steps) and action-oriented.
Thought: "Goal: Open Privacy settings. I see the Chrome Settings sidebar. 'Privacy and security' is clearly listed on the left. Moving mouse to (213, 457). Click. Page loaded. Task completed."

C.2 ScienceBoard transfer case

Figure 6 shows a transfer example on ScienceBoard. The task is "Identify the molecule with the highest dipole moment".



(a) Before training: the agent drifts into irrelevant help pages (failure).

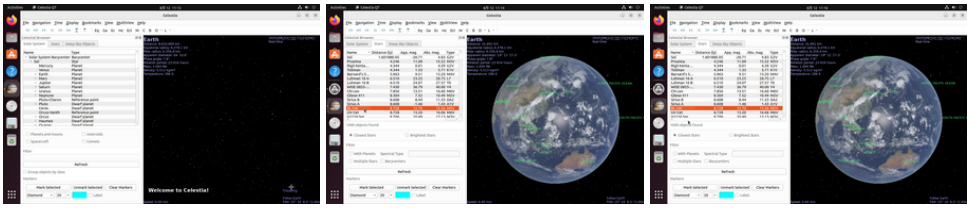


(b) After training: the agent directly navigates to the target setting (success).

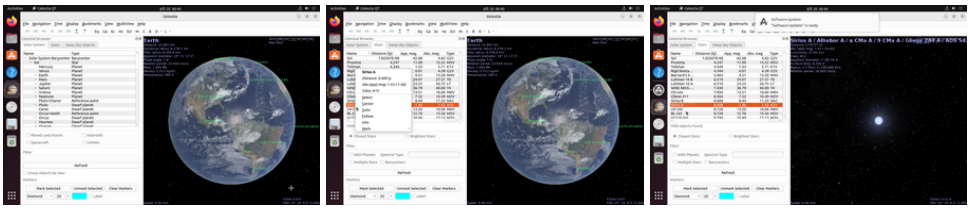
Figure 5: Comparison of agent behavior on an OSWorld task.

Baseline agent (failure). Without reinforcement learning, the agent fails to understand the structured scientific data. It attempts to search for "dipole moment" in a search bar (Step 4) rather than reading the data table directly. It eventually guesses a row randomly and fails.
Thought: "I don't see the answer directly. I will try to use the search function. Typing 'dipole moment'. No results found. This interface is confusing. I'll just click the first row."

I-Judge agent (success). The trained agent parses the table (Step 0), compares values, identifies the target row, and performs the selection action (Step 5).
Thought: "I need to find the highest dipole moment. Scanning the table... Row 1: 1.23D. Row 2: 0.87D. Row 3: 2.48D. Row 3 is highest. Moving to checkbox at (498, 307). Click. Done."



(a) Baseline agent: fails to parse the table, resorts to invalid search (failure).



(b) I-Judge agent: correctly scans the data table and selects target (success).

Figure 6: Generalization comparison on ScienceBoard (novel domain).

Table 9: Detailed hyperparameters for I-Judge training.

Hyperparameter	Value
IDM Pretraining	
IDM / Judge Backbones	Qwen2-VL-style 0.6B encoder / Qwen3-VL-8B judge
Training Data	500K executor-logged transitions + 16,592 AgentNet outcome trajectories [Wang et al., 2025b]
Optimizer	AdamW
Learning Rate	2×10^{-5}
Batch Size	128
Epochs	3
RL Policy Training (GRPO)	
Policy Backbone	Qwen3-VL
Experiment Framework	VERL / ARPO
Parallel Environments	256
GRPO Group Size / Curriculum Rollouts	16 / 16 per task
Total Training Epochs	15
Optimizer	AdamW
Peak Learning Rate	1×10^{-6}
LR Scheduler	Cosine Decay (Min LR 1×10^{-7})
Training Batch Size (Global)	64
Experience Replay Ratio	0.5
GRPO Clip Range (ϵ)	$0.2 \sim 0.3$
KL Penalty (β)	≈ 0
Reward Calibration (α)	0.8 (selected on OSWorld, then frozen)

Table 10: Statistics of evaluation benchmarks.

Benchmark	Description	Tasks
OSWorld	Standard generalist GUI benchmark covering common desktop apps (Office, Chrome, System).	369
OSWorld (Hard)	Filtered subset where "cheating" by simple state-checking is strictly prevented.	369
ScienceBoard	Scientific applications (molecular design and data analysis); IDM uses unlabeled transitions but no outcome labels.	120
AndroidWorld	Mobile applications; not used for reward-model training or hyper-parameter selection.	116

Table 11: Statistics of reward-model evaluation sets. We collected trajectories from diverse agents to ensure coverage of different failure modes and success patterns.

Dataset	Domain	Trajectories	Avg. Steps
AgentRewardBench [Lù et al., 2025]	Web	1,182	<50
OSWorld-Reward (Ours)	Desktop	1,413	50
ScienceBoard-Reward (Ours)	Scientific	833	50

Table 12: Action space definition.

Human Action	Action Description	Agent Action
Click	Click at a specific position	**click**(x, y, button)
Middle Click	Middle click at a specific position	**middleClick**(x, y)
Double Click	Double click at a specific position	**doubleClick**(x, y, button)
Triple Click	Triple click at a specific position	**tripleClick**(x, y, button)
Mouse Move	Move mouse to a specific position	**moveTo**(x, y)
Drag	Drag mouse from one position to another	**dragTo**(x, y)
Scroll	Scroll vertically or horizontally	**scroll**(dx, dy) / **hscroll**(dx, dy)
Type	Type a string of text	**write**(text)
Press	Press a specific key	**press**(key)
Hotkey	Perform a combination of keys	**hotkey**(key1, key2)
Wait	Wait for a few seconds	**wait**()
Terminate	End the task with success or failure	**terminate**('success' or 'failure')

Table 13: Reward computation for each action type in our GUI agent.

Action Type	Description	Distance-based Reward
click, left_single, right_single, hover	Click or hover on a location	Normalized L_1 distance between predicted and ground-truth coordinates
left_double, double_click	Double click on a region	Normalized L_1 distance between clicked coordinates
drag, select	Drag from start box to end box	Intersection over Union (IoU) between predicted and ground-truth boxes
type	Type textual input	Character-level BLEU score between predicted and ground-truth text
hotkey	Press multiple keys at once	Character-level BLEU score between predicted and ground-truth key combinations
press	Press a single key	Character-level BLEU score between predicted and ground-truth key
scroll	Scroll in a certain direction	Character-level BLEU score between predicted and ground-truth direction
move_mouse	Move mouse to a specific location	Normalized L_1 distance between predicted and ground-truth coordinates
highlight	Highlight a rectangular UI region	IoU between predicted and ground-truth region
copy, paste	Clipboard operations	BLEU score between copied/pasted content
wait	Explicit wait command	Fixed reward $+1$
finished, finish_task	Finish current task/trajectory	Fixed reward $+1$

C.3 Reward model calibration analysis

We demonstrate how calibration can suppress a visually unsupported positive judgment. In this example, the agent attempts to "Clear browsing history".

Scenario. The agent clicks a "Clear data" button but a popup blocks the view, and the action fails to register on the UI backend due to a coordinate mismatch. However, visual changes (popup appearing) confuse the standard VLM judge.

Standard VLM judge output (incorrect)

Reasoning: The agent navigated to the settings page. The final screenshot shows a "Clear browsing data" popup window, which implies the agent has successfully initiated the clearing process. The user instruction is fulfilled.
Judgment: Success

I-Judge (IDM-enhanced) output (correct)

Step-by-step analysis:
- Step 4 ($S_3 \rightarrow S_4$): Action = click(point='503 397') $\rightarrow$ *IDM consistency:* High (menu opened).
- Step 5 ($S_4 \rightarrow S_5$): Action = click(point='597 503') $\rightarrow$ *IDM consistency:* **Low ($<$0.08)**. The screen state S_5 is identical to S_4 except for a cursor blink. The intended "confirm clear" action did not trigger a page transition or loading indicator.

Final Judgment: Failure (due to low action consistency in the final step).

The IDM's low consistency score ($\mathcal{C} \approx 0$) identifies that the visible transition does not support the logged critical action, down-weighting the VLM's positive judgment ($R = 1 \cdot 0^{0.8} = 0$).

D Prompt details

We provide the full system prompts used for the different judge models.

D.1 Baseline judge (Qwen3-VL-8B LS)

System prompt (baseline)

You are a GUI Reward Model Evaluator.
Your goal is to determine if a GUI agent has successfully completed a user instruction based on the final screen state.
You will be provided with:

1. The User Instruction (Goal).
2. The Final Screenshot (State t).

You must analyze the screen and provide a judgment:
Step 1: Analysis

- Analyze the content of the Final Screenshot.
- Check if the visual evidence confirms that the User Instruction has been fulfilled.

Step 2: Judgment

- If the task is successfully completed, conclude with "Judgment: Success".
- If the task is incomplete or failed, conclude with "Judgment: Failure".

Format your output as:
Reasoning: <Your detailed reasoning here>
Judgment: <Success/Failure>

User prompt template

User Instruction: {instruction}
Assess the following final screen: <Image Input>

D.2 Standard trajectory judge (SFT)

System prompt (Trajectory SFT)

You are an Expert GUI Agent Evaluator.
Your goal is to determine if a GUI agent has successfully completed a user instruction by analyzing its entire action trajectory.
You will be provided with:

1. The User Instruction (Goal).
2. A sequence of Screenshots (S_0 to S_t) showing the agent's progress.

You must analyze the full sequence and provide a judgment:
Step 1: Trajectory Analysis

- Briefly track the agent's progress through the screenshots.
- Identify key actions taken by the agent.
- Verify if the final state (last screenshot) achieves the goal.

Step 2: Judgment

- If the task is successfully completed, conclude with "Judgment: Success".
- If the task is incomplete, failed, or the agent got stuck, conclude with "Judgment: Failure".

Format your output as:
Reasoning: <Your step-by-step analysis of the trajectory>
Judgment: <Success/Failure>

User prompt template

User Instruction: {instruction}
Here is the execution trajectory:
Screenshot 0 (Step 0): <Image>
Screenshot 1 (Step 1): <Image>
...
Based on the functionality shown in the final screenshot above, has the task been completed?

D.3 I-Judge (IDM-enhanced)

System prompt (I-Judge/ IDM)

You are an Advanced Inverse Dynamic Module (IDM) and Evaluator.
Your goal is to reconstruct the agent's actions by observing the screen transitions and then judge if the task was completed.
You will be provided with:

1. The User Instruction (Goal).
2. A sequence of Screenshots (S_0 to S_t).

You must complete the following analysis:
Step 1: Inverse Dynamics (Action Inference) For each transition (e.g., $S_0 \rightarrow S_1$), you must:

- Describe the visual difference.
- Infer the exact action taken.

Step 2: Final Judgment

- Based on the implied action history and the final state (S_t), determine if the goal is satisfied.

Format your output as:
Step-by-Step Analysis:
- Step 1 ($S_0 \rightarrow S_1$): Action = <Inferred Action>
- Step 2 ($S_1 \rightarrow S_2$): Action = <Inferred Action>
...
Final Judgment: <Success/Failure>

User prompt template

User Instruction: {instruction}
Reconstruct the actions throughout this trajectory:
Screenshot 0 (State 0): <Image>
Screenshot 1 (State 1): <Image>
...
Analyze the steps above and provide your Final Judgment.

E Agent policy prompts

E.1 Qwen3-VL-Agent (base agent)

System prompt (Qwen3-VL-Agent)

Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools> < /tools> XML tags:
<tools>
{
"type": "function",
"function": {
"name": "computer_use",
"description": "Use a mouse and keyboard to interact with a computer... [Detailed instructions omitted for brevity]",
"parameters": { ... }
}
}
</tools>

For each function call, return a json object with function name and arguments within <tool_call> < /tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

Response format

Response format for every step:
1) Action: a short imperative describing what to do in the UI.
2) A single <tool_call>...</tool_call> block containing only the JSON.

Rules:
- Output exactly in the order: Action, <tool_call>.
- Be brief: one sentence for Action.
- Do not output anything else outside those parts.

User prompt template

Please generate the next move according to the UI screenshot, instruction and previous actions.
Instruction: {instruction}
Previous actions:
{previous_actions_str}

E.2 UI-TARS (Ours)

System prompt (UI-TARS)

You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.
Output Format
You should first think about the reasoning process in the mind and then provide the user with the answer. The reasoning process is enclosed within <think> < /think> tags.
After the <think> tags, you should place the final answer, which concludes your summarized thought and your action.
For example,
`<think>detailed reasoning content here</think>`
`Thought: a small plan...`
`Action: ...`
Action Space
click(point='<point>x1 y1</point>')
left_double(point='<point>x1 y1</point>')
right_single(point='<point>x1 y1</point>')
drag(start_point='<point>x1 y1</point>', end_point='<point>x2 y2</point>')
hotkey(key='ctrl c')
type(content='xxx')
scroll(point='<point>x1 y1</point>', direction='down or up or right or left')
wait()
finished(content='xxx')
Note
- Use {language} in 'Thought' part.
- Write a small plan and finally summarize your next action in one sentence in 'Thought' part.

F Broader impacts and LLM usage disclosure

Broader Impacts. I-Judge targets a core bottleneck in training reinforcement-learning-based GUI agents—the lack of reliable, scalable reward signals—by replacing the dependence on costly human annotation with self-supervised inverse-dynamics modeling. Positive impacts include lowering the barrier to using complex specialized software (e.g., scientific tools), accelerating end-to-end RL research on long-horizon agentic tasks, and reducing the need for human-collected demonstrations through automated trajectory synthesis. Potential negative impacts mirror those of computer-use agents in general: more capable autonomous agents could automate harmful UI actions if deployed without oversight, and the same trajectory-generation pipeline could in principle be repurposed to scrape interfaces or perform automation that violates platform terms of service. Our experiments are confined to research-oriented benchmarks (OSWorld, ScienceBoard, AndroidWorld) running inside sandboxed virtual machines, and we do not release tools targeted at production deployment. We encourage follow-up work to incorporate explicit refusal behaviors and human-in-the-loop oversight when deploying I-Judge-style reward models in safety-critical settings.

LLM Usage Disclosure. We leveraged Large Language Models (LLMs) to aid in the preparation and polishing of this manuscript. The use of these models was confined to the role of a sophisticated proofreading tool: correcting grammatical errors, refining sentence structure for clarity, and improving overall readability. All scientific content—including research ideation, experimental design, data analysis, and conclusions—was generated by the human authors. Every suggestion from the LLM was critically evaluated and the authors performed multiple rounds of review and revision and take full responsibility for the final version of the text. The LLM was not used to generate any core scientific insights or substantive parts of the methodology of the proposed I-Judge framework. In addition, several VLMs (Qwen3-VL-8B, UI-TARS-1.5-7B, Qwen-2.5-32B) are core components of the methodology—as the base policy, IDM/judge backbone, and task proposer respectively—and their roles, prompts, and training details are documented in Section 4 and Appendix D.

G Implementation Details for Comparison with Prior Reward Models

The main quantitative comparison with ProgRM [Zhang et al., 2025] and VPT-IDM [Baker et al., 2022], including reward-model precision and end-to-end RL success rates over five seeds, is presented in the main paper (Section 4.4, Table 5). This section provides the implementation details omitted there for space.

ProgRM re-implementation. We re-implement ProgRM's LCS-based self-annotation procedure following the public reference implementation released by [Zhang et al., 2025], but retrain it on the 16,592 AgentNet [Wang et al., 2025b] trajectories used by our supervised component. Using the public WikiHow-Android checkpoint directly would conflate the algorithmic comparison with a desktop-vs-mobile domain gap; matching training data ensures the gap reported in Table 5 reflects the underlying recipe-library mechanism. We use the same architecture (Qwen3-VL-8B backbone), optimiser, and learning-rate schedule as our Direct-FT baseline.

VPT-IDM re-implementation. We follow [Baker et al., 2022] and train an action predictor on a 2K action-labelled subset of AgentNet plus the 500K unlabelled rollouts collected for our IDM pretraining. The reward at inference time is the IDM's action log-likelihood averaged over the trajectory, normalised to $[0, 1]$ per group of GRPO rollouts. We deliberately make this comparison generous to VPT-IDM: it sees the same 500K unlabelled corpus we use, and we tune the temperature on a held-out validation split to maximise downstream SR.

Identical RL settings. All entries in Table 5 use the same Qwen3-VL-8B base policy, ARPO/GRPO [Lu et al., 2025, Shao et al., 2024] optimiser, 256 parallel rollout environments, replay buffer of 2,000 trajectories, and 5 random seeds (seeds 0–4). The relative ranking is stable across the reported seeds.

H Annotation Cost Analysis

We explicitly decompose the human-annotation cost of the supervised component for each method considered in Table 5 to make the cost–quality trade-off precise. We distinguish three label types: **Action labels** (per step, per trajectory; $\sim\$5$/traj at typical vendor pricing for verifying ~ 50 actions), **Outcome / success labels** (one binary judgement per trajectory; $\sim\$0.50$/traj), and **Step / process labels** (per-step success, the most expensive form, requiring an annotator to inspect every screen; $\sim$ \$8/traj). Per-trajectory prices are conservative industry estimates from public crowdsourcing platforms; absolute numbers may shift with provider but the *relative* ordering is robust.

Table 14: **Annotation cost decomposition for different reward modelling methods.** "Auto-emitted" actions in the I-Judge row come from the agent's own `pyautogui` execution log during sandbox rollouts, requiring *zero* human inspection. I-Judge thus achieves the supervision quality of an inverse-dynamics-augmented reward model at the same dollar cost as a basic outcome reward model.

Method	Action label	Outcome label	Step / process label	**Total \$**
LS Judge (zero-shot)	—	—	—	\$0
Direct FT (ORM)	—	16,592 × \$0.50	—	\$8,296
Process RM (hypothetical)	—	16,592 × \$0.50	16,592 × \$8	\$140,992
VPT-IDM [Baker et al., 2022]	2,000 × \$5	—	—	\$10,000
I-Judge (Ours)	500,000, *auto-emitted*	16,592 × \$0.50	—	**\$8,296**

Key observation. I-Judge uses the same human-label budget as a vanilla outcome reward model ($\sim$\$8K) while adding transition-level evidence. The 500K IDM examples are generated during sandbox rollouts, and their action targets come directly from executor logs rather than human annotation.

I Cross-Domain IDM Transferability

An important question for any IDM-based reward model is whether its gain originates from genuine *transferable* understanding of GUI dynamics or from *environment-specific memorisation* (since the

IDM is trained on rollouts from the same environments later used for RL). To disambiguate these two hypotheses, we ablate the IDM's pretraining corpus by restricting it to a *single* domain at a time and measuring both reward-model precision and downstream RL success on all three benchmarks.

Table 15: **Cross-domain IDM transferability.** Reward-model precision (P, %) and end-to-end RL success rate (SR, %; mean ± std over 5 seeds) when the IDM is trained on trajectories from only a single environment. Cells annotated "OOD" indicate that the IDM has *never* seen any trajectory from the corresponding evaluation environment. The *no-IDM* and *full-mix* rows reuse the numbers from Tables 1 and 2.

IDM Pretrain Source	Reward-Model Prec.		End-to-End RL Success Rate (%)		
	OSW-R	SB-R	OSWorld	ScienceBoard	AndroidW. (OOD)
no IDM (Direct FT)	72.3	56.2	$34.1_{\pm 0.8}$	$14.7_{\pm 1.9}$	$47.0_{\pm 1.1}$
OSWorld only (300K)	78.6	63.4 (OOD)	$37.5_{\pm 0.7}$	$18.2_{\pm 1.5}$ (OOD)	$51.7_{\pm 1.0}$
ScienceBoard only (200K)	70.8 (OOD)	76.5	$34.6_{\pm 0.8}$ (OOD)	$22.8_{\pm 1.2}$	$51.2_{\pm 1.0}$
Full mix (500K, current)	**79.3**	**79.2**	$\mathbf{38.6_{\pm 0.5}}$	$\mathbf{23.8_{\pm 1.0}}$	$\mathbf{54.3_{\pm 0.7}}$

Discussion. Two patterns emerge from Table 15. **(1) In-domain dominance.** For each single-domain IDM, reward precision in its training environment approaches that of the full-mix model (e.g., OSWorld-only IDM: 78.6 vs. 79.3 on OSWorld-R). **(2) Cross-domain transfer.** When the IDM has not observed trajectories from the evaluation environment, it still outperforms the no-IDM baseline in these experiments (e.g., OSWorld-only IDM: $18.2_{\pm 1.5}$ vs. $14.7_{\pm 1.9}$ on ScienceBoard and $51.7_{\pm 1.0}$ vs. $47.0_{\pm 1.1}$ on AndroidWorld). This supports partial transfer of action–transition features in the evaluated settings. The full-mix model is best on every cell, while single-domain models retain part of the gain on held-out environments, supporting transfer of learned action–transition regularities in the evaluated settings.

J Action Equivalence Stress Test

By construction, the action consistency score $\mathcal{C}(\tau) = \frac{1}{2T}\sum_t \big(\mathbb{I}[\text{type}(\hat{a}_t) = \text{type}(a_t)] + (1 - \text{dist}(\hat{a}_t, a_t))\big)$ relies on *exact* action-type matching: in real GUIs, multiple syntactically distinct actions can produce the same visual state transition (e.g., pressing `Enter` vs. clicking a `Submit` button; clicking a label vs. clicking the adjacent radio button). If the IDM consistently predicts one canonical form, the agent could in principle be penalised for using the equivalent alternative, stifling exploration of valid but unconventional strategies. We empirically characterise the magnitude of this effect below.

Stress-test set construction. We manually curate 155 trajectory pairs across five categories of semantically equivalent action substitutions, drawn from OSWorld and ScienceBoard. Each pair consists of a *canonical* trajectory τ^c (the action style the IDM has likely seen most often during pretraining, e.g., "click Submit") and an *equivalent* trajectory τ^e achieving an identical visual outcome via a different action (e.g., "press Enter"). Both trajectories complete the same task verified by ground-truth checkers. For each pair we report the mean consistency score $\bar{\mathcal{C}}$ predicted by the IDM, and the resulting *reward gap* $\Delta R = R_{\text{final}}(\tau^c) - R_{\text{final}}(\tau^e)$ at our default $\alpha = 0.8$.

Table 16: **Action equivalence stress test.** Mean consistency on canonical and equivalent trajectories and the resulting reward gap at $\alpha = 0.8$. Larger drops expose sensitivity to syntactically different but semantically equivalent actions.

Equivalence Category	# Pairs	$\bar{\mathcal{C}}(\tau^c)$	$\bar{\mathcal{C}}(\tau^e)$	$\Delta\mathcal{C}\downarrow$	$\Delta R\downarrow$
Click on equivalent UI elements (label ↔ radio)	50	0.83	0.71	0.12	0.08
Keyboard shortcut vs. button click (Ctrl+S vs. Save button)	30	0.84	0.62	**0.22**	**0.18**
Press Enter vs. click Submit	25	0.85	0.78	0.07	0.05
Drag-select vs. Shift-click range selection	20	0.81	0.59	**0.22**	**0.18**
Type+Tab vs. Type+Enter (form-field fill)	30	0.83	0.74	0.09	0.06
Mean (weighted by # pairs)	155	0.83	0.69	**0.14**	**0.11**

Discussion. The mean $\Delta\mathcal{C} = 0.14$ corresponds to an average reward gap of $\Delta R = 0.11$ on a $[0, 1]$-normalised reward scale. While non-zero, this gap is far below the threshold that would convert

a successful trajectory into a perceived failure—at $\alpha = 0.8$, the equivalent trajectory still receives a calibrated reward of $\bar{R} \approx r_{\text{base}} \cdot 0.69^{0.8} \approx 0.74\, r_{\text{base}}$, retaining roughly 74% of the canonical reward. Empirically, this level of penalty does not prevent the GRPO learner from discovering equivalent strategies: across the 5-seed runs underlying Table 2, we observe agents trained with I-Judge spontaneously adopting keyboard shortcuts and Shift-click range selections at later training stages (case studies in Appendix C). The two highest-penalty categories—keyboard shortcuts and drag-select alternatives—both involve actions whose visual signature is strictly weaker (a single keystroke produces no cursor movement; a Shift-click produces no drag rectangle), making them inherently harder for any visual IDM to recognise. We acknowledge this as a residual limitation of the current hard-coded distance metric and view it as an opportunity for future work: replacing the indicator $\mathbb{I}[\text{type}(\hat{a}_t) = \text{type}(a_t)]$ with a learnable action equivalence embedding (e.g., trained on automatically mined equivalence pairs) would directly close the worst-case gaps while preserving the rest of the framework.

K Additional Camera-Ready Comparisons

K.1 ProgRM native WikiHow protocol

We also evaluate under ProgRM's public 150-task WikiHow split, Mobile-Env environment, and programmatic evaluator. ProgRM's checkpoint and exact trajectories are unavailable, so its published Table 2 rows are marked as references; our Qwen2.5-7B actor and reward model use newly generated trajectories on the same public tasks and evaluator. The decision threshold is selected on the WikiHow training split before test evaluation.

Table 17: **WikiHow comparison under ProgRM's native public protocol.** All entries are percentages; published and newly generated trajectories are not treated as paired samples.

Reward model	TP	FN	TN	FP	Prec.	Recall	Acc.
ORM (published)	54.00	4.67	19.33	22.00	71.05	92.04	73.33
ProgRM-LCS (published)	57.33	2.00	33.33	7.33	88.66	96.63	90.67
ProgRM-Env (published)	57.33	4.67	36.67	1.33	97.73	92.47	94.00
I-Judge, no calibration	59.33	0.67	38.67	1.33	97.81	98.88	98.00
I-Judge, full	59.00	1.00	39.33	0.67	**98.88**	98.33	**98.33**

K.2 Direct reward-model comparison

Table 18 evaluates reward models on the same AgentRewardBench trajectories and official protocol. For DigiRL and WebRL, we retain each method's reward construction but replace the original external judge with Qwen2.5-VL-7B to avoid conflating reward design with older backbones. Published closed-model and rule-based rows are references rather than backbone-matched comparisons.

Table 18: **AgentRewardBench comparison.** Precision, recall, and F1 are percentages. The asterisk marks the official rule-based reference, whose high precision comes with low recall.

Reward model / judge	Backbone	Prec.	Recall	F1
Rule-based*	—	83.8	55.9	67.1
AER-C	GPT-4o	67.7	71.9	69.7
AER-V	GPT-4o	67.6	71.5	69.5
NNetNav	Llama-3.3-70B	52.5	82.4	64.1
Claude 3.7 Sonnet judge	Claude 3.7 Sonnet	69.4	76.3	72.7
GPT-4o judge	GPT-4o	68.1	80.3	73.7
GPT-4o Mini judge	GPT-4o Mini	64.5	78.3	70.8
Qwen2.5-VL judge	Qwen2.5-VL	64.5	86.1	73.7
SEAgent [Sun et al., 2025c]	Qwen2.5-VL-7B	71.6	84.5	77.5
DigiRL reward method [Bai et al., 2024]	Qwen2.5-VL-7B	65.1	79.4	71.5
WebRL reward method [Qi et al., 2024]	Qwen2.5-VL-7B	68.3	80.2	73.8
WebJudge [Xue et al., 2025]	Qwen2.5-VL-7B	75.3	82.5	78.7
I-Judge	Qwen2.5-VL-7B	**77.9**	**88.1**	**82.7**
I-Judge	Qwen3-VL-8B	79.1	88.6	83.6

K.3 Integration with complete RL systems

We instantiate each system from the same Qwen3-VL-8B initialization and match task splits and interaction budgets. "+ I-Judge" changes only the reward path, preserving each system's curriculum, replay, and optimization procedure.

Table 19: **Complete-system comparison.** End-to-end success rate (%).

Method	OSWorld	ScienceBoard	AndroidWorld
Qwen3-VL-8B baseline	33.4	13.0	49.4
DigiRL [Bai et al., 2024]	37.2	15.2	52.2
+ I-Judge	38.1	17.9	54.1
WebRL [Qi et al., 2024]	37.7	13.9	51.5
+ I-Judge	39.0	18.5	53.4
SEAgent [Sun et al., 2025c]	39.3	17.2	53.8
+ I-Judge	**40.5**	**23.9**	**56.1**

K.4 Difficulty and component decomposition

Difficulty filtering is an RL-curriculum device, not an IDM data-selection rule. To test whether task difficulty nevertheless affects IDM quality, we train on matched, de-duplicated Easy-, Medium-, and Hard-only corpora for five runs per bin. Differences remain within run-to-run variation.

Table 20: **IDM source-task difficulty.** Mean±standard deviation over five runs. P/R/F1 are measured on OSWorld-Reward (OSW-R) and ScienceBoard-Reward (SB-R).

Corpus	Action acc.	Coord. err.	Text sim.	OSW-R P/R/F1	SB-R P/R/F1
Easy	$81.1_{\pm 0.4}$	$7.99_{\pm 0.95}$	$93.3_{\pm 0.8}$	76.1/82.5/79.2	72.9/80.1/76.3
Medium	$80.9_{\pm 0.3}$	$7.88_{\pm 0.98}$	$94.5_{\pm 1.2}$	75.9/83.3/79.4	73.2/79.5/76.2
Hard	$81.4_{\pm 0.3}$	$8.12_{\pm 1.14}$	$93.4_{\pm 0.9}$	76.2/82.1/79.0	73.9/78.8/76.3
Unfiltered	81.2	7.95	92.3	75.5/83.0/79.1	73.3/79.9/76.5

Finally, Table 21 applies runtime calibration to judges trained without the IDM auxiliary objective, isolating the runtime effect. Malformed joint outputs are filtered by parseability only, and collection continues until each group has 16 valid samples; we report the resulting raw-rollout overhead.

Table 21: **Runtime calibration without IDM auxiliary training.** UI-TARS-1.5-7B policy; success rate is mean±standard deviation over five seeds.

Judge	Calibration	OSWorld	ScienceBoard	AndroidWorld
Direct FT	no	$29.2_{\pm 0.8}$	$11.0_{\pm 1.7}$	$52.6_{\pm 1.0}$
Direct FT	yes (∼9.9% overhead)	$31.1_{\pm 0.4}$	$13.1_{\pm 1.2}$	$53.4_{\pm 0.6}$
LS	no	$26.7_{\pm 0.9}$	$9.1_{\pm 1.3}$	$49.8_{\pm 1.2}$
LS	yes (∼7.5% overhead)	$28.5_{\pm 0.7}$	$14.8_{\pm 1.1}$	$52.8_{\pm 0.9}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state our three contributions—unsupervised knowledge acquisition via IDM, a self-calibrated soft reward, and empirical verification on three benchmarks—which match the methods presented in Section 3 and the experimental results in Section 4. Aspirational extensions (scaling IDM to millions of trajectories, unifying perception–action–verification) are explicitly framed as future work in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated "Limitations" paragraph is included in the conclusion (Section 5), and additional discussion appears in Appendix F. We discuss the bounded benchmark coverage (OSWorld, ScienceBoard, AndroidWorld), the dependence of the IDM signal on the diversity of the unlabeled trajectory pool, and the need for interactive sandboxed VM environments to collect rollouts.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain formal theorems requiring proof. The information-bottleneck discussion in Section 3 is an interpretive analysis used to motivate the IDM objective; the GRPO objective is reproduced from prior work [Shao et al., 2024, Guo et al., 2025] and used as a training algorithm rather than as a theoretical claim.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 describes the IDM objective, action consistency score, and self-calibrated GRPO update. Section 4 and Appendix B report hyperparameters, data sources, the $\alpha = 0.8$ validation protocol, training framework, hardware, and prompt templates. Algorithm 1 provides end-to-end pseudocode.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We commit to releasing the code, IDM-trained reward model, trained agent checkpoints, and data-generation instructions. The source benchmarks are publicly available, and the synthetic trajectory pipeline and dataset sizes are documented so that the corpus can be regenerated.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Hyperparameters (batch sizes, learning rates, schedules, epoch counts), GRPO group size G, KL coefficient β, clipping range ϵ, the calibration factor α, the IDM-pretraining data scale, and the train/test split strategy (in-domain OSWorld training; out-of-domain ScienceBoard and AndroidWorld evaluation) are all reported in Section 4 and Appendix B. The choice of $\alpha = 0.8$ is supported by the ablation in Table 3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Main end-to-end results report the mean and standard deviation over five independent random seeds (0–4), and the difficulty ablation reports the same statistics over five runs per source-task bin. We state the source of variation in each table caption.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 and Appendix B report the hardware for both training and rollout: end-to-end RL is conducted on 4×8 NVIDIA H200 GPUs with 256 parallel OSWorld environments hosted on CPU-only worker nodes, and IDM/judge fine-tuning runs on $8\times$NVIDIA H800 GPUs. Training schedules (3 epochs for IDM pretraining, 15 epochs for GRPO) and batch sizes are also specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly released benchmarks and software within sandboxed VM environments and does not involve human subjects or sensitive personal data. The authors have read the NeurIPS Code of Ethics and confirm conformance.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: A dedicated "Broader Impacts" paragraph in Appendix F discusses both positive impacts (lower barrier to specialized scientific software, accelerating end-to-end RL research on long-horizon agents, reduced demand for human demonstration data) and potential negative impacts (autonomous misuse if deployed without oversight, possible terms-of-service violations from automated interaction with web/desktop interfaces).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The released artifact is an IDM-augmented reward model and a GUI agent fine-tuned with that reward, both of which operate only inside sandboxed virtual machines. They are not general-purpose generative models with elevated misuse potential beyond their open-source bases (Qwen3-VL, UI-TARS-1.5). We do not introduce any new scraped datasets that pose elevated misuse risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All external assets are cited at first use: Qwen3-VL [Bai et al., 2025a,b], UI-TARS [Qin et al., 2025, Wang et al., 2025a], GUI-Actor [Wu et al., 2025a], ARPO [Lu et al., 2025], OSWorld [Xie et al., 2024], ScienceBoard [Sun et al., 2025a], AndroidWorld [Rawles et al., 2024], AgentRewardBench [Lù et al., 2025], AgentNet [Wang et al., 2025b], GRPO [Shao et al., 2024]. These assets are used in accordance with their licenses (Apache-2.0 / Qwen / MIT / research-use as appropriate); benchmark software is open-source and used only for non-commercial research.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new assets we plan to release—the trained I-Judge (IDM-augmented) reward model checkpoint, the OSWorld-Reward and ScienceBoard-Reward evaluation sets (Table 11), the autonomous task-proposal pipeline, and the trained agent checkpoints—are documented in Section 3 and Appendix B–D. License information and a model card will be provided in the released anonymized package.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects; all reward signals are produced by automated judge models, and human-labeled trajectories used in training are taken from publicly available, previously released datasets (e.g., AgentNet [Wang et al., 2025b]).

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

Justification: The core methodology, experimental design, and scientific claims were proposed by the authors. LLMs were not used to generate or decide the methods, nor did they affect the scientific rigor or originality of the research; any LLM assistance was limited to writing, editing, or formatting.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.