LEARN FROM MOVE: CAN GUI AGENTS LEARN HOW THEIR ENVIRONMENTS WORK?

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI benchmarks increasingly emphasize longer workflows, but task horizon alone does not capture whether agents can learn how an environment responds to their actions. We introduce LATENTGUIWORLD, a benchmark of six tasks in three matched pairs organized by the evidence needed for control: I_0 uses prior knowledge and available observations, I_1 requires interaction that directly reveals hidden semantics, and I_2 identifies dynamics from action–response relationships. Frontier models struggle especially on I_2 tasks. We introduce MOVE, which separates movement from button events and exposes intermediate visual feedback, together with demonstrations of iterative exploration and control. Ablations show that both action granularity and multi-step demonstrations improve performance. After SFT, LatentLearner-9B achieves 50.0% overall success on LATENTGUIWORLD, comparable to several frontier models. Joint RL further raises egocentric Drag and Ten-Choice success to 92.0% and 75.3%. Behavioral analysis reveals multi-step exploration. In egocentric Drag, RL increases Pearson correlation between error-normalized commands and ideal gain compensation from 0.610 to 0.803. Together, these behaviors show that models learn to explore through interaction and adapt their control to environment dynamics. The same format improves GUI grounding after RL, reaching 67.37% mean accuracy across five benchmarks versus 66.72% for conventional RL.

1 INTRODUCTION

Multimodal agents combine visual perception, language understanding, and action to assist with a broad range of tasks. Computer use is a practical setting for these capabilities, spanning web browsing, document editing, and workflows across applications (Xie et al., 2024). GUI agents carry out such tasks by interpreting screenshots and operating graphical user interfaces through mouse and keyboard actions, using the same interfaces available to human users.

As GUI agents tackle more complex tasks, benchmarks increasingly emphasize longer workflows and sustained execution across applications (Xie et al., 2024; Yuan et al., 2026). Yet completing long sequences of screen-based actions remains costly and unreliable (Song et al., 2026; Zhou et al., 2026). These difficulties have motivated hybrid agents that combine GUI interaction with command-line execution, code, or application tools. CoAct-1 and OSWorld-MCP show how programmatic actions can reduce reliance on lengthy GUI sequences and improve task completion (Song et al., 2026; Jia et al., 2026). A matched GUI–CLI study likewise finds that CLI agents with verifier-guided skill augmentation outperform screen-only GUI agents (Zhou et al., 2026). This shift motivates evaluating visual interaction beyond workflow length: what can an agent learn by acting on an unfamiliar interface and observing its response?

Coding agents investigate unfamiliar repositories by reading code, running programs, and using test feedback to guide subsequent changes (Yang et al., 2024). GUI agents face an analogous need when the correct action depends on properties that are absent from the current screenshot. Consider a rotation task whose response direction and sensitivity vary across episodes. An agent may know the goal and possible control rules, yet still need to move, observe the resulting rotation, and infer how to complete the task. Existing benchmarks include information acquisition and adaptation, but seldom isolate the evidence required to identify such episode-specific properties. We therefore study whether

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

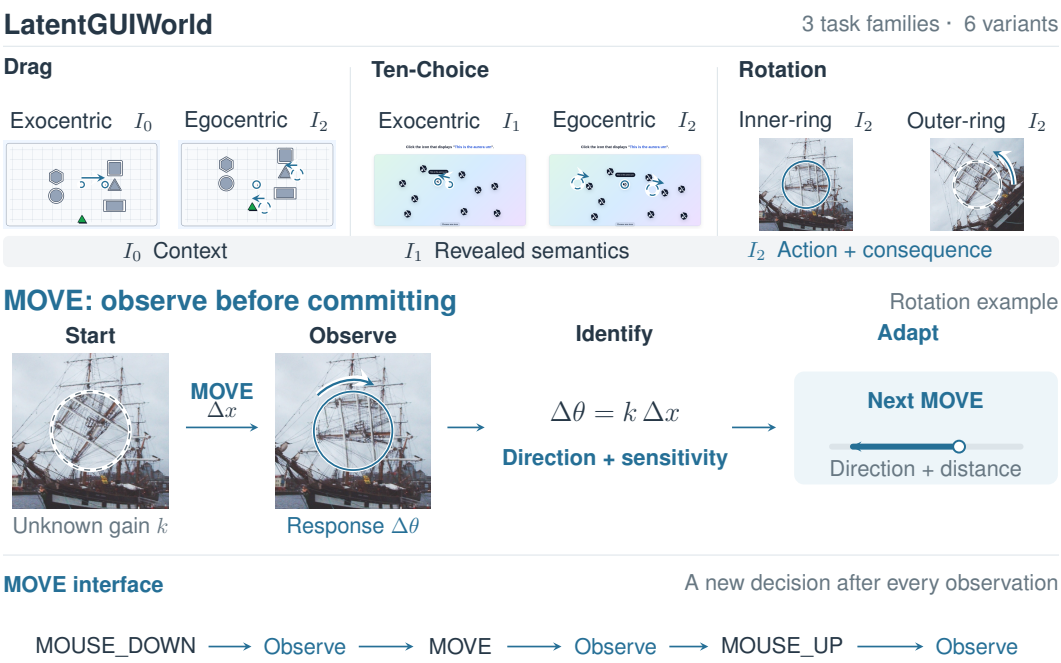


Figure 1: **Task pairs and feedback before commitment.** Matched tasks span I_0–I_2; MOVE exposes feedback after each primitive. Frames and overlays follow replayed interactions; gain inference and the next action are schematic.

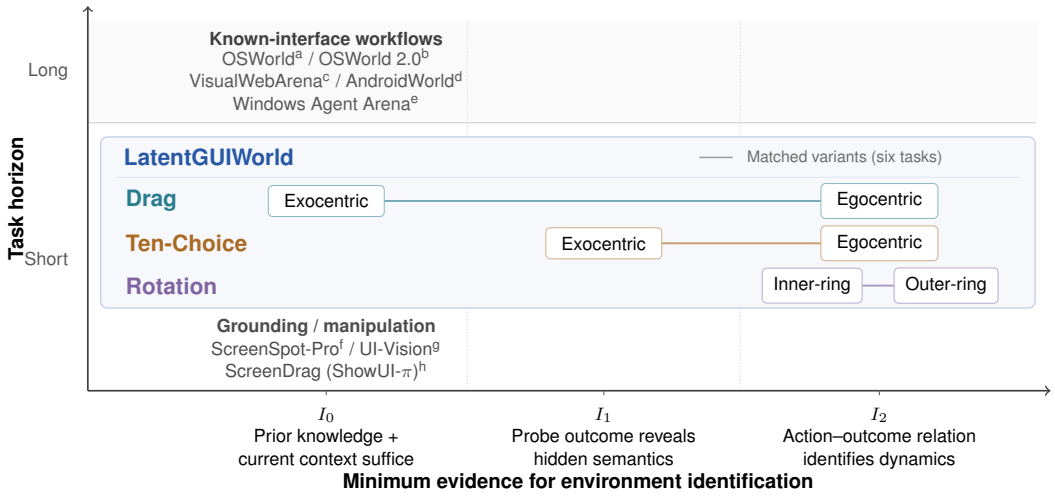
GUI agents can identify these hidden properties from action-induced visual feedback and use what they learn to guide subsequent control.

We introduce LATENTGUIWORLD, a multimodal benchmark of six short GUI tasks organized into three matched pairs. Its taxonomy classifies tasks by whether sufficient knowledge is available before a task-directed action, and by how missing knowledge must be acquired: I_0 uses prior knowledge and observed context; I_1 requires a probe outcome that directly reveals hidden semantics; and I_2 requires interpreting a visual transition together with the action that caused it to identify hidden dynamics. The benchmark pairs exocentric Drag (I_0) with egocentric Drag (I_2), exocentric Ten-Choice (I_1) with its egocentric variant (I_2), and two Rotation geometries (I_2–I_2). These pairs preserve task objectives while varying the interaction mechanism, making environment identification an explicit difficulty dimension alongside task horizon.

Our evaluations reveal a fundamental limitation of current GUI agents: performance degrades sharply when successful control requires inferring hidden environment properties from interaction feedback. We address this by introducing a fine-grained interaction interface that decomposes mouse actions into executable primitives and exposes an observation after each step, together with training trajectories that demonstrate iterative exploration and feedback-conditioned adaptation. LatentLearner-9B trained with this formulation reaches 50.0% overall success, compared with 19.6% under bundled click/drag actions. Moreover, joint RL enables a single policy to control multiple interaction regimes within each task family while retaining strong performance across them, and the same interaction format transfers to standard GUI grounding, achieving 67.37% mean accuracy across five benchmarks. Together, higher success and stronger gain compensation after RL support improved adaptation to episode-specific dynamics.

Our contributions are threefold:

- We introduce **LatentGUIWorld**, a benchmark for studying GUI control under hidden environment properties, exposing substantial failures of agents that cannot adapt from interaction feedback.



[a] OSWorld (Xie et al., 2024)
[b] OSWorld 2.0 (Yuan et al., 2026)
[c] VisualWebArena (Koh et al., 2024)
[d] AndroidWorld (Rawles et al., 2025)
[e] Windows Agent Arena (Bonatti et al., 2025)
[f] ScreenSpot-Pro (Li et al., 2025a)
[g] UI-Vision (Nayak et al., 2025)
[h] ShowUI-π (Hu et al., 2026)

Figure 2: Task horizon and environment-identification evidence are distinct dimensions. Lines connect the three matched task pairs in LATENTGUIWORLD. Named benchmarks illustrate selected task types; bands and positions are qualitative.

- We propose a fine-grained action interface and feedback-rich training formulation that support iterative exploration and adaptation, and show through ablations that both action granularity and trajectory structure are critical.
- We demonstrate that feedback-conditioned control can be learned jointly across interaction regimes and transferred beyond the benchmark to standard GUI grounding, suggesting that the proposed interaction format is broadly applicable.

2 RELATED WORK

2.1 GUI AGENT BENCHMARKS

GUI benchmarks span ScreenSpot-Pro grounding (Li et al., 2025a), realistic execution in OSWorld and MobileBench-OL (Xie et al., 2024; Wu et al., 2026), and long cross-application workflows in OSWorld 2.0 (Yuan et al., 2026). Hybrid GUI–programmatic systems and verifier-guided CLI agents can simplify execution (Song et al., 2026; Jia et al., 2026; Zhou et al., 2026), motivating evaluation beyond workflow length.

ChartAct and CAP examine information revealed through interaction (Huang et al., 2026; Xu et al., 2026b), while DynamicGUIBench evaluates changing interfaces (Liu et al., 2026a). Open CaptchaWorld and Next-Gen CAPTCHAs test interactive perception and manipulation (Luo et al., 2025; Liu et al., 2026b); GUI Knowledge Bench and EvoGUI assess action effects from offline trajectories (Shi et al., 2025; Yang et al., 2026). LATENTGUIWORLD focuses on identifying hidden semantics and episode-specific control laws through the agent's own interactions. Its short, matched tasks isolate evidence requirements as a difficulty dimension alongside task horizon (Figure 2).

2.2 INTERACTION AND ADAPTATION IN GUI AGENTS

Standalone cursor movement is already supported by GUI agents (Bai et al., 2025; Lu et al., 2026). GUI-Cursor uses cursor–target feedback for iterative grounding (Zhao et al., 2026), while ReCAP learns error recovery from self-correction trajectories (Chen et al., 2026). DynamicUI selects informative interaction frames (Liu et al., 2026a); ScreenSearch guides exploration using outcome differences for matched actions at similar screen states (Solodko & Wagle, 2026).

We study how such action-generated evidence supports subsequent control, distinguishing direct semantic revelation from identifying episode-specific dynamics through action–effect comparison. Building on existing movement primitives, MOVE separates movement from mouse-down/up and returns an observation after each primitive. Aligned demonstrations teach agents to use this feedback for exploration and control before committing through a click or release.

3 LATENTGUIWORLD

LATENTGUIWORLD studies episodic GUI environments in which the same task and pre-interaction context can conceal different environment configurations. The object to be identified is a task-relevant *environment latent*: an episode-specific property that determines how the interface responds to an intervention and which subsequent control succeeds. We first formalize this object and classify tasks by the minimum evidence needed to identify it, then construct matched environments that instantiate the resulting taxonomy.

3.1 PROBLEM DEFINITION AND TAXONOMY

Environment identification concerns a task-relevant, episode-specific latent z that affects the control required for success. The agent identifies its current value through interaction-generated evidence. The target is this property, rather than application identity or the complete environment state. Let

$$H_t = (g, O_0, A_0, O_1, A_1, \ldots, O_{t-1}, A_{t-1}), \qquad D_t = (H_t, O_t).$$

Here D_t is the complete observed context before intervention. We assume sufficient prior domain knowledge K: GUI conventions, application rules, and possible control laws, shared across episodes but excluding the current hidden value z. The agent executes a probe A_t^{probe} and receives

$$O_{t+1} \sim T_z(D_t, A_t^{\text{probe}}).$$

The agent knows its executed probe parameters, such as mouse displacement $A_t^{\text{probe}} = (\Delta x_t, \Delta y_t)$. Let $A_{t+1}^*(D_t, z)$ denote the next task-directed action after acquiring the relevant evidence. An environment latent remains fixed within an episode, is not identifiable from (K, D_t), changes the required downstream control, and is identifiable from the consequences of the agent's intervention. At an ambiguous context, prior knowledge alone cannot select the correct downstream control; the probe must supply the missing evidence. Under the same K, two latent values can satisfy

$$z_1 \neq z_2, \qquad D_t(z_1) = D_t(z_2),$$
$$T_{z_1}(D_t, A_t^{\text{probe}}) \neq T_{z_2}(D_t, A_t^{\text{probe}}), \quad A_{t+1}^*(D_t, z_1) \neq A_{t+1}^*(D_t, z_2),$$

Table 1 defines three levels by the minimum evidence sufficient to resolve this ambiguity.

Table 1: Evidence taxonomy for environment identification under fixed prior knowledge K. Levels differ in the minimum evidence needed for task-directed control; D_t denotes observed context.

Level	Before intervention	Minimum identifying evidence	Dependency
$\mathbf{I_0}$	K and D_t suffice.	Prior knowledge and observed context support control; multiple execution steps and screen updates are allowed.	$D_t = (H_t, O_t) \to A_t^*$
$\mathbf{I_1}$	K and D_t do not suffice.	The probe outcome directly reveals a semantic latent: use *what the action revealed*.	$D_t \xrightarrow{A_t^{\text{probe}}} O_{t+1} \to z_{\text{sem}} \to A_{t+1}^*$
$\mathbf{I_2}$	K, D_t, and a probe outcome alone do not identify the dynamics.	The self-action and induced transition jointly identify a dynamics latent: use *what the action caused*.	$(O_t, A_t^{\text{probe}}, O_{t+1}) \to z_{\text{dyn}} \to A_{t+1}^*$

In I_1, the probe result directly reveals the missing semantic value: the action determines *what is revealed*. In I_2, the agent interprets the observation relative to its self-executed action to identify *what its action caused*. We call this *online interaction dynamics identification*; it conditions on the complete intervention–response record, including the executed action parameters.

Let $\Pi_{\epsilon,B}(\mathcal{T})$ be policies with access to K that solve task $\mathcal{T}$ with error at most ϵ under interaction budget B. The task's level is the lowest sufficient for reliable success:

$$I(\mathcal{T}) = \min_{\pi \in \Pi_{\epsilon,B}(\mathcal{T})} \max_{t \in \tau_\pi} I_t. \tag{1}$$

The level measures the evidence required for task-directed control under a fixed observation, action, and interaction-budget contract. When (K, D_t) suffices, long workflows and multi-step execution remain compatible with I_0. Conversely, knowing to hover in Ten-Choice does not determine which candidate to select; an I_1 probe reveals that missing semantic value. Once identified, the latent can be retained in H_t, allowing later decisions to use I_0 evidence. We therefore classify explicit task constructions by their minimum evidence requirement, independently of history length or whether a particular agent explores.

3.2 Benchmark Construction

LatentGUIWorld comprises three task families with two control variants each (Figure 1). Each pair shares its objective while varying the evidence needed for successful control. Detailed control laws and interaction sequences appear in Appendices D and E.

Drag (I_0 vs. I_2). The agent places an object in its matching slot. In exocentric control (I_0), a free cursor moves the object under a known one-to-one mapping, so visible object and slot positions determine the required movement. Egocentric control (I_2) instead uses a center reticle and an episode-specific transfer gain. The agent must relate its executed displacement to the observed scene change to calibrate object placement.

Ten-Choice (I_1 vs. I_2). The agent selects a target message hidden among ten visually identical candidates, with messages permuted each episode. In exocentric reveal (I_1), hovering directly reveals a candidate's message; the observation supplies the missing semantics. Egocentric reveal (I_2) adds an unknown viewpoint gain: movement shifts the candidate field around a fixed interaction point. The agent must infer this gain from action–effect feedback to steer between candidates, while hovering still reveals their messages.

Rotation (I_2 in both variants). The agent aligns an image through inner- or outer-ring control. Inner-ring control maps horizontal movement to rotation through an episode-specific signed gain; outer-ring control maps ring displacement to image rotation through a signed angular coupling. Both require comparing the executed movement with the observed rotation to calibrate the next correction.

Appendix A details episode sampling, data splits, validation-based checkpoint selection, and task pairing.

4 Learning from Move

4.1 Move-Centric Action Schema

$\mathrm{MOVE}(p)$ moves the cursor to p; coordinate-free CLICK, MOUSE_DOWN, and MOUSE_UP act at its current position. Each primitive returns a fresh observation before the next decision. Coordinate-bearing $\mathrm{CLICK}(x, y)$ and $\mathrm{DRAG}(p, q)$ macros remain available without internal decision points; decomposing them exposes intermediate feedback:

$$\mathrm{CLICK}(p) \;\rightsquigarrow\; [\mathrm{MOVE} \to O]^+ \to \mathrm{CLICK} \to O. \tag{2}$$

$$\begin{aligned} \mathrm{DRAG}(p, q) \;\rightsquigarrow\; & [\mathrm{MOVE} \to O]^+ \to \mathrm{MOUSE_DOWN} \to O \\ & \to [\mathrm{MOVE} \to O]^+ \to \mathrm{MOUSE_UP} \to O. \end{aligned} \tag{3}$$

Here O is a fresh observation and $[\mathrm{MOVE} \to O]^+$ denotes one or more individually decided movements. Clicking ends at p; dragging grasps at p and releases at q. Conditioning subsequent actions on $(O_t, A_t^{\mathrm{move}}, O_{t+1})$ supports semantic revelation in I_1 and dynamics identification through action–effect comparison in I_2. We call this feedback-conditioned control *learning from* Move. To train this behavior, we construct multi-step demonstrations in which feedback from one move guides subsequent moves before a click or release.

4.2 Training Data Construction

We construct multi-step MOVE demonstrations across all six LATENTGUIWORLD variants using rule-based scripts, retaining action sequences and their intermediate observations. For interaction-dependent tasks, the scripts inspect revealed content or estimate movement responses, then use this feedback to choose subsequent moves before clicking or releasing. Successful trajectories supervise exploration and correction during free or held movement. A teacher model adds a short Think rationale for each prescribed action from its pre-action observations, preceding responses, and the action itself, without changing the sequence.

Multi-step MOVE data can also be obtained from existing GUI demonstrations by decomposing actions and supplying intermediate observations through replay or intermediate-frame construction, as explored in our grounding experiments (Section 6.4). Appendix B.1 describes SFT and subsequent GRPO training.

5 Experiments

5.1 Experimental Setup

Tasks and data. We evaluate the six environments in Section 3, organized into matched Drag, Ten-Choice, and Rotation pairs. The main evaluation uses 150 episodes per environment; smaller or incomplete runs are documented in the supplementary material. LatentLearner-9B is trained jointly on 24,000 successful demonstration trajectories, with 4,000 per environment, yielding 154,713 supervised examples. Data separation and training details are given in Appendices A and B.2.

Baselines and Implementation. We compare GUI agents based on general-purpose models with GUI-specialized agents, as grouped in Table 2. GPT-6 Astra and Claude Opus 5 follow their official schemas. GPT-5.6-Sol and Kimi-K3 extend their official schemas with the tools needed for task completion. DeepSeek-V4.1-Flash uses our six-action JSON schema, as no official computer-use schema was available. Appendix C specifies the tools and evaluated variants. LatentLearner-9B uses Qwen3.5-9B (Qwen Team, 2026) with joint supervised fine-tuning in the MOVE format. Each training example conditions the next action on the task instruction, up to the latest three screenshots, and preceding assistant actions. The reported benchmark model generates with-think responses and receives no RL updates. One checkpoint is used across all six environments, selected by overall success rate on the separate validation set. Training hyperparameters and checkpoint-selection details are given in Appendix B.2; the task prompt and multimodal history for our SFT models and the raw Qwen baseline are documented in Appendix B.3.

Metrics. We report episode success for each environment, the macro-average over the four I_2 tasks, and the macro-average over all six tasks. Rotation uses strict first-release scoring: a failed first release cannot be recovered by a later release or by final-state success at the action limit. Recorded endpoint, protocol, and executor failures count as nonsuccesses; incomplete coverage is marked explicitly. Drag and Ten-Choice pairs compare different evidence requirements, whereas both Rotation variants are I_2 and compare control geometries.

5.2 Benchmark Performance and Control Behavior

Table 2 shows that strong performance with available observations or directly revealed semantics (I_0/I_1) does not reliably extend to matched tasks requiring dynamics identification (I_2). This gap persists in frontier models and GUI-specialized agents. Most GUI-specialized agents additionally struggle with semantic revelation in I_1 Ten-Choice; Qwen-CUA is an exception on this task but remains weak on the matched I_2 variant.

After SFT, LatentLearner-9B achieves 50.0% overall success, comparable to several frontier models, with strong I_0/I_1 performance and improved egocentric Drag and Ten-Choice control; Rotation remains weak. In egocentric Drag, Pearson correlation between error-normalized commands and ideal gain compensation $1/g$ is $r = 0.610$, indicating a tendency to compensate for gain. Appendix G.1 expands the task comparisons; Appendix F reports behavioral measurements and coverage.

Table 2: Test success rates (%). I_2 Avg. and Overall average four and six tasks, respectively. Δ_{pair} is second-minus-first success (percentage points, before rounding). Bold and underline mark the best and second-best results per column, including ties.

GUI agent	Drag			Ten-Choice			Rotation			I_2 Avg.	Overall
	Exocentric (I_0)	Egocentric (I_2)	Δ_{pair}	Exocentric (I_1)	Egocentric (I_2)	Δ_{pair}	Inner-ring (I_2)	Outer-ring (I_2)	Δ_{pair}		
GUI agents based on general-purpose models											
GPT-5.6-Sol	**100.0**	4.7	−95.3	**100.0**	0.7	−99.3	18.0	5.3	−12.7	7.2	38.1
GPT-6 Astra	**100.0**	2.0	−98.0	**100.0**	**56.0**	−44.0	**65.3**	**56.7**	−8.7	**45.0**	**63.3**
Claude Opus 5	**100.0**	10.0	−90.0	**100.0**	5.0	−95.0	37.3	19.3	−18.0	17.9	45.3
Kimi-K3	**100.0**	6.0	−94.0	**100.0**	3.3	−96.7	25.3	6.0	−19.3	10.2	40.1
DeepSeek-V4.1-Flash	80.0	0.0	−80.0	19.3	1.3	−18.0	0.7	0.7	0.0	0.7	17.0
GUI-specialized agents											
Qwen-CUA	99.3	0.7	−98.7	92.7	1.3	−91.3	0.0	0.7	0.7	0.7	32.4
GUI-Owl-1.5-32B-Think	59.3	0.0	−59.3	8.0	1.3	−6.7	1.3	1.3	0.0	1.0	11.9
UI-TARS-1.5-7B	34.0	0.0	−34.0	8.7	1.3	−7.3	2.7	2.7	0.0	1.7	8.2
DART-GUI-7B	21.3	0.0	−21.3	9.3	0.7	−8.7	3.3	3.3	0.0	1.8	6.3
EvoCUA-32B	7.3	0.0	−7.3	14.7	0.0	−14.7	2.0	2.0	0.0	1.0	4.3
OpenCUA-72B	32.7	0.7	−32.0	9.3	0.7	−8.7	0.0	0.0	0.0	0.3	7.2
Backbone baseline and our models											
Qwen3.5-9B (raw)	0.7	0.0	−0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
LatentLearner-9B	96.0	**41.3**	−54.7	99.3	40.7	−58.7	11.3	11.3	0.0	26.2	50.0

Model references: GPT-5.6-Sol (OpenAI, 2026a); GPT-6 Astra (OpenAI, 2026b); Claude Opus 5 (Anthropic, 2026); Kimi-K3 (Moonshot AI, 2026); DeepSeek-V4.1-Flash (DeepSeek-AI, 2026); Qwen-CUA (Lu et al., 2026); GUI-Owl-1.5-32B-Think (Xu et al., 2026a); UI-TARS-1.5-7B (Qin et al., 2025); DART-GUI-7B (Li et al., 2025b); EvoCUA-32B (Xue et al., 2026); OpenCUA-72B (Wang et al., 2025).

5.3 Frontier-Agent Failure Analysis

Frontier-agent trajectories show failures both before and after feedback. Some agents release a bundled drag before observing its effect; Astra sometimes reaches alignment but keeps moving before release, causing failure. Effective control requires using feedback to adjust movement and decide when to commit. Sol, Astra, Opus, and Kimi also perform worse on outer-ring than inner-ring Rotation, although both are I_2, showing sensitivity to control geometry. Appendix G.2 provides the detailed analysis.

6 Analysis

We first examine whether explicit exploration and perspective instructions improve a frontier agent without training. We then study three aspects of learning from Move: joint RL across interaction regimes, the role of action granularity and intermediate feedback, and compatibility with standard GUI grounding. These analyses connect task success to the evidence available to the model and the behavior learned from that evidence.

6.1 Training-Free Prompt Ablation

We append "Before attempting to solve the task, first explore the environment." and "The current environment is egocentric." to GPT-6 Astra's baseline task instruction (Appendix B.3). We evaluate the first 50 episode IDs from each egocentric environment in the frozen test set and compare with their records in the earlier 150-episode baseline. The model, native tool interface, task instances, initial observations, and other settings remain fixed.

Table 3 shows gains of 26 percentage points on Ten-Choice and 20 on Drag without additional training. These gains reveal a gap between autonomous and explicitly guided adaptation under the same interaction interface. The added cues provide task-specific guidance beyond tool-use instructions; Drag nevertheless reaches only 22% success. These results show that even a frontier model struggles to identify and adapt to environment dynamics, despite explicit guidance.

Table 3: GPT-6 Astra success with baseline versus exploration-and-perspective prompts on 50 matched episodes per task. Changes are in percentage points (pp).

Egocentric task	Baseline	Prompted	Change
Ten-Choice	28/50 (56%)	41/50 (82%)	+26 pp
Drag	1/50 (2%)	11/50 (22%)	+20 pp

6.2 Joint RL Across Interaction Regimes

Can joint RL improve control across a matched pair, and how does its performance compare with environment-specific RL? We compare six single-environment RL specialists of LatentLearner-9B with three paired-RL models, one for each task family. Each specialist receives RL updates on one variant, whereas each paired model receives updates on both. We evaluate success on the same 150 held-out instances per variant, using greedy decoding, a 12-turn budget, and strict first-release scoring. Each paired model is evaluated on both variants with a single checkpoint. Appendix B.4 gives training settings and model selection.

Table 4: Single-environment versus paired RL success (%; $n = 150$ per variant). Δ is paired minus single-environment success (percentage points, before rounding).

Pair	Test variant	Single-env. RL	Paired RL	Δ
Drag	Exocentric	96.67	96.67	0.00
	Egocentric	93.33	92.00	-1.33
Ten-Choice	Exocentric	100.00	100.00	0.00
	Egocentric	78.00	75.33	-2.67
Rotation	Inner-ring	24.00	**27.33**	$+3.33$
	Outer-ring	10.67	**11.33**	$+0.67$

Table 4 shows that joint RL achieves modest gains over single-environment specialists on both Rotation variants, which require I_2 control. Drag and Ten-Choice instead pair an I_0 or I_1 exocentric task with its I_2 egocentric variant. Paired RL matches the I_0/I_1 specialists and remains close to, but slightly below, the I_2 specialists, while substantially improving I_2 success over mixed SFT (Table 2). These results support joint competence across the control regimes within each task family.

The behavioral analysis also shows stronger gain-dependent command scaling after paired RL. In egocentric Drag, Pearson correlation between error-normalized commanded displacement and the ideal gain-compensation coefficient $1/g$ increases from $r = 0.610$ for mixed SFT to $r = 0.803$ for paired RL. Success on the same evaluation suite rises from 41.3% to 92.0%. For a given target error, gain compensation requires larger commands under low gain and smaller commands under high gain. The increased correlation indicates closer alignment with this control requirement. Together, higher success and stronger gain compensation support more effective adaptation to episode-specific control dynamics. Appendix F reports movement coverage, filtered results, and the comparison on shared episode IDs.

6.3 Interaction Paradigm and Training Data Ablations

Table 5 compares interaction-paradigm and training-data ablations of LatentLearner-9B, all using Qwen3.5-9B with joint six-environment SFT and with-think responses.

Replacing Move with atomic click/drag and converting the demonstrations accordingly removes intermediate decision points, reducing overall success from 50.0% to 19.6%. Exocentric Drag remains at 100.0%, but both Ten-Choice variants fall to 2.0%. Retaining Move with only one supervised move per free or held movement phase yields 27.6% overall success; the full model improves on all six environments.

Table 5: Interaction and training-data ablations (%; $n = 150$ per environment). Overall averages six tasks; bold and underline mark the best and second-best results per column.

Setting	MOVE interface	Iterative interaction data	Drag		Ten-Choice		Rotation		Overall
			Exocentric	Egocentric	Exocentric	Egocentric	Inner-ring	Outer-ring	
Paradigm ablation	×	×	**100.0**	0.0	2.0	2.0	7.3	<u>6.0</u>	19.6
Data ablation	✓	×	95.3	<u>12.0</u>	<u>40.7</u>	<u>3.3</u>	<u>8.0</u>	<u>6.0</u>	<u>27.6</u>
LatentLearner-9B	✓	✓	<u>96.0</u>	**41.3**	**99.3**	**40.7**	**11.3**	**11.3**	**50.0**

The data-ablation model also exhibits *multi-step exploration* in exocentric Ten-Choice: 46 of 61 successful episodes (75.4%) use two to ten moves, although all 4,000 demonstrations use one move followed by a click. An audited trajectory switches candidates after a nonmatching tooltip and clicks only upon finding the target. This feedback-conditioned exploration extends beyond the single-move demonstrations provided for this task.

6.4 COMPATIBILITY WITH STANDARD GUI GROUNDING

We evaluate MOVE on standard GUI grounding using 33,353 selected GroundCUA trajectories. Both methods use Qwen3-VL-8B and the same training records, represented as direct clicks or cursor-correction sequences with intermediate visual feedback. We compare SFT and subsequent RL across five benchmarks (Table 6); Appendix B.5 details data construction and training.

Table 6: Standard GUI grounding accuracy (%) on five benchmarks with Qwen3-VL-8B. Average is the unweighted mean; bold denotes the best result per column.

Model	ScreenSpot-Pro	ScreenSpot-v2	MMBench-GUI	UI-Vision	OSWorld-G	Average
Qwen3-VL-8B	55.98	94.42	83.67	27.65	65.29	65.40
Qwen3-VL-8B + SFT	54.84	94.26	83.31	27.76	65.29	65.09
Qwen3-VL-8B + MOVE (SFT)	55.72	94.18	83.11	28.13	62.94	64.82
Qwen3-VL-8B + RL	56.42	94.73	84.59	**31.19**	66.67	66.72
Qwen3-VL-8B + MOVE (RL)	**58.06**	**95.28**	**84.89**	30.37	**68.24**	**67.37**

After SFT, MOVE achieves 64.82% mean accuracy versus 65.09% for direct grounding. After RL, it reaches 67.37% versus 66.72%, improving on four of five benchmarks; UI-Vision favors conventional RL. These results show that the intermediate-feedback format also supports standard GUI grounding, with gains emerging after RL.

7 CONCLUSION

We introduce LATENTGUIWORLD to evaluate whether GUI agents can acquire task-relevant information through interaction and use it to guide subsequent control. Its six matched tasks distinguish available context, revealed semantics, and action–effect evidence, exposing substantial gaps in current agents. The MOVE interface makes intermediate feedback available before commitment, while training enables models to use this feedback for exploration and control. Ablations establish the importance of action granularity and iterative demonstrations. Joint RL supports competence across paired control regimes, and the same interaction format improves standard GUI grounding after RL. Beyond success rates, feedback-conditioned multi-step exploration and gain compensation provide behavioral evidence that the trained models can learn about the environment through interaction and use what they learn to adjust subsequent actions. RL further strengthens this adaptive control, as reflected in stronger gain compensation in egocentric Drag.

A GUI agent should not only learn where to move. It should learn from the move.

AI USE STATEMENT

In this work, we used generative AI writing assistants to polish the English of manuscript drafts, revise the text, and prepare LaTeX. Generative AI coding assistants supported development and debugging of experimental, preprocessing, and analysis code. A teacher model generated Think rationales for prescribed actions in the training demonstrations (Section 4.2). All reported numerical results were computed by the training, evaluation, and offline analysis pipelines described in the paper; gain-compensation statistics were recomputed from saved evaluation records. We have reviewed all AI-assisted code and text, and the final manuscript was revised and approved by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our models build on pretrained backbones. Appendix A describes benchmark construction and data separation; Appendix B and Table 10 specify training settings and validation-based checkpoint selection. Appendix B.3 gives task prompts and multimodal history, and Appendix C documents model-specific evaluation interfaces. Appendix F defines the control-behavior measurements, filtering rules, and sample coverage.

REFERENCES

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, 2026. Accessed September 8, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows Agent Arena: Evaluating Multi-Modal OS Agents at Scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 4874–4910. PMLR, 2025. URL https://proceedings.mlr.press/v267/bonatti25a.html.

Yuxi Chen, Haoyu Zhai, Chenkai Wang, Rui Yang, Lingming Zhang, Gang Wang, and Huan Zhang. CAPTCHA Solving for Native GUI Agents: Automated Reasoning–Action Data Generation and Self-Corrective Training. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=HWTRHS26ZF.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash, 2026. Accessed September 11, 2026.

Siyuan Hu, Kevin Qinghong Lin, and Mike Zheng Shou. ShowUI-π: Flow-based Generative Models as GUI Dexterous Hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8130–8140, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Hu_ShowUI-p_Flow-based_Generative_Models_as_GUI_Dexterous_Hands_CVPR_2026_paper.html.

Muye Huang, Wu Lin, Lingling Zhang, Hang Yan, Zhiyuan Wang, Yumeng Fu, Zesheng Yang, and Jun Liu. Chartact: A benchmark for dynamic chart understanding. *arXiv preprint arXiv:2605.26994*, 2026.

Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. OSWorld-MCP: Benchmarking MCP Tool Invocation In Computer-Use Agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rceD6wwt4B.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905. Association for

Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786. Association for Computing Machinery, 2025a. doi: 10.1145/3746027.3755688. URL https://doi.org/10.1145/3746027.3755688.

Pengxiang Li, Zechen Hu, Zirui Shang, et al. Efficient multi-turn RL for GUI agents via decoupled training and adaptive data curation. *arXiv preprint arXiv:2509.23866*, 2025b. URL https://arxiv.org/abs/2509.23866.

Enqi Liu, Liyuan Pan, Zhi Gao, Yan Yang, Chenrui Shi, Yang Liu, Jingrong Wu, and Qing Li. Benchmarking and improving gui agents in high-dynamic environments. *arXiv preprint arXiv:2604.25380*, 2026a.

Jiacheng Liu, Yaxin Luo, Jiacheng Cui, Xinyi Shang, Xiaohan Zhao, and Zhiqiang Shen. Next-Gen CAPTCHAs: Leveraging the Cognitive Gap for Scalable and Diverse GUI-Agent Defense. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=wPKGirrtyz.

Dunjie Lu, Shuai Bai, Tianyi Bai, et al. Qwen-CUA: Native computer use for (almost) everything. *arXiv preprint arXiv:2608.02352*, 2026. URL https://arxiv.org/abs/2608.02352.

Yaxin Luo, Zhaoyi Li, Jiacheng Liu, Jiacheng Cui, Xiaohan Zhao, and Zhiqiang Shen. Open Captcha-World: A Comprehensive Web-based Platform for Testing and Benchmarking Multimodal LLM Agents. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-3006. URL https://papers.nips.cc/paper_files/paper/2025/hash/8196be81e68289d7a9ece21ed7f5750a-Abstract-Datasets_and_Benchmarks_Track.html.

Moonshot AI. Kimi K3. https://platform.kimi.ai/docs/guide/kimi-k3-quickstart, 2026. Official model documentation. Accessed September 8, 2026.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. UI-Vision: A Desktop-centric GUI Benchmark for Visual Perception and Interaction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 45817–45851. PMLR, 2025. URL https://proceedings.mlr.press/v267/nayak25a.html.

OpenAI. GPT-5.6 Sol. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026a. Official model documentation. Accessed September 8, 2026.

OpenAI. GPT-6 Astra. https://developers.openai.com/api/docs/models/gpt-6-astra, 2026b. Official model documentation. Accessed September 8, 2026.

Yujia Qin, Yining Ye, Junjie Fang, et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. URL https://arxiv.org/abs/2501.12326.

Qwen Team. Qwen3.5-9B model card. https://huggingface.co/Qwen/Qwen3.5-9B, 2026. Accessed September 8, 2026.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. In *International Conference on Learning Representations*, pp. 406–441, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html.

Chenrui Shi et al. Gui knowledge bench: Revealing the knowledge gap of vlms in gui tasks. *arXiv preprint arXiv:2510.26098*, 2025.

Michael Solodko and Justin Wagle. Screensearch: Uncertainty-aware os exploration. *arXiv preprint arXiv:2605.16024*, 2026.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Silvio Savarese, Zeyuan Chen, Jieyu Zhao, Ran Xu, and Caiming Xiong. CoAct-1: Computer-using Multi-Agent System with Coding Actions. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=l1MQVgIKEU.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Y. Charles, Zhilin Yang, and Tao Yu. OpenCUA: Open Foundations for Computer-Use Agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 139756–139806. Curran Associates, Inc., 2025. doi: 10.52202/085713-4669. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/cc7ae529e945226b0d52ea4ac478c4f3-Abstract-Conference.html.

Qinzhuo Wu, Zhizhuo Yang, Hanhao Li, Pengzhi Gao, Wei Liu, and Jian Luan. Mobilebench-ol: A comprehensive chinese benchmark for evaluating mobile gui agents in real-world environment. *arXiv preprint arXiv:2601.20335*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Haiyang Xu, Xi Zhang, Haowei Liu, et al. Mobile-Agent-v3.5: Multi-platform fundamental GUI agents. *arXiv preprint arXiv:2602.16855*, 2026a. URL https://arxiv.org/abs/2602.16855.

Zejun Xu et al. Cap: A scalable benchmark for evaluating cross-site browser agents with complex actions and perception. *arXiv preprint arXiv:2608.08392*, 2026b. URL https://arxiv.org/abs/2608.08392. Accepted to the Conference on Language Modeling (COLM), 2026.

Taofeng Xue, Chong Peng, Mianqiu Huang, et al. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026. URL https://arxiv.org/abs/2601.15876.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Yaohan Yang, Minglei Shi, Borui Zhang, Jie Zhou, and Jiwen Lu. Evogui: An evolution-aware benchmark for gui state-transition understanding. *arXiv preprint arXiv:2607.17050*, 2026.

Mengqi Yuan et al. Osworld 2.0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026.

Yu Zhao, Wei-Ning Chen, Huseyin Atahan Inan, Samuel Kessler, Lu Wang, Lukas Wutschitz, Fangkai Yang, Chaoyun Zhang, Pasquale Minervini, Saravan Rajmohan, and Robert Sim. Learning GUI Grounding with Spatial Reasoning from Visual Feedback. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=dqmt0Yd4rc.

Xiao Zhou, Siyue Zhang, Yilun Zhao, Jinbiao Wei, Tingyu Song, Arman Cohan, and Chen Zhao. GUI vs. CLI: Execution bottlenecks in screen-only and skill-mediated computer-use agents. *arXiv preprint arXiv:2606.24551*, 2026.

A Benchmark Construction Details

The fixed test suite contains 150 paired cases per environment, each instantiated in both variants, yielding 900 evaluation episodes at 1280×720 resolution. Evaluation covers episode-specific configurations. Table 7 distinguishes variation in layouts and control parameters from explicit holdout of shape categories, message content, or background images. Held-out factors are defined relative to task-training data, not to the unknown pretraining data of external models.

Table 7: Fixed test-set composition and generalization factors. Counts are per variant; each environment has two paired variants. Episode-specific configuration variation and explicit factor holdout are reported separately.

Environment	Test episodes	Episode-varying factors	Explicitly held-out factors
Drag	150	Object/slot geometry, appearance, and layout; egocentric sensitivity	Shape categories in 75 episodes; the other 75 use training shape categories
Ten-Choice	150	Candidate positions and message assignments; egocentric sensitivity	Candidate message content
Rotation	150	Background image, initial angle, sensitivity, and rotation direction	Background images

Validation set and model selection. A separate validation set contains 150 episodes per variant, totaling 900 episodes, matching the test-set size. Validation episodes are disjoint from the training and test episodes. Checkpoint selection uses validation success rates; the selected checkpoints are evaluated on the test set for the reported benchmark results. Test success rates are not used to choose checkpoints.

Drag sampling and shape holdout. Each variant has 75 test episodes using training shape categories and 75 using held-out shape categories. The former use six categories for movable objects and matching slots: circle, square, triangle, hexagon, wide rectangle, and cross. The latter use four disjoint categories: pentagon, trapezoid, star, and L-shape. Both subsets vary scene geometry and layout, covering 5–8 candidate slots and four layout families: scattered, jittered grid, ring/arc, and two clusters. The evaluated egocentric variant uses equal positive gains on both axes, with sensitivity drawn from $[0.5, 0.8] \cup [1.2, 1.5]$, also used in training. Thus, using training shape categories does not remove the need to handle episode-specific configurations; shape-category holdout and within-range control-parameter variation are distinct evaluation factors.

Ten-Choice sampling and splits. The 150 test scenes use a pool of 240 distinct messages disjoint from the 80 messages in the task-training data. Each scene specifies ten candidate messages, their positions, and a target, all shared by its two variants. The egocentric variant uses equal positive axis gains, with 75 test cases in each sensitivity band, $[0.5, 0.8]$ and $[1.2, 1.5]$. Training uses the same bands. Thus, all test episodes use held-out message content, while episode-specific gain variation remains within the training support.

Rotation sampling and splits. The 150 test pairs use 150 distinct background images, disjoint from the 1,000 backgrounds in the task-training set. Initial angular misalignment, sensitivity, and rotation direction vary across cases. Within each pair, the image, initial relative angle, alignment target, and underlying challenge parameters are shared; the responsive control region differs. This split holds out image content, without requiring the control-parameter ranges themselves to be out of distribution.

Episode parameterization and train–test separation. Semantic assignments and dynamics parameters are assigned during dataset construction and remain fixed throughout each trajectory. Resetting a fixed test case restores its assigned configuration rather than drawing a new latent. The egocentric Drag and Ten-Choice test variants use positive gains; Rotation varies both response sensitivity and direction. Test cases use separate seeds from task-training cases. Drag additionally separates source cases, Ten-Choice separates message pools, and Rotation separates background images.

Pairing and initial observations. Pairing preserves scene content and task objectives but need not imply identical initial observations. Drag pairs share object and slot identities and world positions; 100 pairs share the initial layout, while 50 place the egocentric target initially offscreen (25 using training shape categories and 25 using held-out shape categories). Ten-Choice pairs share candidate layouts, message assignments, and targets; Rotation pairs share the image and initial alignment challenge. With a common mouse marker, all Ten-Choice and Rotation pairs and the 100 shared-layout Drag pairs have identical first frames. These counts distinguish scene pairing from strict visual matching. Cross-variant matching is distinct from the within-variant latent ambiguity used to define environment-identification levels.

B Training Details

B.1 Model Training

We first train the policy through supervised fine-tuning on demonstration trajectories. Following Section 3.1, let $D_t = (H_t, O_t)$ denote the observed context before action A_t. Each training example pairs this context with the demonstrated response y_t, and we minimize the autoregressive objective:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(D_t, y_t)\sim\mathcal{D}} \log \pi_\theta(y_t \mid D_t). \tag{4}$$

For LatentGUIWorld, y_t contains a Think rationale followed by the primitive action; the grounding extension uses action-only targets. This objective supervises the next response using observations and responses available up to the current decision.

We further explore reinforcement learning initialized from the SFT policy. The policy samples trajectories through online interaction, receiving a new observation after each action and conditioning subsequent decisions on the updated context. We use Group Relative Policy Optimization (GRPO) to optimize trajectory rewards based on task success and action-format compliance. Experiment-specific setups are described in Sections 5.1 and 6.4; optimization settings are detailed below.

B.2 Benchmark SFT

LatentLearner-9B initializes from Qwen3.5-9B and is fine-tuned on 24,000 successful trajectories, with 4,000 from each environment. These trajectories yield 154,713 supervised examples. Each example contains the task instruction, up to the latest three screenshots, and preceding assistant actions; the target is a Think rationale followed by one primitive action. The loss supervises the demonstrated response, while the preceding context serves as input.

Training uses VERL with FSDP2, gradient checkpointing, and a maximum sequence length of 16,384 tokens. The configured budget is three epochs; Table 10 lists the optimization settings alongside those of the grounding extension. We select one of 30 checkpoints by overall success rate on the separate six-environment validation set (150 episodes per environment, 900 in total) and use it for all six test environments. The benchmark model in Table 2 receives no RL updates. Benchmark paired RL and grounding RL are detailed separately below.

MOSS-VL SFT. We additionally fine-tune MOSS-VL-Instruct-0708 on the preserved mixed-think demonstration data adapted to its native processor and template. Training uses BF16, DeepSpeed ZeRO-3, a trainable vision encoder, global batch size 64, learning rate 3×10^{-6}, maximum sequence length 16,384, and seed 42. The three-epoch run retains 30 model checkpoints. All 30 are evaluated on the separate six-environment validation set (150 episodes per environment, 900 in total); one checkpoint is selected by overall validation success and used for all six test environments. This model receives no RL updates. The Benchmark SFT column of Table 10 describes the original Qwen run.

B.3 Main-Result Policy Prompt and Conversation Format

This section specifies the policy prompt used for the six-environment main results. LatentLearner-9B uses the same task prompt, action schemas, and history layout during supervised training and evaluation. The public Qwen3.5-9B row uses this evaluation prompt without project fine-tuning; MOSS-VL uses the same task text and history content through its native processor and chat template.

Environment task instructions. Table 8 gives the exact text substituted for `{task_instruction}`. Both variants in each task family use the same instruction. In Ten-Choice, `{target_label}` is replaced by the episode's requested message. These task instructions are separate from model-specific tool instructions.

Table 8: Task instructions for the six evaluation environments. Quotation marks around the Ten-Choice target are part of the instruction.

Environment	Task instruction
Exocentric Drag	`Place the solid colored shape into the matching gray outline.`
Egocentric Drag	`Place the solid colored shape into the matching gray outline.`
Exocentric Ten-Choice	`Click the icon that displays "{target_label}".`
Egocentric Ten-Choice	`Click the icon that displays "{target_label}".`
Inner-ring Rotation	`Drag the slider to complete verification.`
Outer-ring Rotation	`Drag the slider to complete verification.`

Prompt-ablation suffix. For the prompted condition in Section 6.1, GPT-6 Astra receives its baseline task instruction followed by the two sentences below, separated by blank lines. The baseline condition contains neither added sentence; model-specific tool instructions remain unchanged.

```
Before attempting to solve the task, first explore the environment.

The current environment is egocentric.
```

Complete task prompt. The following text begins the user message at every decision. The placeholder `{task_instruction}` is replaced by the episode's task instruction. The six action schemas below are exposed together; each response contains one selected action. The template is shared across the six variants, with the task instruction, observations, and history supplying the episode-specific input. Line wrapping below is only for typesetting.

```
Task requirement: {task_instruction}
Action space for this task:
Allowed action kinds: move_to, mouse_down, mouse_up, left_click, drag,
↪  click
Action schemas:
- move_to: {"kind":"move_to","x":<integer 0-1000>,"y":<integer 0-1000>}
- mouse_down: {"kind":"mouse_down"}
- mouse_up: {"kind":"mouse_up"}
- left_click: {"kind":"left_click"}
- drag: {"kind":"drag","points":[[<integer 0-1000>,<integer
↪  0-1000>],[<integer 0-1000>,<integer 0-1000>]]}
- click: {"kind":"click","points":[[<integer 0-1000>,<integer 0-1000>]]}
Choose exactly one next action from the current screenshot and causal
↪  history.
Coordinates must be integers in the 0-1000 screenshot-relative system: x
↪  increases left to right and y top to bottom.
Put one or two short English sentences grounded only in visible evidence
↪  and causal history inside <think>...</think>, at most 60 words.
Then output one compact JSON object: {"action":{...}}.
Do not mention hidden targets, sensitivities, oracle values,
↪  ground-truth actions, formatting, or validation.
Do not add Markdown or any other text.
```

Multimodal input and history. We construct one user message containing the task prompt followed by image and text blocks. At decision t (counting from one), we retain up to the three most recent pre-action screenshots, including the current observation O_t. Every preceding assistant response $R_1, \ldots, R_{t-1}$ is included verbatim, preserving both its Think block and action JSON. Responses

whose screenshots have fallen outside the three-image window appear first as text; the retained screenshots are then interleaved with their corresponding responses in chronological order. Screenshots are supplied as image blocks, not as textual scene descriptions. No separate benchmark-specific system message is added; model-native chat serialization is applied by the corresponding processor.

The first decision contains the complete task prompt, the initial screenshot, and this suffix:

```
No previous actions are present in this trajectory.
The last attached image is the current observation.
Return the Think block and JSON action for step 1.
```

For example, the fourth decision is assembled as follows. Bracketed entries denote inserted content rather than literal prompt text; all other lines reproduce the history labels and suffix.

```
[Complete task prompt above]
Previous step 1 assistant response (context only):
[R_1, verbatim]
[Image O_2]
Previous step 2 assistant response (context only):
[R_2, verbatim]
[Image O_3]
Previous step 3 assistant response (context only):
[R_3, verbatim]
[Image O_4: current observation]
3 previous assistant response(s) are included in chronological order.
The last attached image is the current observation.
Return the Think block and JSON action for step 4.
```

Training targets and evaluation responses. During SFT, the user message is followed by the demonstrated assistant response, which is the supervised target; earlier responses embedded in the user message provide context. During evaluation, the assistant response is generated from this user message, its action is executed, and the next observation updates the context. Thus, training history contains demonstration responses and evaluation history contains the model's own preceding responses. The response format is a `<think>...</think>` block followed by one compact JSON object with the top-level key `action`. The prompt requests one or two short English reasoning sentences, at most 60 words, grounded in visible evidence and prior interaction. Hidden environment parameters, target annotations, and evaluator-only state are not supplied as additional input fields.

B.4 Benchmark Paired RL Training and Model Selection

Each paired model starts from the mixed-SFT LatentLearner-9B checkpoint and is trained with GRPO on 5,120 prompts, comprising 2,560 from each variant. The combined dataset is shuffled and used for one epoch. Each prompt generates five rollouts; advantage normalization groups rollouts from the same prompt and environment instance. Training retains the with-think response format, the latest three screenshots, and at most 12 interaction turns. Table 9 lists the training settings.

The reward combines task success (1) and trajectory-format validity (0.1). In Drag and Ten-Choice, trajectories terminated by generation or context-budget exhaustion receive zero reward. KL regularization is applied as a separate loss, with no KL term added to the reward. All three runs use BF16 and FSDP2.

Each turn generates at most 384 tokens, subject to the remaining context and trajectory response budgets and a 32-token safety reserve.

Checkpoint selection uses the separate validation set, with 150 validation episodes per variant, matching the test-set size. Each single-environment specialist in Table 4 is selected by validation success rate on the variant it is trained on, with ties resolved toward the later checkpoint. The paired Drag and Ten-Choice models are selected by mean validation success rate across the two variants with the same tie rule. Paired Rotation is selected by inner-ring validation success. Each paired model supplies both test results within its family.

Table 9: Benchmark paired-RL hyperparameters.

Hyperparameter	**All task families**
Training prompts (per variant)	2,560
Training budget (epochs)	1
Optimizer	AdamW
Learning rate	10^{-6}
Learning-rate schedule	Cosine
Warmup ratio	0.08
Minimum learning-rate ratio	0.1
Adam β_1, β_2	$(0.9, 0.95)$
Weight decay	0.05
Gradient clipping norm	1.0
Generation batch (prompts)	16
Rollouts per prompt	5
Rollouts per generation batch	80
Optimizer minibatch (action samples)	80
Micro-batch size per GPU	1
PPO epochs per batch	1
PPO clipping ratio	0.2
KL loss coefficient	0.005
Entropy loss coefficient	0
Rollout temperature / top-p	$1.0/0.95$
Context window (tokens)	16,384
Trajectory response budget (tokens)	8,192
Per-turn generation cap (tokens)	384
Maximum interaction turns	12
Retained screenshots	3

B.5 GROUNDING SFT AND RL

Training data. For grounding SFT, we use 33,353 training trajectories, including 30,018 one-move, 1,668 two-move, and 1,667 three-move trajectories, giving an approximately $90{:}5{:}5$ distribution. The conventional direct-click baseline and the MOVE variant use the same selected records with different action representations. Subsequent RL uses 10,000 grounding instructions excluded from the SFT instruction set.

Rejection sampling for coordinate supervision. We construct coordinate supervision by generating and verifying candidate grounding actions with Qwen3-VL-8B-Instruct. The model receives the task instruction, screenshots, and preceding actions, while the annotated target box remains hidden and is used only for offline verification. Predicted coordinates are integers in $[0, 1000]$ on each axis and are mapped back to the original image for the target-box check. A coordinate is accepted as the terminal target when it falls inside the annotated box, including its boundary. Malformed actions and out-of-range coordinates trigger retries.

If the first valid prediction misses the target, its coordinate is retained as an intermediate cursor position. We render the cursor at that position on the unchanged screenshot and request a correction conditioned on the updated observation and action history. The second valid prediction is checked in the same way. If both predictions miss, or format retries are exhausted, the pipeline supplies a recovery coordinate that lies inside the target box after coordinate quantization. Recovery coordinates are recorded separately from model predictions. Valid misses can therefore remain as intermediate correction steps, while every completed trajectory ends at a verified target position.

Trajectory construction and serialization. For the MOVE variant, coordinate-bearing clicks are converted into `move_to` actions without changing their coordinates, followed by a coordinate-free `left_click`. Intermediate observations update only the cursor overlay, while the underlying GUI screenshot remains fixed. During rebalancing, some three-move trajectories are shortened by removing the second miss while preserving the first miss and the successful endpoint. We then select training trajectories according to the length distribution described above.

The direct-click baseline retains only the final coordinate-bearing click for each selected record. The MOVE variant preserves intermediate actions and observations and uses training windows containing at most three screenshots together with the preceding textual action history. Windowing yields 35,020 training examples from the 33,353 trajectories. Actions already supervised in an earlier window are masked from the loss in subsequent windows. Both settings use action-only targets without Think annotations or assistant prefilling.

Supervised fine-tuning. Both SFT settings initialize from Qwen3-VL-8B-Instruct and use full-parameter fine-tuning with VERL and FSDP2. Training uses BF16 precision, FlashAttention 2, gradient checkpointing, and random seed 1. Both settings maintain a global batch size of 128 and train for two epochs. Optimization settings are summarized in Table 10.

Reinforcement learning. RL initializes from the corresponding SFT checkpoint and uses GRPO with eight rollouts per prompt. Rewards are -0.2 for invalid format, 0 for a valid but incorrect result, and 1 for a valid and correct result. We apply a KL regularization loss with coefficient 0.01, without adding KL to the reward. The MOVE policy receives updated cursor observations between actions, with at most four actions and three retained screenshots; the direct-click policy predicts one coordinate-bearing click. Rejection sampling constructs the offline SFT supervision, whereas RL optimizes trajectories newly generated by the current policy.

Table 10: Training hyperparameters. Benchmark SFT uses a three-epoch budget, with the reported checkpoint selected by overall validation success rate. Grounding RL uses GRPO on 10,000 instructions excluded from grounding SFT.

Hyperparameter	**Benchmark SFT**	**Grounding SFT**	**Grounding RL**
Training budget (epochs)	3	2	1
Learning rate	3×10^{-6}	3×10^{-6}	5×10^{-7}
Learning-rate schedule	Cosine	Cosine	Cosine
Warmup ratio	0.08	0.05	0.05
Weight decay	0.05	0.1	0.01
Gradient clipping norm	1.0	1.0	1.0
Training batch size	64 examples	128 examples	8 prompts
Micro-batch size per GPU	1	1	1
Maximum sequence length	16,384	32,768	16,384
Rollouts per prompt	–	–	8
Rollout temperature	–	–	1.0
Rollout top-p	–	–	0.95
KL loss coefficient	–	–	0.01

C FRONTIER EVALUATION PROTOCOLS

GPT-5.6-Sol. Egocentric Drag and both Rotation variants use the official `computer` tool plus independent `mouse_down` and `mouse_up` functions. Each episode allows up to 12 model responses, including complete action batches unless the environment terminates. The added functions return a completion message without a screenshot; native computer-tool feedback is preserved. These three variants yield 7, 27, and 8 successes out of 150 each. Exocentric Drag and both Ten-Choice variants retain the earlier results. In main-table order, the six success counts are 150, 7, 150, 1, 27, and 8, giving 343/900 overall and 43/600 on I_2.

Kimi-K3. Egocentric Drag, both Ten-Choice variants, and both Rotation variants use the 13 official KimiCU MCP tools plus four local extensions: `move_to`, `mouse_down`, `mouse_up`, and `left_click`. The adapter supplies screenshots after actions, and the prompt explicitly instructs the agent to inspect feedback while holding a drag. Exocentric Drag retains the earlier six-action-interface result. The aggregate includes argument-compatibility handling and differing output-token limits; remaining protocol errors count as nonsuccesses. In main-table order, the six success counts are 150, 9, 150, 5, 38, and 9, each out of 150, giving 361/900 overall and 61/600 on I_2.

Both rows combine the specified evaluations with retained results. Models are evaluated with their respective interfaces, feedback, and interaction budgets.

D BENCHMARK CONTROL MECHANISMS

DRAG

The agent must drag a solid object into its matching outline among several candidate slots.

Variant A: Exocentric control (I_0). A free cursor moves a visible object toward its matching slot under a fixed camera and a globally known one-to-one screen-space mapping. There is no task-relevant environment latent to identify: the visible object and slot locations already determine the next held-object movement, so the task is I_0.

Variant B: Egocentric control (I_2). The matched variant uses a center-reticle, viewpoint-relative control law. Its dynamics environment latent is the two-axis held-object transfer gain $z_{\text{dragEgo}} = (g_x, g_y)$: for an executed cursor displacement Δu_t, the held object moves by $p_{t+1} - p_t = \text{diag}(g_x, g_y)\Delta u_t$. The evaluated variant uses equal positive gains on both axes, whose magnitude varies across episodes and is not shown in the initial frame. The same egocentric movement can therefore transport the held object by different amounts in otherwise identical samples. The agent must identify z_{dragEgo} by relating its executed displacement to the resulting visual transition, $(O_t, \Delta u_t, O_{t+1})$, before it can place the object efficiently, making the variant I_2.

Each Drag variant contains 75 IID cases using training shapes and 75 OOD cases using held-out shapes.

TEN-CHOICE

Variant A: Exocentric reveal (I_1). Ten visually identical candidates hide messages from $\mathcal{M} = \{m_1, \ldots, m_{10}\}$. The semantic environment latent is an episode-specific permutation $z_{10}^{\text{sem}} = \pi$ that maps candidate indices to messages. The instruction specifies a target message m^*. Before interaction, neither the initial frame nor the history identifies which candidate contains m^*. Moving onto candidate i renders $\pi(i)$ directly in O_{t+1}. Because the probe result itself reveals the task-relevant value without requiring action–effect comparison, this variant is I_1; after finding $\pi(i) = m^*$, the final click is an I_0 decision.

Variant B: Egocentric reveal (I_2). The matched variant retains the hidden permutation π and adds an episode-specific egocentric viewpoint law. Its dynamics environment latent is the gain $z_{\text{10Ego}}^{\text{dyn}} = (s_x, s_y)$, which maps movement Δu_t to viewpoint change $v_{t+1} - v_t = \text{diag}(s_x, s_y)\Delta u_t$. The complete environment latent is $(\pi, z_{\text{10Ego}}^{\text{dyn}})$. The permutation still requires I_1 revelation, but the task's required level is I_2: the agent must compare its movement with the displacement of the candidate field to infer (s_x, s_y) before it can steer efficiently to the candidates and reveal their messages.

ROTATION

The agent must rotate the controlled image region into alignment with the surrounding image.

Variant A: Inner-ring control (I_2). The inner-ring variant uses an episode-specific signed scalar gain $z_{\text{rot}} = k$. A horizontal movement Δx_t produces object rotation $\Delta\theta_t = k\Delta x_t$, so the same initial frame can require different movement magnitudes or directions. From one probe, the agent can estimate k by comparing the observed rotation change with its executed displacement and use that estimate to calibrate the next correction. Although O_{t+1} alone can verify whether the object is under- or over-rotated, identifying z_{rot} requires the full transition $(O_t, \Delta x_t, O_{t+1})$, making the task I_2.

Variant B: Outer-ring control (I_2). The matched variant moves the control to an outer ring. Its dynamics environment latent is the signed angular coupling $z_{\text{outer}} = k_{\text{outer}}$, which maps the ring displacement $\Delta\phi_t$ to object rotation $\Delta\theta_t = k_{\text{outer}}\Delta\phi_t$. The magnitude of k_{outer} determines response sensitivity, while its sign determines whether the object follows or opposes the outer-ring motion. The agent must compare the applied ring displacement with the observed object rotation to infer z_{outer}; a post-move image without the self-action is sufficient for verification but not calibrated control.

E ILLUSTRATED BENCHMARK INTERACTION SEQUENCES

Full replay sequences complement the environment overview in Figure 1 and the control laws in Appendix D.



Figure 3: **Shared first frames, different interaction mechanisms.** Action headers apply to both variants; coordinates may differ. Each numbered primitive executes separately, with intermediate frames omitted. Middle and right panels follow the actions above them. Arrows mark motion; dashed marks indicate pre-MOVE positions (Ten-Choice: action 1; Drag: action 3) or Rotation's fixed inner boundary. Scripted successes illustrate mechanics, not model performance.

F GAIN-DEPENDENT CONTROL AFTER SFT AND PAIRED RL

We analyze the saved egocentric Drag evaluation of the jointly trained Qwen3.5-9B SFT policy reported in Table 2. The evaluation contains 150 complete episodes, including 62 successes, with no infrastructure or model protocol errors. This analysis uses the same evaluated policy and requires no additional inference.

Measurement. For each `move_to` action that produces nonzero held-object movement, let u_t be the commanded displacement in screen pixels and e_t the pre-action target-minus-object residual, also in screen pixels. We measure the error-normalized command component along the residual,

$$\alpha_t = \frac{u_t^\top e_t}{e_t^\top e_t}, \qquad \bar{\alpha}_i = \mathrm{median}_{t \in \mathcal{M}_i}\, \alpha_t,$$

where $\mathcal{M}_i$ contains the eligible movements in episode i. Pearson correlations are computed across episodes between $\bar{\alpha}_i$ and either hidden gain g_i or its inverse $1/g_i$, giving each episode equal weight. Under the evaluated positive isotropic gain law, ideal unclamped correction has $\alpha = 1/g$. Hidden gain and geometry are read from evaluator-only logs for offline measurement and are not supplied to the policy.

Coverage and robustness. We apply no terminal-success filter. The full movement cohort contains 155 actions from 80 episodes: 62 successful and 18 failed, with 30 low-gain ($g < 1$) and 50 high-gain ($g > 1$) episodes. The remaining 70 episodes contain no held-object movement and all fail; their movement coefficient is undefined. The correlation therefore describes episodes with observed held-object movement.

A filtered cohort retains movements with a visible target, residual norm at least 20 pixels, an ideal correction within the action-coordinate range, and no clamping. We recompute the episode medians after filtering. Table 11 reports both correlations for the full and filtered cohorts of each policy. Correlations with $1/g$ describe alignment with ideal compensatory scaling; filtered results check robustness.

Table 11: Gain-dependent control after mixed SFT and paired RL in egocentric Drag. Pearson correlations relate episode-median normalized command to hidden gain g and ideal compensation $1/g$, using the same aggregation and movement filters for both policies.

Policy	Cohort	Episodes	Moves	$r(\bar{\alpha}, g)$	$r(\bar{\alpha}, 1/g)$
Mixed SFT	All held moves	80	155	-0.592	0.610
	Filtered	72	85	-0.557	0.616
Paired RL	All held moves	145	270	-0.820	0.803
	Filtered	135	177	-0.749	0.745

For mixed SFT, median $\bar{\alpha}$ is 1.232 in the low-gain band and 0.755 in the high-gain band; after filtering, the corresponding values are 1.375 and 0.755. For the full cohort, median ideal $1/g$ is 1.528 in the low-gain band and 0.729 in the high-gain band. The association reflects compensatory command scaling across the two gain bands.

Comparison with paired RL. The paired-RL evaluation succeeds on 138/150 episodes (92.0%), compared with 62/150 (41.3%) for mixed SFT. It contains no infrastructure errors; two episodes with protocol errors remain in the evaluation. Its 145 movement episodes comprise 138 successes and seven failures; the other five episodes have no held movement and fail.

On the 79 episode IDs with held movement under both policies, correlations are -0.591 for SFT and -0.813 for paired RL (154 and 143 moves, respectively). The stronger association therefore persists on shared episode IDs.

G Additional Benchmark Results and Failure Analysis

G.1 Performance across paired tasks

Table 2 shows that strong performance on I_0/I_1 tasks does not ensure success on their matched I_2 variants. GPT-6 Astra achieves 100% on exocentric Drag and Ten-Choice, but only 2.0% and 56.0% on their egocentric counterparts, despite having the highest I_2 average. GUI-specialized agents exhibit similar gaps. Several achieve only 8.0–14.7% on exocentric Ten-Choice (I_1), indicating difficulty acquiring the hidden target semantics. Qwen-CUA is an exception at 92.7%, yet falls to 1.3% on the paired I_2 variant; its Drag success likewise drops from 99.3% (I_0) to 0.7% (I_2). The matched comparisons expose distinct difficulties in acquiring hidden semantics and adapting movement to hidden dynamics, even when the task objective is unchanged.

LatentLearner-9B achieves 50.0% overall success and 26.2% on I_2, with 96.0% and 99.3% on the I_0/I_1 variants and 41.3% and 40.7% on egocentric Drag and Ten-Choice. Rotation remains difficult at 11.3% on each variant. In egocentric Drag, the Pearson correlation between error-normalized commanded displacement and the ideal gain-compensation coefficient $1/g$ is $r = 0.610$ across the 80 episodes containing held-object movement. This positive association indicates a tendency to compensate for gain: relative to target error, the policy issues larger commands at lower gains and smaller commands at higher gains. Appendix F gives the measurement protocol and coverage.

G.2 Frontier-agent failure analysis

To interpret the paired performance gaps in Section 5.2, we inspect frontier-agent trajectories.

Acting before observing feedback. In the inspected trajectories, agents sometimes commit before movement-induced feedback informs their next decision. In Rotation, a bundled drag releases before the model observes its effect. In egocentric Drag, agents may move and immediately press without checking how the scene has shifted relative to the fixed interaction center. Such sequences can suffice under known mappings, but leave no opportunity to adapt the next action when the movement response is uncertain.

Feedback-based correction remains unreliable. Sol, Astra, Opus, and Kimi also exhibit feedback-conditioned control: maintaining the press, observing the resulting rotation, adjusting subsequent movements, and then releasing. These behaviors show that agents can use intermediate feedback, but their corrections and stopping decisions remain unreliable. Some Astra trajectories reach an aligned state yet continue moving before release, turning a potentially successful attempt into failure. Effective control therefore requires coordinating observation, adjustment, and commitment, rather than simply obtaining more feedback.

Sensitivity to control geometry. Sol, Astra, Opus, and Kimi all perform worse on outer-ring than inner-ring Rotation (Table 2), although both variants are I_2 and share the alignment objective. Adaptation remains sensitive to control geometry.