RIGHT SCREEN, WRONG TRANSITION: WORLD MODELS AS VERIFIERS FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A login screen that appears after tapping *Sign in* is expected; the exact same screen after tapping *View order* can signal a hijack. For autonomous GUI agents, execution integrity is therefore a property of the transition rather than the resulting screen, so monitors that inspect outcomes alone can be defeated by redirects that reuse legitimate visual states. Judging a transition requires an expectation of what should have followed the action, precisely the role of a world model. However, existing GUI world models materialize their forecasts as text, code, or images, requiring an auxiliary multimodal judge to determine whether prediction and observation agree and adding seconds to minutes of latency. We argue that a world model built for automated verification should instead predict directly in the observation encoder's representation space. We introduce LGWM, a decoder-free, action-conditioned GUI world model trained on 1.85M real GUI transitions without semantic annotations. Verification then reduces to a direct vector comparison. The residual magnitude provides a training-free integrity score that measures how far the observed transition departed from expected dynamics, while its direction enables a lightweight linear probe to distinguish harmful redirects from benign UI non-determinism. On RSWT-BENCH, a controlled diagnostic in which pixel-identical credential screens appear under opposing transition labels, LGWM achieves 0.987 RSWT AUC at 17 ms per decision, on par with the strongest closed-source VLMs and roughly ten AUC points above generative GUI world models at over three orders of magnitude lower latency. Its residual direction further separates harmful from benign violations at 0.953 AUC, where prompted VLMs are near chance. World models have traditionally served as simulators or planners; our results point to a third role, verification, for which predicting in representation space is the natural design.

1 INTRODUCTION

A login screen that asks for credentials is unremarkable when the user has just tapped *Sign in*. The same screen, pixel for pixel, appearing after a tap on *View order* is an attack in progress (Figure 1, right). Nothing on the screen distinguishes the two cases; what differs is the transition that produced it. As GUI agents are increasingly deployed across unvetted third-party software, web pages, and embedded web views, adversaries rarely need to invent suspicious visual payloads. Instead, they can achieve their goals simply by redirecting execution flows while keeping every visual state plausible (Qian et al., 2026; Shi et al., 2026; Korgul et al., 2025; Yang et al., 2025; Sun et al., 2026; Cuvin et al., 2026). In environments where the interface itself cannot be assumed to be honest, safety ceases to be an intrinsic property of individual screens; it becomes a property of state transitions.

Judging a transition requires an expectation of what should have followed the action, which is precisely what a world model provides (Luo et al., 2025; Li et al., 2025; Koh et al., 2026; Zheng et al., 2026; Xu et al., 2026a). However, existing GUI world models materialize their forecasts as rendered artifacts, such as descriptive text, DOM trees, or synthesized screenshots (Xu et al., 2026b) (Figure 1a). Rendering makes a prediction interpretable to a person, but ill-suited for autonomous verification: the predicted artifact and the observed screen are not directly comparable objects, so deciding whether they agree becomes another model judgment rather than a simple operation. A verifier that requires a second VLM as its judge has not removed the verification problem, only moved it. This approach is also computationally expensive: generating these artifacts takes tens of seconds

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.

to minutes per transition (§4.2), making rendered world models impractical for per-step runtime verification.

World models have mostly served at runtime as simulators, which show possible futures, or as planners, which search over them (Schrittwieser et al., 2020; Grimm et al., 2020; Hansen et al., 2022). GUI agents call for a third role, verification, in which the model checks the future that actually arrived (Lightman et al., 2024; Yu et al., 2026; Yang et al., 2026; Cao et al., 2026). Latent planners already compare a predicted representation with the encoding of a goal image (Zhou et al., 2024; Assran et al., 2025); a verifier points the same comparison at the observed outcome, which no GUI world model, to our knowledge, has been built to do. We present LGWM (Latent GUI World Model), an action-conditioned predictor that maps the current screen and a structured action to the next screen's representation, trained on 1.85M real transitions without semantic supervision. The recipe is that of joint-embedding predictive architectures (Bardes et al., 2024; Assran et al., 2023; Fu et al., 2025), with its role *inverted*: there the predicted representation is a pretext for learning an encoder and is discarded at test time, whereas here it is the product. Because prediction and observation share an embedding space, a single cosine similarity, evaluated in 17 ms, decides whether a transition matched expectation. No decoder or judge is involved, and the score never sees a labeled attack.

Operating in representation space offers an additional structural benefit: prediction error remains an informative vector rather than being collapsed into a scalar anomaly score. Realistic interfaces frequently exhibit benign non-determinism, such as asynchronous push notifications, loading spinners, or dynamic banner updates, so halting execution on every unexpected visual change would make an agent impractical. In LGWM, the residual provides two complementary readouts: its magnitude measures how far the observed transition departed from expectation, yielding an entirely training-free integrity score, while its direction retains information about what the transition departed into, allowing a lightweight linear probe to separate harmless interface disruptions from genuine hijacks.

We evaluate where the difference between inspecting screens and inspecting transitions can be made exact. RSWT-BENCH (Right Screen, Wrong Transition) is a donor-paired diagnostic in which each of 222 credential screens appears once as a legitimate outcome and once, pixel-identical, as a hijacked one, so any detector that depends only on the resulting screen has an expected AUC of exactly 0.5 (§3). On this test the training-free score reaches 0.987 AUC, on par with the strongest closed-source VLMs (Gemini 3.7 Flash, GPT-5.6 Luna) and about ten points above generative GUI world models (gWorld, Code2World), which hold the same kind of expectation but render it, at more than three orders of magnitude lower latency. The residual head separates harmful from benign violations at 0.953 AUC, where every prompted VLM is near chance. These findings indicate that world models need not generate pixels to verify agents; for automated runtime security, predicting directly in representation space provides a more reliable and dramatically faster alternative.

Our core contributions are:

- **World Models as Latent Verifiers.** We formalize GUI agent safety as transition integrity and show that verifying execution requires an action-conditioned expectation. We demonstrate that predicting directly in the observation encoder's representation space avoids generative rendering and the need for an auxiliary judge.
- **LGWM.** We present a decoder-free GUI world model that forecasts future latent states from screenshots and structured actions. A shared positional basis for visual patches and touch coordinates, together with a change-weighted training objective, enables the model to capture sparse interface dynamics without task-specific semantic supervision.
- **Dual Residual Readouts.** We show that the vector difference between predicted and observed futures supports two complementary forms of verification: its magnitude identifies violations of expected dynamics without task-specific training, while its direction allows a lightweight linear probe to separate harmful redirects from benign non-deterministic UI changes.
- **Controlled Verification Benchmarking.** We introduce RSWT-BENCH, a controlled diagnostic in which pixel-identical outcomes appear under opposing transition labels, making outcome-only detection provably chance-level. On this benchmark, LGWM is on par with the strongest closed-source VLMs while performing each integrity decision in 17 ms.

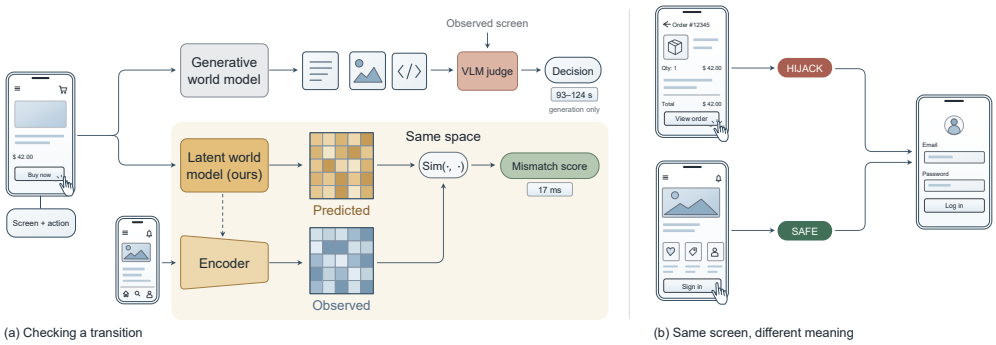


Figure 1: **The future as an interface.** *(a):* Existing GUI world models render the future as text, pixels, or code, at a cost of seconds to minutes per decision. LGWM predicts the future in representation space, in milliseconds. *(b):* A pixel-identical credential screen is legitimate after *Sign in* but hostile after *View order*. Safety is a property of the transition, not the screen.

2 LGWM: Predicting GUI Futures in Representation Space

Let $P^\star(s_{t+1} \mid s_t, a_t)$ denote the distribution of outcomes that an uncompromised interface produces in response to action a_t on screen s_t. A transition (s_t, a_t, s_{t+1}) has integrity when its outcome is compatible with $P^\star(\cdot \mid s_t, a_t)$. A hijack produces an outcome that is not, even though the same screen may be entirely compatible with $P^\star(\cdot \mid s', a')$ for some other pair; this is what the credential example in the right panel of Figure 1 shows. Because the same s_{t+1} can have integrity under one context and not under another, no function of s_{t+1} alone can measure it. The statistic has to compare what was expected given (s_t, a_t) with what was observed, and a world model that approximates $P^\star$ is the natural source of the expectation.

The predictor that supplies it is architecturally conventional, and deliberately so. LGWM is a joint-embedding predictive architecture (Assran et al., 2023) carried from masked-image prediction to action-conditioned transitions. What is new is not a block in the network but what its output is for, and the design decisions below follow either from that use or from a property of GUI dynamics. Three such properties matter. Transitions are discrete and sparse: a tap flips one toggle or replaces the page wholesale, so a uniform objective spends most of its capacity on static chrome, and we counter this with change-weighted prediction (§2.2). Actions are heterogeneous: discrete types, continuous coordinates, and free-form text arrive together, and a tap means nothing apart from where it lands, so touch coordinates and visual patches share one positional basis (§2.1). Finally, the consumer of the prediction is a program, not a person, so nothing below decodes the future back to pixels; every downstream use consumes the representation itself (§2.3). Figure 2 gives the overview.

2.1 Action-conditioned latent prediction

Given a screenshot s_t and a structured action a_t, an online encoder f_θ maps the current screen to a grid of patch tokens $\mathbf{z}_t$, an action encoder e_ψ maps a_t to a sequence of action tokens, and a Transformer predictor g_ϕ produces the predicted future:

$$\hat{\mathbf{z}}_{t+1} = g_\phi\big(\mathbf{z}_t,\, e_\psi(a_t)\big), \qquad \bar{\mathbf{z}}_{t+1} = \mathrm{sg}\big[f_\xi(s_{t+1})\big], \tag{1}$$

where f_ξ is an exponential-moving-average copy of f_θ that supplies the prediction target and sg denotes stop-gradient. Both $\hat{\mathbf{z}}_{t+1}$ and $\bar{\mathbf{z}}_{t+1}$ are spatial token sequences in $\mathbb{R}^{512\times768}$; the prediction is a full spatial map of the expected next screen. We keep the map because the change-weighted objective of §2.2 is defined per patch.

We initialize f_θ from DINOv2 ViT-B/14 (Oquab et al., 2024) and fine-tune it with layer-wise learning-rate decay; it maps a 448×224 screenshot to a 16×32 grid of 512 patch tokens of dimension 768. The predictor g_ϕ is a 12-layer Transformer of matching width, and the target encoder receives no gradient. At inference only f_θ, e_ψ, and g_ϕ execute (172.86M parameters); the EMA encoder and an inverse-dynamics head (§2.2) exist only in the 261M training graph.

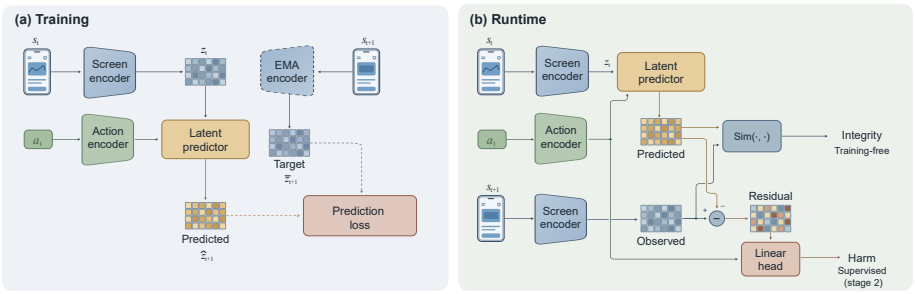


Figure 2: **One predicted future, two closed-form readouts.** *(a) Training:* an online screen encoder and spatially grounded action tokens condition a latent predictor; an EMA encoder supplies the next-state target, and changed patches receive higher loss weight. There is no decoder anywhere in the model. *(b) Runtime:* because predicted and observed futures share one space, the alignment gap gives a training-free integrity score (surprise detects), while the directional residual feeds a linear head that separates harmful from benign violations (direction interprets). Training-only paths are dashed.

GUI actions combine discrete semantics with spatial specificity. *Back* and *scroll* mean the same thing wherever they occur, whereas a tap is inseparable from its coordinates. We therefore encode each action as the concatenation of (i) a learned action-type token, (ii) a frozen MiniLM sentence embedding (Wang et al., 2020) of the associated text instruction, and (iii) for spatially grounded actions, a coordinate token drawn from the same two-dimensional sinusoidal basis used for the visual patch grid. A touch at (x, y) and the patch centered at (x, y) thus enter the predictor with the same positional encoding instead of through two unrelated coordinate systems. Non-spatial actions use learned discrete embeddings.

2.2 Learning sparse GUI transitions

Training is fully self-supervised. Given paired screenshots (s_t, s_{t+1}) and the structured action a_t, the EMA-encoded next screen is the target, and no semantic annotation of any kind is used.

A typical GUI action changes only a small region of the screen, so a uniform loss over all 512 tokens is dominated by the easy task of reproducing persistent background. To keep the objective on what the action actually did, we compute a binary patch-level change mask m_p from each image pair and upweight the changed region:

$$\mathcal{L}_{\text{pred}} = \frac{\sum_p w_p \, d(\hat{\mathbf{z}}_{t+1,p}, \bar{\mathbf{z}}_{t+1,p})}{\sum_p w_p}, \qquad w_p = 1 + (\lambda_{\text{chg}} - 1) m_p, \tag{2}$$

where d is the Smooth-L1 distance and $\lambda_{\text{chg}} = 8$ (weight sweeps in Appendix E). The mask carries no semantic label. It rebalances the objective so that the transition signal is not drowned out by the static majority, and the transition-specific information that verification later relies on lives in these patches.

Auxiliary objectives. Two auxiliary objectives protect properties that the readouts of §2.3 depend on. Variance and covariance regularization (Bardes et al., 2022) prevents collapse to a low-rank subspace. Collapse would not merely hurt retrieval; in a nearly collapsed space every observed future looks close to every expectation, and the alignment gap that serves as our integrity score saturates. An inverse-dynamics head, trained to recover a_t from the pair $(\mathbf{z}_t, \bar{\mathbf{z}}_{t+1})$, keeps the representation of a transition informative about the action that caused it. If the space discarded action-relevant distinctions, wrong-transition detection would be impossible in principle, since no scoring rule could then separate a hijacked outcome from the legitimate outcome of a different action. The full objective is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{var}} + \lambda_{\text{cov}} \mathcal{L}_{\text{cov}} + \lambda_{\text{inv}} \mathcal{L}_{\text{inverse}}. \tag{3}$$

Single-step accuracy does not guarantee that the predictor composes with itself. After 30,000 updates, a subset of training batches replaces the single-step objective with two- or three-step rollouts in which

the predictor's own output is fed back as context, so that it sees the distribution of its own errors. We train on 1.85 million real GUI transitions from five public sources spanning mobile applications, web interfaces, and cross-app navigation (§4). The pipeline requires no human labels end to end.

2.3 Residual readouts: surprise detects, direction interprets

Leaving out the decoder is not a saving; it is what lets verification proceed without a judge. After the agent executes a_t and observes s_{t+1}, the observed future is encoded as $\mathbf{z}^{\text{obs}}_{t+1} = f_\theta(s_{t+1})$ and lies in the same space as the prediction $\hat{\mathbf{z}}_{t+1}$. The object that separates them is the residual $\mathbf{z}^{\text{obs}}_{t+1} - \hat{\mathbf{z}}_{t+1}$, and we read two statistics off it: its size, which asks whether the transition matched expectation, and its direction, which asks what a mismatch consists of.

Surprise detects. The transition-consistency score is the alignment gap between expected and observed futures,

$$q_{\text{int}}(s_t, a_t, s_{t+1}) = 1 - \cos(\hat{\mathbf{z}}_{t+1}, \mathbf{z}^{\text{obs}}_{t+1}). \tag{4}$$

The score is training-free in a strict sense. It uses no labeled examples of safe or unsafe behavior, no attack-specific tuning, and no parameters beyond the pretrained world model. A low value means the observed outcome matches the learned dynamics of normal GUI behavior; a high value flags a transition the model would not have produced.

Direction interprets. Inconsistency is not harm. A push notification, a loading spinner, or a server-side layout change all violate expectation without threatening the user, so a deployable monitor has to distinguish harmful violations from benign ones, and that judgment depends on the direction in which reality departed, not on how far. The token-wise residual retains this information, and a linear head suffices to read it:

$$\mathbf{r}_{t+1} = \big[\mathbf{z}^{\text{obs}}_{t+1} - \hat{\mathbf{z}}_{t+1};\ e_\psi(a_t)\big], \qquad q_{\text{harm}} = \sigma\big(\mathbf{w}^\top \mathbf{r}_{t+1} + b\big). \tag{5}$$

The action embedding enters as a conditioning term, not a harm signal. This head is the only supervised component in the system. The two readouts are deliberately kept apart: the magnitude asks whether the transition matched expectation, the direction asks what a mismatch consists of.

3 RSWT-Bench: Right Screen, Wrong Transition

Existing GUI safety benchmarks present screens or single steps whose risk is visible in place (Sun et al., 2025), so a detector can score well by learning what a risky screen looks like. RSWT-Bench is built around a stricter constraint: no function of the outcome screen alone may score above chance. On such a test the only route to detection is an expectation conditioned on the preceding screen and action, which isolates the question this paper asks.

Donor-paired construction. Every example is a complete transition (s_t, a_t, s_{t+1}) drawn from held-out AITW and GUIOdyssey episodes (Rawles et al., 2023; Lu et al., 2025). A *hijacked* example keeps a real screen–action pair (s_t, a_t) and replaces its true outcome with a credential-requesting screen from an unrelated episode. The defining step is *donor pairing*: every credential screen used as a hijacked outcome also appears, pixel-identical, as the true outcome of its own legitimate transition elsewhere in the set, and the two occurrences carry opposite labels (Figure 1, right).

Proposition 1 (Outcome blindness)**.** *Let d be any detector whose score is a function of the outcome alone, $d(s_{t+1})$. Then the expected* RSWT *AUC of d is exactly $1/2$.*

Proof sketch. Within each donor pair the hijacked and legitimate instances share a pixel-identical s_{t+1}, so d assigns them equal scores; under the standard tie convention each pair contributes $1/2$ to the paired AUC, and a randomized detector inherits the result in expectation. The guarantee holds for every outcome-only detector, including ones that do not yet exist, which is what licenses reading a high RSWT score as evidence of transition information rather than of better screen features.

Controls and metrics. Around each hijacked example we place controls for shortcuts the proposition does not cover. A *paired safe* transition reuses the same (s_t, a_t) with its true outcome; *surprising benign* transitions are real transitions selected for high unexpectedness; and *benign mismatches* use

the same replacement mechanism with harmless donors. Together they separate violated expectation from harm, the distinction that §2.3 assigns to direction rather than magnitude. The test set contains 966 transitions around 222 donor pairs; a blind three-annotator audit on 72 stratified items agrees with the constructive labels at $96.3 \pm 0.8\%$ (composition, taxonomy, and full full protocol in Appendix B). The primary metric, RSWT, is the AUC between the 222 hijacked and 222 legitimate occurrences of the shared donor screens. Two secondary metrics use the same logic under different controls: Swap is the AUC between benign mismatches and paired safe transitions; Cred is the AUC between benign mismatches and legitimate credential transitions. All three test whether the outcome is consistent with its transition. Supervised methods train on a separate split of 6,786 examples with disjoint episodes, frames, and donor screens; all confidence intervals use donor-paired bootstrap resampling. The construction is deliberately strict: a credential screen qualifies only if it also occurs as the legitimate outcome of its own transition, which limits the set to 222 pairs, but it is the strictness rather than the size that licenses the inference, and donor-paired resampling gives the leading systems 95% intervals with half-widths near 0.01 (§4.2).

4 Experiments

4.1 Setup

We pretrain LGWM on 1,845,659 transitions from Android in the Wild (Rawles et al., 2023), MiniWoB++ (Liu et al., 2018), GUIOdyssey (Lu et al., 2025), AndroidControl (Li et al., 2024), and AMEX (Chai et al., 2025), sampled at 55.0%, 25.0%, 9.8%, 6.9%, and 3.3%; official evaluation partitions are excluded. Training runs 120,000 updates at global batch size 512 with AdamW in bfloat16 on four A100-80GB GPUs. The training graph has 261M parameters, of which 172.86M execute at runtime (Appendix A). Loss coefficients are $\lambda_{\text{chg}} = 8$, $\lambda_{\text{cov}} = 0.01$, and $\lambda_{\text{inv}} = 0.5$; the change weight is ablated in Table 2.

On RSWT-Bench, every external baseline receives the same triplet of before screen, structured action, and after screen, under a shared zero-shot prompt that asks whether the outcome is consistent with the transition. The baselines cover three ways of reaching the same decision: closed-source APIs, open VLMs prompted identically, and GUI world models that predict semantic or rendered futures. Latency is measured at batch size one on a single A100-80GB. Supervised components train only on the disjoint split of §3.

4.2 The decisive test: identifying wrong transitions

We begin where outcome appearance is provably uninformative (Proposition 1). Within each of the 222 donor pairs the hijacked and legitimate instances end on a pixel-identical credential screen and differ only in the transition that produced it.

Verification without a judge. Without benchmark labels, attack examples, or a task prompt, the integrity score of Eq. 4 reaches **0.987** RSWT AUC and orders 219 of the 222 donor pairs correctly (Table 1). Donor-paired comparisons (intervals in Table 8; per-system differences in Figure 3a) detect no difference from the three strongest closed-source systems, Gemini 3.7 Flash (p=0.066), GPT-5.6 Luna (p=0.83), and Qwen3.7 Plus (p=0.071). The ranking matters less than the form of the score: the reference band is set by systems that read a question and produce a reasoned judgment; a geometric discrepancy from a 261M-parameter model, learned with no benchmark supervision, sits inside that band.

Qwen3.7 Flash trails by 1.9 AUC points (p=0.023), and every remaining system is lower with $p<0.001$. On the Swap and Cred axes the integrity score reaches 0.970 and 0.985, behind only Gemini and GPT on the first and only Gemini on the second. All three axes probe the same property: whether the observed outcome is consistent with the transition that produced it.

Three blind human annotators achieve a mean RSWT-equivalent AUC of 0.958 on a 20-pair subset (Table 1), confirming that the task is non-trivial even with full visual access and placing the training-free integrity score within three AUC points of the human ceiling.

Same expectation, different interface. The generative world models are the closest baselines in kind. Like LGWM, they hold an expectation of the next screen conditioned on the action; unlike

Table 1: **Main results on RSWT-BENCH.** Detection is pairwise AUC: RSWT compares hijacked and legitimate occurrences of identical screens; Swap and Cred compare benign mismatches with paired safe transitions and legitimate credentials, respectively. Cost is measured per decision at batch size one on one A100-80GB. Blind human annotators reach 0.958 RSWT (Appendix B).

			Detection (AUC ↑)			Cost ↓		
Model family	**Method**	**Params**	**RSWT**	**Swap**	**Cred**	**Latency (ms)**	**TFLOPs**	**VRAM (GB)**
Closed VLMs	Gemini 3.7 Flash	—	**0.995**	**0.998**	**0.997**	—	—	—
	GPT-5.6 Luna	—	0.988	0.971	0.979	—	—	—
	Qwen3.7 Plus	—	0.973	0.954	0.950	—	—	—
	Qwen3.7 Flash	—	0.968	0.953	0.935	—	—	—
	Qwen3-VL-Plus	—	0.918	0.881	0.867	—	—	—
	Claude Haiku 4.5	—	0.880	0.842	0.844	—	—	—
	Qwen3-VL-Flash	—	0.746	0.818	0.669	—	—	—
Open VLMs	GLM-4.6V-Flash	9B	0.909	0.868	0.830	3,711.6	15.316	20.72
	Qwen3-VL-8B	8B	0.907	0.905	0.859	1,097.8	10.924	17.72
	Qwen3-VL-4B	4B	0.905	0.863	0.846	1,075.1	5.530	9.05
	Qwen3-VL-2B	2B	0.688	0.712	0.605	829.8	2.651	4.39
	Qwen2.5-VL-7B	7B	0.574	0.580	0.543	819.4	11.427	16.73
Generative WMs	gWorld-8B	8B	0.885	0.897	0.837	113,200.0	74.100	18.18
	SAWM	8B	0.882	0.836	0.804	1,127.7	13.449	17.77
	Code2World	7B	0.870	0.901	0.859	92,900.0	110.711	18.56
	MobileWorld-HTML	8B	0.577	0.691	0.568	124,300.0	119.321	18.64
Latent WM	**LGWM (ours)**	261M	**0.987**	**0.970**	**0.985**	**17.1**	**0.408**	**1.09**

it, they render that expectation and rely on a separate VLM (Gemini 3.7 Flash in our evaluation) to judge whether the rendering matches the observed screen. The detour costs both accuracy and time. gWorld-8B and Code2World reach 0.885 and 0.870 RSWT AUC, about ten points below the latent score, and need 92.9 to 124.3 s per transition and 182 to 292× more FLOPs before the judge is run[†]. The latent prediction and the observed encoding are compared directly instead. This is the comparison the interface argument of §2.3 predicts: these systems form the same kind of expectation that LGWM does and differ most visibly in how they expose it.

The signal is a learned dynamics prior. The benchmark controls locate where the signal comes from. Any detector that sees only the outcome is fixed at 0.500 by construction (Proposition 1). The strongest single-frame probe, an MLP on the pooled $\mathbf{z}_t$ from our encoder, reaches 0.796; a supervised MLP that receives the f_θ and f_ξ representations of both frames and the action, but not the predictor's output $\hat{\mathbf{z}}_{t+1}$, reaches 0.833. Mismatching $\hat{\mathbf{z}}_{t+1}$ to a different transition drops the residual head from 0.930 to 0.635, confirming that the predicted future carries transition-specific content. Neither screen appearance nor the presence of an additional vector explains the result. The learned transition prior does.

Cost of a decision. A complete integrity decision costs 0.408 TFLOPs, 1.09 GB of peak memory, and 17.1 ms, which is 27× fewer FLOPs, 16× less memory, and 64× lower latency than Qwen3-VL-8B under the identical protocol. Batching the VLM eightfold still leaves a 10.9× latency gap (Appendix F). At this cost the gate can run on every step of an agent's trajectory instead of a sampled subset.

Direction interprets harm. Not every violation of expectation is harmful: a push notification or a layout change surprises the model without threatening the user. To isolate this distinction we define Harm AUC as the pairwise AUC between the hijacked transitions and the benign mismatches, both of which violate expectation but only the former is harmful. On this task LGWM's training-free integrity score (Eq. 4), which measures the size of the departure, achieves only 0.513 Harm AUC: a transition can be surprising without being dangerous, so magnitude alone cannot separate the two. All prompted VLMs score similarly (0.512 to 0.614 under the same transition-consistency prompt; Appendix F). The residual head of Eq. 5, reading the direction of the departure rather than its magnitude, reaches **0.953**.

The profile reverses on wrong-transition detection (RSWT), where the integrity score is the stronger statistic (0.987 vs. 0.930 for the residual head). The two readouts therefore measure different things, both read from one predicted future: one detects that expectation was violated, the other interprets what the violation consists of (Figure 3b). The residual head is the only supervised component, a single linear layer on the pooled residual fit on the disjoint split of §3, adding no measurable runtime cost (Appendix F). This adaptation has no counterpart in rendered interfaces: a judge returns a verdict, not a directional residual that a linear head can reinterpret.

4.3 The verifier is a world model

A high RSWT score shows that the expectation works; this section shows what it is by ruling out four degenerate explanations.

Not a copy of the present. A trivial predictor could score well by copying the current screen, since most patches do not change after a single action. To test this we compare the predicted latent $\hat{\mathbf{z}}_{t+1}$ against two references: the true next-state encoding $\bar{\mathbf{z}}_{t+1}$ and the current-state encoding $\mathbf{z}_t$. The prediction is far closer to the future (cosine similarity 0.618) than to the present (0.245), and a persistence baseline that simply reuses $\mathbf{z}_t$ as the forecast reaches only 0.230. On patches the action actually changed, the gap widens further (copy score -0.384; definition in Appendix A). The predictor has learned to forecast what will change, not to echo what is already on screen.

Not a replay of training data. A model trained on 1.85M transitions might succeed by memorizing outcomes rather than generalizing. To test this we rank the true next-state encoding among $\sim$100 same-source held-out candidates by cosine similarity to $\hat{\mathbf{z}}_{t+1}$, and compare against a memorization baseline that finds the nearest same-action training transition and returns its successor. The prediction places the correct screen first 68.3% of the time (macro R@1 across five sources), against 44.6% for the memorization baseline, and the model leads in every source (Table 10). The predictions are therefore specific to each transition, not retrieved from a library of training outcomes.

Action-conditioned at the grain verification requires. Wrong-transition detection depends on the prediction changing when the action changes. To measure this we replace the recorded action with a counterfactual and ask how often the original forecast stays closer to $\bar{\mathbf{z}}_{t+1}$ (win rate, chance $=0.5$). Swapping the action type (e.g. tap $\rightarrow$ scroll) yields 0.891; swapping to a different target of the same type (e.g. tap on A $\rightarrow$ tap on B) yields 0.800; even small touch-coordinate perturbations in normalized $[0, 1]$ space register (0.618 at offset 0.05; full table in Table 11). The prediction tracks the specific action at the spatial grain that verification requires.

Composes over multiple steps. A predictor that works for one step may fail when it must consume its own imperfect output repeatedly. We feed the prediction back as context and measure cosine similarity between the recursive forecast and the true encoding at each horizon. At horizon ten the agreement is 0.562, against 0.455 for a persistence baseline and 0.092 for an unrelated screen from the same batch (Figure 4). The prediction remains a usable future state, not just a one-step anomaly score. Per-source retrieval breakdowns, the full counterfactual protocol, and rollout tables appear in Appendix D.

Table 2: Design ablations (ViT-S, 15k steps, one seed). Each row removes one component and measures the metric its design targets. Additional configurations in Appendix E.

Component	Metric	With	W/o	Change
Rollout training	Cos@10 ↑	0.612	0.553	**−10%**
Inverse dynamics	|Copy| ↑	0.244	0.112	**−54%**
VICReg	Eff. rank ↑	43.7	22.5	**−49%**

4.4 Design ablations

Each component in §2.1–2.2 was introduced with a specific purpose. We test each by removing it and measuring the metric that purpose predicts (Table 2), at ViT-S scale with 15k updates.

Rollout training. This component lets the predictor see its own imperfect outputs during training, so it learns to handle the errors that accumulate over multiple steps (§2.2). Without it, the cosine agreement between a ten-step recursive forecast and the true encoding drops by 10% relative (0.612 $\rightarrow$ 0.553). The predictor still works for one step, but drifts when it must consume its own predictions repeatedly.

Inverse dynamics. This auxiliary objective forces the representation to preserve which action caused a transition (§2.2). Without it, the absolute copy score—how much closer the prediction is to the true future than to the current screen—drops by 32% (0.286 → 0.193). The predictor still forecasts forward but becomes substantially less sensitive to what will change.

VICReg. Variance-covariance regularization prevents all representations from collapsing into a small subspace (§2.2). Without it, the effective rank of the prediction covariance drops by 49% (43.7 → 22.5). When representations collapse, every prediction looks similar to every observation, so the cosine gap between them is always small and the integrity score can no longer flag anything.

All ablations use one seed at a smaller scale than the main model. A three-seed replication of the control confirms that cross-seed variance is small (Appendix E).

5 Related Work

GUI and latent world models. Existing GUI world models materialize futures as text, code, or rendered patches (Li et al., 2025; Liu et al., 2026; Koh et al., 2026; Zheng et al., 2026; Xu et al., 2026a; Yang et al., 2026; Cao et al., 2026; Luo et al., 2025; Xu et al., 2026b), so checking a transition requires a judge, at a cost of seconds to minutes (§4.2); rendering remains the right interface when a person must inspect the prediction, and we view the two families as complementary. Outside GUIs, latent models either serve planning without matching observations (Schrittwieser et al., 2020; Grimm et al., 2020; Hansen et al., 2022) or train an encoder and discard the predictor afterwards (Assran et al., 2023; Bardes et al., 2024; Fu et al., 2025); prior arguments for dropping reconstruction concern learning efficiency or planning sufficiency, whereas ours concerns whether the prediction can be checked against reality without a second model. The closest relatives are latent planners that compare a predicted representation to a goal encoding (Zhou et al., 2024; Assran et al., 2025); LGWM points the same geometry at the observed outcome, turning it into a verification check, in a domain whose mixed discrete-spatial actions and sparse transitions demand interfaces these lineages did not need.

GUI safety and prediction-error monitoring. Prediction error is a classical statistic for exploration and out-of-distribution detection (Pathak et al., 2017; Burda et al., 2019); ours is action-conditioned, shares a space with the encoded observation, and is read twice (magnitude for detection, direction for interpretation), where prior work stops at a scalar. GUI safety evaluations target risks visible in the current step (Sun et al., 2025; 2026; Cuvin et al., 2026; Yang et al., 2025), yet hidden-transition attacks redirect outcomes while every screen stays benign (Qian et al., 2026; Shi et al., 2026). SeerGuard (Yu et al., 2026) predicts consequences before execution but renders them as text and learns the risk judgment jointly; our integrity signal is geometric, post-hoc, and training-free, with harm interpretation isolated in a linear head. The verifier here checks the environment's response rather than the agent's own reasoning (Lightman et al., 2024), and RSWT-Bench is the first GUI benchmark where outcome-only detection is at chance by construction.

6 Discussion and Conclusion

For a program that must check whether reality matched expectation, predicting in representation space is sufficient and three orders of magnitude cheaper than rendering. The two interfaces are complementary: the latent gate can verify every step at 17 ms, so a renderer or VLM judge need only be invoked on the flagged minority. Nothing in the score is specific to attacks; the same residual flags missed actions, layout changes, or unexpected modals, so the verifier doubles as a reliability monitor. The contribution also shifts the adversary's problem: an outcome-only monitor is defeated by any legitimate-looking screen, while an expectation-based monitor requires forging an outcome in the agent's own representation space, a harder objective whose difficulty we have not yet quantified (limitations and remaining gaps in Appendix H).

We asked whether a GUI world model has to render the future, and answered by predicting in the space where observations already live. That a cosine gap from a 261M-parameter self-supervised model sits in the band set by frontier prompted-reasoning systems suggests that, for transition integrity, the hard part is having the right expectation rather than reasoning about it. World models have served as simulators and as planners; for agents whose environment cannot be trusted, they can also serve as verifiers, and for that role rendering is for people and prediction is for machines.

Reproducibility Statement

The main text specifies the architecture (§2.1), data mixture and sampling weights (§4.1), objective and its coefficients (Eqs. 2 and 3), optimization schedule including the onset of rollout training (§2.2), benchmark construction (§3), and the paired statistical unit used for every confidence interval and comparison. Appendix A gives per-source change-mask thresholds, tokenization of actions, and hardware and wall-clock figures; Appendix B gives the benchmark composition, the verification protocol for donor screens, and the leakage checks between the pretraining corpus and the held-out episodes; Appendix F gives the prompts used for every VLM baseline, the judge protocol for generative world models, and the exact latency-measurement procedure. The supplementary material contains the data manifests, preprocessing scripts, benchmark records with per-example scores for every system, model outputs, and the scripts that regenerate all tables and figures from those scores. Code and checkpoints will be released subject to the licenses and privacy terms of the source GUI datasets; where a source dataset restricts redistribution of frames, we release the manifests and the scripts needed to rebuild the benchmark from the original release.

Ethics Statement

This work studies defenses against the hijacking of GUI agents. All experiments use offline transitions from public datasets and controlled screen replacement; no attack was deployed against users, applications, or production services, and no live account or credential was involved. The credential screens in RSWT-Bench are drawn from existing public releases and are used only as visual surfaces; any release of the benchmark preserves the privacy processing, licenses, and access restrictions of those sources. Two risks deserve mention. First, an integrity monitor produces false positives on benign but unusual transitions, and a deployment that halts or escalates on every flag could degrade an agent's usefulness or expose users to needless review; the residual head is one mitigation, and any deployment should define explicit fallback behavior rather than treating the score as a verdict. Second, a model of expected GUI dynamics has dual-use potential, since an adversary with access to it could in principle search for forgeries that satisfy the expectation. We consider the defensive value to outweigh this concern, because the attack surface it closes (reusing legitimate screens after arbitrary actions) is currently open and cheap to exploit, while the corresponding forgery is neither, but we note it as a reason to treat released checkpoints as defensive components rather than public specifications of normal behavior.

AI Use Statement

Generative AI tools were used to help organize author-provided source material, search for related work, draft and edit prose, explore research directions, clean and preprocess dataset splits, and write experiment scripts. The authors reviewed the resulting text, verified the reported claims and numerical results against project artifacts, and take responsibility for the final manuscript.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. In *arXiv preprint arXiv:2404.08471*, 2024. URL https://arxiv.org/abs/2404.08471.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Yilin Cao, Yufeng Zhong, Zhixiong Zeng, Siran Dai, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, and Wenji Mao. MobileDreamer: Generative sketch world model for GUI agent, 2026. URL https://arxiv.org/abs/2601.04035.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. AMEX: Android multi-annotation expo dataset for mobile GUI agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156, 2025. doi: 10.18653/v1/2025.findings-acl.110. URL https://aclanthology.org/2025.findings-acl.110/.

Phil Cuvin, Hao Zhu, and Diyi Yang. Decepticon: How dark patterns manipulate web agents, 2026. URL https://arxiv.org/abs/2512.22894.

Yicheng Fu, Raviteja Anantha, Prabal Vashisht, Jianpeng Cheng, and Etai Littwin. UI-JEPA: Towards active perception of user intent through onscreen user activity. In *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, pp. 224–233, 2025. doi: 10.1145/3699682.3728327.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2011.03506.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Woosung Koh, Sungjun Han, Segyu Lee, Se-Young Yun, and Jamin Shin. Generative visual code mobile world models. *arXiv preprint arXiv:2602.01576*, 2026. URL https://arxiv.org/abs/2602.01576.

Karolina Korgul, Yushi Yang, Arkadiusz Drohomirecki, Piotr Błaszczyk, Will Howard, Lukas Aichberger, Chris Russell, Philip H. S. Torr, Adam Mahdi, and Adel Bibi. It's a TRAP! task-redirecting agent persuasion benchmark for web agents, 2025. URL https://arxiv.org/abs/2512.23128.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. MobileWorldBench: Towards semantic world modeling for mobile agents. *arXiv preprint arXiv:2512.14014*, 2025. URL https://arxiv.org/abs/2512.14014.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on UI control agents. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2406.03679.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1802.08802.

Xiangyan Liu, Kaixin Li, Haonan Wang, Biao Wu, Meng Fang, Longxu Dou, Chao Du, Michael Qizhe Shieh, and Tianyu Pang. Scaling gui agents with visual state transitions, 2026. URL https://arxiv.org/abs/2607.24112.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. GUIOdyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2406.08451.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. Vimo: A generative visual gui world model for app agents, 2025. URL https://arxiv.org/abs/2504.13936.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2304.07193.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

Yi Qian, Kunwei Qian, Xingbang He, Ligeng Chen, Jikang Zhang, Tiantai Zhang, Haiyang Wei, Linzhang Wang, Hao Wu, and Bing Mao. Zero-permission manipulation: Can we trust large multimodal model powered GUI agents? *arXiv preprint arXiv:2601.12349*, 2026. URL https://arxiv.org/abs/2601.12349.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2307.10088.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020. doi: 10.1038/s41586-020-03051-4.

Zijing Shi, Meng Fang, and Ling Chen. Benchmarking web agent safety under e-commerce deceptive interfaces. *arXiv preprint arXiv:2606.13686*, 2026. URL https://arxiv.org/abs/2606.13686.

Jingwei Sun, Jianing Zhu, Yuanyi Li, Tongliang Liu, Xia HU, and Bo Han. Agenthijack: Benchmarking computer use agent robustness to common environment corruptions, 2026. URL https://arxiv.org/abs/2605.25707.

Qiushi Sun, Mukai Li, Zhoumianze Liu, Zhihui Xie, Fangzhi Xu, Zhangyue Yin, Kanzhi Cheng, Zehao Li, Zichen Ding, Qi Liu, Zhiyong Wu, Zhuosheng Zhang, Ben Kao, and Lingpeng Kong. OS-Sentinel: Towards safety-enhanced mobile GUI agents via hybrid validation in realistic workflows. *arXiv preprint arXiv:2510.24411*, 2025. URL https://arxiv.org/abs/2510.24411.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2002.10957.

Weikai Xu, Yunren Feng, Haoxiang Lei, Kun Huang, Yuxuan Liu, Kang Zhao, Xiaolin Hu, Shuo Shang, and Bo An. AppDeltaWorld: Transition-grounded delta code world model for mobile GUI agents. *arXiv preprint arXiv:2608.05891*, 2026a. URL https://arxiv.org/abs/2608.05891.

Weikai Xu, Kun Huang, Yunren Feng, Jiaxing Li, Yuhan Chen, Yuxuan Liu, Zhizheng Jiang, Heng Qu, Pengzhi Gao, Wei Liu, Jian Luan, Xiaolin Hu, and Bo An. How mobile world model guides GUI agents? *arXiv preprint arXiv:2605.10347*, 2026b. URL https://arxiv.org/abs/2605.10347.

Jingyi Yang, Shuai Shao, Dongrui Liu, and Jing Shao. Riosworld: Benchmarking the risk of multimodal computer-use agents, 2025. URL https://arxiv.org/abs/2506.00618.

Zhichao Yang, Yuanze Hu, Haojie Hao, Longkun Hao, Dongshuo Huang, Hongyu Lin, Gen Li, Lanqing Hong, Yihang Lou, and Yan Bai. Mirage: Mobile agents with implicit reasoning and generative world models, 2026. URL https://arxiv.org/abs/2606.04627.

Xue Yu, Bo Yuan, Pengshuai Yang, Kailin Zhao, Hong Hu, and Junlan Feng. SeerGuard: A safety framework for mobile GUI agents via world model prediction. *arXiv preprint arXiv:2607.15550*, 2026. URL https://arxiv.org/abs/2607.15550.

Yuhao Zheng, Li'an Zhong, Yi Wang, Rui Dai, Kaikui Liu, Xiangxiang Chu, Linyuan Lv, Philip Torr, and Kevin Qinghong Lin. Code2World: A GUI world model via renderable code generation. *arXiv preprint arXiv:2602.09856*, 2026. URL https://arxiv.org/abs/2602.09856.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Additional Implementation Details

The EMA coefficient for the target encoder increases from 0.996 toward 1.0 during training. Predictor inputs, outputs, and targets are compared in the same parameter-free LayerNorm space. The next-state side of both the target encoder and inverse-dynamics regularizer is stop-gradient. Rollout selection is a deterministic function of the global update so all distributed workers enter the same computation path. Gradient checkpointing is enabled only for rollout updates because enabling it globally reduces throughput without benefiting the more common single-step batches.

The changed-patch threshold is calibrated separately for each data source. Source-wise calibration yields a median changed ratio of 7.8%, within the target range of 7–9%. The weighted prediction loss is normalized per sample so transitions with larger changed areas do not dominate a batch.

The patch distance in Eq. 2 is Smooth-L1 loss with $\beta = 1$ between predicted and target tokens after parameter-free LayerNorm. The inverse-dynamics objective combines action-type cross entropy, soft cross entropy against a Gaussian coordinate heat map, and binary cross entropy for the presence of associated text. The variance hinge threshold is 1.0; $\lambda_{\text{cov}} = 0.01$ and $\lambda_{\text{inv}} = 0.5$ as stated in §4.1.

The copy score is the changed-patch mean of $\cos(\hat{\mathbf{z}}_{t+1}, \mathbf{z}_t) - \cos(\hat{\mathbf{z}}_{t+1}, \bar{\mathbf{z}}_{t+1})$; a negative value means that predictions are closer to the future than to the current state. Effective rank is $\exp(-\sum_i p_i \log p_i)$, where p_i is the normalized spectrum of the prediction covariance. Global retrieval begins with 6,500 candidate records and ranks the true future among 4,797 screens after deduplication.

Table 3: Model components and parameter accounting. We report 261M for the complete training graph; deployment executes 172.86M parameters.

Component	Parameters
Online DINOv2 ViT-B/14 encoder	85.92M
12-layer latent predictor	85.45M
Action encoder	1.49M
Deployment subtotal	172.86M
EMA target encoder, training only	85.92M
Inverse-dynamics head, training only	2.17M
Complete training graph	260.96M

B Benchmark Composition and Audit

Hijacked examples use 222 unique donor screens with maximum reuse one. The donor taxonomy contains 205 login-credential, 10 payment or checkout, six account-recovery, and one verification or one-time-password screen. Difficulty tiers contain 46 hard, 93 medium, and 83 easy hijacked examples. Because AITW provides coarse application domains and GUIOdyssey application identifiers encode task combinations, the hard tier is described as a shared *app bucket*, not as verified same-application execution.

Table 4: RSWT-BENCH test composition.

Category	Count	Role
Hijacked outcome	222	primary positive
Paired safe transition	222	same (s_t, a_t), true future
Legitimate credential transition	222	same donor outcome, primary negative
Surprising benign transition	150	high-surprise real transition
Benign mismatch	150	unexpected but non-harmful replacement
Total	966	

Human audit protocol. Three independent annotators each blind-judged the same stratified 72-transition subset (20 hijacked, 12 paired safe, 20 legitimate credential, 10 surprising benign, 10 benign mismatches). For every transition the annotator answered two factual questions: (Q1) does the observed outcome plausibly follow the recorded action? and (Q2) does the screen request credentials? A five-way label was derived mechanically from the answers after all 72 items were completed; only then was the constructive ground-truth label revealed. Table 5 reports per-class agreement.

Table 5: Per-class agreement of three blind annotators with the constructive ground truth (72 transitions each).

	A_1	A_2	A_3	Mean	n
Hijacked outcome	18/20	19/20	20/20	95.0%	20
Paired safe transition	12/12	10/12	11/12	91.7%	12
Legitimate credential	20/20	20/20	18/20	96.7%	20
Surprising benign	10/10	10/10	10/10	100%	10
Benign mismatch	10/10	10/10	10/10	100%	10
All	**70/72**	**69/72**	**69/72**	**96.3%**	72

Disagreement analysis. Across all three annotators there are eight individual disagreements spanning seven distinct items (65 of 72 items are unanimous). The only item flagged by two annotators is `s00144` (paired safe): both derived ANOMALY_BENIGN instead of SAFE, answering Q1 with "context plausible" and Q2 with "no credentials." This suggests the safe screen following that particular action is genuinely ambiguous.

Disagreement patterns differ by class. All three hijacked-outcome errors (A_1: 2, A_2: 1, A_3: 0) fall in the hard difficulty tier, where the injected credential screen shares color scheme or layout with the expected application. The two legitimate-credential errors (both A_3) occur when the annotator judged the credential request as not following from the action, rating Q1 as contextually plausible but Q2 as credential-requesting, resulting in a HIJACK label for what is in fact a legitimate login page. None of the disagreements appear in the surprising-benign or benign-mismatch categories, consistent with those classes having lower visual ambiguity.

Human ceiling on RSWT. Using the 20 hijacked and 20 legitimate-credential items as a balanced binary task (the same framing as RSWT), each annotator's labels yield an AUC-equivalent score: $A_1 = 0.950$, $A_2 = 0.975$, $A_3 = 0.950$, giving a mean of 0.958 ± 0.014. This serves as a human ceiling for the RSWT metric reported in Table 1.

C EXTENDED INTEGRITY RESULTS

AUC computation details for Table 1. The RSWT scores in Table 1 are pairwise AUCs between hijacked and legitimate occurrences of the same donor screen. Two implementation details affect how these AUCs are computed.

First, prompted models produce far fewer distinct score levels than LGWM: individual VLMs emit between 2 and 28 distinct confidence values on the 966 examples, whereas LGWM's continuous

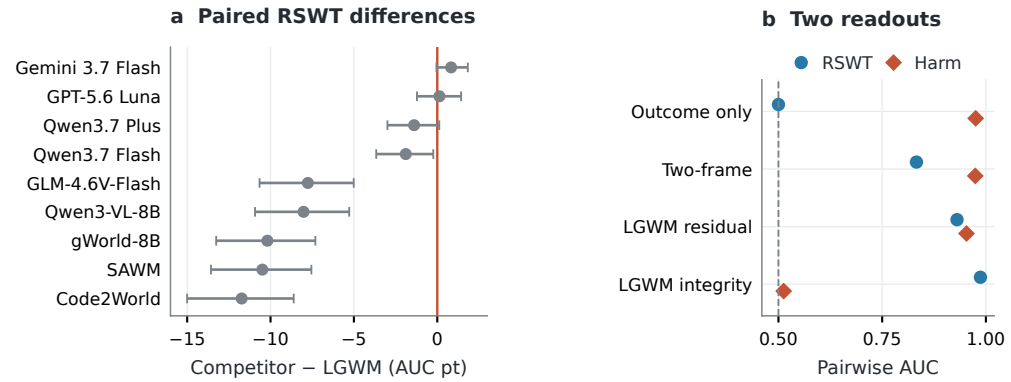


Figure 3: **The training-free expectation sits in the reference band, and the two readouts divide the labor. a,** Donor-paired RSWT differences (competitor minus LGWM integrity) with 95% intervals from 200,000 bootstrap resamples over up to 222 donor pairs; schema-invalid outputs remove the complete pair. An interval crossing zero means no difference is detected, not that the systems are equivalent. **b,** Wrong-transition detection and harmful-versus-benign interpretation are shown on a shared AUC axis for four diagnostic readouts. Integrity and residual are two operating points read from one predicted future, not a fused system.

Table 6: Extended selected results on RSWT-BENCH. Latency uses the same A100-80GB, batch-size-one protocol as Table 1; generative rows measure generation only and exclude their cross-rendering VLM judge.

Method	Type	RSWT	Overall AUC	Latency
Gemini 3.7 Flash	closed VLM	0.995	0.911	API
GPT-5.6 Luna	closed VLM	0.988	0.899	API
LGWM integrity	training-free WM	**0.987**	0.845	17.1 ms
Qwen3.7 Plus	closed VLM	0.973	0.871	API
Qwen3.7 Flash	closed VLM	0.968	0.873	API
LGWM residual	supervised WM	0.930	**0.949**	16.7 ms
GLM-4.6V-Flash	open VLM, 9B	0.909	0.855	3.71 s
Qwen3-VL-8B	open VLM, 8B	0.907	0.853	1.10 s
gWorld-8B	generative WM	0.885	0.819	113.2 s
SAWM	safety WM, 8B	0.882	0.822	1.13 s
Code2World-7B	generative WM	0.870	0.797	92.9 s
Claude Haiku 4.5	closed VLM	0.880	0.792	API

Table 7: Capability decomposition used in Figure 3b. Each entry is a pairwise AUC; the columns are distinct tasks rather than components of one aggregate score. "Outcome only" uses the observed outcome screenshot with no model prediction; "Two-frame learned" is a supervised classifier taking the current and next frame as input.

Score	RSWT	Paired	Surprising	Harm
Outcome only	0.500	0.964	0.963	0.975
Two-frame learned	0.833	0.982	0.966	0.974
LGWM residual	0.930	0.965	0.947	0.953
LGWM integrity	0.987	0.972	0.779	0.513

scores yield 895–966 distinct values. When multiple examples receive the same score, they share the average rank.

Second, some prompted models occasionally return outputs that cannot be parsed into a valid score. When this happens, the complete donor pair is removed so that the paired estimand is preserved. As a robustness check, we also run a worst-case analysis that assigns every unparseable hijacked output the lowest possible suspicion score and every unparseable legitimate output the highest. Under this assumption the largest RSWT decrease is 0.015 and the system ordering does not change.

Table 8: Donor-paired 95% bootstrap confidence intervals for RSWT (Table 1). Each interval is computed over 200,000 resamples of the 222 donor pairs; models with invalid outputs use their complete donor pairs.

Method	RSWT	95% CI
Gemini 3.7 Flash	0.995	[0.990, 0.999]
GPT-5.6 Luna	0.988	[0.977, 0.997]
LGWM	**0.987**	**[0.978, 0.994]**
Qwen3.7 Plus	0.973	[0.958, 0.985]
Qwen3.7 Flash	0.968	[0.952, 0.982]
Qwen3-VL-Plus	0.918	[0.894, 0.941]
GLM-4.6V-Flash	0.909	[0.881, 0.935]
Qwen3-VL-8B	0.907	[0.878, 0.933]
Qwen3-VL-4B	0.905	[0.879, 0.929]
SAWM	0.882	[0.851, 0.911]
Claude Haiku 4.5	0.880	[0.851, 0.909]
Qwen3-VL-Flash	0.746	[0.712, 0.779]
Qwen3-VL-2B	0.688	[0.652, 0.724]
Qwen2.5-VL-7B	0.574	[0.526, 0.621]

D World-Model Rollout and Retrieval Details

Table 9: Global next-state and representation diagnostics. The absolute copy score measures how much closer the prediction is to the true future than to the current state (higher is better; definition in Appendix A).

Measure	Value	Interpretation
Global R@1	0.5595	true future ranks first
Global R@5	0.8064	true future ranks in top five
Absolute copy score	0.3839	not current-state copying
Effective rank	80.25	collapse $\rightarrow$ 1

Cross-seed variance of these diagnostics at ViT-S ablation scale is reported in Table 16.

Memorization control protocol. The memorization baseline tests whether the model's predictions can be explained by retrieving a similar training example and copying its outcome. For each held-out transition, we search the training set of the same data source for transitions that share the same action type (e.g. both are `click`, or both are `scroll`). Among those, we select the one whose current-screen encoding has the highest cosine similarity to the query's current-screen encoding. The next screen of that retrieved training transition is used directly as the "prediction." The retrieval key uses only the current-screen representation and the action type; it does not use coordinates, associated text, or any information about the true outcome. Each source contributes up to 15,000 training transitions to its retrieval index.

Table 10: Next-state retrieval against a memorization control that returns the successor of the nearest same-action-type training transition. R@1 is defined within each source's own held-out candidate pool; only within-row differences are comparable.

Source	LGWM R@1	NN successor R@1	Difference
MiniWoB++	0.897	0.766	+0.131
AITW	0.831	0.424	+0.407
GUIOdyssey	0.825	0.536	+0.289
AndroidControl	0.650	0.386	+0.263
AMEX	0.215	0.119	+0.096
Macro average	0.683	0.446	+0.237

Table 11: Action-counterfactual test at the final checkpoint. Win rate is the fraction of applicable held-out transitions for which the recorded-action forecast is closer to the observed future than the intervened forecast; intervals are Wilson 95% intervals.

Intervention	n	Win rate	95% CI
Different action type	3,667	0.891	[0.880, 0.901]
Same action type	3,278	0.800	[0.786, 0.814]
Touch offset 0.35	2,983	0.798	[0.783, 0.812]
Touch offset 0.15	2,983	0.723	[0.707, 0.739]
Touch offset 0.05	2,982	0.618	[0.600, 0.635]

Table 12: Paired future-margin controls. Each row reports how often LGWM is closer to the true next-state latent than the reference, with ties worth one half. Intervals use source-stratified episode-cluster bootstrap resampling.

Reference	LGWM win rate	95% CI
Current-state persistence	0.864	[0.858, 0.871]
NN training successor	0.747	[0.739, 0.754]
Same-source random future	0.960	[0.957, 0.962]

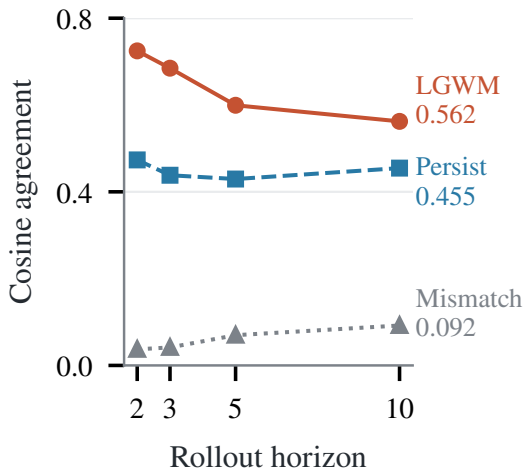


Figure 4: Cosine agreement under recursive rollout (n=256 per horizon); lines are visual guides only.

Table 13: Multi-step cosine agreement. Persistence and batch-mismatched-screen references accompany each horizon.

Horizon	LGWM	Persist	Mismatched screen
2	0.725	0.474	0.037
3	0.685	0.438	0.042
5	0.599	0.429	0.070
10	0.562	0.455	0.092

Table 14: Operational cost of producing a forecast on one A100-80GB at batch size one in bfloat16. Output interfaces differ, so these rows are cost references rather than matched-output comparisons. gWorld uses its official HTML-generation recipe and excludes the downstream VLM judge.

Forecast path	Output	Latency	TFLOPs
LGWM one step	latent	12.39 ms	0.311
LGWM ten steps	latent	58.31 ms	—
Qwen3-VL-8B	text	7.44 s	7.82
gWorld-8B	HTML	113.2 s	74.1

E Design Ablations

Table 2 in the main text summarises three key ablation rows; this appendix reports the full set of configurations with all world-model diagnostics and downstream evaluation metrics.

Table 15: Full design ablations at ViT-S scale (15k updates, one seed). Each row changes one factor from the **control** configuration. Cos@k: cosine agreement between the k-step recursive forecast and the true encoding (higher is better). |Copy|: absolute deviation from identity-copy behaviour. RSWT and Harm AUC are the downstream evaluation metrics from Section 4.

Configuration	R@1	R@5	Cos@3	Cos@10	|Copy|	Eff. rank	Overall AUC	Harm AUC
Control	**0.190**	**0.490**	**0.700**	**0.612**	**0.286**	**43.7**	**0.878**	**0.527**
Coordinate MLP	0.193	0.496	0.703	0.621	0.300	43.2	0.881	0.541
No inverse dynamics	0.231	0.572	0.765	0.709	0.193	46.4	0.900	0.544
No rollout training	0.199	0.487	0.660	0.553	0.302	44.6	0.869	0.496
Change weight 1	0.194	0.490	0.703	0.618	0.281	42.7	0.873	0.520
Change weight 32	0.184	0.460	0.660	0.571	0.298	46.4	0.868	0.507
No VICReg	0.179	0.472	0.706	0.663	0.299	22.5	0.857	0.513

The ablations isolate the contribution of each design factor and complement the summary in Table 2. Rollout training provides the largest effect on multi-step forecast quality: removing it drops Cos@10 from 0.612 to 0.553 (-10%) and reduces downstream Overall AUC from 0.878 to 0.869. Removing VICReg collapses representation diversity from effective rank 43.7 to 22.5 (-49%) and lowers Overall AUC from 0.878 to 0.857. Removing inverse dynamics improves retrieval (R@1 rises from 0.190 to 0.231) but halves |Copy| from 0.244 to 0.112, indicating increased reliance on identity-copy behaviour. Change weighting trades retrieval against recursion: both weight 1 and weight 32 reduce downstream RSWT relative to the default weight 8. All effects are directional: each configuration uses one seed at a smaller scale than the main model.

Cross-seed variance. Table 16 replicates the control configuration across three seeds. All metrics vary by less than one standard deviation of their cross-source spread, confirming that the single-seed ablation conclusions are not fragile.

Table 16: Three-seed replication of the control configuration (ViT-S, 15k steps). Std is computed across the three seeds.

Metric	seed 2027	seed 42	seed 123	std
R@1	0.190	0.188	0.193	0.003
Copy score	−0.286	−0.307	−0.316	0.015
Cos@3	0.700	0.682	0.695	0.009
Eff. rank	43.7	44.8	44.7	0.6
Overall AUC	0.878	0.866	0.872	0.006
Harm AUC	0.527	0.509	0.517	0.009

F Auxiliary Robustness and Fairness Controls

F.1 Harm interpretation across model families

The primary prompt asks whether an observed outcome is consistent with the preceding transition. It does not explicitly train or prompt external models to separate harmful from benign inconsistencies. With that qualification, the held-out scores provide a useful capability decomposition: off-the-shelf prompted models identify transition violations but remain near chance when both alternatives are violations and the remaining question is which one is harmful. The supervised residual readout supplies that missing interpretation (Table 17). These rows are therefore diagnostic controls, not a task-matched harm leaderboard.

Table 17: Harm-interpretation diagnostics. External models share a prompt about transition consistency, not a task-matched harm prompt; their Harm AUC and macro FPR at at least 90% hijack recall are therefore diagnostic controls rather than a harm leaderboard.

Score	Harm AUC ↑	mFPR@90R ↓
Gemini 3.7 Flash	0.585	0.225
GPT-5.6 Luna	0.575	0.233
Qwen3.7 Plus	0.512	0.253
Qwen3-VL-8B	0.608	0.387
SAWM	0.614	0.375
LGWM integrity	0.513	0.372
LGWM residual	**0.953**	**0.163**

F.2 Batching and second-stage overhead

The batch-size-one comparison in Table 1 models an online monitor, where transitions arrive sequentially. To give the VLM baseline the most favorable batching condition, we additionally benchmark Qwen3-VL-8B at batch size eight and divide whole-batch latency by eight. Its amortized latency falls from 1097.8 to 186.8 ms per transition, a 5.9× throughput gain, but remains 10.9× the 17.1 ms latency of the unbatched LGWM integrity score. FLOPs remain 10.924 versus 0.408 TFLOPs per decision.

The supervised harm readout adds no measurable systems burden: integrity and harm decisions take 17.1 and 16.7 ms, respectively, and both measure 0.408 TFLOPs and 1.09 GB peak memory. The residual classifier is a single 1536-dimensional linear readout, so the second semantic stage is effectively free relative to the shared encoder and predictor forward pass.

For the generative baselines, the reported 92.9–124.3 s covers generation only and excludes the cross-rendering VLM judge required for a final decision. MobileWorld-HTML also reaches its 4096-token generation cap, so its 124.3 s measurement is a lower bound. These qualifications favor the generative baselines and do not account for the full cost of converting their rendered output into a transition score.

G TIER-STRATIFIED RSWT RESULTS

A natural concern is whether LGWM's aggregate RSWT score is driven by the easier donor pairs. Table 18 breaks the result down by difficulty tier, where Hard means the hijacked screen shares an application bucket with the legitimate transition and Medium and Easy denote progressively less related applications (Appendix B).

Table 18: RSWT AUC by difficulty tier. Hard denotes a shared application bucket, not verified execution within the same application workflow. Sample sizes count donor pairs.

Method	Type	Hard (n=46)	Medium (n=93)	Easy (n=83)
Gemini 3.7 Flash	closed VLM	0.993	0.997	0.997
LGWM integrity	**training-free WM**	**0.990**	**0.985**	**0.986**
GPT-5.6 Luna	closed VLM	0.964	0.997	0.993
Qwen3.7 Flash	closed VLM	0.957	0.969	0.979
Qwen3.7 Plus	closed VLM	0.936	0.980	0.992
Qwen3-VL-Plus	closed VLM	0.920	0.913	0.925
Qwen3-VL-4B	open VLM, 4B	0.913	0.902	0.907
GLM-4.6V-Flash	open VLM, 9B	0.909	0.930	0.889
LGWM residual	supervised WM	0.901	0.959	0.917
Qwen3-VL-8B	open VLM, 8B	0.898	0.913	0.910
gWorld-8B	generative WM	0.897	0.855	0.916
SAWM	safety WM, 8B	0.866	0.874	0.904
Code2World-7B	generative WM	0.856	0.863	0.883
Claude Haiku 4.5	closed VLM	0.829	0.903	0.883
Qwen3-VL-Flash	open VLM	0.756	0.729	0.759
Qwen3-VL-2B	open VLM, 2B	0.677	0.669	0.716
MobileWorld-HTML-8B	generative WM	0.634	0.604	0.526
Qwen2.5-VL-7B	open VLM, 7B	0.531	0.652	0.508

LGWM's values are stable across all three tiers (0.985–0.990), confirming that the aggregate score is not driven by the easier pairs. The per-tier samples are small, so the individual point estimates do not support cross-system equivalence claims.

H LIMITATIONS

Scale of evidence. The design ablations use one seed at ViT-S scale (a three-seed replication of the control confirms low variance; Table 16), and the main model is a single training run at ViT-B scale. Full-scale replication with multiple seeds remains future work.

Benchmark scope. RSWT-BENCH draws on two Android-centric sources and tests controlled screen replacement with credential screens. The hard tier is defined by shared application bucket rather than verified same-application execution (Appendix B). We have not measured performance on web or desktop interfaces, on non-credential attack types, or in live adaptive settings where an adversary can observe and react to the monitor.

Deployment considerations. An integrity monitor produces false positives on benign but unusual transitions such as push notifications or server-side layout changes. The residual head is one mitigation, and any deployment should define explicit fallback behavior rather than treating the score as a verdict. Quantifying how much harder forgery becomes under an expectation-based monitor is an open question.