WORLD MODEL-INDUCED DUAL-VIEW INCONSISTENCY FOR GUI AGENT TRAJECTORY EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Benefiting from large-scale, high-quality interaction trajectories, GUI agents have achieved substantial progress in perception, planning, and interaction. As trajectory synthesis and automated collection scale up, data quantity is becoming less of a bottleneck, while trajectory quality is increasingly important. However, large-scale trajectory collections often contain substantial redundancy and quality variation, making reliable supervision difficult to identify. Trajectory evaluation is further complicated by rich interface and state information and unexpected factors during execution. Existing filtering methods mainly assess realized state transitions and cannot effectively distinguish action quality from deviations in execution outcomes, potentially discarding valuable supervision. Since identifying whether a realized transition matches the expected outcome requires predicting future states, we introduce WM-IncPRM, a world-model-guided inconsistency-aware process reward framework for GUI trajectory evaluation. WM-IncPRM combines a world model with PRM to predict future semantic transitions and compare them with realized transitions. Reliable and supported prediction–reality discrepancies are used to calibrate step-level quality, which is further aggregated with recoverable supervision coverage to classify trajectories into Keep, Repair, or Reject. Experiments demonstrate that WM-IncPRM improves trajectory filtering quality and downstream performance on AndroidWorld. We will release the source code on GitHub in the future.

1 INTRODUCTION

The rapid development of GUI agents has increased the demand for reliable interaction data. As mobile agents operate across diverse tasks and interfaces, their trajectories exhibit substantial variation in states, actions, and execution outcomes, making it difficult to identify useful supervision and filter noisy data.

A key difficulty is that an undesirable realized outcome does not always imply an incorrect action. Execution failures or environmental disturbances may prevent an appropriate action from producing its expected transition. A PRM that evaluates the realized interaction may therefore underestimate the action quality, causing conventional filtering methods to discard useful supervision. The central challenge is to distinguish incorrect actions from appropriate actions whose effects are not properly realized.

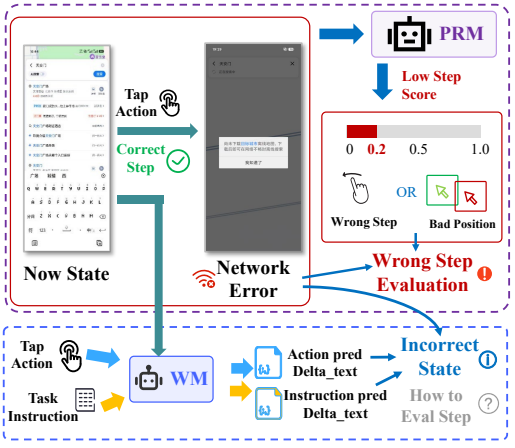

Figure 1: Outcome-only PRMs cannot separate action quality from environmental failures; future-transition prediction adds complementary signals.

Outcome-based evaluation cannot directly address this issue because it only observes the realized state. While PRMs provide fine-grained evaluation, they lack a reference for what should happen after an action. World models provide such a reference by predicting future semantic state transitions

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.

conditioned on the task instruction and action. We therefore jointly consider instruction-conditioned and action-conditioned predictions with the realized transition. When these estimates are substantially inconsistent and the realized quality is lower than the predictive quality, the discrepancy provides evidence of underestimated action quality.

However, world-model predictions can also be unreliable due to prediction errors, representation mismatches, and out-of-distribution interfaces. We therefore introduce empirical Support from real mobile transitions and World Model Reliability based on prediction consistency. Only sufficiently supported and reliable predictions are used for selective quality recovery.

Motivated by this observation, we propose WM-IncPRM, a world-model-guided inconsistency-aware process reward framework for mobile GUI trajectory evaluation and filtering. It evaluates instruction-conditioned predictions, action-conditioned predictions, and realized transitions, then uses support-aware and reliability-aware gating to recover underestimated step-level quality. We further aggregate recovered step quality and recoverable step coverage to obtain trajectory-level scores and assign trajectories to Keep, Repair, or Reject.

We evaluate filtering methods on OpenMobile under the same data scale and training configuration, and further validate filtering quality on a 500-trajectory human-annotated validation set. Finally, we evaluate downstream performance on AndroidWorld. Results show that WM-IncPRM improves filtered dataset quality and downstream mobile-agent training performance. The human-annotated dataset will be publicly released in the future.

Our contributions are summarized as follows:

- We identify a critical limitation of PRM-based mobile trajectory filtering: a correct action may receive a low realized score when the environment fails to realize its expected consequence, causing useful supervision to be discarded.
- We propose WM-IncPRM, a world-model-guided framework that uses prediction–reality inconsistency as a forward-looking signal and selectively recovers underestimated PRM scores according to empirical Support and World Model Reliability.
- We conduct extensive experiments to verify the effectiveness of WM-IncPRM, demonstrating improved trajectory dataset quality and strong performance on a human-annotated dataset, together with better downstream training performance on AndroidWorld.

2 Related Work

2.1 GUI Agents

Recent multimodal GUI agents enable autonomous interaction with graphical interfaces through visual perception, language understanding, task planning, and tool execution. Mobile-Agent combines visual perception, task planning, and tool execution for autonomous smartphone interaction Wang et al. (2024). Mobile-Agent-v3 further strengthens perception and planning for general GUI interaction Ye et al. (2025), while AppAgent explores visual understanding and action grounding for autonomous app interaction Zhang et al. (2025). Ferret-UI focuses on fine-grained GUI understanding through referring, grounding, and reasoning You et al. (2024). SeeAct extends multimodal GUI interaction to real-world web navigation, demonstrating the applicability of GUI agents beyond mobile environments Zheng et al. (2024).

Large-scale datasets and interactive environments have also become important for developing and evaluating GUI agents. Android in the Wild provides human demonstrations across diverse tasks, applications, devices, and touch actions Rawles et al. (2023), while AndroidWorld offers a reproducible environment for standardized GUI-agent training and evaluation Rawles et al. (2025). OS-Genesis further automates GUI trajectory construction through reverse task synthesis and evaluates trajectory quality with a graded Trajectory Reward Model, showing that useful supervision can remain in imperfect trajectories Sun et al. (2025).

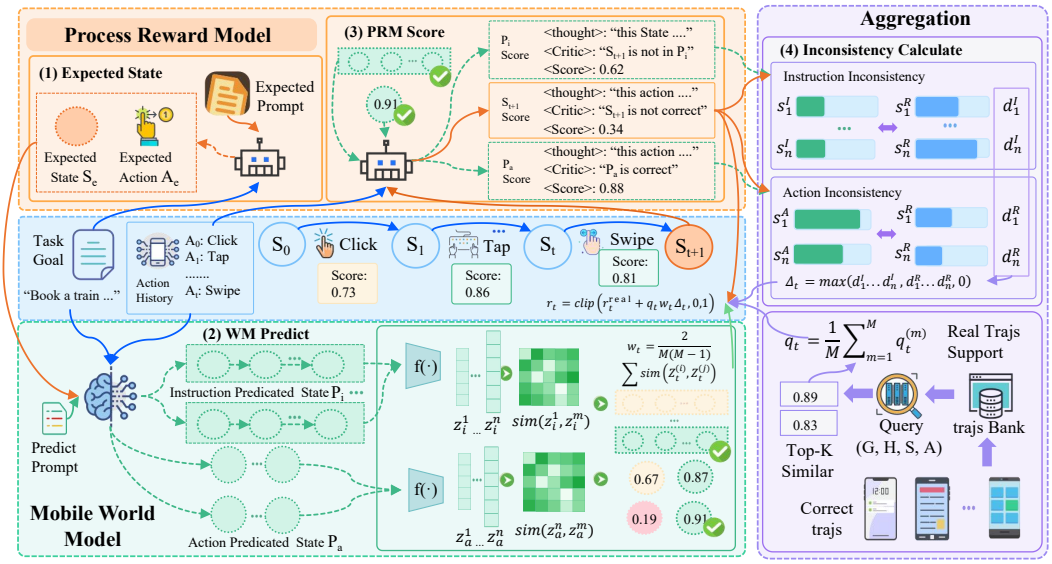

Figure 2: Overview of the proposed world-model–PRM collaborative evaluation framework. Given the current state, task instruction, and executed action, the world model predicts action-conditioned and instruction-conditioned semantic transitions.

2.2 Process Reward Models

Process supervision evaluates intermediate steps rather than only final outcomes, providing fine-grained signals for multi-step tasks Lightman et al. (2024). Chen et al. develop an uncertainty-aware self-trained trajectory evaluator and show that evaluator-based filtering improves downstream GUI-agent training Chen et al. (2025b). GUI-Shepherd provides dense step-level rewards for long-horizon GUI tasks Chen et al. (2025a), while CUARewardBench systematically evaluates outcome and process reward models on computer-use trajectories Lin et al. (2025).

These studies establish the value of process-level evaluation, but realized-outcome scoring cannot distinguish incorrect actions from correct actions affected by environmental failures. WM-IncPRM addresses this gap with an action-conditioned predictive view.

2.3 Mobile World Models

World models predict future states conditioned on current states and actions and have been explored for GUI-agent planning, simulation, and training. Recent work studies different future-GUI representations, including delta text, full text, diffusion images, and code-to-image, together with their roles in perception, criticism, self-reflection, and imagination-based training Xu et al. (2026). ViMo investigates visual world modeling for interactive decision making Luo et al. (2026), while Code2World/GUI World Model explores code-based generation of future GUI states Zheng et al. (2026).

World models have also been integrated into planning. AndroidWMSearch uses a world model to simulate Android transitions within tree search Jia et al. (2025), and gWorld explores code-based visual world modeling for future-state generation and reasoning Koh et al. (2026). These studies demonstrate the value of forward-looking state prediction, but predicted transitions may be unreliable under model uncertainty, representation mismatch, or out-of-distribution interfaces. WM-IncPRM therefore uses action-conditioned predictions as references rather than independent evaluators, and employs Support and Reliability to assess the resulting prediction–reality inconsistency.

3 METHOD

3.1 PROBLEM FORMULATION

We consider offline mobile GUI trajectories of the form

$$\tau = \{(G, H_t, S_t, a_t, S_{t+1})\}_{t=1}^{T}, \tag{1}$$

where G denotes the task instruction, H_t denotes the interaction history up to step t, S_t is the current GUI state, a_t is the executed action, and S_{t+1} is the realized next state.

Our goal is to identify useful supervision from failed trajectories. A trajectory-level failure does not necessarily imply that all executed actions are incorrect: an agent may select an appropriate action while the environment fails to produce the expected transition. However, a PRM that evaluates only the realized transition can assign a low score to such a step because the observed state shows little or no task progress. This conflates action quality with environmental realization and may incorrectly discard otherwise useful supervision.

For each step, we construct the pre-action context

$$c_t = (G, H_t, S_t, a_t). \tag{2}$$

The realized next state S_{t+1} is withheld during future prediction. Thus, all predictive signals are generated before observing the actual outcome, while the realized transition is introduced only for subsequent comparison and quality recovery.

3.2 DUAL-VIEW FUTURE PREDICTION

For each step, a frozen PRM first provides task-related guidance:

$$M_t^{P\to W} = PRM_{\text{guide}}(G, H_t, S_t, a_t), \tag{3}$$

which summarizes information such as the current subgoal, task progress, and task-relevant interface elements. This guidance is provided to the frozen world model and constitutes the first information exchange, $PRM \to WM$.

The world model then predicts two complementary future views. The instruction-conditioned branch predicts the task-oriented expected future:

$$E_t^{(m)} \sim p_\theta\left(E \mid G, H_t, S_t, M_t^{P\to W}\right), \qquad m = 1, \ldots, M, \tag{4}$$

while the action-conditioned branch predicts the consequence of the executed action:

$$P_t^{(m)} \sim p_\theta\left(P \mid G, H_t, S_t, a_t, M_t^{P\to W}\right). \tag{5}$$

The instruction-conditioned predictions describe where the task should progress, whereas the action-conditioned predictions estimate what should happen if the selected action is successfully realized. Multiple samples are retained to estimate the stability of the predicted outcome.

3.3 PREDICTED AND REALIZED PROGRESS

After the actual transition is available, we extract its task-relevant semantic representation:

$$R_t = \Phi(S_t, a_t, S_{t+1}). \tag{6}$$

Using the same frozen PRM, we score the predicted and realized transitions:

$$r_t^{\text{exp}} = PRM(G, H_t, S_t, E_t), \tag{7}$$

$$r_t^{\text{pred}} = PRM(G, H_t, S_t, a_t, P_t), \tag{8}$$

$$r_t^{\text{real}} = PRM(G, H_t, S_t, a_t, R_t). \tag{9}$$

Here, r_t^{pred} measures the quality of the selected action under its predicted consequence, whereas r_t^{real} measures the quality observed after actual execution. Therefore, the gap between them reveals potential cases in which an otherwise appropriate action is undervalued because its intended consequence was not realized.

3.4 QUALITY RECOVERY

We define the quality discrepancy among the predicted and realized transitions as

$$\Delta_t = \max\left(r_t^{\text{exp}}, r_t^{\text{pred}}, r_t^{\text{real}}\right) - \min\left(r_t^{\text{exp}}, r_t^{\text{pred}}, r_t^{\text{real}}\right). \tag{10}$$

We consider the discrepancy recoverable only when

$$r_t^{\text{real}} < \max\left(r_t^{\text{exp}}, r_t^{\text{pred}}\right). \tag{11}$$

A large Δ_t indicates that the action was expected to produce substantially more progress than was observed. Importantly, this gap is not directly interpreted as a prediction error. When the selected action is appropriate but the environment fails to realize the expected transition, r_t^{pred} remains high while r_t^{real} becomes low, creating exactly the signal required to recover the underestimated action quality.

To avoid trusting arbitrary world-model predictions, we introduce two independent confidence signals.

First, an external real-transition bank $\mathcal{B}_{\text{real}}$ is used to assess whether the predicted consequences are supported by previously observed mobile interactions. The resulting Support score is defined as

$$q_t = \frac{1}{M}\sum_{m=1}^{M} \text{Support}\left(P_t^{(m)}, \mathcal{B}_{\text{real}}\right), \tag{12}$$

where $\text{Support}(\cdot, \mathcal{B}_{\text{real}})$ measures the empirical support of a predicted consequence against the real-transition bank. The detailed construction and computation of the Support score are provided in Appendix B.2.

Second, to measure the stability of the World Model predictions, we perform M samples with the same input and compute the pairwise semantic similarity among the resulting predictions. The World Model Reliability is then defined as

$$w_t = \frac{2}{M(M-1)} \sum_{1 \le i < j \le M} \text{Sim}_{\text{sem}}\left(P_t^{(i)}, P_t^{(j)}\right), \tag{13}$$

where $P_t^{(i)}$ denotes the predicted future state after the current action generated by the World Model in the i-th sampling, and $\text{Sim}_{\text{sem}}(\cdot,\cdot) \in [0,1]$ denotes the semantic similarity between two predictions. A higher w_t indicates greater consistency among the sampled predictions, implying that the World Model provides a more stable prediction for the corresponding state transition.

The support and reliability signals jointly determine whether the predicted action quality can be trusted. We therefore recover the PRM score as

$$\tilde{r}_t = \text{clip}\left(r_t^{\text{real}} + q_t w_t \Delta_t, 0, 1\right). \tag{14}$$

When the realized outcome is poor but the selected action has a highly supported and reliable predicted outcome, WM-IncPRM increases the score toward the predicted action quality. Conversely, an incorrect action usually produces a low r_t^{pred}, resulting in a small recovery gap and preventing erroneous rescue.

The resulting information flow realizes the second information exchange, $WM \rightarrow PRM$, because the PRM jointly considers predicted and realized consequences when estimating the step quality.

3.5 TRAJECTORY-LEVEL QUALITY AND FILTERING

The purpose of trajectory aggregation is to estimate how much reliable action-level supervision a trajectory contains, rather than simply whether the final task succeeds.

We first compute the mean recovered step quality:

$$R_{\text{mean}}(\tau) = \frac{1}{T}\sum_{t=1}^{T} \tilde{r}_t. \tag{15}$$

We additionally measure the coverage of recoverable high-quality steps:

$$C_{\mathrm{rec}}(\tau) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\left[r_t^{\mathrm{pred}} \geq \tau_a \wedge q_t w_t \geq \tau_c \wedge \Delta_t > 0\right], \tag{16}$$

where τ_a and τ_c are fixed thresholds selected on validation data. C_{rec} measures the fraction of steps for which the trajectory provides evidence of a reliable action whose realized outcome was potentially disrupted.

The final trajectory score is defined as

$$R(\tau) = (1-\beta)R_{\mathrm{mean}}(\tau) + \beta C_{\mathrm{rec}}(\tau), \tag{17}$$

where $\beta \in [0,1]$ balances the overall step quality and the coverage of recoverable high-quality steps.

Given two trajectory-level thresholds $\tau_{\mathrm{repair}} < \tau_{\mathrm{keep}}$, each trajectory is assigned to one of three outcomes:

$$\mathcal{D}(\tau) = \begin{cases} \mathrm{Keep}, & R(\tau) \geq \tau_{\mathrm{keep}}, \\ \mathrm{Repair}, & \tau_{\mathrm{repair}} \leq R(\tau) < \tau_{\mathrm{keep}}, \\ \mathrm{Reject}, & R(\tau) < \tau_{\mathrm{repair}}. \end{cases} \tag{18}$$

In our experiments, the recovery thresholds are set to $\tau_\Delta = 0.4$, $\tau_q = 0.78$, and $\tau_w = 0.82$ for discrepancy, Support, and World Model Reliability, respectively. The trajectory-level thresholds are set to $\tau_{\mathrm{keep}} = 0.73$ and $\tau_{\mathrm{repair}} = 0.52$. These threshold values are determined based on the experimental results on the validation set and fixed for all subsequent evaluations. The algorithmic details are provided in Appendix B.

4 Experiments

4.1 Experimental Setup

Datasets and Benchmark. *OpenMobile* Cheng et al. (2026) is used for filtering and training, while *AndroidWorld* Rawles et al. (2025) provides 116 evaluation tasks. **Process Reward Model.** A frozen UI-Genie-RM PRM Xiao et al. (2025) provides step-level progress scores for PRM-based methods. **World Model.** MobileWorld-Delta-TEXT-8B Xu et al. (2026) predicts future state changes from pre-execution information. **Training and Annotation.** Qwen3.5-4B Qwen Team (2026) is trained on equally sized subsets with five random seeds. Five domain experts annotate environmental errors in the validation set. **Evaluation Metrics.** Success Rate and Avg. Steps measure task performance and efficiency. AOC measures trajectory ranking quality, while EEPR measures preservation of human-labeled environmental-error trajectories; higher values indicate better performance.

4.2 Baselines

We compare WM-IncPRM with six baselines: *Ordered Raw Data*, *Random*, *AgentRM* Xia et al. (2025), *AgentTrek-VLM* Xu et al. (2025), *GUI-Critic-R1* Wanyan et al. (2025), and *URST* Chen et al. (2025b), covering different trajectory evaluation and selection paradigms.

All baselines retain the same number of trajectories as WM-IncPRM and use the same downstream Qwen3.5-4B training configuration, isolating the effect of trajectory selection quality.

4.3 Main Results

Table 1 summarizes the *Keep*, *Repair*, and *Reject* subsets. On 500 human-labeled trajectories, WM-IncPRM achieves the highest AOC for all three categories: 56.7%, 59.1%, and 50.3%, respectively, indicating better trajectory quality ranking.

Table 2 shows downstream results on the *Keep* subset under a 20% retention budget. WM-IncPRM achieves $26.0 \pm 1.1\%$ success, outperforming Ordered Raw Data ($17.8 \pm 1.0\%$), Random ($10.7 \pm$

Table 1: OpenMobile filtering results and AOC performance on 500 human-labeled trajectories. For AgentTrek-VLM and URST, the original *Accepted* and *Rejected* outputs are mapped to *Keep* and *Reject*, respectively. Higher AOC indicates better ranking quality.

	OpenMobile Filtering			Human-Labeled AOC		
Method	Keep	Repair	Reject	Keep AOC ↑	Repair AOC ↑	Reject AOC ↑
URST	2237	–	1647	40.2%	–	41.3%
AgentTrek-VLM	3203	–	681	52.4%	–	42.5%
AgentRM	999	2308	577	38.9%	46.7%	44.6%
GUI-Critic-R1	835	2325	724	41.4%	50.9%	45.7%
WM-IncPRM (Ours)	**823**	**2075**	**986**	**56.7%**	**59.1%**	**50.3%**

Table 2: Downstream performance on AndroidWorld using filtered Keep trajectories. Higher Success and Success Rate indicate better Keep data quality, while lower Failed indicates better performance. Results are mean ± standard deviation over 5 random seeds.

Method	Data Percent	Success ↑	Failed ↓	Avg. Steps ↓	Success Rate (%) ↑
Ordered Raw Data	20%	20.6 ± 1.2	95.4 ± 1.2	14.1 ± 0.9	17.8 ± 1.0
Ordered Raw Data	50%	12.4 ± 1.1	103.6 ± 1.1	14.6 ± 0.6	10.7 ± 1.0
Random	20%	12.4 ± 1.3	103.6 ± 1.3	14.5 ± 0.8	10.7 ± 1.0
Random	50%	13.4 ± 1.1	102.6 ± 1.1	13.4 ± 1.1	11.6 ± 1.0
Random	100%	18.8 ± 1.3	97.2 ± 1.3	14.3 ± 0.9	16.2 ± 1.1
URST	20%	22.8 ± 1.8	93.2 ± 1.8	13.2 ± 0.6	19.7 ± 1.5
AgentTrek-VLM	20%	28.0 ± 0.7	88.0 ± 0.7	14.9 ± 0.7	24.1 ± 0.6
AgentRM	20%	19.8 ± 1.6	96.2 ± 1.6	14.6 ± 0.9	17.1 ± 1.4
GUI-Critic-R1	20%	22.4 ± 1.9	93.6 ± 1.9	14.1 ± 0.7	19.3 ± 1.7
WM-IncPRM (Ours)	20%	$\mathbf{30.2 \pm 1.3}$	$\mathbf{85.8 \pm 1.3}$	$\mathbf{13.1 \pm 0.8}$	$\mathbf{26.0 \pm 1.1}$

1.0%), URST (19.7±1.5%), AgentRM (17.1±1.4%), GUI-Critic-R1 (19.3±1.7%), and AgentTrek-VLM ($24.1 \pm 0.6\%$). Compared with Random, it improves success by 15.3 percentage points and reduces average steps from 14.5±0.8 to 13.1±0.8. Table 3 shows that WM-IncPRM Reject achieves $12.4 \pm 1.0\%$ success, lower than all compared baselines, indicating better isolation of lower-quality trajectories. The 13.6-point gap between Keep and Reject further demonstrates distinct downstream supervision quality. Table 6 shows that WM-IncPRM achieves the highest overall EEPR of 78.9%, compared with AgentTrek-VLM (70.5%) and GUI-Critic-R1 (53.1%). It also achieves the highest or tied highest EEPR across all five error types, reaching 86.9% for Network Error and 82.4% for Target Missing.

To evaluate the ranking quality of WM-IncPRM beyond a single retention budget, we further vary the retention rate from 10% to 100% and retrain the downstream model using the top-ranked Keep trajectories at each operating point. The corresponding AndroidWorld results are reported in Appendix E.1.

4.4 Ablation Studies

We compare four variants, PRM-only, WM-A+I, WM-IncPRM w/o Reliability, and WM-IncPRM w/o Support, with the complete WM-IncPRM framework. PRM-only uses realized-transition evaluation, while WM-A+I uses instruction- and action-conditioned World Model predictions. All variants are evaluated under the same downstream configuration with five random seeds.

For the Keep subset in Table 4, the success rates are $22.2 \pm 1.3\%$, $16.2 \pm 0.7\%$, $19.3 \pm 1.0\%$, $21.4 \pm 0.7\%$, and $26.0 \pm 1.1\%$, respectively. WM-IncPRM achieves the highest success rate and the lowest average steps (13.1 ± 0.8), compared with 13.7 ± 1.0, 16.2 ± 0.9, 14.2 ± 0.7, and 14.1 ± 0.9 for the other variants.

For the Reject subset in Table 5, the corresponding success rates are $14.5 \pm 1.1\%$, $17.2 \pm 1.2\%$, $15.3 \pm 0.7\%$, $15.3 \pm 0.7\%$, and $12.4 \pm 1.0\%$. WM-IncPRM achieves the lowest success rate, with 14.4 ± 1.1 successful tasks and 101.6 ± 1.1 failed tasks, indicating more effective separation of low-quality trajectories.

Table 3: Downstream performance on AndroidWorld using filtered Reject trajectories. Each Reject subset contains the same number of trajectories and is trained separately. Lower Success and Success Rate indicate better rejection quality, as they imply that the selected Reject trajectories are of lower quality. Results are mean ± standard deviation over 5 random seeds.

Method	Data Percent	Success ↓	Failed ↑	Avg. Steps ↑	Success Rate (%) ↓
URST	20%	18.4 ± 1.1	97.6 ± 1.1	14.3 ± 0.9	15.9 ± 1.0
AgentTrek-VLM	20%	18.6 ± 2.3	97.4 ± 2.3	15.1 ± 1.1	16.0 ± 2.0
AgentRM	20%	19.6 ± 1.7	96.4 ± 1.7	15.0 ± 0.9	16.9 ± 1.4
GUI-Critic-R1	20%	20.6 ± 1.5	95.4 ± 1.5	15.2 ± 0.7	17.8 ± 1.3
WM-IncPRM (Ours)	20%	$\mathbf{14.4 \pm 1.1}$	$\mathbf{101.6 \pm 1.1}$	$\mathbf{15.5 \pm 0.9}$	$\mathbf{12.4 \pm 1.0}$

Table 4: Ablation results on AndroidWorld using filtered Keep trajectories. All variants use equal-sized 20% subsets. Higher Success and Success Rate indicate better Keep data quality, while lower Failed indicates better performance. Results are mean ± standard deviation over 5 random seeds.

Variant	Success ↑	Failed ↓	Avg. Steps ↓	Success Rate (%) ↑
PRM-only	25.8 ± 1.5	90.2 ± 1.5	13.7 ± 1.0	22.2 ± 1.3
WM-A+I	18.8 ± 0.8	97.2 ± 0.8	16.2 ± 0.9	16.2 ± 0.7
WM-IncPRM w/o Reliability	22.4 ± 1.1	93.6 ± 1.1	14.2 ± 0.7	19.3 ± 1.0
WM-IncPRM w/o Support	24.8 ± 0.8	91.2 ± 0.8	14.1 ± 0.9	21.4 ± 0.7
WM-IncPRM (Ours)	$\mathbf{30.2 \pm 1.3}$	$\mathbf{85.8 \pm 1.3}$	$\mathbf{13.1 \pm 0.8}$	$\mathbf{26.0 \pm 1.1}$

WM-IncPRM denotes the complete framework with PRM guidance, instruction- and action-conditioned prediction, Prediction–Reality comparison, empirical support, world-model reliability, and support-aware calibration.

5 Analysis

5.1 Data Quality vs. Data Quantity

Compared with the baseline methods in Table 1, WM-IncPRM also achieves the best ranking quality on the human-labeled trajectory set, with the highest AOC across Keep, Repair, and Reject categories.

Table 2 shows that both trajectory quality and data quantity affect downstream performance, while the quality distribution of the raw corpus is not uniform. At the same 20% data ratio, Ordered Raw Data achieves a success rate of 17.8%, substantially higher than Random at 10.7%, suggesting that the original trajectory order contains different-quality regions rather than uniformly distributed supervision. When the data ratio increases to 50%, Ordered Raw Data and Random achieve 10.7% and 11.6% success, respectively, while Random reaches only 16.2% even with the full dataset. Thus, increasing the amount of randomly selected data does not consistently improve downstream performance.

Under the same 20% data budget, quality-aware filtering also shows clear differences from raw and random selection. URST, GUI-Critic-R1, AgentRM, and AgentTrek-VLM achieve success rates of 19.7%, 19.3%, 17.1%, and 24.1%, respectively, while WM-IncPRM Keep reaches 26.0%. Compared with Random at 20%, WM-IncPRM improves the success rate by 15.3%. Overall, these results indicate that, under a limited data budget, selecting higher-quality trajectories can be more effective than simply increasing the quantity of randomly sampled training data.

5.2 Baseline Analysis

Under the same 20% data budget, Table 2 shows that WM-IncPRM achieves the highest downstream success rate among all compared filtering methods. WM-IncPRM Keep reaches 26.0%, outperforming Ordered Raw Data (17.8%), Random (10.7%), URST (19.7%), AgentRM (17.1%), GUI-Critic-R1 (19.3%), and AgentTrek-VLM (24.1%). The improvement over AgentTrek-VLM is

Table 5: Ablation results on AndroidWorld using filtered Reject trajectories. All variants use equal-sized 20% subsets. Lower Success and Success Rate indicate better rejection quality, as they imply that the selected Reject trajectories are of lower quality. Results are mean ± standard deviation over 5 random seeds.

Variant	Success ↓	Failed ↑	Avg. Steps ↑	Success Rate (%) ↓
PRM-only	16.8 ± 1.3	99.2 ± 1.3	15.8 ± 0.7	14.5 ± 1.1
WM-A+I	20.0 ± 1.4	96.0 ± 1.4	14.6 ± 0.8	17.2 ± 1.2
WM-IncPRM w/o Reliability	17.2 ± 0.8	98.2 ± 0.8	15.6 ± 1.1	15.3 ± 0.7
WM-IncPRM w/o Support	17.8 ± 0.8	98.8 ± 0.8	15.4 ± 0.9	15.3 ± 0.7
WM-IncPRM (Ours)	$\mathbf{14.4 \pm 1.1}$	$\mathbf{101.6 \pm 1.1}$	$\mathbf{15.5 \pm 0.9}$	$\mathbf{12.4 \pm 1.0}$

Table 6: Environmental Error Preservation Rate (EEPR) of each filtering method on human-labeled environmental-error trajectories from OpenMobile ($n = 3884$). Higher is better.

Error Type (n)	URST	AgentTrek-VLM	AgentRM	GUI-Critic-R1	**WM-IncPRM (Ours)**
App Freeze (n=32)	37.5%	50.0%	9.4%	34.4%	**62.5%**
Network Error (n=23)	34.8%	82.6%	21.7%	30.4%	**86.9%**
Blank / Black Screen (n=10)	70.0%	80.0%	40.0%	50.0%	**80%**
UI Interruption (n=124)	44.4%	68.5%	20.2%	54.8%	**71.8%**
Target Missing (n=353)	49.6%	72.0%	16.7%	55.8%	**82.4%**
Total (n=542)	**47.4%**	**70.5%**	**17.7%**	**53.1%**	**78.9%**

1.9 percentage points, while the gain over the equal-sized Random subset reaches 15.3 percentage points.

Compared with URST, AgentRM, and GUI-Critic-R1, WM-IncPRM improves success by 6.3%, 8.9%, and 6.7% , respectively. Its improvement over AgentTrek-VLM further shows that the action-conditioned future view can provide useful information beyond evaluation based only on the observed interaction sequence.

The Reject results in Table 3 provide a complementary comparison. WM-IncPRM Reject achieves a success rate of 12.4%, compared with 15.9%, 16.0%, 16.9%, and 17.8% for URST, AgentTrek-VLM, AgentRM, and GUI-Critic-R1, respectively. The resulting gap between WM-IncPRM Keep and Reject (26.0% vs. 12.4%) indicates a clearer separation in the quality of the selected subsets. Overall, WM-IncPRM identifies more useful training trajectories under the same data budget and yields stronger downstream performance.

5.3 Ablation Analysis

For the Keep subset in Table 4, WM-IncPRM achieves 26.0% success, compared with 19.3% and 21.4% for WM-IncPRM w/o Reliability and w/o Support, corresponding to drops of 6.7 and 4.6 percentage points. This shows that Prediction–Reality comparison is essential, while Support further strengthens the predictive view.

WM-IncPRM also outperforms WM-A+I (16.2%) and PRM-only (22.2%), confirming the importance of integrating World Model predictions with realized transitions. For the Reject subset in Table 5, WM-IncPRM achieves the lowest success rate (12.4%), compared with 14.5%, 17.2%, 15.3%, and 15.3% for the four variants. Table 6 further shows improved preservation of environmental-error trajectories, with an 8.4 percentage-point EEPR gain over AgentTrek-VLM.

6 Conclusion

We present WM-IncPRM, a world-model-guided process reward framework for offline mobile GUI trajectory filtering. By modeling prediction–reality inconsistency and combining empirical Support and World Model Reliability, WM-IncPRM recovers underestimated PRM scores for correct actions affected by environmental failures and improves downstream mobile-agent training.

7 AI USE STATEMENT

In this work, we used generative AI tools to assist with the development and refinement of the research methodology, including discussing and refining the proposed framework; editing and improving the readability of the manuscript; writing and modifying experimental and implementation code; and creating and refining scientific figures. All AI-assisted outputs were reviewed, verified, and, where applicable, tested by the authors. The authors take full responsibility for the final content of this work, including the methodology, experimental design, implementation, results, figures, and written presentation.

REFERENCES

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks, 2025a. URL https://arxiv.org/abs/2509.23738.

Gongwei Chen, Lirong Jie, Lexiao Zou, Weili Guan, Miao Zhang, and Liqiang Nie. Enhancing GUI agent with uncertainty-aware self-trained evaluator. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=cuUsD5FJbe.

Kanzhi Cheng, Zehao Li, Zheng Ma, Nuo Chen, Jialin Cao, Qiushi Sun, Zichen Ding, Fangzhi Xu, Hang Yan, Jiajun Chen, Anh Tuan Luu, Jianbing Zhang, Lewei Lu, and Dahua Lin. Openmobile: Building open mobile agents with task and trajectory synthesis, 2026. URL https://arxiv.org/abs/2604.15093.

Xianqing Jia, Li Zhang, and Mengwei Xu. Androidwmsearch: Mobile agents tree search with world model. In *2025 IEEE 31th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 1–8, 2025. doi: 10.1109/ICPADS67057.2025.11323094.

Woosung Koh, Sungjun Han, Segyu Lee, Se-Young Yun, and Jamin Shin. Generative visual code mobile world models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZI6dgwPpoE.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, Ke Li, and Xing Sun. Cuarewardbench: A benchmark for evaluating reward models on computer-using agent, 2025. URL https://arxiv.org/abs/2510.18596.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mWoMyDEfbM.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728. Curran Associates, Inc., 2023. doi: 10.52202/075280-2609. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/bbbb6308b402fe909c39dd29950c32e0-Paper-Datasets_and_Benchmarks.pdf.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. OS-genesis: Automating GUI agent trajectory construction via reverse task synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5555–5579, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.277. URL https://aclanthology.org/2025.acl-long.277/.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=jE6pDYCnVF.

Yuyang Wanyan, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Jiabo Ye, Yutong Kou, Ming Yan, Fei Huang, Xiaoshan Yang, Weiming Dong, and Changsheng Xu. Look before you leap: A GUI-critic-r1 model for pre-operative error diagnosis in GUI automation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=theJ4Qiz55.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.945. URL https://aclanthology.org/2025.acl-long.945/.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents, 2025. URL https://arxiv.org/abs/2505.21496.

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?, 2026. URL https://arxiv.org/abs/2605.10347.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 79822–79843, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c681fb2bf1d785fbc766f3ea14758aab-Paper-Conference.pdf.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. Mobile-agent-v3: Fundamental agents for gui automation, 2025. URL https://arxiv.org/abs/2508.15144.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with nbsp;multimodal llms. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIV*, pp. 240–255, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73038-2. doi: 10.1007/978-3-031-73039-9_14. URL https://doi.org/10.1007/978-3-031-73039-9_14.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. AppAgent: Multimodal Agents as Smartphone Users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 70:1–70:20. ACM, 2025. doi: 10.1145/3706598.3713600. URL https://doi.org/10.1145/3706598.3713600.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024. URL https://arxiv.org/abs/2401.01614.

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, 2026. URL https://arxiv.org/abs/2602.09856.

A OpenMobile Dataset

A.1 Dataset Overview

OpenMobile Cheng et al. (2026) is an open-source framework for constructing mobile-agent training data through task and trajectory synthesis. The statistics of the trajectory splits used in our experiments are summarized in Table 7. Its task synthesis pipeline builds a global environment memory from exploration and uses this information to construct diverse and grounded mobile-task instructions. Its trajectory synthesis pipeline adopts a policy-switching strategy that alternates between learner and expert policies during rollout, allowing the collected trajectories to contain error-recovery behaviors that may be underrepresented in standard imitation-learning data.

In our experiments, OpenMobile is used as the offline trajectory corpus to be filtered before downstream mobile-agent training. We do not modify the original trajectory-generation procedure. Instead, the same OpenMobile corpus is provided to all trajectory filtering methods, including AgentRM, AgentTrek-VLM, GUI-Critic-R1, URST, and WM-IncPRM. This controlled setting isolates the effect of trajectory selection from differences in data-generation procedures.

A.2 Trajectory Organization

Each OpenMobile interaction trajectory is represented as a sequence of task-conditioned GUI interactions:

$$\tau = \{(G, H_t, S_t, a_t, S_{t+1})\}_{t=1}^{T}, \tag{19}$$

where G denotes the task instruction, H_t denotes the interaction history, S_t is the current GUI state, a_t is the executed action, and $St + 1$ is the realized next state.

For the original-data analysis, trajectories are first organized according to their native sequence order. We refer to this setting as *Sequence-Ordered Raw Data (Seq)*. Seq does not introduce any quality model or filtering operation; it is used only to characterize the intrinsic quality distribution of the original OpenMobile corpus and examine whether trajectories of different quality are unevenly distributed in the raw data.

In our experimental protocol, the OpenMobile corpus is consistently processed under the same split organization for all filtering methods. The filtering models receive the same candidate trajectories, and the downstream training stage uses the same retention budget. Thus, a difference in downstream performance reflects the quality of the selected trajectories rather than a difference in the amount of training data.

B Algorithm Details

B.1 Implementation and Information Boundaries

The overall filtering procedure follows Algorithm 1. During target-trajectory filtering, the PRM and World Model remain frozen, while the real-transition bank is used as an external resource for Support

Table 7: Statistics of the OpenMobile trajectory splits used for offline trajectory filtering and downstream training.

Split	Total	1–10	11–20	21–30	Single-App	Cross-App
Split 1	898	507	268	123	850	48
Split 2	962	531	323	108	894	68
Split 3	1066	307	378	381	1005	61
Split 4	958	324	382	252	914	44
Total	3884	1669	1351	864	3663	221

Algorithm 1 WM-IncPRM Trajectory Filtering

Require: Trajectory τ; frozen PRM F_ϕ; frozen WM W_θ; real-transition bank $\mathcal{B}_{\text{real}}$; sampling number M
Ensure: Filtering decision $\mathcal{D}(\tau)$
1: **for** each step $t = 1, \ldots, T$ **do**
2: Construct context and obtain PRM guidance.
3: Generate dual-view WM predictions with M samples.
4: Compute Support q_t and Reliability w_t.
5: Compare predicted and realized transitions with the PRM.
6: Recover the step score $\tilde{r}_t$.
7: **end for**
8: Compute trajectory score $R(\tau)$.
9: Assign KEEP, REPAIR, or REJECT according to $R(\tau)$.
10: For REPAIR, retain steps with $\tilde{r}_t \geq \tau_{\text{step}}$.
11: **return** $\mathcal{D}(\tau)$

estimation. For each step, all predictive components, including PRM guidance, dual-view WM prediction, Support estimation, and Reliability estimation, are computed before the realized next state S_{t+1} is introduced. The realized transition is used only for PRM evaluation and subsequent step-score recovery, preventing outcome leakage. After obtaining the recovered step scores, the trajectory is assigned to KEEP, REPAIR, or REJECT; for REPAIR, only steps satisfying the step-level threshold are retained. All filtering methods use the same OpenMobile candidate corpus and the same downstream training configuration. The selected trajectories are used to train Qwen3.5-4B and evaluated on AndroidWorld, with five random seeds and results reported as mean $\pm$ standard deviation.

B.2 SUPPORT COMPUTATION

The Support score evaluates whether a predicted consequence is supported by previously observed real mobile interactions. We construct an external real-transition bank

$$\mathcal{B}_{\text{real}} = \{(c_i, z_i)\}_{i=1}^{N}, \tag{20}$$

where c_i denotes the interaction context and $z_i = \Phi(S_i, a_i, S_{i+1})$ denotes the semantic representation of the realized state transition.

For each prediction $P_t^{(m)}$, we retrieve relevant real transitions from $\mathcal{B}_{\text{real}}$ according to the current context. We then compute the semantic similarity between the prediction and the retrieved real consequences:

$$q_t^{(m)} = \frac{1}{K} \sum_{z_i \in \mathcal{N}_t^{(m)}} \text{Sim}_{\text{sem}} \left(P_t^{(m)}, z_i \right), \tag{21}$$

where $\mathcal{N}_t^{(m)}$ contains the K most similar retrieved transitions. Finally, the step-level Support score is obtained by averaging over the M predictions:

$$q_t = \frac{1}{M} \sum_{m=1}^{M} q_t^{(m)}. \tag{22}$$

System Prompt

You are a mobile GUI task analyzer. Analyze the current step and describe what the task really cares about, so that a world model can predict the right state change.

User Prompt

Return STRICT JSON only, no explanation:

```
{
    "task_progress": "short sentence", "current_subgoal": "short sentence",
    "important_elements": ["short phrase", "..."], "expected_change": "short sentence"
}
```

Write every field in English, reusing the wording that appears on screen.

Task goal: {context.instruction}

Executed history: {_history_text(context)}

Current screen: {context.current_state.state_summary or '(unknown)'}

Current action: {_action_text(context)}

Figure 3: Prompt template used by the PRM to generate Expected Progress from the current task context and interaction.

C Prompt Templates

This appendix provides the prompt templates used in the WM-IncPRM trajectory evaluation pipeline. The prompts cover three key stages of the framework: world-model future-state prediction, PRM-based Expected Progress generation, and step-level trajectory quality scoring. Unless otherwise specified, the task instruction, interaction history, current GUI state, and action representation follow a consistent input format across the corresponding components. The exact prompt instances used in our experiments are shown in the following figures.

C.1 World-Model Prediction Prompt

Figure 4 presents the prompt used by the mobile world model to predict future semantic state changes. The prompt provides the model with the information available before action execution and is used to generate both instruction-conditioned and action-conditioned future predictions.

C.2 PRM Expected and Step-Scoring Prompts

The PRM is used in two complementary ways in our pipeline. First, the Expected prompt extracts the expected task-related progress from the current interaction context, providing task-oriented guidance for future prediction. Second, the five-level scoring prompt evaluates the quality of an individual trajectory step based on the corresponding interaction transition. The two prompt templates are shown in Figures 3 and 7, respectively.

D Case Study

In Figure 5, a network failure prevents the expected recipe webpage from being reached, resulting in a low realized PRM score. In contrast, both instruction- and action-conditioned World Model predictions indicate the expected transition to the “Shrimp Avocado Salad” recipe page. The re-

System Prompt

Return STRICT JSON only, no explanation, with exactly these fields: {

```
"change": "one sentence describing how the screen changes",
"new_elements": ["up to 5 short phrases"],
"removed_elements": ["up to 5 short phrases"],
"changed_elements": ["up to 5 short phrases"],
"subgoal_progress": "one sentence",
"side_effects": ["up to 3 short phrases"],
"confidence": 0.0
```

}

confidence is a number between 0 and 1 meaning how sure you are.
Write every field in English, and reuse the wording that appears on screen.

Instruction Prompt

You are a mobile GUI world model. Do NOT execute any concrete action now. Based only on the task goal and the current screen, predict what the screen should change into next if the task keeps moving in the right direction.
Task goal: {context.instruction}
ask guidance:{guidance.to_prompt()}
Current screen: {context.current_state.state_summary or '(unknown)'}
Describe the task-expected state change, not the consequence of any specific tap.

Action Prompt

You are a mobile GUI world model. Predict what the screen will change into immediately after the action below is executed.
Task goal: {context.instruction}
Task guidance:\n{guidance.to_prompt()}
Current screen: {context.current_state.state_summary or '(unknown)'}
Action to execute: {context.action.type} {context.action.description}
Describe only the direct consequence of this action.

Figure 4: Prompt template used for world-model future-state prediction, covering instruction-conditioned and action-conditioned prediction.

sulting inconsistency suggests an execution failure rather than insufficient task progress, allowing WM-IncPRM to recover the underestimated step score.

In Figure 6, the task requires opening the Calendar application, but the system UI becomes unresponsive, causing a low realized PRM score. Meanwhile, the World Model predicts the expected transition toward the Calendar application. This inconsistency indicates a system-level execution

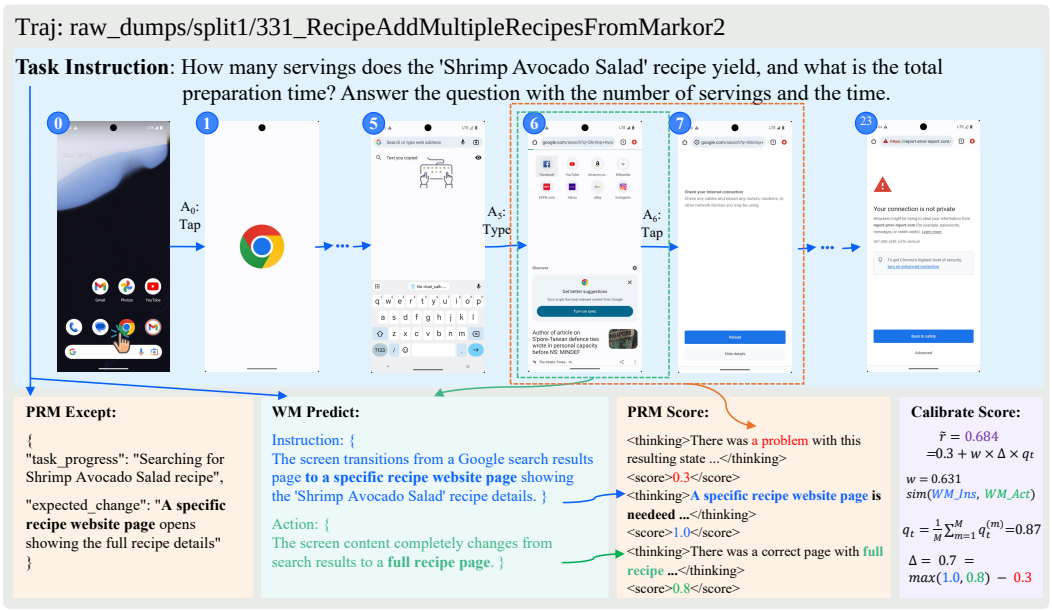

Figure 5: Network Failure Causes PRM Underestimation, Corrected by World-Model Prediction and Quality Recovery

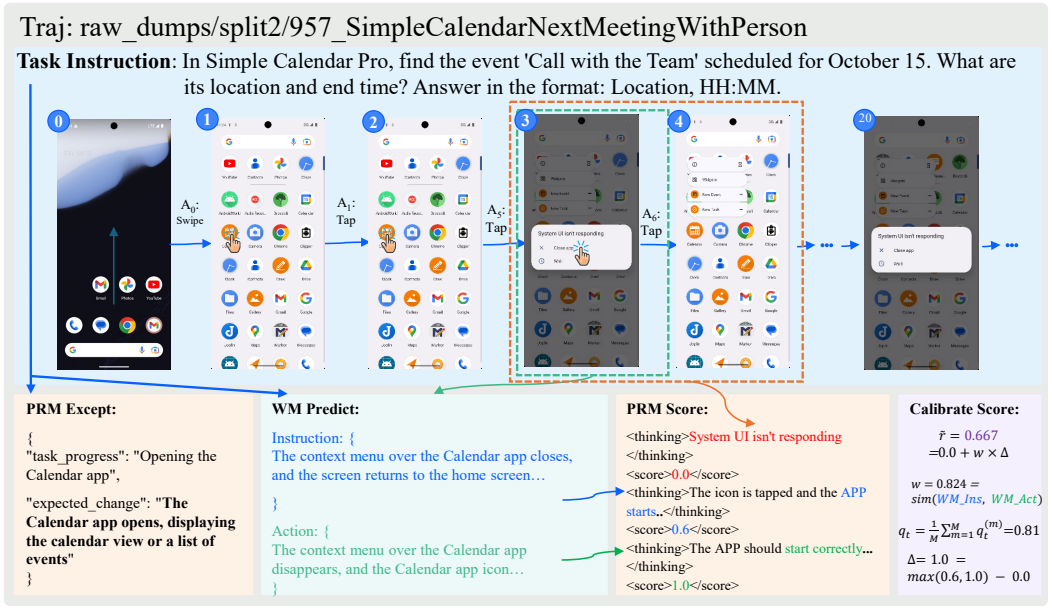

Figure 6: System UI Freeze Causes PRM Underestimation and WM-Guided Quality Recovery

failure rather than an inappropriate action, enabling WM-IncPRM to recover the underestimated step score.

Table 8: Downstream performance on AndroidWorld using the WM-IncPRM-selected Keep subset under different retention rates. Results are reported as mean ± standard deviation over five runs.

Retention Rate	Success	Failed	Avg. Steps	Success Rate (%)
10%	9 ± 1	107 ± 1	15.4	7.76 ± 0.86
20%	9 ± 1	107 ± 1	15.6	7.76 ± 0.86
30%	11 ± 2	105 ± 2	15.4	9.48 ± 1.72
40%	14 ± 1	102 ± 1	14.9	12.07 ± 0.86
50%	18 ± 1	98 ± 1	15.0	15.52 ± 0.86
60%	22 ± 2	94 ± 2	14.7	18.97 ± 1.72
70%	26 ± 2	94 ± 2	13.9	18.97 ± 1.72
80%	28 ± 2	88 ± 2	13.4	24.14 ± 1.72
90%	29 ± 1	87 ± 1	13.1	25.00 ± 0.86
100%	31 ± 1	85 ± 1	13.1	26.72 ± 0.86

E EXPERIMENTALN DETAILS

E.1 ADDITIONAL RESULTS

Table 8 reports the downstream performance of the WM-IncPRM-selected Keep subset under different retention rates. Overall, performance improves steadily as more ranked trajectories are retained, with the success rate increasing from 7.76% at 10% retention to 26.72% at full retention. The largest improvements occur from 30% to 60%, suggesting that retaining additional high-ranked trajectories progressively increases the available training signal. Meanwhile, the average number of steps decreases from 15.4% to 13.1%, indicating more efficient task execution at higher retention rates. These results show that the downstream performance varies consistently with the retained data scale, supporting the use of retention-rate analysis to characterize the ranking behavior of WM-IncPRM.

E.2 IMPLEMENTATION DETAILS

This appendix summarizes the key training and evaluation configurations used in our experiments. Detailed hyperparameters not listed here follow the default settings of the corresponding models or frameworks.

The PRM is trained on a subset of UI-Genie-RM data and remains frozen during trajectory filtering. The MobileWorld-Delta-TEXT-8B world model is also frozen and is used only for future-state prediction. Qwen3.5-4B is fine-tuned on the trajectories selected by each filtering method.

All WM-IncPRM parameters are fixed before evaluation and are shared across the corresponding experimental settings.

All filtering methods retain the same number of OpenMobile trajectories and use the same Qwen3.5-4B training configuration. The trained agents are evaluated on the 116 AndroidWorld tasks using the same execution protocol. We report Task Success, Task Failed, average steps, average execution time, and Success Rate.

Each experiment is independently repeated with five random seeds, and results are reported as mean ± standard deviation. The same seed set and downstream training budget are used across competing methods.

System Prompt

You are a strict GUI action evaluator. Given the task background, action, pre-state, and result state, judge whether the result correctly advances the task goal.

User Prompt

Please evaluate the result state of the following GUI operation step, and judge whether it correctly advances the task goal.

Task background: {context.instruction}
Action: {_action_text(context)}
State before execution: {context.current_state.state_summary or '(none)'}
Result state ({_source_label(future.source)}): {future.to_text() or '(none)'}

Scoring rules:
<thinking>Must cite specific UI elements and text on the interface, and analyze them in combination with the task goal
<critique>When the score in this section is <0.6, problems must be pointed out; when ≥0.8, confirm correctness
<score>Output a number between 0.0 and 1.0
Five-level scoring criteria:
- 1.0 Completely correct: The result state is highly consistent with the task goal.
- 0.8 Clear advancement: The result state clearly advances toward the task goal, but has not yet reached the final target page
- 0.6 Reasonable transition: A reasonable intermediate transition, such as opening a necessary page or pop-up, switching entries, or preparing for subsequent goals.
- 0.3 Weakly related or partially incorrect: The result state is weakly related to the task goal.
- 0.0 Wrong or invalid: The result state is opposite to the task goal, stays on an unrelated page, causes obvious regression, or cannot support continued progress toward the task.

Judging requirements:
Prioritize the task goal, then check whether the result state moves closer to it
Distinguish among "direct completion," "clear advancement," and "reasonable transition"; do not mark reasonable rollbacks and necessary intermediate steps as wrong
The result state may be **predicted** or **actually occurred** (see the label above), but the scoring criteria are exactly the same
Output format:
<thinking>Write the analysis here</thinking>
<critique>Write the evaluation here</critique>
<score>0.0~1.0</score>

Figure 7: Five-level PRM scoring prompt used to evaluate step-level trajectory quality.