Alibaba Cloud

ICML International Conference On Machine Learning

Recovering Policy-Induced Errors: Benchmarking and Trajectory Synthesis for Robust GUI Agents

Tianpeng Bu* **Xin Liu*** **Qihua Chen*** Hao Jiang Shurui Li
Hongtao Duan Lu Jiang Lulu Hu Bin Yang Minying Zhang

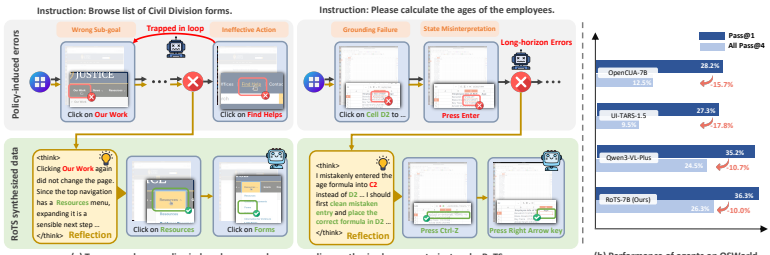
Alibaba Cloud Computing

★ **ICML 2026 Spotlight**

https://github.com/AlibabaResearch/RoTS

Motivation & Overview

GUI agents frequently make **policy-induced errors** during execution and often **cannot recover**, hindering real-world deployment.



(a) Two examples on policy-induced errors and corresponding synthesized recovery trajectory by RoTS

(b) Performance of agents on OSWorld

Contributions:

- **Identify two train-test gaps**: **coverage mismatch** (low-level vs. planning & perception errors) and **horizon mismatch** (immediate vs. multi-step recovery)
- **GUI-RobustEval**: 1,216 test cases across 11 error types $\times$ 4 error depths
- **RoTS**: tree-based data synthesis framework $\rightarrow$ 800k high-quality samples
- **RoTS-7B/32B**: SOTA robustness & task success on OSWorld & GUI-RobustEval

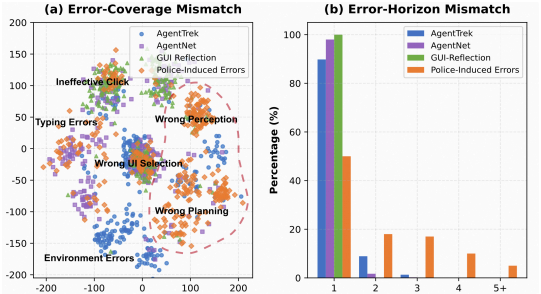
Analysis: Coverage & Horizon Gaps in Existing Data

Methodology:

- Analyze 1.5k failed trajectories from 12 SOTA agents
- Compare error distributions vs. 3 training datasets

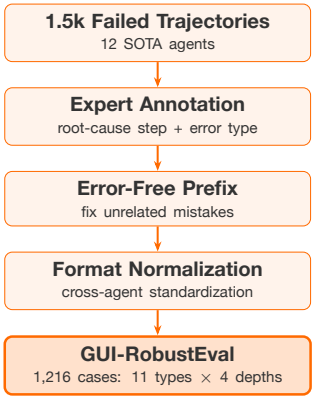
Findings:

- **Coverage gap:** training → low-level; real → **planning & perception**
- **Horizon gap:** training → horizon=0; real → **multi-step**



Existing data misaligns with real policy-induced error distribution in both **type coverage** and **detection horizon**.

GUI-RobustEval: Benchmark Construction



Key Design Choices

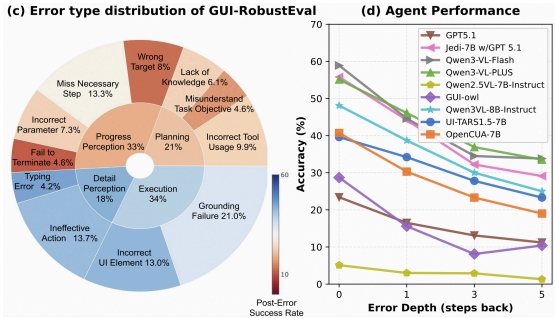
- **Error taxonomy:** 11 types covering planning, perception, execution
- **Controlled depth:** $d \in \{0, 1, 3, 5\}$ steps after root-cause
- **Error-free prefix:** human-verified correction before root-cause
- **Format norm.:** agent-agnostic $\rightarrow$ native format at test time

Evaluation Protocol

- Replay prefix $\rightarrow$ inject error $\rightarrow$ agent takes over
- **Error-Awareness Rate:** does agent recognize error?
- **Post-Error Success Rate:** can agent recover?

GUI-RobustEval: Benchmark Insights

Model	*d*=0	*d*=1	*d*=3	*d*=5	Aw.
Proprietary					
GPT 5.1	23.4	16.5	13.1	11.2	33.9
Jedi w/ GPT 5.1	55.8	44.2	32.2	29.1	34.6
Qwen3-VL-Plus	55.1	46.1	36.9	33.5	65.4
Open-Weights					
UI-TARS1.5-7B	39.6	34.2	27.8	23.3	38.0
OpenCUA-7B	40.7	30.3	23.3	19.0	46.3
OpenCUA-32B	45.5	37.2	28.6	25.9	50.3
RoTS-7B	43.5	36.6	30.1	26.7	51.9
RoTS-32B	**49.7**	**41.8**	**36.5**	**33.2**	**58.8**



- **RoTS-32B**: lowest drop (↓33%) at *d*=5
- Highest error-awareness (58.8%)
- Outperforms all open-weights at every depth

- **Planning & perception** errors hardest to recover
- These types are least covered in training data
- Performance drops sharply with error depth

RoTS: ***Ro***bustness-driven ***T***rajectory ***S***ynthesis

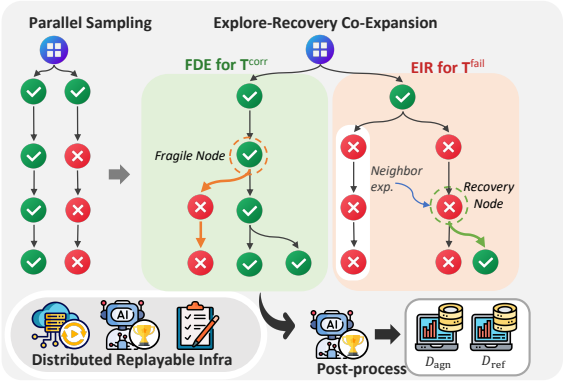
Explore-Recovery Co-Expansion:

- N parallel rollouts $\rightarrow$ trajectory tree T
- Partition T via reward model $\mathcal{R}$ into T^{corr} (successful) and T^{fail} (failed)
- Iterate K rounds: FDE expands T^{corr}, EIR expands T^{fail}
- Post-process: filter noisy steps via progress critic $\mathcal{R}_p$ and action critic $\mathcal{R}_a$

FDE (on T^{corr}): UCB selects high-fragility nodes

$$f_i = (1 - r_i) + c\sqrt{\frac{\ln(V^f_{p(i)} + 1)}{V^f_i + 1}}$$

Replay to fragile node, deploy π_θ to discover new failure modes.



EIR (on T^{fail}): neighbor experiences localize errors

$$\{(i, g_i, p_i)\} \sim \pi_\theta^{er}(u, \tau^{\text{fail}}, \mathcal{E}(\tau^{\text{fail}}))$$

UCB selects recovery node; π_θ^{rec} launches advice-conditioned rollouts.

Results: SOTA Task Success

OSWorld (Ubuntu, 369 tasks)

Model	AP@4	SR@15	SR@50
Proprietary			
OpenAI CUA	–	26.0	31.3
Qwen3-VL-Plus	24.5	33.1	35.2
Open-Weights			
OpenCUA-7B	12.5	24.3	28.2
OpenCUA-32B	15.5	29.7	34.1
Qwen3-VL-32B-Think	21.1	28.1	41.0
RoTS-7B	**26.3**	**31.7**	**36.3**
RoTS-32B	**33.8**	**42.8**	**47.4**

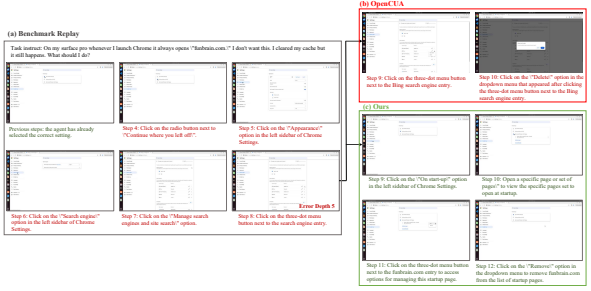
WindowsAgentArena (154 tasks)

Model	SR@15	SR@50
Baselines		
Claude 3.7 Sonnet	7.1	6.4
Jedi-7B w/ GPT4-o	30.2	32.9
UI-TARS-1.5-7B	11.1	15.9
UI-TARS-72B-DPO	11.1	17.9
ScaleCUA-7B	18.0	20.7
ScaleCUA-32B	21.4	24.2
RoTS-7B	**24.9**	**28.2**
RoTS-32B	**35.9**	**39.1**

- **RoTS-32B**: SOTA open-weights on both OSWorld (47.4%) and WindowsAgentArena (39.1%)
- Cross-platform generalization: trained on mixed Ubuntu/Windows data, excels on both
- AP@4 = 33.8% demonstrates consistent robustness across multiple runs

Case Study: Baseline vs. RoTS on GUI-RobustEval

Task: Disable a specific website from opening at startup in Chrome settings.



Baseline (OpenCUA)

- Agent commits a **Fail-to-Terminate** error
- Continues harmful operations (removes "Bing" from search engines)
- **No reflection** — drifts into irrelevant actions

RoTS (Ours)

- Detects task deviation and **reflects on the error**
- Navigates back to "On start-up" settings page
- **Verifies** target website removal — task success

Analysis: Ablation & Scaling

Data Quality

Data (100k)	AP@4	SR@50
AgentNet (human)	7.8	15.3
+ Human refl.	8.4	16.1
+ RoTS refl.	11.6	18.8
RoTS (full)	**14.1**	**21.4**

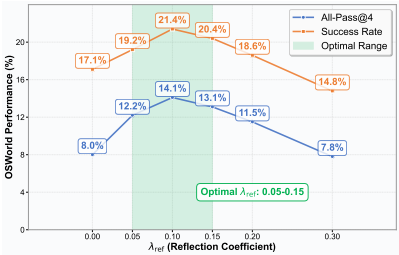
Policy-induced reflections $\gg$ human reflections

Co-Expansion Ablation

Strategy	AP@4	SR@50
PS (baseline)	8.6	18.1
+ FDE	9.1	19.6
+ EIR	12.1	19.5
+ FDE + EIR	**14.1**	**21.4**

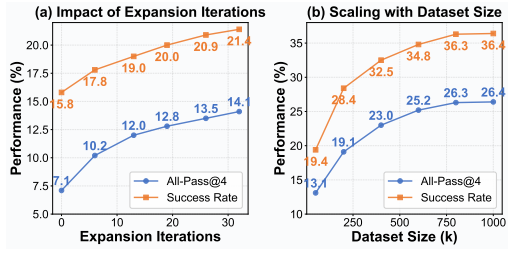
Co-expansion outperforms PS by +5.5% AP@4

λ_{ref} Sensitivity



Best at $\lambda_{\text{ref}}=0.1$; reflective & agnostic complementary

Scaling Curves



0→32 rounds: AP@4 8%→14.1%; 50k→1M: 36.4%

Conclusion & Takeaways

- **Error recovery on policy-induced errors is the key to robust GUI agents.**
- **RoTS enables tree-based synthesis closing coverage & horizon gaps.**

- **GUI-RobustEval:** First benchmark for policy-induced error recovery (11 types $\times$ 4 depths, 1,216 cases)
- **RoTS:** Scalable tree-based synthesis closing coverage & horizon gaps (800k samples)
- **RoTS-32B:** 47.4% OSWorld, 33.8% AP@4, lowest robustness drop

Code & Data

github.com/AlibabaResearch/RoTS

Thank You! **Questions?**