GUI0: Self-Evolving Foundational GUI Agents in Super App Ecosystems

Xinyi Wang[1,2*], Wei Dai[1], Kyle Qiao[1], Ke Wang[3], Peng Chen[4], Gang Cao[1], Kang Qin[1], Zhongpu Wang[1], Xiaode Zhang[1], Yanming Liu[5], Jihao Gu[6], Jingtao Xu[1], Gong Zhi[1†]

[1]WeChat, Tencent Inc., China, [2]Nankai University, [3]Renmin University of China, [4]University of Chinese Academy of Sciences, [5]Zhejiang University, [6]Beijing University of Posts and Telecommunications

2120230741@mail.nankai.edu.cn

Abstract

Automated interaction with graphical user interfaces (GUIs) is central to General Artificial Intelligence yet remains challenging within Super App ecosystems, characterized by non-standard rendering and absent accessibility metadata. While GUI agents often rely on explicit accessibility trees or static imitation, they are less explored for dynamic environments marked by sparse feedback and implicit visual cues. We present GUI0, a framework synergizing autonomous data synthesis with dual-agent co-evolution. GUI0 establishes a domain-aware foundation model via synthesized corpora and employs curriculum-driven reinforcement learning, where a curriculum agent generates boundary tasks to optimize an actor agent. Empirical results demonstrate three key advantages: (1) State-of-the-art performance on the SuperAPP benchmark, outperforming Gemini-2.5-Pro and Claude-4-Sonnet; (2) universal efficacy across diverse base models, consistently yielding substantial improvements on both Qwen2.5-VL and GUI-Owl variants; and (3) robust zero-shot generalization to standard GUIs (e.g., +62.7% on ScreenSpot Pro).

1 Introduction

Automated interaction with graphical user interfaces (GUIs) is pivotal for advancing general artificial intelligence, enabling mobile assistance, accessibility, and end-to-end workflow execution (Ye et al., 2025; Jiang et al., 2025; Zhang et al., 2025b; Wang et al., 2024; Li and Huang, 2025; Zhang et al., 2024). Beyond standard native applications, Super Apps (e.g., WeChat and Revolut) introduce unique challenges. Functioning as "operating systems within operating systems," they host tens of thousands of mini apps within a dynamic, hybrid runtime environment. Unlike native applications with well-defined XML accessibility trees (Li et al., 2024c), Super App ecosystems often expose incomplete or missing accessibility metadata and rely on non-standard rendering technologies such as Canvas and WebViews (Shi et al., 2025; Wang et al., 2025). Consequently, agents must infer actionable UI structures directly from pixels and perform long-horizon planning under sparse and delayed feedback, making automation in these non-standard Android scenarios substantially more difficult than in traditional Android environments.

Pioneering works (Qin et al., 2025; Ye et al., 2025; Gu et al., 2025b) have established strong GUI interaction capabilities via progressive training paradigms. However, these approaches are predominantly premised on the availability of well-formed UIs and stable accessibility metadata, encountering challenges due to data scarcity in non-standard and dynamic environments.

Self-evolving methods (Xia et al., 2025; Liu et al., 2025; Yuan et al., 2025; Jiang et al., 2025) enhance autonomy through self-play or self-correction and demonstrate promising generalization. Real-world Super Apps feature highly dynamic interfaces, near-infinite state spaces, and limited supervised data for non-standard UI scenarios, the lack of scene-specific priors can hinder the generation of effective curriculum tasks in complex GUI environments.

To bridge this gap, we introduce **GUI0**, an autonomous framework designed for robust interaction in Super Apps through data synthesis and dual-agent co-evolution. As illustrated in Fig. 1, GUI0 framework integrates three core components: *Agent-based Data Synthesis*, which autonomously constructs a hierarchical training corpus; a *Foundational Model* trained on large-scale agentic data to instantiate task-specific agents; and a *Dual-Agent Co-Evolution* mechanism, in which a curriculum agent generates tasks at the boundary of an actor agent's capabilities, employing Group Relative Policy Optimization (GRPO) to align the actor agent

*Work was done when interned at WeChat, Tencent Inc.
†Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44348–44361
July 2-7, 2026

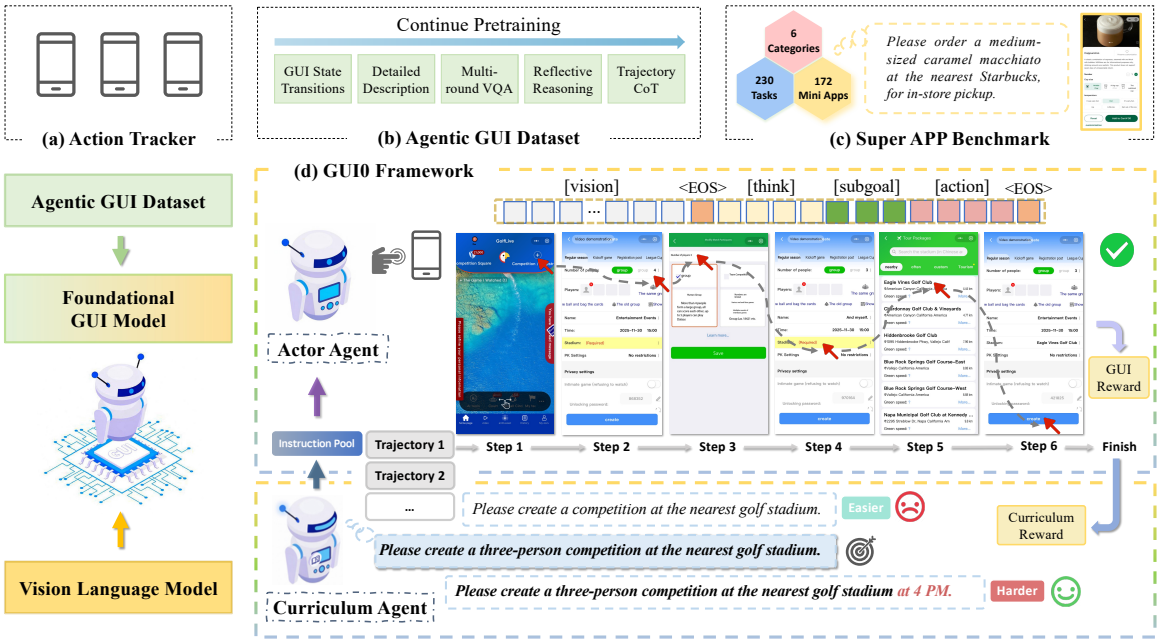

Figure 1: **Overview of the GUI0 framework. (a) Action Tracker** captures raw action logs. **(b) Agentic GUI Dataset** constructs diverse GUI corpus. **(c) Super APP Benchmark** evaluates agent robustness within complex ecosystems. **(d) GUI0 Framework** trains from cognitive grounding to dual-agent co-evolution.

with sparse, task-level objectives. This synergy between domain-specific foundational knowledge and curriculum-driven reinforcement learning enables GUI0 to achieve stronger alignment between domain-aware foundations and execution logic.

Our contributions are summarized as follows:

- **Agentic Data Synthesis:** We implement an automated pipeline to synthesize large-scale, high-quality GUI data without manual annotation, effectively mitigating data scarcity in non-standard rendering environments.
- **The GUI0 Framework:** We propose a dual-agent co-evolution framework on domain-aware foundations. By instantiating a curriculum agent to generate boundary tasks and co-evolve with an actor agent, we enable robust long-horizon execution on Super App tasks.
- **SuperAPP Benchmark:** We establish a specialized SuperAPP benchmark comprising 230 carefully designed tasks across 172 mini apps and six categories, assessing agent adaptability in hybrid UI ecosystems.
- **State-of-the-Art Performance:** Experiments show that GUI0 achieves a peak success rate of 68.7% on the SuperAPP benchmark, outperforming Gemini-2.5-Pro (+24.3%) and the backbone model (+44.8%), while maintaining robust zero-shot generalization on ScreenSpot Pro.

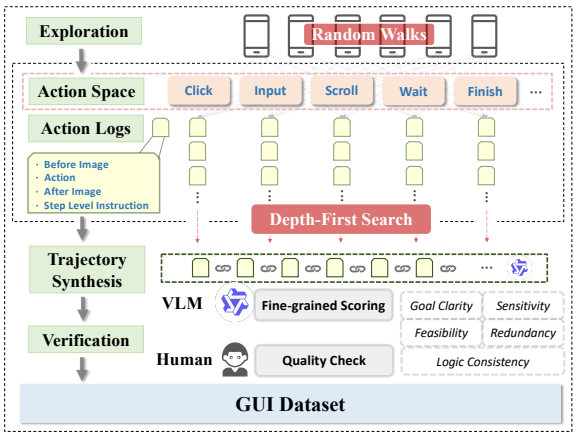

Figure 2: Pipeline of Data Synthesis.

2 Tracker and Benchmark

We formulate GUI interaction as a POMDP defined by (S, A, O, R, γ). To address the inefficiency of static instructions in open-ended, sparse-reward environments, we develop an automated **Action Tracker** for data collection and establish the SuperAPP Benchmark for robust assessment.

Exploration and Trajectory Construction. Navigating the state space S employs a hybrid strategy of random walks and Depth-First Search (DFS). At step t, the tracker maps observation o_t (screenshot I_t) to an action from the unified space A, comprising ten operations: `click`, `input`, `scroll`, `wait`, `finish`, `long_press`,

Table 1: Statistics of the SuperAPP Benchmark.

Category	Tasks	Mini Apps	Percentage
Commerce & Local Services (C&L)	61	43	26.5%
Productivity & Public Services (P&P)	51	39	22.2%
Communication & Social (C&S)	33	25	14.3%
Navigation & Mobility (N&M)	33	21	14.3%
Media & Browser (M&B)	26	21	11.3%
Games & Entertainment (G&E)	26	23	11.3%
Total	**230**	**172**	**100.0%**

`open_app`, `navigate_home`, `navigate_back`, and `call_user` (details in Appendix A). Coordinates for UI-grounded actions are derived from the centers of OmniParser-detected bounding boxes. The resulting triplets $\langle o_t, a_t, o_{t+1} \rangle$ are assembled into coherent trajectories τ by matching sequential image hashes and pruning loops. This ensures strict on-policy supervision reflecting the true state distribution.

Reverse Task Synthesis and Data Collection. We employ a VLM to annotate trajectories with latent instructions x, approximating $P(x|\tau)$. To ensure quality, a separate VLM filters pairs across five dimensions: *Redundancy*, *Sensitivity*, *Feasibility*, *Logic Consistency*, and *Goal Clarity* (refer to Appendix B). VLM scoring serves as a scalable automatic pre-filter and all retained trajectories are subsequently reviewed by human annotators as the final quality and privacy gate.[1]

SuperAPP Benchmark. Real-world Super Apps present unique challenges, including non-standard widgets, deep navigation, and hybrid rendering. To assess agent robustness within such complex ecosystems, we establish the SuperAPP Benchmark (Tab.1), comprising 230 high-quality tasks across 172 mini apps and 6 categories. Derived from real-world user queries, these tasks cover multi-step interactions and retrieval. Data quality is guaranteed through a rigorous validation process, involving over five rounds of simulation and cross-validation by a ten-person expert team. All evaluations are conducted on standard Android emulators to maintain consistency.

3 Methodology

We propose GUI0, an autonomous data synthesis and dual-agent co-evolution framework, guided by constructing a robust cognitive foundation via Agentic Continuous Pre-training (Agentic CPT) and fostering capability breakthroughs through Dual-Agent Co-Evolution. Fig. 1 illustrates the overall pipeline.

3.1 Preliminaries

Input/Output Format. At step t, the model input c_t contains: the instruction x, a textual history summary $\{h_1, ..., h_{t-1}\}$, and visual context $\{I_{t-1}, I_t\}$ encoded as visual tokens. The agent outputs a structured response y_t with `<think>`, `<subgoal>`, `<answer>` blocks, ensuring reasoning precedes action. Templates are in Appendix E.

Training Paradigm. Adapting general VLMs to specific GUI tasks typically involves a progressive three-stage pipeline. First, the model undergoes **Continual Pre-training (CPT)** to inject domain knowledge via auto-regressive modeling on GUI corpora. Subsequently, **Supervised Fine-tuning (SFT)** on expert trajectories aligns the agent with instruction-following behaviors. Finally, offline/online **Reinforcement Learning (RL)** employs GRPO to maximize expected rewards. Specifically, for a given input x, GRPO samples a group of outputs $\{y_i\}_{i=1}^{G}$ from the old policy π_{old} and optimizes the current policy π_θ by maximizing the group-normalized advantage $\hat{A}_i$, while constraining deviation from the reference policy π_{ref} (typically the SFT model):

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim D,\{y_i\}\sim\pi_{\text{old}}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Big(\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{old}}(y_i \mid x)}\hat{A}_i, \text{clip}\Big(\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{old}}(y_i \mid x)}, 1-\epsilon, 1+\epsilon\Big)\hat{A}_i\Big) - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Bigg] \tag{1}$$

where $\hat{A}_i$ represents the advantage normalized across the group reward distribution, ϵ and β are the PPO clipping hyper-parameter and the coefficient controlling the Kullback-Leibler (KL) -divergence penalty, and $\mathbb{D}_{\text{KL}}$ is calculated at the token level to ensure stability.

3.2 Agentic Data Synthesis

Leveraging raw interaction logs from the Action Tracker (Sec. 2), we synthesize a verified GUI corpus to strengthen OCR, grounding, and reasoning. The pipeline includes screenshot acquisition, automated annotation (Gemini-2.5-Pro), and two-stage verification (VLM filter and human check). We construct five task types (details in Appendix B).

(1) GUI State Transitions. From triplets $\langle o_{pre}, a, o_{post} \rangle$, we create forward dynamics (predict o_{post} given o_{pre}, a) and inverse dynamics (infer a from $o_{pre} \rightarrow o_{post}$) to teach causal GUI dynamics.

[1]The synthetic data was used exclusively for training and strictly isolated from the evaluation set.

(2) Detailed Screen Descriptions. Dense captions describing components, text, layout, and functions provide fine-grained perceptual grounding.
(3) Multi-turn VQA. We filter screens by OmniParser elements (< 3 items) and generate multi-turn QA emphasizing spatial grounding (e.g., *text2bbox*, *bbox2text*) and functional understanding.
(4) Reflective Reasoning. We generate complex queries, followed by a *post-hoc analysis* of the agent's responses to provide alternative reasoning paths. This injects a "reflection" mechanism into the training data, guiding the model to decompose problems and generalize solutions.
(5) Trajectory CoT. We synthesize step-by-step CoT rationales to align perception with decision-making and clarify action principles.

3.3 GUI0 Training Framework

GUI0 adopts a progressive learning paradigm, evolving from cognitive grounding to dual-agent co-evolution. This pipeline effectively bridges general vision-language capabilities and specialized GUI execution.
Stage 1: Cognitive Grounding via CPT. We conduct CPT on the synthesized corpus (Sec. 3.2) to inject domain priors. This establishes the essential perceptual and reasoning capabilities, serving as the shared initialization for both subsequent agents.
Stage 2: Actor Agent Specialization. Building on the CPT backbone, the Actor Agent π_ϕ undergoes specialization via SFT on high-quality synthetic CoT trajectories. Followed by a two-phase GRPO strategy to bridge atomic visual grounding with long-horizon planning.
(1) Phase 1: Single-step GRPO. To refine atomic operations, we optimize the agent on offline trajectories with step-level ground truth a_{gt}. The output y adheres to the structured format `<think><subgoal><answer>`. The step-wise reward R_{step} is formulated as:

$$R_{\text{step}}(y, a_{gt}) = \underbrace{R_{\text{fmt}} + R_{\text{align}} + R_{\text{subgoal}}}_{\text{Reasoning Compliance}} + \underbrace{\mathbb{I}_{\text{type}} \cdot (r_{\text{base}} + r_{\text{coord}} + r_{\text{content}})}_{\text{Action Precision}} \tag{2}$$

The first term ensures reasoning integrity, R_{fmt} validates the format, R_{align} employs a VLM evaluator (Qwen2.5-VL-72B) to verify consistency between think and action, and R_{subgoal} measures the F1 alignment of subgoals. The second term incentivizes execution precision conditioned on action type matching ($\mathbb{I}_{\text{type}} = 1$). This comprises a base reward r_{base}, r_{coord} validating coordinate falls within bounding box, and a character-level text matching score r_{content}. Detailed formulation in Appendix C.
(2) Phase 2: Multi-turn GRPO. To improve task completion, we transition to online trajectory-level supervision. We sample G parallel rollouts for instruction x and employ a larger model as an unbiased evaluator. The trajectory reward is defined as:

$$R_{\text{traj}}(\tau) = R_{\text{fmt}} + R_{\text{success}} \tag{3}$$

where $R_{\text{success}} \in \{0, 1\}$ denotes the binary success status assessed by the evaluator based on the instruction x and trajectory history.
Stage 3: Dual-Agent Co-Evolution. To continually synthesize training tasks, we co-train a Curriculum Agent π_ψ (initialized from the Stage 1 CPT model) together with the Actor π_ϕ. Given an instruction x, we estimate the Actor's empirical success rate $S_x \in [0, 1]$ via repeated rollouts scored by the trajectory-level evaluator (Eq. 3). The Curriculum Agent is optimized to generate instructions that maximize:

$$\begin{cases} S_x = \dfrac{1}{G}\displaystyle\sum_{i=1}^{G} R_{\text{success}}(\tau_i) \\ R_{\text{diff}}(x) = \min(S_x, 1 - S_x) \\ R_{\text{len}}(x) = \log(\text{len}(x) + 1) \\ R_{\text{curric}}(x) = \mathbb{I}_{\text{valid}}(x) \cdot \Big[R_{\text{diff}}(x) + R_{\text{len}}(x)\Big] \end{cases} \tag{4}$$

where $R_{\text{diff}}(x)$ peaks at $S_x = 0.5$ and thus favors boundary-difficulty tasks while down-weighting tasks which always solved or failed. $R_{\text{len}}(x)$ encourages semantically richer instructions, capped at $L_{\max}$. $\mathbb{I}_{\text{valid}}(x) \in \{0, 1\}$ filtering out non-executable instructions ensuring training stability. Concretely $\mathbb{I}_{\text{valid}}(x) = 1$ iff at least one of G parallel rollouts under the current actor succeeds.

4 Experimental Setup

Dataset. Following UI-Tars and Mobile-Agent-v3 methodologies (Qin et al., 2025; Ye et al., 2025), we utilized the WeChat Super APP (covering 200+ sub-industries) as our experimental platform. By deploying action trackers across 300 miniapps for one month, we yielded 700k screenshots and 10,000 high-quality trajectories (avg. step 7.8, 95%/5% train/val split) for Agentic

Data Synthesis (Sec. 3.2). Synthesized data is organized into three subsets: (1) **Agentic CPT Corpus.** 2.34M samples comprising GUI state transitions (52.1%), screen descriptions (9.1%), multi-turn VQA (27.2%), reflective reasoning (7.3%), and CoT actions (4.4%); (2) **SFT Cold-Start.** 9,700 trajectories enriched with CoT reasoning; and (3) **Co-evolution (RL).** A selection of 7,000 trajectories for single-step GRPO, alongside 500 representative tasks for multi-step curriculum expansion.

Benchmarks and Metrics. We evaluate baselines on the SuperAPP Benchmark, Android Control (AC) (Li et al., 2024a), ScreenSpot Mobile (SSM) (Cheng et al., 2024), ScreenSpot Pro (SSP) (Li et al., 2025b), and Android World (AW) (Rawles et al., 2024). We report official metrics. Step Accuracy for AC (Low/High splits), Accuracy for SSM (Text/Icon splits) and SSP, Success Rate (SR) for AW, Task Success Rate (TSR) for SuperAPP Benchmark, which is the fraction of tasks completed within a 50-step budget. A task succeeds iff its final UI state meets task-specific criteria (e.g., reaching the target page or returning the required information). For curriculum analysis, we report the fraction of executable generated instructions (Inst. Val (%)), mean trajectory length (Avg. Step), the Pass@16 divergence between the initial ($\text{Actor}_{\textbf{Init}}$) and curriculum-trained ($\text{Actor}_{\textbf{Curr}}$) actors, where a widening gap signifies increasing yet learnable difficulty.

Baselines. We compare three model categories. Closed-source VLMs, GPT-4o (Hurst et al., 2024), Claude-4-Sonnet, and Gemini-2.5-Pro (Comanici et al., 2025). Open-source VLMs, Qwen2.5-VL (7B/32B/72B)(Bai et al., 2025) and GLM-4.5V(Team et al., 2025). Specialized GUI Agents, UI-TARS-1.5 (7B) (Qin et al., 2025), GUI-OWL-32B (Ye et al., 2025), and UI-Venus-72B (Gu et al., 2025b). All baselines use official prompts or recommended configurations for fair comparison.

Training Settings. We initialize the Actor Agent with Qwen2.5-VL (7B/32B) or GUI-OWL-32B, and the Curriculum Agent with Qwen2.5-VL-32B. Training proceeds in three stages under the GUI0 framework. Stage 1 (Agentic CPT) trained for 1 epoch (LR 1e-5, global batch size 1,024) on 256 NVIDIA H20 GPUs. Stage 2 (SFT) fine-tuned for 1 epoch with a cosine schedule (peak LR 1e-5) on 32 NVIDIA H20 GPUs. In Stage 3 (Co-evolution) both agents are optimized via GRPO for 3 epochs ($G = 8$ rollouts per instruction, temperature 0.7, top-p 0.9, top-k -1, $\epsilon = 0.2$, $\beta = 0.04$, $L_{\max} = 40$) on 32 NVIDIA H20 GPUs.

Implementation Details. Task-level testing is performed on standard Android Emulators with test accounts. For general benchmarks, we strictly adhere to official evaluation protocols and action spaces. Inference employs temperature 0, top-k 1 with a 512-token output limit.

5 Experiment Results

5.1 Main Results 1: SuperAPP Benchmark Evaluation

Tab. 2 compares our GUI0-enhanced models against baselines on the SuperAPP Benchmark, which assesses agent robustness in dynamic, hybrid Android environments. Task success is determined by unanimous agreement between an automatic evaluator (Gemini-2.5-Pro comparing against expert-annotated gold trajectories) and human verification. The table presents detailed scores across six sub-categories along with the average performance. Key observations are as follows.

GUI0 enhanced models establish new state-of-the-art performance across complex application ecosystems, achieving remarkable gains over both closed-source and open-source baselines. As shown in Tab. 2, Qwen2.5-VL-32B-GUI0 attains a leading average score of 68.7%, surpassing the strong open-source baseline UI-Venus-72B by +23.0 points and outperforming its backbone model by a substantial margin of +34.3 points. Notably, even the smaller 7B variant of GUI0 demonstrates exceptional efficiency, achieving an average score of 36.5%, which surpasses the much larger GPT-4o model by +1.3 points, validating the effectiveness of our specialized training pipeline.

Performance gains are particularly significant in long-horizon and dynamic tasks, where deep multi-level interaction is paramount. For instance, in the challenging *Navigation & Mobility* category, Qwen2.5-VL-32B-GUI0 achieves a 3.1× improvement compared to GPT-4o and a 2.5× enhancement over its base model. Similarly, in *Commerce & Local Services*, the model exhibits a +37.3% relative improvement compared to Gemini-2.5-Pro. These results confirm GUI0's capacity to handle dynamic GUI elements in specific domains, significantly enhancing the semantic understanding and execution capabilities of general visual-language models.

Table 2: Performance comparison on **SuperAPP Benchmark**. GUI0 enhanced models inherit base architecture identifiers with the -GUI0 suffix. The highest scores in each column are highlighted in **bold**.

Model	SuperAPP Benchmark (TSR (%))						
	Category Scores						Avg.
	C&L	P&P	C&S	N&M	M&B	G&E	
Closed-Source Large Vision Language Models							
Gemini-2.5-Pro	52.5	51.0	45.5	33.3	42.3	26.9	44.4
GPT-4o	37.7	25.5	42.4	27.3	42.3	42.3	35.2
Claude-4-Sonnet	39.3	52.9	**63.6**	45.5	**76.9**	38.5	40.4
Open-Source Large Vision Language Models							
Qwen2.5-VL-7B	16.4	19.6	0.0	12.1	0.0	3.8	10.9
Qwen2.5-VL-32B	44.3	33.3	24.2	33.3	53.8	7.7	34.4
Qwen2.5-VL-72B	45.9	45.1	27.3	42.4	53.8	7.7	39.1
GLM-4.5V	34.4	52.9	15.2	27.3	34.6	26.9	33.9
Open-Source GUI Agents							
UI-TARS-1.5-7B	24.6	37.3	12.1	39.4	38.5	23.1	29.1
GUI-Owl-32B	23.0	27.5	3.0	24.2	46.2	23.1	23.9
UI-Venus-72B	50.8	49.0	42.4	33.3	69.2	23.1	45.7
GUI0 (Ours)							
Qwen2.5-VL-7B-GUI0	50.8	27.5	21.2	27.3	46.2	42.3	36.5
GUI-Owl-32B-GUI0	68.9	**66.7**	54.5	81.8	65.4	**57.7**	66.5
Qwen2.5-VL-32B-GUI0	**72.1**	**66.7**	57.6	**84.8**	69.2	**57.7**	**68.7**

5.2 Main Results 2: General GUI Capability

To verify that our specialized training preserves fundamental capabilities and avoids catastrophic forgetting, we evaluated GUI0 on standard GUI benchmarks. Results are presented in Tab. 3.

Models enhanced with GUI0 demonstrate robust zero-shot generalization to OS and Web environments. Despite being trained exclusively on Android SuperApp scenarios, GUI0 exhibits remarkable transferability to unseen domains. On ScreenSpot Pro, Qwen2.5-VL-32B-GUI0 achieves 64.1%, representing relative improvements of +62.7% over its base model (39.4%) and +10.5% over the previous SOTA, GUI-Owl-32B (58.0%). Furthermore, on Android World, it scores 64.0%, outperforming its backbone (22.0%) by a substantial +190.9%. These results confirm that GUI0 establishes a robust cognitive foundation that effectively transfers to general GUI environments.

5.3 Main Results 3: Curriculum Agent Analysis

A core component of our *Dual-Agent Co-Evolution* framework is the Curriculum Agent, which generates training tasks for the Actor. Tab. 4 compares the instruction generation quality of our evolved agents against general VLM baselines.

The GUI0 Curriculum Agent progressively learns to generate more valid and challenging tasks. As shown in Tab. 4, general-purpose VLMs struggle to produce valid executable SuperAPP instructions (e.g., Qwen3-VL: 32.1% validity). In contrast, our evolved agent reaches 92.6% validity at GRPO Iter 3, a +104% relative improvement over the strongest baseline (Gemini-2.5-Pro). Task complexity also increases substantially: the average trajectory length grows from 7.0 (Iter 0) to 15.8 (Iter 3), 2.4× longer than Gemini-2.5-Pro. While validity stays high, the initial actor's pass rate drops from 93.8% to 75.3%, reflecting the emergence of progressively harder tasks; meanwhile, actor trained on curriculum tasks improves markedly (ΔPass 11.9). Together, these results indicate that the Curriculum Agent provides an adaptive learning signal that avoids overfitting to trivial tasks and drives robust gains in dynamic environments.

Table 3: Performance comparison on **General Benchmark**.

Model	General Benchmark					
	AC (Step Acc.)		SSM (Acc.)		SSP (Acc.)	AW (SR)
	Low	High	Text	Icon		
Open-Source Large Vision Language Models						
Qwen2.5-VL-7B	85.0	62.9	71.1	55.0	27.6	27.6
Qwen2.5-VL-32B	93.3	69.6	84.6	77.3	39.4	22.0
Qwen2.5-VL-72B	93.7	67.4	93.7	80.5	43.6	35.0
Open-Source GUI Agents						
UI-TARS-1.5-7B	90.8	72.5	**95.5**	84.5	35.7	30.0
GUI-Owl-32B	–	76.6	93.6*	81.2*	58.0	**73.3**
UI-Venus-72B	92.9	77.2	92.8*	85.3*	61.9	65.9
GUI0 (Ours)						
Qwen2.5-VL-7B-GUI0	89.2	68.5	88.4	74.4	37.5	30.0
GUI-Owl-32B-GUI0	92.4	75.7	91.2	**86.1**	61.1	65.3
Qwen2.5-VL-32B-GUI0	**94.2**	**78.2**	93.4	83.8	**64.1**	64.0

Results marked with * are our reproduced results. "–" indicates result is unavailable.

Table 4: Curriculum instruction quality and solvability evaluation.

Method	Inst. Val (%)	Avg. Step	$\text{Actor}_{\text{Init}}$	$\text{Actor}_{\text{Curr}}$	ΔPass
Static Sampling	–	6.8	94.2	96.1	1.9
General VLMs (Few-shot generation)					
Gemini-2.5-Pro	45.3	6.6	82.1	86.8	4.7
Qwen3-VL-235B-A22B	32.1	5.6	81.0	84.6	3.6
Qwen2.5-VL-32B	26.2	6.0	83.2	85.4	2.2
GUI0-Curriculum Agent (Ours)					
CPT Init (Iter 0)	92.7	7.0	93.8	95.5	1.7
GRPO Iter 1	90.0	9.8	76.5	83.8	7.3
GRPO Iter 2	91.9	13.4	76.5	85.4	8.9
GRPO Iter 3	**92.6**	**15.8**	75.3	**87.2**	**11.9**

5.4 Ablation Study

We perform ablation studies on the SuperAPP Benchmark to quantify the contribution of each component in GUI0. The results show that both the synthesized data composition (Stage 1 CPT) and the progressive training pipeline are necessary, and

Table 5: Ablation on Data Composition (Stage 1). Task Success Rate (TSR) on SuperAPP Benchmark.

CPT Data Configuration	TSR (%)
Full Synthesized Corpus	**68.70**
w/o Trajectory CoT	67.83 (-0.87)
w/o Detailed Description	67.39 (-1.31)
w/o Multi-turn VQA	66.52 (-2.18)
w/o Reflective Reasoning	66.09 (-2.61)
w/o GUI State Transitions	65.04 (-3.66)

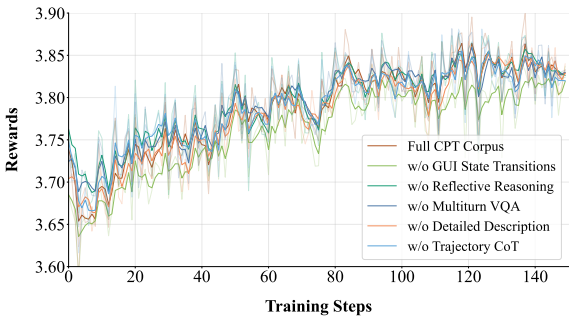

Figure 3: Reward convergence analysis of different CPT data compositions.

their combination yields the best performance.

5.4.1 Data Composition Ablation

We systematically removed specific data components from the Stage 1 CPT corpus. Tab. 5 reports downstream performance, and Fig. 3 shows the smoothed average reward curves during the training steps. All variants use the same SFT initialization and identical RL hyperparameters.

All five CPT data types are essential for optimal GUI comprehension. The full-data configuration achieves best performance, while removing any component degrades results. Notably, excluding *GUI State Transitions* leads to the most severe performance drop (-3.66), indicating that modeling causal dynamics is crucial. The drops from removing *Reflective Reasoning* (-2.61) and *Multi-turn VQA* (-2.18) underscore the critical role of reasoning traces in error correction and alternative planning, and the importance of long-context understanding for handling interruptions. Consistently, the full dataset also leads faster and higher reward convergence in GRPO.

5.4.2 Progressive Training Process Ablation

Tab. 6 presents performance across training stages.

Complete training pipeline synergistically enhance performance. Stages 1–2 provide a necessary cognitive foundation (+10.0), after which single-step GRPO (Stage 3a) delivers a substantial

Table 6: Ablation on Training Stages. Performance evolution from Base to GUI0.

Training Stage	TSR (%)
Base (Qwen2.5-VL-32B)	34.35
+ Stage 1–2: Agentic CPT + SFT	44.35 (+10.00)
+ Stage 3a: Single-step GRPO	61.74 (+17.39)
+ Stage 3b Strategies:	
Multi-step GRPO (Static)	65.65 (+3.91)
Co-Evolution (Ours)	**68.70** (+6.96)

+17.39 improvement. To isolate the contribution of our curriculum design, we compare two multi-turn strategies: a static Multi-turn GRPO baseline with random instruction sampling achieves a modest +3.91% gain, whereas our Co-Evolution gains +6.96%. These results suggest that while trajectory-level optimization is effective, an adaptive curriculum is pivotal for pushing the agent's capability boundary in complex scenarios.

5.5 Reward Evaluator Robustness Evaluation

We analyze whether our GRPO learning signal is overly dependent on a particular reward model by performing a reward re-labeling sensitivity test. Specifically, we fix a set of 300 GRPO rollouts randomly sampled across training epochs and obtain reference binary success/failure labels via majority vote from six human annotators, then re-label the same rollouts with several VLM evaluators under identical inputs (instruction, full action and screenshot history) to compare their success judgments. Qwen2.5-VL-72B serving as the reward model used during training.

The quality of trajectory-level judgments is robust. As shown in Fig. 5, the estimated success rates from different evaluators are close and largely overlap within 95% confidence intervals. This indicates that the trajectory-level reward signal used in training is not brittle to the evaluator choice, alleviating concerns about reward model specific bias.

5.6 Qualitative Visualization

Fig. 4 illustrates the Actor Agent's progression under the Curriculum Agent's guidance. Initially (top row), the agent employs explicit CoT to decompose a pricing query, validating its reasoning efficacy. As the Curriculum Agent introduces temporal constraints (middle row), the Actor demonstrates robustness by navigating non-standard UI elements like calendar widgets. Finally, the task evolves into

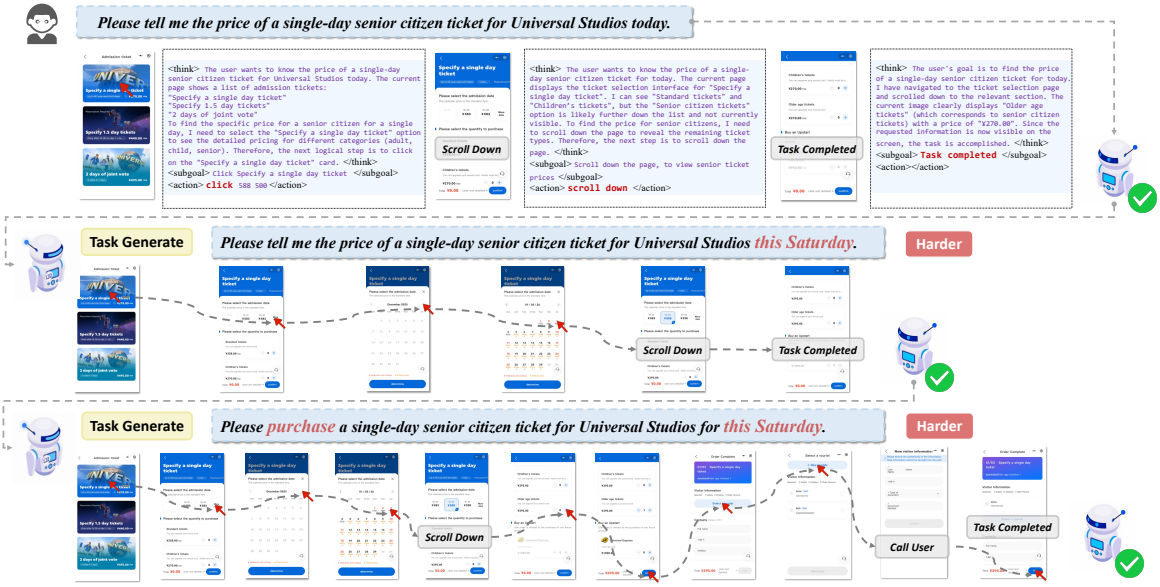

Figure 4: Qualitative visualization of GUI0: a case study on Universal Studios ticket booking.

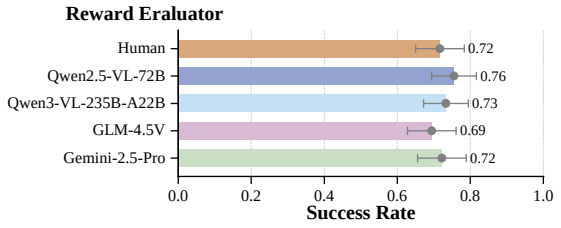

Figure 5: Sensitivity of trajectory-level success labeling to reward evaluator choice. Error bars denote 95% bootstrap confidence intervals (1,000 resamples), reflecting finite-sample uncertainty.

a long-horizon "purchase" workflow (bottom row), where the agent successfully coordinates date selection, quantity adjustment, and payment. Overall, GUI0 exhibits reliable self-evolution, effectively bridging atomic actions with complex task logic.

6 Related Work

6.1 Agentic Continual Pre-training and GUI Agents

Agentic Continual Pre-training (CPT) (Parmar et al., 2024; Su et al., 2025; Chen et al., 2025b; Li et al., 2025a; Lin et al., 2025; Zhuang et al., 2025) has become a key bridge between general pre-training and post-training alignment, injecting domain knowledge and strengthening planning while mitigating catastrophic forgetting. For GUI Agents, standard Vision-Language Models (VLMs) remain limited in fine-grained widget grounding (e.g., OCR) (Lu et al., 2024; Yu et al., 2025; Zhang et al., 2025a; Chen et al., 2025a; Gu et al., 2025a; Wu et al., 2024; Chen et al., 2025b; Xie et al., 2025; Yan et al., 2025). Recent systems such as UI-TARS, Mobile-Agent-v3 and Ui-venus (Qin et al., 2025; Ye et al., 2025; Gu et al., 2025b) address this gap by incorporating large-scale, high-quality GUI data, substantially improving perception and action precision.

Super Apps and their mini app ecosystems, however, pose distinct challenges (Wang et al., 2025; Shi et al., 2025). Compared with native apps, they heavily rely on hybrid runtimes (e.g., Canvas, WebView), which often yield missing Accessibility Trees (Li et al., 2024b) and thus force agents to depend primarily on visual perception. Meanwhile, high-quality GUI data tailored to such non-standard rendering remains scarce. The high dynamics and deep nesting of Super Apps further stress long-horizon planning under sparse feedback. Existing Android benchmarks (Rawles et al., 2024; Li et al., 2024a) also struggle to cover these hybrid environments, highlighting the need for specialized evaluation protocols and data-efficient solutions.

6.2 Agent Self-Evolution and Curriculum Learning

To circumvent the scalability bottleneck of human annotation, methods such as Self-Instruct (Wang et al., 2023) and Evol-Instruct (Xu et al., 2024) show that progressively synthesized data can improve generalization (Gao et al., 2025; Fang et al., 2025). Building on this idea, Agent0 (Xia et al., 2025; Liu et al., 2025) employs self-play and self-correction, enabling models to act as both solver and evaluator for self-improvement (Yuan

et al., 2025; Xi et al., 2025). In GUI settings, SEAgent (Sun et al., 2025), AppAgentX (Jiang et al., 2025), UI-Evol (Zhang et al., 2025c) and CRAFT-GUI (Nong et al., 2025) further explore experience-driven optimization via trial-and-error, shortcut evolution, and trajectory backtracking to enhance execution efficiency.

7 Conclusion

In this paper, we introduce GUI0, a framework designed to master the non-standard environments of Super Apps through autonomous data synthesis and dual-agent co-evolution. By effectively mitigating data scarcity and exploration inefficiencies, GUI0 achieves state-of-the-art performance on the proposed SuperAPP benchmark, significantly surpassing existing proprietary models. Ultimately, this work establishes a robust paradigm for bridging foundational knowledge with execution logic, advancing the development of generalist agents for complex digital ecosystems.

Limitations

GUI0 introduces a novel foundational dual-agent co-evolution framework for mastering dynamic GUI interactions. However, it is important to recognize the limitations of our current approach.

Firstly, our reliance on a single VLM judge (Qwen2.5-VL) for both reward modeling and success evaluation creates a risk of circular reasoning. While this enables scalable training, the actor may over-optimize to the judge's inherent biases rather than achieving ground-truth utility, highlighting the need for future work to incorporate diverse judge ensembles and human-in-the-loop calibration to prevent reward hacking.

Secondly, ensuring strict fairness in benchmarking against closed-source models (e.g., GPT-4o, Gemini-2.5-Pro) remains challenging. Differences in action schemas, tool wrappers, and allowable retry budgets mean that comparisons may reflect engineering inconsistencies rather than pure model capability. Furthermore, observed risks of negative transfer (e.g., on AndroidWorld) suggest a need for better regularization during domain specialization.

Finally, we acknowledge that while we implement multi-stage filtering for sensitive content, more granular privacy-preserving techniques and robust data synthesis pipelines are required for broader real-world application. The additional overhead of co-evolution mainly occurs at the instruction generation level which is token-light and performed at a lower frequency than actor rollouts, in wall-clock terms this cost is typically smaller than actor-environment interaction costs.

Ethical considerations

Our research strictly adheres to ethical guidelines regarding data privacy, model and tool usage, and labor standards. The goal of this work is to improve the reliability of agentic behaviors in super-app scenarios, rather than to bypass platform safeguards or enable harmful automation.

Data Privacy and Protection. We ensure that data collection complies with WeChat's privacy and compliance requirements, including user consent, data minimization, secure storage, and anonymization. In constructing the Agentic CPT corpus and the SuperAPP benchmark, we implemented a rigorous multi-stage privacy protection protocol. First, data collection was conducted using dedicated research test accounts in a sandboxed emulator environment, ensuring that we do not access, collect, or use real-user data, real social graphs, or any real financial assets. Second, prior to storage, all raw screenshots and view hierarchies underwent automated PII (Personally Identifiable Information) decontamination, using OCR and object detection to identify and redact sensitive entities such as real names, phone numbers, and addresses. Finally, the "Sensitivity" filtering step (Sec. 2) combined VLM-based auditing with human verification to discard any trajectories containing residual private, harmful, or otherwise inappropriate content. Furthermore, we do not generate, store, or infer sensitive personal attributes (e.g., health status, political views), nor do we target or profile any specific demographic groups.

Model, Tool Usage, and Safety. We utilize open-source models (e.g., Qwen2.5-VL) and commercial APIs (e.g., Gemini-2.5-Pro) for data synthesis and evaluation, strictly complying with their respective licenses and terms of use. We also adhere to the licenses associated with all datasets, benchmark platforms, and tool environments used, ensuring compliance with applicable platform developer policies and automation testing guidelines. Recognizing that VLM-based agents may exhibit hallucinations or unpredictable behaviors, all training and evaluation are confined to isolated virtual environments to prevent unintended real-world consequences. Inter-

actions with external tools are limited to simulated environments or publicly documented APIs with appropriate usage permissions; we do not perform unauthorized access, circumvent access controls, or weaken platform safety mechanisms.

Human Annotation and Labor Standards. The human verification process involved professional annotators who provided informed consent regarding the nature of the data and tasks. All annotators were compensated fairly, exceeding local minimum wage standards. We employed clear task protocols and quality control mechanisms to reduce unnecessary exposure to potentially sensitive content.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2025a. Octopus: On-device language model for function calling of software apis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 329–339.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2025b. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, and 1 others. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*.

Jihao Gu, Qihang Ai, Yingyao Wang, Pi Bu, Jingxuan Xing, Zekun Zhu, Wei Jiang, Ziming Wang, Yingxiu Zhao, Ming-Liang Zhang, and 1 others. 2025a. Mobile-r1: Towards interactive reinforcement learning for vlm-based mobile agent via task-level rewards. *arXiv preprint arXiv:2506.20332*.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, and 1 others. 2025b. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. 2025. Appagentx: Evolving gui agents as proficient smartphone users. *arXiv preprint arXiv:2503.02268*.

Hongxin Li, Jingran Su, Jingfan Chen, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025a. Uipro: Unleashing superior interaction capability for gui agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1613–1623.

Jiahao Li and Kaer Huang. 2025. A survey on gui agents with foundation models enhanced by reinforcement learning. *arXiv preprint arXiv:2504.20464*.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025b. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8778–8786.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024a. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Wei Li, Borui Yang, Hangyu Ye, Liyao Xiang, Qingxiao Tao, Xinbing Wang, and Chenghu Zhou. 2024b. Minitracker: Large-scale sensitive information tracking in mini apps. *IEEE Transactions on Dependable and Secure Computing*, 21(4):2099–2114.

Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024c. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508.

Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. 2025. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning. *arXiv preprint arXiv:2511.19900*.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*.

Songqin Nong, Jingxuan Xu, Sheng Zhou, Jianfeng Chen, Xiaoxuan Tang, Tao Jiang, and Wenhao Xu. 2025. Craft-gui: Curriculum-reinforced agent for gui tasks. *arXiv preprint arXiv:2508.11360*.

Jupinder Parmar, Sanjev Satheesh, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv preprint arXiv:2407.07263*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Yizhe Shi, Guangliang Yang, Zhemin Yang, Yifan Yang, Min Yang, Kangwei Zhong, and Xiaohan Zhang. 2025. The skeleton keys: A large scale analysis of credential leakage in mini-apps.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, and 1 others. 2025. Scaling agents via continual pre-training. *arXiv preprint arXiv:2509.13310*.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. 2025. Seagent: Self-evolving computer use agent with autonomous learning from experience. *arXiv preprint arXiv:2508.04700*.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 69 others. 2025. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Shenao Wang, Yuekang Li, Kailong Wang, Yi Liu, Hui Li, Yang Liu, and Haoyu Wang. 2025. Miniscope: Automated ui exploration and privacy inconsistency detection of miniapps via two-phase iterative hybrid analysis. *ACM Transactions on Software Engineering and Methodology*, 34(6):1–29.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, and 1 others. 2024. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Liujian Liujianfeng, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10231–10251.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, and 1 others. 2025. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. 2025. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, and 1 others. 2025. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Haolong Yan, Yeqing Shen, Xin Huang, Jia Wang, Kaijun Tan, Zhixuan Liang, Hongxin Li, Zheng Ge, Osamu Yoshie, Si Li, and 1 others. 2025. Gui exploration lab: Enhancing screen navigation in agents via multi-turn reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Wenwen Yu, Zhibo Yang, Jianqiang Wan, Sibo Song, Jun Tang, Wenqing Cheng, Yuliang Liu, and Xiang Bai. 2025. Omniparser v2: Structured-points-of-thought for unified visual text parsing and its generality to multimodal large language models. *arXiv preprint arXiv:2502.16161*.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and 1 others. 2025. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, and 1 others. 2024. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025a. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, Jie Xie, Wei Zhou, Wang Xu, Yuanheng Zhang, Zhou Su, Zhongwu Zhai, Xiaoming Liu, Yudong Mei, Jianming Xu, and 6 others. 2025b. AgentCPM-GUI: Building mobile-use agents with reinforcement fine-tuning. *arXiv preprint arXiv:2506.01391*.

Ziyun Zhang, Xinyi Liu, Xiaoyi Zhang, Jun Wang, Gang Chen, and Yan Lu. 2025c. Ui-evol: Automatic knowledge evolving for computer use agents. *arXiv preprint arXiv:2505.21964*.

Yuchen Zhuang, Jingfeng Yang, Haoming Jiang, Xin Liu, Kewei Cheng, Sanket Lokegaonkar, Yifan Gao, Qing Ping, Tianyi Liu, Binxuan Huang, and 1 others. 2025. Hephaestus: Improving fundamental agent capabilities of large language models through continual pre-training. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6041–6068.

A Action Space Definition

As discussed in Sec. 2, we define a unified action space A consisting of 10 atomic operations tailored for mobile GUI interaction. To facilitate precise agent control, each action primitive is associated with specific parameter constraints. Tab. 7 details the definitions and semantic descriptions of these operations.

Table 7: Definition of the Unified Action Space.

Action Primitive	Parameters	Semantics and Constraints
`click`	x, y	Tap at the specific pixel coordinates (x, y).
`long_press`	x, y	Perform a long-press gesture at coordinates (x, y).
`input`	x, y, text	Focus on the input field at (x, y) and enter the specified string `text`.
`scroll`	direction	Scroll the screen. The parameter is strictly restricted to $\{$`up`, `down`, `left`, `right`$\}$.
`open_app`	name	Launch a specific application or mini-app identified by `name`.
`wait`	t	Pause execution for t seconds (e.g., awaiting page load).
`navigate_home`	*None*	Return to the system home screen.
`navigate_back`	*None*	Return to the previous page or state.
`finish`	*None*	Terminate the session, marking the task as successfully completed.
`call_user`	*None*	Request human intervention for tasks exceeding autonomous capabilities.

B Data Synthesis and Statistics

B.1 VLM-based Quality Filtering

To guarantee the high fidelity of synthesized instructions and trajectories, we employed a Visual Language Model (VLM) to evaluate data samples across five distinct dimensions. We established rigorous thresholds to filter out low-quality instances. Tab. 8 summarizes the evaluation metrics, the average scores of our retained dataset, and the specific filtering criteria applied.

Table 8: VLM Evaluation Dimensions, Average Scores, and Filtering Thresholds.

Dimension	Description	Avg. Score	Threshold
Redundancy	Repetition in instructions or actions	1.9	≤ 1.0
Sensitivity	Presence of PII or sensitive content	1.2	< 5.0
Feasibility	Executability of the instruction	4.2	≥ 3.0
Logic Consistency	Causal link between observation & action	3.8	> 3.0
Goal Clarity	Ambiguity of the task description	4.8	> 3.0

B.2 Continual Pre-training (CPT) Dataset

The dataset constructed for the Agentic CPT stage comprises approximately 2.3 million samples. These are categorized into five types designed to build robust cognitive foundations for GUI agents. The detailed distribution is provided in Tab. 9.

C Reward Function Formulation

In the Single-step GRPO phase, the total reward R_{step} is formulated as a composite of format com-

Table 9: Composition of the Agentic CPT Dataset.

Data Category	Sample Count	Percentage
GUI State Transitions	1,218,241	52.1%
Visual Question Answering (VQA)	636,413	27.2%
Detailed Screen Description	212,763	9.1%
Reasoning	170,289	7.3%
Trajectory Chain-of-Thought (CoT)	102,935	4.4%
Total	**2,340,641**	**100.0%**

pliance, logical alignment, semantic accuracy, and execution precision.

C.1 Composite Reward Structure

The total step reward is defined as:

$$R_{\text{step}} = R_{\text{fmt}} + R_{\text{align}} + R_{\text{subgoal}} + R_{\text{action}} \quad (5)$$

The components are calculated as follows:

- **Format Reward (R_{fmt})**: A binary indicator that equals 1.0 if the output strictly matches the regex pattern, and 0.0 otherwise.
- **Alignment Reward (R_{align})**: Determined by the evaluator prompt (Appendix E). If the VLM outputs "Yes", $R_{\text{align}} = 1.0$, otherwise 0.0.
- **Sub-goal Reward (R_{subgoal})**: Calculated as the Character-level F1 Score between the generated sub-goal text and the ground truth sub-goal text.

C.2 Action Precision Reward (R_{action})

We quantify the precision of the predicted action a_{pred} against the ground truth a_{gt}. A base reward $r_{\text{base}} = 0.2$ is awarded if the action types match ($\text{type}_{\text{pred}} = \text{type}_{\text{gt}}$). The remaining reward is contingent on the action category:

1. Coordinate-based Actions For actions involving spatial interaction (`click`, `long_press`), we compute the pixel deviation Δ for each dimension (x, y). The axis-specific reward $r_{\text{axis}}(\Delta)$ follows a piecewise decay function:

$$r_{\text{axis}}(\Delta) = \begin{cases} 0.4 & \text{if } \Delta \leq 5 \\ 0.1 \cdot \frac{1}{\Delta - 4} & \text{if } 5 < \Delta \leq 100 \\ 0 & \text{if } \Delta > 100 \end{cases} \quad (6)$$

The total action reward is $R_{\text{action}} = r_{\text{base}} + r_{\text{axis}}(\Delta_x) + r_{\text{axis}}(\Delta_y)$. The maximum possible reward is $0.2 + 0.4 + 0.4 = 1.0$.

2. Text Input Actions For `input` operations, precision is evaluated on both spatial accuracy and textual content:

- **Coordinates:** Calculated similarly to clicks, but with a maximum of 0.25 per axis. If $\Delta \leq 5$, the reward is 0.25; otherwise, it follows the same decay function $0.1 \cdot \frac{1}{\Delta - 4}$ up to 100 pixels.
- **Content:** A content bonus $r_{\text{content}} = 0.3$ is added if the predicted text matches the ground truth (case-insensitive, ignoring whitespace).

Maximum reward: $0.2(\text{base}) + 0.25(\text{x}) + 0.25(\text{y}) + 0.3(\text{content}) = 1.0$.

3. Discrete Actions For actions with discrete parameters, the reward logic is:

- `scroll`: $+0.8$ if the direction string (up, down, left, right) strictly matches.
- `open_app`: $+0.8 \times \text{F1}(\text{name}_{\text{pred}}, \text{name}_{\text{gt}})$, where F1 is the character-level F1 score of the app name.
- `wait`, `navigate_home`, `navigate_back`, `finish`: $+0.8$ directly upon type match.

In all discrete cases, the total reward sums to 1.0 (0.2 base + 0.8 specific).

D Training Algorithm Details

To provide a comprehensive view of our training pipeline, Alg. 1 details the execution flow of the Dual-Agent Co-Evolution strategy. This process establishes a symbiotic learning loop: the Actor Agent focuses on mastering GUI interactions through reinforcement learning, while the Curriculum Agent evolves to synthesize progressively challenging instructions tailored to the Actor's current capability frontier. By iteratively updating the instruction pool based on the Actor's success rates, the system ensures a dynamic balance between task difficulty and agent ability.

E Prompt Templates

E.1 System Prompt for Actor Agent

The Actor Agent utilizes the following system prompt to generate structured reasoning paths and executable actions.

Algorithm 1 GUI0 Dual-Agent Co-Evolution Training

Require: Initial Actor π_ϕ, Initial Curriculum Agent π_ψ, Seed Instructions D_{seed}

Initialization: Pre-train π_ϕ (Stage 1 & 2) and π_ψ (Stage 1).
Set current instruction pool $D_{\text{curr}} \leftarrow D_{\text{seed}}$
for round $k = 1$ to K **do**
  // Step 1: Actor Rollout & Evaluation
  Collect trajectories $T_k = \{(x, \tau_i)\}_{i=1}^{G}$ for $x \in D_{\text{curr}}$ using π_ϕ
  $S_x = \frac{1}{G}\sum_{i=1}^{G} R_{\text{success}}(\tau_i)$ for each x
  // Step 2: Policy Optimization
  Update Actor π_ϕ via GRPO on T_k to maximize $R(\tau)$ (Eq. 3)
  Compute curriculum reward $R_{\text{curric}}(x)$ using S_x (Eq. 4)
  Update Curriculum Agent π_ψ via GRPO to maximize R_{curric}
  // Step 3: Curriculum Generation for Next Round
  Sample contexts and generate new instructions: $D_{\text{new}} \sim \pi_\psi(\cdot)$
  Filter D_{new} for validity and merge: $D_{\text{curr}} \leftarrow D_{\text{curr}} \cup D_{\text{new}}$
end for
return Optimized Actor π_ϕ^*

Actor Agent System Prompt

You are a top-tier GUI operation assistant. Your task is to generate a rigorous reasoning process and plan the most reasonable next step based on the given high-level instruction, analyzing screenshots and operation history.

Input Data: High-level Instruction: {instruction} History Summary: {history} Image: A sequence of operation screenshots; the latest one is the current state.

Output Format: Strictly follow the format:
<think>...</think>
<sub_goal>...</sub_goal>
<answer>...</answer>.
<think>: Place your reasoning process here.
<sub_goal>: Plan the specific sub-goal for this step based on the reasoning. It must be simple, clear, and executable. Do not be ambiguous, do not add extra intentions, and do not include extra explanations.
<answer>: The operation actions can only be selected from the following 10 categories:

1. `click x y` – tap at pixel (x, y)
2. `input x y t` – focus at (x, y) and enter text t
3. `finish` – task completed successfully
4. `scroll direction` – scroll up/down/left/right
5. `wait t` – pause for t seconds
6. `long_press x y` – long-press at (x, y)
7. `open_app name` – launch the named app
8. `navigate_home` – return to home screen
9. `navigate_back` – return to previous screen
10. `call_user` – request human intervention

E.2 Prompt for Alignment Evaluator

We employ a VLM (Qwen2.5-VL-72B) as a reward model to assess the logical alignment between the reasoning process and the sub-goal.

Alignment Judge Prompt

You are a Mini-Program operation consistency expert. Your task is to: Strictly evaluate whether the "Think Process" explicitly, reasonably, and logically derives the given "Sub-goal". Alignment is established only if the thinking process clearly points to and supports the generation of the sub-goal.

Judgment Criteria: 1. Clarity: The thinking process must contain key judgments or reasoning sufficient to derive the sub-goal, without being vague, general, or jumping to conclusions. 2. Logic: There should be a clear causal or procedural derivation relationship from the thinking process to the sub-goal, without disconnection. 3. Consistency: The action and target object in the sub-goal must be completely consistent with the analysis results in the thinking process. 4. Necessity: The thinking process should explain "why" the sub-goal is executed, rather than just describing the action itself.

Input Data: Page screenshot (for context understanding, but judgment is primarily text-based) Think Process: {content_think} Sub-goal: {content_summary}

Output Requirements: Strictly output in the following format without adding extra content: <result>Yes</result> or <result>No</result>

Note: If the thinking process does not explicitly explain the source of the sub-goal or has logical gaps, even if the intention is similar, it should be judged as "No". Do not judge as aligned solely based on semantic similarity; the focus is on "whether it is derived".