TUTORIALGUI: TOWARDS GENERALIZABLE GUI AGENTS VIA TUTORIAL-DRIVEN SKILL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI agents have made rapid progress in operating general-purpose software, yet they struggle to generalize to unfamiliar specialized applications due to limited application-specific knowledge and operational experience. Collecting data and fine-tuning agents for each application is costly and difficult to scale to long-tail applications. Official tutorials provide a readily available source of domain knowledge and procedural guidance, but they are human-oriented and cover only a subset of application capabilities. To address these limitations, we introduce **TutorialGUI**, a training-free framework that transforms official tutorials into executable, reusable skills for GUI agents through an iterative **Replay–Exploration** loop. Starting from tutorial-derived candidate skills, Replay completes and refines their procedures through real GUI execution and verifies whether they achieve the intended effects. Exploration builds on verified anchor skills to discover new candidate skills via three exploration directions: forward, backward, and lateral. The Replay-Exploration loop can run multiple iterations, progressively extending skills beyond the original tutorials. The resulting skills and associated Knowledge Cards can be retrieved and reused by different GUI agents without updating the model parameters. On six ScienceBoard applications, TutorialGUI improves task success by an average of 13.42 percentage points across three executor models over their corresponding baselines. These results support tutorial-driven skill learning as an effective approach to improving GUI agents' generalization to specialized applications without per-application fine-tuning.

1 INTRODUCTION

GUI agents have made rapid progress in interacting with general-purpose applications. Native models such as UI-TARS (Qin et al., 2025) improve perception, reasoning, and action generation through large-scale interaction data and iterative training, while benchmarks such as OSWorld (Xie et al., 2024) support evaluation in real desktop environments. Despite this progress, unfamiliar specialized applications remain challenging: agents may lack the application-specific knowledge needed to select tools, configure parameters, and follow specialized workflows (Xie et al., 2026; Sun et al., 2026). Application-specific training can help address this gap (Sun et al., 2025a), but repeatedly collecting data and updating models introduces adaptation and maintenance costs, limiting scalability to long-tail applications. This motivates complementary approaches that acquire application knowledge and operational experience without per-application fine-tuning.

Scientific software provides a concrete setting for studying this challenge. The tasks in these applications require understanding domain-specific objects, parameter semantics, and operational preconditions while executing multi-step procedures with strict dependencies (Sun et al., 2026). Successful execution therefore requires both conceptual understanding and procedural knowledge: agents must understand what application-specific concepts mean, when particular operations are appropriate, and how to carry them out in the correct sequence. These complementary forms of knowledge are important for reliable interaction with unfamiliar specialized applications.

Official tutorials offer a readily available source of such knowledge through functional explanations and procedural examples. Existing work has explored synthesizing training trajectories from web tutorials (Xu et al., 2025) and extracting video demonstrations for inference-time guidance (Liu et al., 2025). Turning tutorials into reusable skills poses two challenges: executability and coverage.

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

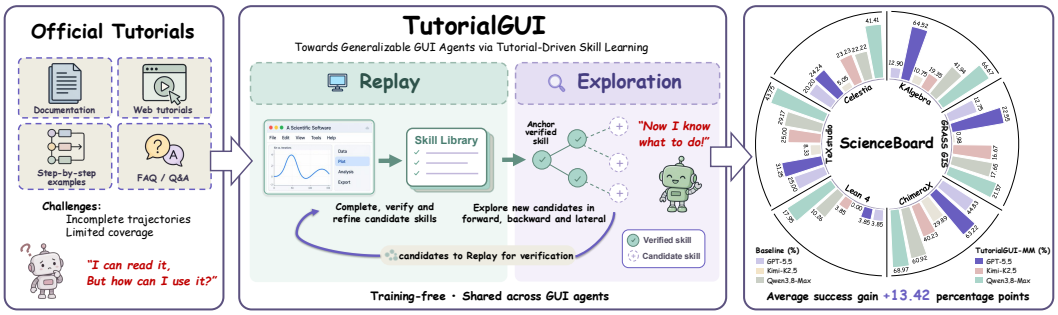


Figure 1: TutorialGUI turns official tutorials into reusable skills through Replay and Exploration, improving GUI agents' performance on scientific software without parameter updates.

First, **ensuring executability** requires more than extracting a sequence of steps. Tutorials written for human users may omit intermediate actions or assume prerequisite states; the extracted procedures therefore need to be completed and verified in the actual application. Second, **extending coverage** requires discovering operations beyond those demonstrated in tutorials. Since tutorials typically cover selected workflows, extraction alone leaves related operations and alternative procedures unexplored.

To address these challenges, we introduce **TutorialGUI**, a training-free framework that organizes tutorial-derived procedures and application knowledge into a shared external library. It extracts executable and extended skills describing operational procedures and Knowledge Cards providing static knowledge, such as tool functionality and parameter semantics. At the core of skill acquisition is a **Replay–Exploration loop** that improves skill executability and expands skill coverage. Replay uses real GUI execution to complete missing steps, refine candidate procedures, and verify their intended effects. Exploration uses verified skills as anchors to explore new candidate skills in forward, backward and lateral directions. The Replay-Exploration loop can run multiple iterations, progressively extending skills beyond the original tutorials. During task execution, different GUI agents retrieve relevant skills and Knowledge Cards as contextual augmentation, without parameter updates.

On scientific software tasks in ScienceBoard (Sun et al., 2026), TutorialGUI improves task success rates by an average of **13.42 percentage points** across multiple models over baselines without tutorials. These results demonstrate the effectiveness of the overall framework in supporting scientific software use through tutorial-driven knowledge and skill acquisition without model fine-tuning.

Our main contributions are as follows:

- **We introduce TutorialGUI**, a training-free framework that transforms official tutorials into reusable operational skills and Knowledge Cards capturing non-procedural application knowledge. These resources form a shared external library reusable across GUI agents to improve generalization to unfamiliar specialized applications without per-application fine-tuning.

- **We propose a Replay-Exploration loop to improve skill executability and expand skill coverage.** Replay completes, refines, and verifies candidate procedures through real GUI execution, while Exploration builds on verified skills to explore in forward, backward and lateral directions to discover new skills beyond the original tutorials.

- **We evaluated TutorialGUI across multiple open- and closed-source models on ScienceBoard**, demonstrating an average improvement of 13.42 percentage points in task success rates over baselines without tutorials.

2 Related Work

2.1 GUI Agents

GUI agents aim to accomplish user goals by interpreting interface observations and executing sequences of actions. Benchmarks including OSWorld (Xie et al., 2024; Yuan et al., 2026), Windows Agent Arena (Bonatti et al., 2024), and MacArena (Muryn et al., 2026) evaluate this ability in real desktop environments, while models such as UI-TARS (Qin et al., 2025; Wang et al., 2025) and Mobile-Agents (Xu et al., 2026) advance GUI perception and action generation. Successful task execution requires these capabilities to be coordinated across a workflow: agents must select appropriate operations, track changes in application state, and determine how intermediate results support subsequent steps.

2.2 GUI Agents for Scientific Software

Operating scientific software requires agents to understand domain-specific tools, parameter semantics, and workflow dependencies. ScienceBoard (Sun et al., 2026) provides a benchmark for evaluating agents on such scientific workflows. Training-based approaches acquire application-specific capabilities from interaction data (Fan et al., 2025; Liu et al., 2026a;b). For example, SEAgent (Sun et al., 2025b) combines autonomous exploration with reinforcement learning, while CODA (Sun et al., 2025a) trains scientific-software planners through reinforcement learning and supervised fine-tuning. TutorialGUI instead uses official tutorials and GUI interaction to build a reusable external library of application knowledge and operational skills, keeping model parameters fixed.

2.3 Learning from Tutorials for GUI Agents

Software tutorials provide procedural knowledge and demonstrations that GUI agents can learn from. One line of work converts these resources into training supervision. AgentTrek (Xu et al., 2025) extracts tasks and instructions from web tutorials and generates training trajectories through guided replay in real environments. Subsequent work scales tutorial-based supervision by identifying interaction steps and aligning them with screen observations to construct large-scale training corpora across diverse applications (Zhang et al., 2026; Lu et al., 2026; Xiong et al., 2026).

Another line of work uses tutorial-derived knowledge as external context during task execution. GUIDE (Xie et al., 2026) extracts planning and grounding knowledge from tutorial videos, while other methods convert videos into demonstration trajectories that agents consult during execution (Liu et al., 2025; Song et al., 2026). WebLearner (Long et al., 2026) further trains agents to retrieve and apply multimodal tutorials. More closely related work organizes these resources into reusable knowledge and skills. MMG2Skill (Che et al., 2026) refines guide-derived skills through task-rollout feedback, VisualSkill (Jiang et al., 2026) enriches multimodal guides from official documentation through UI exploration, and Resource2Skill (Fan et al., 2026) distills multimodal resources into skill libraries. Distinct from the above lines of research, TutorialGUI combines per-skill Replay with exploration in real GUI environments to address incomplete procedures and limited tutorial coverage. Replay refines candidate procedures and verifies their intended effects before library admission, while Exploration discovers additional skills beyond those directly extracted from tutorials.

3 Method

TutorialGUI extracts candidate skills and Knowledge Cards from official tutorials. Its Replay–Exploration loop refines and verifies candidate procedures through GUI interaction, then uses verified skills in the Skill Graph to discover further candidates. The resulting library provides retrievable context for downstream agents. Figure 2 provides an overview.

3.1 Tutorial-Based Skill and Knowledge Extraction

Given an application's official tutorials, a vision-language model (VLM) reads their text and images subsection by subsection. It extracts operational procedures as candidate skills and complementary knowledge as Knowledge Cards, organizing the candidate skills into a graph.

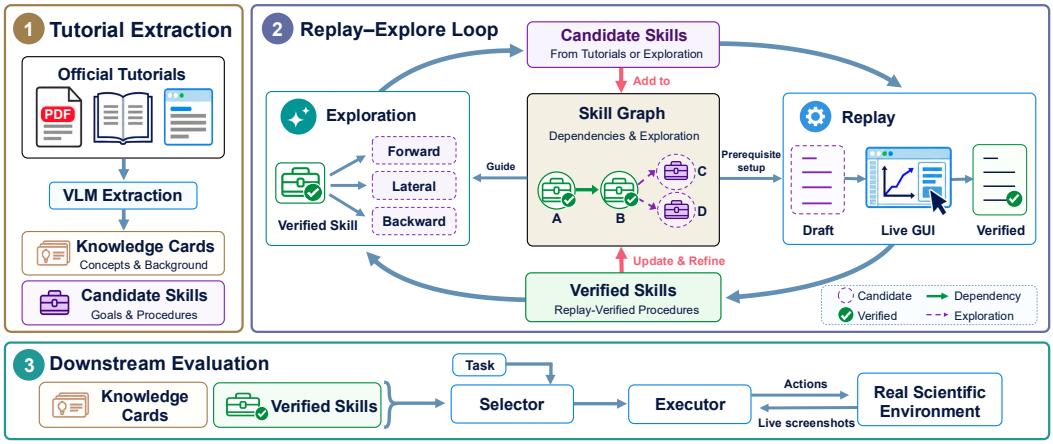


Figure 2: Overview of TutorialGUI. Official tutorials provide candidate skills and Knowledge Cards. Replay refines and verifies candidates through GUI execution, while Exploration proposes related candidates from verified skills. The resulting resources are frozen before evaluation and selected as context for downstream agents without parameter updates.

Candidate skills. The skills defined in this work are anchored to an operational goal. Operational procedures for different goals are extracted as distinct skills. We formulate a skill as $s = \langle d, P, \Theta, A, Q \rangle$, where d describes the goal, P specifies preconditions that may be established by other skills, Θ defines the parameter schema for reuse with different inputs, A describes the operational procedure, and Q specifies expected effects. For example, a KAlgebra plotting skill can bind its expression parameter to $\ln(x)$ or $\sin(2x)$, reusing the procedure for different inputs. We prompt a VLM to extract these fields from tutorials, yielding an initial candidate set $\mathcal{S}_0 = \{s_i\}$. Replay further refines these candidates to resolve omitted operational details and ambiguous preconditions.

Skill Graph. We organize relationships among skills in a continuously updated Skill Graph. Nodes represent individual skills, and edges capture two types of relationships: execution dependencies and exploration relations. Execution dependency edges are determined by skill preconditions and guide the execution order during Replay. Exploration edges link newly discovered candidates to their anchor skills, recording how the library expands through Exploration.

Knowledge Cards. Beyond the operational procedures for each function, a GUI agent requires essential background knowledge to operate the application correctly. We therefore extract non-executable knowledge records from tutorials to form Knowledge Cards, which contain software-related concepts, functional explanations, applicability conditions, and interface knowledge. Knowledge Cards can also be retrieved to augment the context of the GUI agent, and remain fixed after extraction.

3.2 Replay: Execution, Verification, and Refinement

Replay establishes whether a candidate procedure can achieve its intended effect and uses the resulting execution evidence to refine it. Guided by dependency relations in the Skill Graph, Replay selects a candidate skill $s_i = \langle d_i, P_i, \Theta_i, A_i, Q_i \rangle$. A dependency identifies a procedure for establishing a required precondition; it does not imply that the current GUI state already satisfies that precondition.

The Execution Agent first checks P_i against the current application state in the environment $\mathcal{E}$ and performs preparation operations when necessary. It instantiates the parameters in Θ_i using the replay inputs and current application state, yielding bindings b_i, then executes A_i once the required preconditions and parameter values are established.

Replay returns the bindings b_i and execution trace τ_i, including preparation, actions, screenshots, and any executor-supplied file probes; a VLM-based postcondition verifier V checks the declared effects under these bindings against the execution evidence:

$$(b_i, \tau_i) = \text{Replay}(s_i, \mathcal{E}), \tag{1}$$

$$v_i = \mathbf{1}[V(Q_i, b_i, \tau_i) = \texttt{verified}] \in \{0, 1\}. \tag{2}$$

The verifier V returns `verified`, `failed`, or `inconclusive`; the indicator is one only when all required postconditions pass. For admission, (b_i, τ_i) refers to the first verified attempt, or the final unsuccessful attempt if the Replay budget is exhausted. Verification applies only to the observed instance.

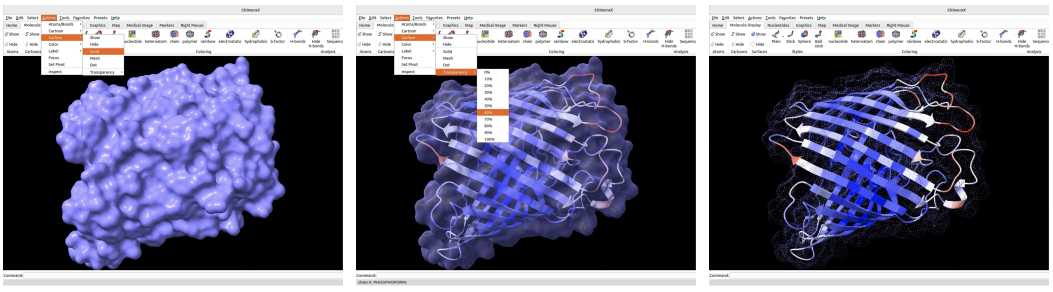
After successful verification, refinement records the procedure $A_i^\star$ and preconditions $P_i^\star$ supported by that trajectory as $s_i^\star = \langle d_i, P_i^\star, \Theta_i, A_i^\star, Q_i \rangle$. Direct Replay of tutorial-derived candidates yields $\mathcal{S}_0^\star = \{s_i^\star \mid s_i \in \mathcal{S}_0,\ v_i = 1\}$. Candidates with no verified attempt within the Replay budget are not admitted and receive no further Replay attempts in subsequent rounds.

3.3 Exploration: Iterative Skill Expansion

Another challenge in extracting skills from tutorials is incomplete coverage: tutorials typically cover only selected workflows, leaving many functions and procedures uncovered. Accordingly, we propose the Exploration module to discover skills beyond the scope of tutorials. Starting from a verified anchor skill $s_i^\star$, Exploration considers three directions:

- Forward exploration identifies downstream operations that consume an object produced by the anchor or a state established by it.
- Backward exploration identifies procedures that establish the anchor's required inputs or preconditions, including missing or alternative preparation paths.
- Lateral exploration identifies operations with shared inputs, related goals, or neighboring functions, including alternative modes or configurations within the same skill family, without implying an execution dependency.

Figure 3 illustrates these relations in ChimeraX: starting from an anchor that sets the surface style to `Solid`, forward exploration considers adjusting the surface transparency, while lateral exploration considers the alternative `Dot` surface style.



(a) Anchor
Set surface style to Solid

(b) Forward
Adjust surface transparency

(c) Lateral
Set surface style to Dot

Figure 3: Exploration from a ChimeraX surface-style anchor: (a) the anchor sets the style to `Solid`; (b) forward exploration considers adjusting surface transparency, with available settings shown; (c) lateral exploration considers the alternative `Dot` style.

When selecting anchor skills for exploration, verified skills that have not yet been used as exploration anchors are prioritized. Exploration proposes candidates from these anchors, application observations, and relevant tutorial or manual evidence. Candidates remain unverified until Replay (Appendix A.4.6).

For round $t \geq 1$, let $\mathcal{S}^{\star}_{t-1}$ and $\mathcal{S}^{\star}_{t}$ denote the verified skill sets at the start and end of the round, respectively, and let $\mathcal{G}$ denote the current Skill Graph. Candidate generation and the round-level verified-set update are summarized as

$$\mathcal{U}_t = \mathrm{Explore}(\mathcal{G}, \mathcal{E}), \tag{3}$$

$$\mathcal{S}^{\star}_{t} = \mathcal{S}^{\star}_{t-1} \cup \{ s^{\star}_i \mid s_i \in \mathcal{U}_t,\ v_i = 1 \}. \tag{4}$$

Here, Explore summarizes successive candidate-generation calls interleaved with Replay, and $\mathcal{U}_t$ collects all proposals made during the round. Only candidates that pass verification are refined and admitted as new records. The graph is updated upon admission, making newly verified skills available for subsequent exploration within the same round. Previously verified skills are retained unchanged and are not replayed as part of this update. Each exploration round stops when a preset target number of newly verified skills has been admitted.

3.4 Skill-Augmented Task Execution

After resource construction, the resulting skills augment downstream GUI agents.

Selector. Given the task instruction, the Selector first examines skill summaries and Knowledge Card topics and summaries to identify applicable resources. It selects at most two skills and one Knowledge Card, which are provided to the Executor as task context. If no resources are suitable, the Selector abstains. The Executor binds skill parameters using the task instruction and current GUI observations.

Executor. We use a standard ReAct agent loop for task execution. Each evaluated VLM serves as the Executor, using the selected skills and Knowledge Cards as additional context to guide GUI actions.

Text and Multimodal Skills. TutorialGUI supports text and multimodal variants. For Replay-verified skills, the multimodal variant supplements the textual record with one final-state image from the verified trajectory to limit token overhead. Unverified candidates use original tutorial images when available.

4 Experiments

We evaluate TutorialGUI on ScienceBoard (Sun et al., 2026) to answer three questions: (1) Can TutorialGUI resources improve downstream GUI task performance across different executor models? (2) How does downstream performance change across resource-construction stages? (3) How does continued exploration affect task performance as the skill library grows?

4.1 Experimental Setup

Benchmark. We evaluate TutorialGUI on six scientific domains from ScienceBoard (Sun et al., 2026), each instantiated by a corresponding software application: Algebra (KAlgebra), Biochemistry (ChimeraX), Geographic Information Systems (GRASS GIS), Automated Theorem Proving (Lean 4 within VS Code), Astronomy (Celestia), and Scientific Documentation (TeXstudio). We use the official ScienceBoard task sets (Sun et al., 2026), comprising 169 tasks across these applications; per-application task counts are listed in Table 5. The applications are deployed in AgentBay sandboxes[1] running Ubuntu 22.04 at a fixed resolution of 1920×1080, following the ScienceBoard software configurations. Replay settings and downstream evaluation details are provided in Appendix A.1.

Baselines. We compare TutorialGUI against two categories of baselines: (1) *Published trained GUI agents*, including CODA Stage-2 (Sun et al., 2025a), SEAgent with UI-TARS-1.5-7B-DPO (Sun et al., 2025b), and ScaleCUA with Qwen3.5-9B (Lv et al., 2026); and (2) *Foundation-model baselines*, including GPT-5.5 (OpenAI, 2026), Kimi-K2.5 (Team et al., 2026), and Qwen3.8-Max (Qwen Team, 2026), which execute tasks without access to TutorialGUI Knowledge Cards or skills. We rerun all three foundation-model baselines using the same infrastructure and evaluation protocol as TutorialGUI. The reproduced GPT-5.5 and Kimi-K2.5 baselines achieve slightly higher average

[1] https://www.alibabacloud.com/help/en/agentbay

success rates than those reported in ScienceBoard. Detailed comparison protocols are provided in Appendix A.3.

Evaluation Metric. We determine task success using the official ScienceBoard evaluators (Sun et al., 2026) and report the average success rate over three runs for results evaluated by us.[2] Appendix A.5 reports the means and sample standard deviations over the three runs. Performance gains are reported in percentage points (pp).

TutorialGUI configurations. All three foundation models share a library of 1,715 Knowledge Cards and 569 verified skills, frozen after direct Replay and one Exploration round per application. Each model serves as both Selector and Executor. TutorialGUI-Text supplies selected Cards and skills as text; TutorialGUI-MM additionally supplies one reference image per selected skill. Both receive live GUI screenshots, and all model parameters remain fixed. Model roles and per-application library counts are detailed in Appendix A.1.

4.2 Main Results

Table 1 presents the main results of TutorialGUI across three foundation models and six scientific domains, together with published results from trained GUI agents. We highlight three key findings.

Table 1: Task success rates (%) on ScienceBoard. Δ is the gain over the corresponding Baseline in percentage points. †Published results.

Method	Alg	Biochem	GIS	ATP	Astron	Doc	Avg	Δ
Published trained GUI agents								
CODA Stage-2†	20.16	32.23	14.71	—	17.05	—	21.04	—
SEAgent (UI-TARS-1.5-7B-DPO)†	21.29	23.45	10.59	—	11.52	—	16.71	—
ScaleCUA (Qwen3.5-9B)†	67.7	62.1	35.3	19.0	60.6	86.7	55.23	—
GPT-5.5								
Baseline	12.90	44.83	12.75	3.85	20.20	25.00	19.92	—
TutorialGUI-Text	63.44	60.92	17.65	3.85	29.29	27.08	33.70	+13.78
TutorialGUI-MM	64.52	63.22	22.55	3.85	24.24	31.25	**34.94**	+15.02
Kimi-K2.5								
Baseline	10.75	29.89	0.98	0.00	5.05	8.33	9.17	—
TutorialGUI-Text	19.35	35.63	15.69	6.41	15.15	27.08	19.89	+10.72
TutorialGUI-MM	19.35	40.23	16.67	3.85	23.23	25.00	**21.39**	+12.22
Qwen3.8-Max								
Baseline	41.94	60.92	17.65	10.26	22.22	29.17	30.36	—
TutorialGUI-Text	59.14	67.82	20.59	14.10	40.40	39.58	40.27	+9.91
TutorialGUI-MM	66.67	68.97	21.57	17.95	41.41	43.75	**43.39**	+13.03

(1) Consistent Gains Across Foundation Models. TutorialGUI consistently improves downstream task performance across different foundation models. With textual resources alone, TutorialGUI improves the average success rates of GPT-5.5, Kimi-K2.5, and Qwen3.8-Max by 13.78, 10.72, and 9.91 pp, respectively. With multimodal resources, the corresponding gains further increase to 15.02, 12.22, and 13.03 pp, reaching average success rates of 34.94%, 21.39%, and 43.39%. These consistent gains demonstrate the effectiveness of tutorial-derived knowledge and skills across different executor models.

(2) Strong Performance Without Model Training. On the four domains reported by CODA Stage-2 and SEAgent, TutorialGUI-MM achieves higher average success rates with all three evaluated foundation models. These results show competitive performance without model-specific training.

[2] Avg is the unweighted mean across reported applications, excluding missing scores (Appendix A.3).

(3) Additional Gains from Multimodal Resources. Compared with Text, MM improves Avg by 1.23, 1.50, and 3.11 pp for GPT-5.5, Kimi-K2.5, and Qwen3.8-Max, respectively. The gains vary across applications: GPT-5.5 improves on Algebra, Biochemistry, GIS, and Documentation but declines on Astronomy, while Kimi-K2.5 declines on ATP and Documentation. Reference images therefore provide complementary guidance on average.

4.3 Stage-wise Ablation

To examine how Replay and Exploration address the executability and coverage limitations of official tutorials, we evaluate four cumulative resource settings using GPT-5.5: *Baseline*, without TutorialGUI resources; *Candidate*, using resources directly extracted from official tutorials; *Replayed*, using resources refined and verified through Replay; and *Explored*, further expanding the Replayed library through Exploration and subsequent Replay verification.

Table 2 reports the results for these four resource settings. The Knowledge Card library remains fixed across Candidate, Replayed, and Explored; Baseline receives no TutorialGUI resources. Candidate includes original tutorial images when available, although most candidate skills have no image because the source tutorials contain few illustrations. Both Replayed and Explored include multimodal references derived from verified trajectories. Thus, the Candidate-to-Replayed comparison reflects both procedural refinement and changes in visual references.

Table 2: GPT-5.5 task success rates (%) across resource-construction stages. Baseline matches Table 1; Δ is the gain over Baseline in percentage points.

Domain	Baseline	Candidate	Replayed	Explored
Algebra	12.90	56.99	59.14	64.52
Biochemistry	44.83	49.43	57.47	63.22
GIS	12.75	6.86	8.82	22.55
ATP	3.85	3.85	3.85	3.85
Astronomy	20.20	17.17	25.25	24.24
Documentation	25.00	20.83	25.00	31.25
Avg	**19.92**	**25.85**	**29.92**	**34.94**
Δ vs. Baseline	—	+5.93	+10.00	+15.02

(1) Direct Extraction Is Useful but Insufficient. Candidate improves average success from 19.92% to 25.85%, showing that official tutorials provide useful application knowledge and procedural guidance for downstream execution. However, performance decreases on GIS, Astronomy, and Documentation, showing that directly extracted resources do not consistently improve downstream performance. These mixed results motivate further execution-based refinement and verification.

(2) Replay Improves Downstream Task Performance. Replay further increases average success from 25.85% to 29.92%, with performance improving or remaining unchanged across all six domains. By executing candidate procedures in real GUI environments, Replay completes missing steps, refines operational details, and verifies their intended effects before they enter the skill library. The improvement supports the downstream utility of Replay-refined resources.

(3) Exploration Extends Useful Skill Coverage. Exploration further increases average success from 29.92% to 34.94%. The additional gains show that expanding the verified library beyond procedures directly extracted from tutorials provides further downstream utility. This supports the role of Exploration in extending the coverage of useful capabilities while retaining execution-based verification.

4.4 Scaling Continued Exploration

We further study how TutorialGUI behaves as Exploration continues and the verified skill library grows. Using KAlgebra as a case study, we start from 95 verified skills obtained through Direct Replay of tutorial-derived candidates and conduct seven consecutive rounds of Exploration. Each

round targets a preset number of newly verified skills; proposed candidates count toward this target only after passing Replay and entering the library. The R1 library contains 123 verified skills and corresponds to the KAlgebra library used by GPT-5.5 with TutorialGUI-MM in Table 1, with the same mean success rate of 64.52%. R2 expands the library to 140 verified skills, and each subsequent round adds 20, reaching 240 skills at R7.

With the Knowledge Card library fixed and only the verified skill library expanded, we evaluate GPT-5.5 with TutorialGUI-MM at each stage over three runs on the same 31 KAlgebra tasks. Table 3 reports the successful-task counts for each run and the mean task success rate.

Table 3: Continued Exploration on KAlgebra. Tasks Solved gives each run's successful-task count (out of 31); success rates are three-run means.

Stage	Verified Skills	Tasks Solved (Runs 1/2/3)	Success Rate (%)
Direct Replay	95	19 / 18 / 18	59.14
+ Explore R1	123	19 / 20 / 21	64.52
+ Explore R2	140	20 / 20 / 21	65.59
+ Explore R3	160	22 / 23 / 22	72.04
+ Explore R4	180	23 / 24 / 24	76.34
+ Explore R5	200	24 / 26 / 25	80.65
+ Explore R6	220	23 / 22 / 23	73.12
+ Explore R7	240	21 / 20 / 21	66.67

(1) Continued Exploration Initially Improves Performance. Direct Replay achieves a mean success rate of 59.14% with 95 verified skills. Exploration raises this to 64.52% at R1 and 80.65% at R5, with 123 and 200 skills, respectively. The R5 result is 21.51 percentage points above Direct Replay and 16.13 points above R1. This trend indicates that continued Exploration can discover additional useful capabilities beyond the initial expanded library and provide further downstream utility as the verified skill set grows.

(2) Performance Does Not Scale Monotonically with Library Size. Mean success decreases to 73.12% at R6 and 66.67% at R7, despite the library growing to 220 and 240 verified skills, respectively. One possible explanation is increasing selection difficulty as the library grows: a larger library may contain more skills with overlapping functionality or similar applicability conditions, making it more difficult for the Selector to identify the most appropriate guidance for a given task.

5 Conclusion

We introduced TutorialGUI, a training-free framework that turns official tutorials into reusable skills and Knowledge Cards for GUI agents. Its Replay–Exploration loop addresses both incomplete procedures and limited tutorial coverage: Replay refines and verifies candidate skills through real GUI interaction, and Exploration uses verified skills to discover further capabilities. On six ScienceBoard applications, TutorialGUI improves average task success by 13.42 percentage points across three evaluated models relative to their corresponding baselines. Continued Exploration further extends the verified skill library beyond the original tutorials, demonstrating the potential for sustained capability expansion through GUI interaction.

Limitations and Future Work

Official tutorials may provide incomplete coverage of application capabilities, which can limit the scope of the initial resource library. More extensive Exploration may therefore be needed for applications with sparse documentation.

As the skill library grows, effective skill selection may become more challenging. Our continued-exploration study suggests that the benefits of library expansion may increasingly depend on effective skill selection, motivating future work on jointly improving skill acquisition and selection and studying their scaling behavior across applications.

AI USE STATEMENT

We used generative AI tools, including OpenAI Codex, to assist with writing and literature retrieval. Writing assistance included Chinese-to-English translation, drafting and polishing text, organizing methodological explanations and discussions of reported results, and LaTeX formatting. Retrieval assistance supported finding related work and locating published results for comparison. The models used within TutorialGUI and downstream evaluation are described in Section 4.1 and Appendix A.1. The authors take responsibility for the final manuscript, citations, and experimental claims.

REFERENCES

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024. URL https://arxiv.org/abs/2409.08264.

Xinyu Che, Junqi Xiong, Yunfei Ge, Xinping Lei, Shihao Li, Hang Yan, Han Li, Yuanxing Zhang, Zhiqi Bai, Jinhua Hao, Ming Sun, Han Li, and Jiaheng Liu. Mmg2skill: Can agents distill in-the-wild guides into self-evolving skills?, 2026. URL https://arxiv.org/abs/2606.01993.

Yijia Fan, Zonglin Di, Zimo Wen, Yifan Yang, Mingxi Cheng, Qi Dai, Bei Liu, Kai Qiu, Yue Dong, Ji Li, and Chong Luo. Resource2skill: Distilling executable agent skills from human-created multimodal resources, 2026. URL https://arxiv.org/abs/2606.29538.

Yue Fan, Handong Zhao, Ruiyi Zhang, Yu Shen, Xin Eric Wang, and Gang Wu. GUI-bee: Align GUI action grounding to novel environments via autonomous exploration. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33261–33278, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1688. URL https://aclanthology.org/2025.emnlp-main.1688/.

Ziyan Jiang, Li An, Yujian Liu, Jiabao Ji, Qiucheng Wu, Jacob Andreas, Yang Zhang, and Shiyu Chang. Visualskill: Multimodal skills for computer-use agents, 2026. URL https://arxiv.org/abs/2606.18448.

Jiateng Liu, Zhenhailong Wang, Rushi Wang, Bingxuan Li, Jeonghwan Kim, Aditi Tiwari, Pengfei Yu, Denghui Zhang, and Heng Ji. Osexpert: Computer-use agents learning professional skills via exploration, 2026a. URL https://arxiv.org/abs/2603.07978.

Yujian Liu, Ze Wang, Hao Chen, Ximeng Sun, Xiaodong Yu, Jialian Wu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Shiyu Chang. Learning from online videos at inference time for computer-use agents, 2025. URL https://arxiv.org/abs/2511.04137.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Zeyue Tian, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. Scalecua: Scaling open-source computer use agents with cross-platform data. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 62707–62766, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/666c1861d709bd84e20b6e0e02a2c223-Paper-Conference.pdf.

Xinwei Long, Kai Tian, Peng Xu, Weibo Gao, Yihua Shao, Guoli Jia, Haozhe Geng, Sa Yang, Jingxuan Li, Huayong Hu, Kaiyan Zhang, Jiaqi Wang, and Bowen Zhou. Guiding the blind: Generalizing gui agents to unseen websites via multimodal tutorials. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 373–391, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37490-5.

Dunjie Lu, Yiheng Xu, Junli Wang, Haoyuan Wu, Xinyuan Wang, Zekun Wang, Junlin Yang, Hongjin SU, Jixuan Chen, Junda Chen, Yuchen Mao, Junyang Lin, Binyuan Hui, and Tao Yu. Videoagenttrek: Computer-use pretraining from unlabeled videos. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 122696–122720, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c78b7a0016da6bf5be2a0691c17943ee-Paper-Conference.pdf.

Bowen Lv, Xiao Liu, Yanyu Ren, Hanyu Lai, Bohao Jing, Hanchen Zhang, Yanxiao Zhao, Shuntian Yao, Jie Tang, and Yuxiao Dong. Scalecua: Scaling computer use agents with verifiable task synthesis and efficient online rl, 2026. URL https://arxiv.org/abs/2607.11185.

Victor Muryn, Maksym Shamrai, Sofiia Mazepa, and Yehor Khodysko. Macarena: Benchmarking computer use agents on an online macos environment, 2026. URL https://arxiv.org/abs/2606.06560.

OpenAI. GPT-5.5 System Card, April 2026. URL https://openai.com/index/gpt-5-5-system-card/.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Qwen Team. Qwen3.8, 2026. URL https://qwen.ai/blog?id=qwen3.8.

Chan Hee Song, Yiwen Song, Palash Goyal, Yu Su, Oriana Riva, Hamid Palangi, and Tomas Pfister. Watch and learn: Learning to use computers from online videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5421–5431, June 2026.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, Lingpeng Kong, and Zhiyong Wu. Scienceboard: Evaluating multimodal autonomous agents in realistic scientific workflows. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 75694–75731, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7a9745f251508a053425a256490b0665-Paper-Conference.pdf.

Zeyi Sun, Yuhang Cao, Jianze Liang, Qiushi Sun, Ziyu Liu, Zhixiong Zhang, Yuhang Zang, Xiaoyi Dong, Kai Chen, Dahua Lin, and Jiaqi Wang. Coda: Coordinating the cerebrum and cerebellum for a dual-brain computer use agent with decoupled reinforcement learning, 2025a. URL https://arxiv.org/abs/2508.20096.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience, 2025b. URL https://arxiv.org/abs/2508.04700.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Rui Xie, Zhi Gao, Chenrui Shi, Zirui Shang, Lu Chen, and Qing Li. Guide:resolving domain bias in gui agents through real-time web video retrieval and plug-and-play annotation. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 581–598, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37026-6.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

Weimin Xiong, Shuhao Gu, Bowen Ye, Zihao Yue, Lei Li, Feifan Song, Sujian Li, and Hao Tian. Video2gui: Synthesizing large-scale interaction trajectories for generalized gui agent pretraining, 2026. URL https://arxiv.org/abs/2605.14747.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, Zhiyuan Chen, Jitong Liao, Qi Zheng, Jiahui Zeng, Ze Xu, Shuai Bai, Junyang Lin, Jingren Zhou, and Ming Yan. Mobile-agent-v3.5: Multi-platform fundamental gui agents, 2026. URL https://arxiv.org/abs/2602.16855.

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.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lu, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. Osworld 2.0: Benchmarking computer use agents on long-horizon real-world tasks, 2026. URL https://arxiv.org/abs/2606.29537.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. Tongui: Internet-scale trajectories from multimodal web tutorials for generalized gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12367–12375, 2026.

A Supplementary Experimental Details and Prompts

A.1 Experimental Environment and Implementation Details

Environment and Software. Replay and downstream evaluation are conducted in AgentBay sandboxes running Ubuntu 22.04, each configured with 8 vCPUs, 16 GiB of memory, and a fixed screen resolution of 1920×1080. We follow the application versions specified by ScienceBoard (Sun et al., 2026). Table 4 maps the domain abbreviations used in Table 1 to the corresponding applications and software configurations in our sandboxes.

Table 4: Application configurations in our AgentBay sandboxes for Replay and downstream evaluation. Software versions follow ScienceBoard (Sun et al., 2026), and the rows correspond to the domain columns in Table 1.

Domain	Application	Configuration
Alg	KAlgebra	21.12.3
Biochem	ChimeraX	1.9
GIS	GRASS GIS	8.4.2
ATP	Lean 4 within VS Code	VS Code 1.127.0; Lean 4 extension 0.0.237
Astron	Celestia	1.6.4
Doc	TeXstudio	4.8.6

Resource Construction. A unified model configuration is used across extraction, Replay, and Exploration. Candidate-skill and Knowledge Card extraction, parameter instantiation and action planning, visual outcome verification, and candidate generation during Exploration are performed with Qwen3.8-Max. Action grounding is handled by GUI-Plus[3], which is specialized for GUI understanding and interaction. Candidates proposed during Exploration are subsequently executed and verified through Replay before entering the skill library.

Replay. Each candidate is allowed at most three Replay attempts, including the initial attempt. Replay terminates once verification succeeds. Candidates that do not pass verification within this budget are discarded and are not retried in subsequent Exploration rounds.

Exploration. Each Exploration round alternates candidate generation and Replay until a preset target number of newly verified skills is reached. Failed or inconclusive candidates do not count toward this target. In the KAlgebra continued-exploration study, R1 and R2 add 28 and 17 verified skills, respectively, while R3–R7 each add 20. Candidate-generation budgets apply to individual proposal calls and are independent of the round-level target for newly verified skills.

Downstream Evaluation. After resource construction, the skill library is frozen and shared across all evaluated models. Both the foundation-model baselines and TutorialGUI are evaluated in the AgentBay environments described above, using the same evaluation protocol and the official ScienceBoard task-success evaluators (Sun et al., 2026). GPT-5.5, Kimi-K2.5, and Qwen3.8-Max each serve as both Selector and Executor following the resource-selection protocol in Section 3.4.

TutorialGUI-Text provides the selected Knowledge Cards and skills in textual form. *TutorialGUI-MM* follows the same protocol and additionally provides one reference image for each selected skill. Complete abstention provides no external image. Both settings retain live screenshots of the current GUI state throughout task execution. All model parameters remain fixed during resource construction and downstream evaluation. Appendix A.3 details the sources and evaluation protocols of the baseline and published comparison results, including which results are evaluated in our environment and which are taken from official reports.

Library Statistics. Table 5 reports the official ScienceBoard task counts (Sun et al., 2026) together with the verified skills used in the main experiments. The final resource library contains 1,715 Knowledge Cards and 569 verified skills across the six domains. It includes both tutorial-derived

[3] https://help.aliyun.com/en/model-studio/gui-automation

candidates admitted through direct Replay and skills obtained after one round of Exploration followed by Replay for each application. The additional KAlgebra Exploration rounds in Section 4.4 are used only for the continued-exploration study and are not included in the library used for the main results.

Table 5: Official ScienceBoard task counts and verified skill counts in the main experimental library. Applications follow the column order of Table 1; each skill count includes one round of Exploration.

Application	Tasks	Verified skills
KAlgebra	31	123
ChimeraX	29	111
GRASS GIS	34	100
Lean 4	26	83
Celestia	33	115
TeXstudio	16	37
Total	169	569

A.2 Tutorial Sources

Table 6 lists the tutorial and documentation sources used for knowledge extraction, together with their resource formats. For the Celestia guides page, MS Word denotes the linked guide documents rather than the format of the landing page.

Table 6: Tutorial and documentation sources by application. URLs identify the source pages or repositories; formats describe the corresponding resources.

Application	Source URL	Format
KAlgebra	`https://docs.kde.org/stable_kf6/en/kalgebra/kalgebra/`	HTML
ChimeraX	`https://www.cgl.ucsf.edu/chimerax/docs/user/`	HTML
GRASS GIS	`https://grass.osgeo.org/grass84/manuals/`	HTML
Lean 4	`https://leanprover-community.github.io/mathematics_in_lean/`	HTML
	`https://leanprover-community.github.io/learn.html`	HTML
	`https://github.com/leanprover/vscode-lean4/blob/master/vscode-lean4/manual/manual.md`	Markdown
Celestia	`https://en.wikibooks.org/wiki/Celestia`	HTML
	`https://celestiaproject.space/guides.html`	MS Word
TeXstudio	`https://texstudio-org.github.io`	HTML

A.3 Baseline Sources and Reporting Conventions

Rerun foundation-model baselines. We rerun GPT-5.5, Kimi-K2.5, and Qwen3.8-Max without TutorialGUI Knowledge Cards or skills, using the same infrastructure and evaluation protocol as their TutorialGUI counterparts (Appendix A.1). Their six-application average success rates are 19.92%, 9.17%, and 30.36%, respectively. The GPT-5.5 and Kimi-K2.5 results are slightly higher than the corresponding ScienceBoard baselines of 18.31% and 7.76%; ScienceBoard does not report a Qwen3.8-Max baseline.

Published comparison results. CODA Stage-2 results are the Average@8 scores from Table 1 of Sun et al. (2025a), using a trained Qwen2.5-VL-32B planner and a UI-TARS-1.5-7B executor.

SEAgent results are the UI-TARS-1.5-7B-DPO+SEAgent success rates from Table 5 of Sun et al. (2025b). ScaleCUA results are the screenshot-observation Qwen3.5-9B scores from Table 3 of Lv et al. (2026), which reports four independent rollout passes per task. These published results retain their original backbone, training, and evaluation settings.

Aggregation and performance gains. For our evaluations, application-level success rates are averaged over three runs. Avg gives equal weight to each reported application: four for CODA and SEAgent, and six for ScaleCUA and our evaluated models. Missing entries (—) are excluded. Published per-application scores retain their original precision; applying this averaging convention to ScaleCUA yields 55.23%, compared with its reported Overall of 54.0%. For our results, averages and performance differences are computed from unrounded success proportions and rounded to two decimal places only for presentation. In Tables 1 and 2, gains are measured against the corresponding rerun foundation-model baseline; Δ denotes the difference in Avg, expressed in percentage points.

A.4 Key Prompts for Resource Construction and Task Execution

The prompts below retain the core instructions, with output schemas and formatting requirements omitted for brevity.

A.4.1 Source-Grounded Knowledge Extraction

Knowledge extraction produces source-grounded Skill candidates, Knowledge Cards, and Unknown Slots, and may yield multiple records. Knowledge Cards provide non-executable decision support, while Unknown Slots identify information requiring further evidence rather than guessed values.

> Mine the following approved source units for App {app}. Produce {skill_count} distinct, replayable, low-risk Skill candidates when the evidence supports them; never pad the count.
>
> A Skill must have one primary effect, semantic rather than coordinate targets, a required postcondition verifiable from exact application/file evidence or stable UI-semantic/declared visual-relation evidence, and no unresolved required field. One Skill may use one or several primitive actions. Frequently reused hotkey, menu, and button effects may stand alone when independently useful and verifiable.
>
> Put variable values in typed runtime parameter slots instead of cloning literal-specific Skills. Exclude command-line-only actions unless the source explicitly presents them as an operation through the GUI.
>
> Also extract source-supported, nonredundant Knowledge Cards when they contribute useful decision-relevant knowledge. Do not impose a fixed card count or generate cards merely to fill a quota. Return an empty card list when the sources contain no suitable knowledge.
>
> A Knowledge Card is a non-executable knowledge record. It may explain when an operation is appropriate, how to interpret a parameter or application state, which constraints or prerequisites apply, how to recognize a valid result, or how to diagnose a failure. State a clear claim and preserve its relevant conditions, qualifiers, and exclusions. Explain which decision the knowledge supports. Do not duplicate an entire Skill procedure as a Card merely to create another record, and do not label a Card as a replay-verified executable Skill.
>
> Return genuine Unknown Slots for information the sources do not establish. Do not invent missing facts. Candidates with unresolved required information must not be presented as replay-ready Skills.
>
> Paraphrase rather than copying long source passages. Every extracted record must cite valid supplied source-unit IDs.

A.4.2 Replay Postcondition Verification Prompt

The verifier assesses declared postconditions for a single Replay attempt. The fixture contract and transient screenshots are used only when supplied and applicable, as specified below.

> You are an independent postcondition verifier for one source-bound Atomic Skill Replay in a frozen desktop application.
>
> Use only the declared postconditions, bound parameters, setup trace, action trace, stable before screenshot, current stable post-action screenshot, and any explicitly labelled immediate post-action transient full screenshot. The declared postconditions and actually bound parameters are authoritative. Planner or grounder prose cannot add a residue identity, object name, value, chain, target, or other success constraint that is absent from those declarations; when an optional parameter is unbound, a visibly valid runtime exemplar is acceptable if it satisfies the declared relation. Do not trust planner or grounder claims of completion. Do not use benchmark task IDs, answers, evaluator verdicts, historical pass/fail, or external facts. Evaluate only visible UI/application relations; file probes supplied by the executor take precedence for file-state assertions.
>
> A `fixture_verification_contract` is trusted only as a description of an attempt-local runner-generated fixture when its content SHA matches the `fixture_create` setup trace. It may explain whether a visible before/after transition is consistent with that exact input, but it cannot replace live evidence, prove an assertion by itself, or become part of the reusable Skill.
>
> For a source-declared transient UI effect such as an OSD or toast, a sharp immediate post-action full screenshot may establish the assertion when it directly and unambiguously displays the declared effect; the required stable after screenshot proves the application subsequently settled, but the transient text is not required to remain visible in that later frame.
>
> Assess every declared postcondition exactly once, assigning pass, fail, or inconclusive and explaining the supporting evidence. Return `verified` only when every required assertion passes. Use `failed` for clear contradictory evidence and `inconclusive` for unavailable or ambiguous evidence.

A.4.3 Shared Knowledge-Selection Prompt

The shared Selector selects up to two skills and one Knowledge Card from the library for the current task.

> You select reusable tutorial knowledge for a GUI executor. This is selection only; parameter binding and execution are handled by the executor.
>
> Select at most two Skills and at most one Card, with at most three records total. A Card is optional: select one only when it supplies relevant, compatible knowledge that helps the executor choose an operation, interpret or bind parameters, understand the current state, respect constraints, verify results, or handle a failure. Do not select a Card merely because its topic matches the task.
>
> One useful partial operation is sufficient. Decompose the task into necessary operations; choose records that explain how to perform one or more of them in the relevant app. Do not require a candidate to solve the whole scientific task, guarantee success, contain the final answer, name the current object, or already know runtime values.
>
> A source example's object names and values are replaceable parameters unless the source explicitly fixes them as preconditions. Unknown GUI state is checked by the executor; an explicit incompatible precondition is grounds for rejection.
>
> Use the source procedure and coverage to decide applicability. Prefer the most direct operation for the requested outcome, then complementary steps. A Card containing the needed command or UI guidance can be more useful than an unrelated executable Skill, but its selection does not establish that the operation has been executed or verified.
>
> Do not fill the quota with irrelevant records. Abstain only when none of the candidates supplies a useful compatible operation or knowledge. For cross-app tasks, select at most one record per owner app, except when an explicit application-specific policy replaces this rule.

> Identify the required task stages and indicate which are covered by the selected records. Do not bind parameters during selection. The executor binds parameter values using the task instruction and current GUI observations. Required values must be resolved before executing the corresponding operation. Selection is not execution or task success.

A.4.4 TEXSTUDIO CROSS-APPLICATION SELECTION POLICY

When the TeXstudio cross-application branch is enabled, this policy replaces the shared rule of at most one record per owner app. It is not an unconditional addition for all tasks, and the global budget of at most two Skills and one Card remains unchanged.

> For TeXstudio cross-app tasks, select only from the allowed owner apps. Complementary records may share an owner, including a Card and Skills from the same app.
>
> The global limits remain at most two Skills and one Card, three distinct records total. No owner must be represented; do not substitute or add records to balance owners.
>
> Report uncovered required stages explicitly. Owner membership does not establish stage coverage.

A.4.5 EXECUTION-TIME KNOWLEDGE CONTEXT

This additional method context guides the GUI executor in using selected knowledge, historical procedures, and reference images. It is not the complete executor system prompt and does not replace the underlying action or coordinate protocol.

> TutorialGUI method context: selected tutorial guidance and eligible Skill programs.
>
> In the multimodal setting, each selected Skill is accompanied by one reference image when available. Source steps and reference images are historical examples, not current observations or answers. Follow compatible source operations; bind targets from the current task and live GUI.
>
> Use Knowledge Cards as contextual guidance for decisions, parameters, constraints, state interpretation, and verification. A Card is not an executable program, and its claim alone does not establish the current GUI state or prove that an action succeeded. Check its conditions against current observations when applicable.
>
> Missing values may be acquired through the GUI, but unresolved placeholders must never be executed. Partial coverage is valid; complete uncovered stages yourself. Check current preconditions and results.
>
> Reference-image coordinates are not live coordinates. Follow the system action/coordinate protocol unchanged.

A.4.6 EXPLORATION CANDIDATE-GENERATION PROMPT

This prompt takes an anchor selected by the outer exploration loop and proposes candidates for subsequent Replay. The outer loop stops when the preset number of newly verified skills has been admitted; candidate generation itself does not execute or verify proposals.

> You are exploring reusable operations in a desktop application.
>
> Given a verified anchor skill, the existing skill library, current application observations, available initial-state setup, relevant tutorial or manual evidence, a per-call candidate budget, and the remaining target for newly verified skills in this round, propose exploration candidates in the following directions:
>
> **FORWARD** — Discover downstream operations that use an object produced by the anchor or a result state established by it. Identify the output or state that connects the anchor to the candidate.

BACKWARD — Discover operations that establish an input or precondition required by the anchor. These may supply a missing preparation process or provide an alternative preparation path. An existing verified predecessor is not required. When the initial-state setup already satisfies the preconditions, explain whether further preparation exploration is useful.

LATERAL — Discover operations that share an input, serve a related goal, or provide a neighboring function. These may include alternative operating modes or configurations within the same skill family. Explain what changes in the procedure, application state, or observable outcome, without assuming an execution dependency.

Requirements:

- Associate every candidate with the supplied anchor ID.
- Specify its input slots, preconditions, procedure, and expected effect.
- Ground proposals in the supplied evidence and distinguish observed facts from hypotheses.
- Compare each candidate with existing skills. State whether it may represent a distinct operation, an alternative mode, a parameterized variant, or an already covered capability.
- Prioritize useful, evidence-supported candidates with checkable effects. Avoid redundant proposals that add no exploration value.
- Keep exploration relations separate from execution dependencies. Each candidate admitted after separate Replay and verification is added as a new skill record. Retain existing skill records unchanged; do not merge candidates into them or extend their parameter coverage.
- Do not infer dependencies from semantic similarity alone.
- Do not invent controls, sources, execution traces, or successful results.
- Do not use benchmark tasks, answers, or evaluation outcomes.
- Do not force coverage of all three directions. Explain any direction for which no useful candidate is supported.
- Do not execute or verify proposals. Candidates remain proposed and unverified pending separate Replay and verification.
- Stay within the supplied per-call candidate budget. Proposed candidates do not count toward the verified-skill target; only admission after separate Replay and verification counts. If the remaining target is zero, return no new candidates.

For each direction, report findings and supporting evidence. For each candidate, report its relation to the anchor and existing skills, evidence, uncertainties, and the checks required during Replay. Identify related existing skills and their supporting evidence.

A.5 Run-to-Run Variability

Tables 7–9 report the mean and sample standard deviation (SD) across three evaluation runs. SD is computed with denominator $n - 1$, where $n = 3$. For Avg, we first compute the unweighted mean across the six applications within each run, then summarize the three resulting values. All statistics use unrounded success proportions and are rounded to two decimal places for presentation; means are percentages and SDs are in percentage points.

Table 7: Main experimental results as mean $\pm$ sample SD over three runs, complementing Table 1. Applications follow the same order as its columns.

Application	Baseline	TutorialGUI-Text	TutorialGUI-MM
GPT-5.5			
KAlgebra	12.90 ± 3.23	63.44 ± 4.93	64.52 ± 3.23
ChimeraX	44.83 ± 3.45	60.92 ± 1.99	63.22 ± 1.99
GRASS GIS	12.75 ± 1.70	17.65 ± 2.94	22.55 ± 1.70
Lean 4	3.85 ± 0.00	3.85 ± 0.00	3.85 ± 0.00
Celestia	20.20 ± 1.75	29.29 ± 1.75	24.24 ± 3.03
TeXstudio	25.00 ± 0.00	27.08 ± 3.61	31.25 ± 0.00
Avg	19.92 ± 0.48	33.70 ± 0.56	34.94 ± 1.39
Kimi-K2.5			
KAlgebra	10.75 ± 1.86	19.35 ± 3.23	19.35 ± 3.23
ChimeraX	29.89 ± 1.99	35.63 ± 5.27	40.23 ± 1.99
GRASS GIS	0.98 ± 1.70	15.69 ± 1.70	16.67 ± 1.70
Lean 4	0.00 ± 0.00	6.41 ± 2.22	3.85 ± 0.00
Celestia	5.05 ± 1.75	15.15 ± 3.03	23.23 ± 1.75
TeXstudio	8.33 ± 3.61	27.08 ± 3.61	25.00 ± 0.00
Avg	9.17 ± 0.77	19.89 ± 1.58	21.39 ± 0.52
Qwen3.8-Max			
KAlgebra	41.94 ± 3.23	59.14 ± 4.93	66.67 ± 1.86
ChimeraX	60.92 ± 3.98	67.82 ± 3.98	68.97 ± 3.45
GRASS GIS	17.65 ± 0.00	20.59 ± 0.00	21.57 ± 1.70
Lean 4	10.26 ± 2.22	14.10 ± 5.88	17.95 ± 2.22
Celestia	22.22 ± 1.75	40.40 ± 7.63	41.41 ± 1.75
TeXstudio	29.17 ± 7.22	39.58 ± 3.61	43.75 ± 6.25
Avg	30.36 ± 2.35	40.27 ± 0.70	43.39 ± 1.77

Table 8: GPT-5.5 stage-wise ablation as mean ± sample SD over three runs, complementing Table 2. Baseline and Explored reuse the GPT-5.5 Baseline and TutorialGUI-MM results in Table 7, respectively.

Application	Baseline	Candidate	Replayed	Explored
KAlgebra	12.90 ± 3.23	56.99 ± 8.12	59.14 ± 1.86	64.52 ± 3.23
ChimeraX	44.83 ± 3.45	49.43 ± 7.18	57.47 ± 5.27	63.22 ± 1.99
GRASS GIS	12.75 ± 1.70	6.86 ± 1.70	8.82 ± 2.94	22.55 ± 1.70
Lean 4	3.85 ± 0.00	3.85 ± 0.00	3.85 ± 0.00	3.85 ± 0.00
Celestia	20.20 ± 1.75	17.17 ± 3.50	25.25 ± 1.75	24.24 ± 3.03
TeXstudio	25.00 ± 0.00	20.83 ± 3.61	25.00 ± 0.00	31.25 ± 0.00
Avg	19.92 ± 0.48	25.85 ± 1.45	29.92 ± 0.85	34.94 ± 1.39

Table 9: Continued Exploration on KAlgebra: per-run success rates (%) and mean ± sample SD. The underlying successful-task counts are reported in Table 3; each run contains 31 tasks.

Stage	Run 1	Run 2	Run 3	Mean ± SD
Direct Replay	61.29	58.06	58.06	59.14 ± 1.86
+ Explore R1	61.29	64.52	67.74	64.52 ± 3.23
+ Explore R2	64.52	64.52	67.74	65.59 ± 1.86
+ Explore R3	70.97	74.19	70.97	72.04 ± 1.86
+ Explore R4	74.19	77.42	77.42	76.34 ± 1.86
+ Explore R5	77.42	83.87	80.65	80.65 ± 3.23
+ Explore R6	74.19	70.97	74.19	73.12 ± 1.86
+ Explore R7	67.74	64.52	67.74	66.67 ± 1.86

B EXAMPLES OF SKILLS AND KNOWLEDGE CARDS

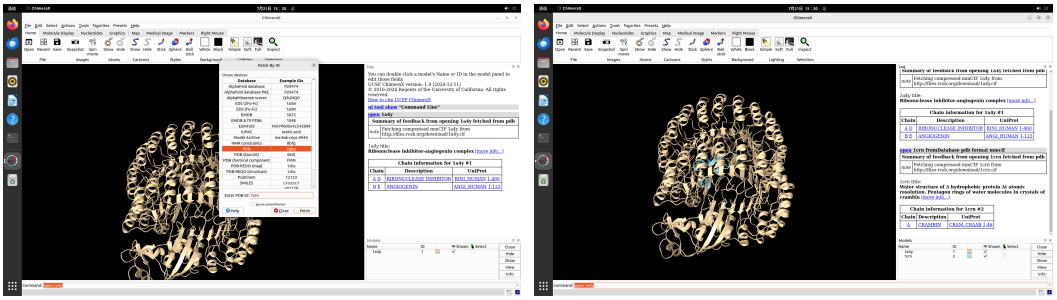
The following examples complement Appendix A.4 with a ChimeraX skill and a Celestia Knowledge Card.

B.1 SKILL EXAMPLE: FETCHING A STRUCTURE IN CHIMERAX

Table 10 shows a parameterized Fetch by ID skill and its recorded verification results. Figure 4 shows the dialog and the fetched model.

Table 10: Selected original fields of the ChimeraX Fetch by ID skill record.

Field	Value
`name`	Fetch a structure through the Fetch by ID dialog
`app`	ChimeraX
`source`	`direct_replay`
`preconditions`	ChimeraX is running with an English, maximized main window.
`input_schema`	`database`, `identifier`: required strings
`operation_flow`	`open_menu`: File `click_menu_item`: Fetch by ID `select_control`: database `type_text`: identifier `confirm`: Fetch
`gui_path`	`path`: [“File”, “Fetch by ID”] `visible_result`: The requested database entry is fetched and opened as a model.
`success_checks`	The fetched model appears in the Model Panel and the Log identifies the completed fetch.
`source_citations`	`https://www.cgl.ucsf.edu/chimerax/docs/user/commands/open.html`
`replay`	`fresh_session`: `true`; `success`: `true`
`semantic_judgment`	`status`: `semantic_pass` `judge`: qwen3.8-max screenshot evaluator `reason`: 1CRN loaded as new visible model (blue structure) while preparation model 1A4Y remains present (beige structure), both shown in Models panel
`verified`	`true`



(a) Fetch by ID dialog **(b) Fetched model loaded**

Figure 4: Fetch by ID in ChimeraX. (a) PDB is selected and `1crn` entered. (b) Fetched `1CRN` (blue) appears alongside preparation model `1A4Y` (beige). The Models panel lists both entries, and the Log records the PDB fetch.

B.2 KNOWLEDGE CARD EXAMPLE: UNDERSTANDING OUTDATED ISS ORBITS IN CELESTIA

Table 11 presents selected original fields of a Celestia Knowledge Card explaining why ISS orbit data can become outdated. This example illustrates how Knowledge Cards support diagnosis without specifying an executable procedure.

Table 11: Selected original fields of the Celestia Knowledge Card record.

Field	Value
app	Celestia
artifact_kind	knowledge_card
card_type	troubleshooting
content	The orbit of the International Space Station changes continuously in ways that are almost impossible to predict due to things like atmospheric drag, light pressure, cargo ship docking, etc.
citation	source_document_id: wikibooks_faqs section_id: wikibooks_faqs:s00031 source_locator: https://en.wikibooks.org/wiki/Celestia/FAQs
family_ids	[celestia.time]
capability_tags	[celestia.time, celestia.time_rate]
runtime_channel	phase0