IDENTIFIABLE PROTOCOL STATES FOR CROSS-ENVIRONMENT GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI agents need to reuse task structure when interfaces change, yet most policies bind progress to screen-specific actions. We introduce Identifiable Protocol States (IPS), an operational abstraction that groups observations by shared preconditions, admissible next subgoals, and recovery options. A proposed match is therefore testable through the behavior it permits. IPS learns this abstraction from successful trajectories, a human-written template of 4–8 subgoals per task family, and automatic language-model slot assignment. A state matcher, invariant miner, protocol-constrained planner, and recovery compiler separate task structure from interface-specific grounding. We also give a conditional sample-complexity argument for reusing a transferred protocol. On SPA-Bench and AndroidWorld, IPS reaches 58.9% and 54.1% task success, compared with 53.7% and 47.4% for GUI-explorer. Controls with the same templates and slot supervision isolate the contribution of cross-environment matching. Matching, recovery, and cost analyses explain where the gains arise and where the abstraction is most useful. The evidence supports protocol-based transfer for structured workflows and shows smaller gains on open-ended browsing.The code is available at https://anonymous.4open.science/r/GUI_Agent-2EE0/.

1 INTRODUCTION

A task stage is defined by what must already hold, what can happen next, and how the agent can recover when a transition fails. In flight booking, date selection, fare selection, passenger details, and confirmation recur across sites, although their screens and controls vary. Similar-looking forms can still require different next actions: passenger and billing details have different prerequisites and recovery options. Cross-interface reuse therefore requires task-stage equivalence rather than visual resemblance.

Current GUI agents combine visual grounding, language-model action generation, exploration, and planning (Hong et al., 2024; Xu et al., 2024; Xie et al., 2025; Li et al., 2025). Their policies still tend to attach progress to a particular screen, so interface changes can break a useful sequence of actions. SPA-Bench and AndroidWorld make this pressure visible in multi-step smartphone tasks (Chen et al., 2024; Rawles et al., 2024). We ask whether an explicit representation of shared task stages can make transfer and recovery more reliable.

We define a *protocol state* through three operational signatures: preconditions, admissible next subgoals, and recovery structure. Two observations share a state when these signatures agree. The definition makes a learned match falsifiable: a different valid continuation or recovery option counts as evidence against it. We call the state *identifiable* in this operational sense, which is distinct from uniqueness results in formal statistical identifiability.

IPS learns from successful trajectories, one human-written template of 4–8 subgoal descriptions per task family, and automatic large language model (LLM) slot assignment. An LLM screen summarizer supplies candidate invariant predicates. This template-supervised abstraction transfers to a new interface within a supported family; a new family requires its own template and suitable trajectories. Human audits used to evaluate the representation remain separate from this training input.

The framework factors execution into a protocol layer and an interface-specific grounding layer. Cross-environment matching learns the shared representation, invariant mining checks state constraints,

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.

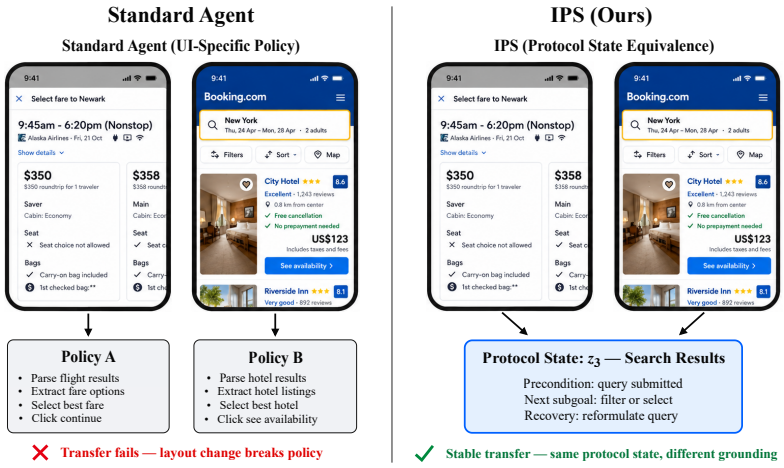

Figure 1: Protocol abstraction across interfaces. The same task stage can admit different interface-specific actions. The success and failure indicators illustrate the mechanism.

planning selects a successor, and recovery maps transition failures to corrective actions. A conditional learning argument shows how freezing a suitable transferred protocol can reduce an adaptation upper bound; the experiments then test the representation and its execution behavior.

We evaluate task success, operational agreement, recovery, and cost. IPS improves SPA-Bench and AndroidWorld success by 5.2 and 6.7 percentage points over GUI-explorer. A same-template standard operating procedure (SOP) baseline and a hierarchy-only control with the same slot labels isolate the role of matching. Correctly matched states reach 61.8% success versus 34.3% for mismatches, linking representation quality to execution. Together, the results establish protocol-based transfer for the evaluated English-language, structured workflows and identify open-ended browsing as a harder setting.

2 Related Work

GUI policies and grounding. ReAct (Yao et al., 2023) couples reasoning with action, while AppAgent, AutoDroid, CogAgent, and Mobile-Agent-v2 build smartphone and GUI interaction pipelines (Zhang et al., 2023; Wen et al., 2024; Hong et al., 2024; Wang et al., 2024a). SeeAct, Aguvis, Aria-UI, and UGround target visual action grounding (Zheng et al., 2024; Xu et al., 2024; Yang et al., 2025; Gou et al., 2024). IPS builds on these grounding capabilities and asks which task-state information remains reusable across interfaces.

Procedural planning and exploration. ChatSOP uses standard operating procedures to guide planning (Li et al., 2025). AdaptAgent adapts from demonstrations, while GUI-explorer and Open-WebVoyager acquire interaction knowledge through exploration (Verma et al., 2025; Xie et al., 2025; He et al., 2025); GUICourse studies GUI training data (Chen et al., 2025). IPS shares this reliance on demonstrations and procedural supervision but adds an explicit cross-environment matching objective over preconditions, continuations, and recovery signatures. Our controls keep the templates fixed, and the hierarchy-only control also keeps the slot labels.

Recent transfer and recovery mechanisms. ShowUI-Aloha structures human screen recordings into executable tasks (Zhang et al., 2026), and GUIDE retrieves tutorial-video knowledge to address application-specific domain bias (Xie et al., 2026). Continual GUI Agents studies changing domains and resolutions (Liu et al., 2026); HyMEM organizes reusable experience with symbolic nodes and trajectory embeddings (Zhu et al., 2026); UI-Voyager learns from failed experience (Lin et al., 2026); and Code2World predicts action-conditioned interfaces through renderable code (Zheng et al., 2026). These directions motivate explicit reuse and recovery. IPS tests operational agreement under supplied

templates, complementing retrieval-, recording-, and world-model-based approaches. Appendix I places the evaluation alongside these directions.

Temporal and state abstraction. Options provide a framework for temporally extended behavior (Sutton et al., 1999), and hierarchical planners and task automata represent intermediate structure. IPS instantiates this idea with an operational signature for GUI transfer: preconditions, admissible continuations, and recovery options. The experiments compare IPS with hierarchical planning and evaluate the signature directly. Formal identifiability in nonlinear independent component analysis (ICA) addresses a different problem from these operational tests.

3 Protocol States and Execution

3.1 Operational Equivalence and Transfer

Let $o_{\le t}$ be the observation history, g the goal, and a_t an interface action. IPS introduces a protocol state $z_t \in \mathcal{Z}$ and models

$$p(z_t, a_t \mid o_{\le t}, g) = p_{\theta_z}(z_t \mid o_{\le t}, g)\, p_{\theta_a}(a_t \mid z_t, o_t, g). \tag{1}$$

The first factor represents task stage; the second grounds that stage in the current interface. Transfer is therefore appropriate when source and target tasks share the relevant structure.

For an observation in its task context, define

$$\Phi(o) = \big(c(o), n(o), r(o)\big), \qquad o^{(i)} \sim_z o^{(j)} \iff \Phi(o^{(i)}) = \Phi(o^{(j)}), \tag{2}$$

where c denotes preconditions, n the admissible next-subgoal class, and r the recovery structure. We suppress task context in Φ for readability. This ideal relation defines the target of the learned assignment; agreement rates measure how closely the matcher approximates it.

Variation across sites. Signature agreement does not require identical screen layouts or identical end-to-end workflows. Sites with different mandatory information requirements map to different states at that stage. Optional-slot tags record optional steps during trajectory segmentation. An optional seat-selection stage and a mandatory passenger-information stage therefore remain distinct even when both precede payment. Context-dependent branches create ambiguity when the observation or template omits the condition that changes the valid continuation.

Conditional adaptation argument. Write $\theta = (\theta_z, \theta_a) \in \Theta_z \times \Theta_a$, where d_z and d_a denote the dimensions of the protocol and grounding parameter classes. The argument assumes (A1) that a source-learned, fixed protocol θ_z^* permits target risk at most $\epsilon/2$ for some grounding parameters, (A2) that the loss lies in $[0, 1]$ and target training episodes are independent and identically distributed, and (A3) that the restricted grounding class has Rademacher complexity $\widetilde{O}(\sqrt{d_a/n})$. Uniform convergence then gives the sufficient sample scaling

$$n = \widetilde{O}(d_a/\epsilon^2) \tag{3}$$

for empirical risk minimization over the grounding class. A joint class with complexity $\widetilde{O}(\sqrt{(d_z + d_a)/n})$ has the analogous sufficient bound $\widetilde{O}((d_z + d_a)/\epsilon^2)$. These are conditional upper bounds under the stated capacity assumptions. Appendix B gives the full scope of the argument and separates its assumptions from the empirical diagnostics.

3.2 State Matching and Required Supervision

A frozen CLIP ViT-L/14 encoder (Radford et al., 2021) produces observation features h_t. Three learned projections represent the signature components:

$$p_t = W_c h_t, \qquad q_t = W_n h_t, \qquad r_t = W_r h_t, \qquad \nu_t = f([p_t; q_t; r_t]), \tag{4}$$

where f is a two-layer multilayer perceptron (MLP) with layer normalization. The embedding ν_t is the shared protocol-state representation.

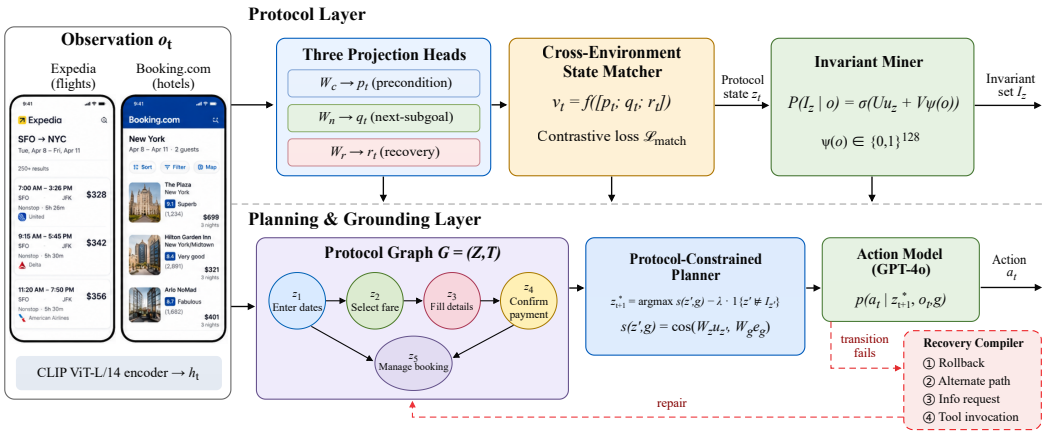

Figure 2: IPS execution loop. Learned matches determine the protocol state; invariant checks and recovery respond to assignment and execution errors. The checks and recovery module share learned components with the matcher.

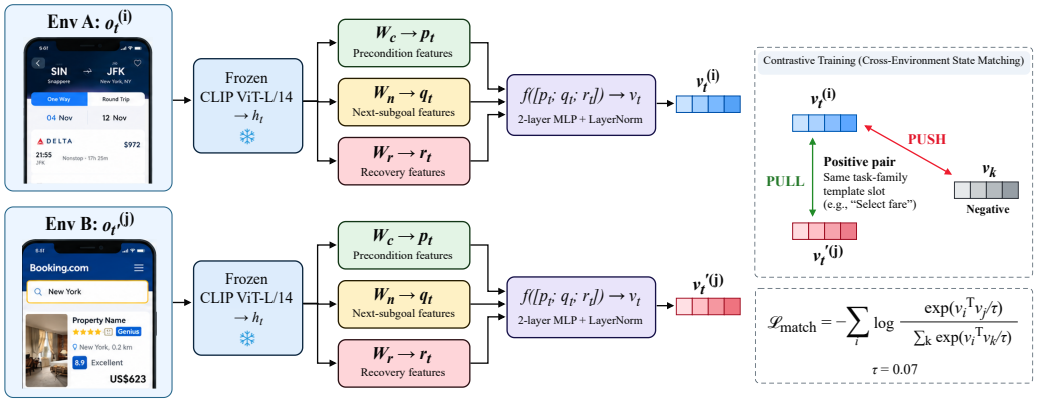

Figure 3: State-matching module. The three feature projections represent preconditions, next subgoals, and recovery structure before fusion into a protocol embedding. The interface panels illustrate the matching problem; they do not add evaluated cases to Table 3.

Successful trajectories are segmented with the family template. An LLM assigns observation–action pairs to subgoal slots by majority vote over three calls. Same-slot assignments form positive pairs, while different-protocol assignments form negative pairs for contrastive training. Optional and branching steps carry optional-slot tags. An audit of 500 human-annotated pairs estimates about 12% label noise. Template construction supplies the manual supervision; the audit and evaluation annotations are separate human inputs.

The matching objective draws positive embeddings together within the contrastive candidate set, using temperature $\tau = 0.07$. Figure 3 shows the three signature projections. Slot labels provide supervision, while agreement in admissible continuations and recovery options provides the operational test. Section 4.3 evaluates both.

3.3 Invariant Checks, Planning, and Recovery

Invariant mining. The invariant miner associates each state z with a constraint set I_z. An LLM screen summarizer produces binary predicates such as "query submitted" and "required field filled." We retain the 128 most frequent training predicates. With centroid u_z and predicate vector $\psi(o)$, the



Figure 4: Illustrative matching examples: a cross-interface positive pair and a visually similar but operationally different negative pair.

miner computes

$$P(I_z \mid o) = \sigma\big(U u_z + V\psi(o)\big). \tag{5}$$

The additive projections condition each constraint score on the state and observation. Predicate predictions agree with human labels on 89.3% of a 200-observation validation set. An invariant violation can flag an incorrect match or an unmet task requirement; the observation alone does not always distinguish these cases.

Protocol-constrained planning. The protocol graph $G = (\mathcal{Z}, \mathcal{T})$ stores legal transitions. Given the current state, the planner selects

$$z^*_{t+1} = \arg\max_{z' \in \mathcal{N}(z_t)} \left[s(z', g) - \lambda\, \mathbf{1}\{\text{preconditions of } z' \text{ are unmet}\} \right], \tag{6}$$

where $\mathcal{N}(z_t)$ contains legal successors and $s(z', g)$ is cosine similarity between projected state and goal embeddings in a shared 256-dimensional space. The projections train with demonstration next-state labels. The action backbone grounds the selected successor in the current interface. Reusing a protocol graph removes the need to relearn task order while leaving interface grounding to the action model.

Recovery compilation. A failed transition triggers one of four corrective categories: roll back to a previous valid state and try another grounding, choose an alternate successor, request missing information, or invoke an external tool. The compiler conditions its decision on the protocol state and failure signal. Invariant checks and recovery handle execution errors, while an incorrect match can still misdirect both modules. Appendix A gives the execution algorithm, and Section 4.3 quantifies the resulting errors.

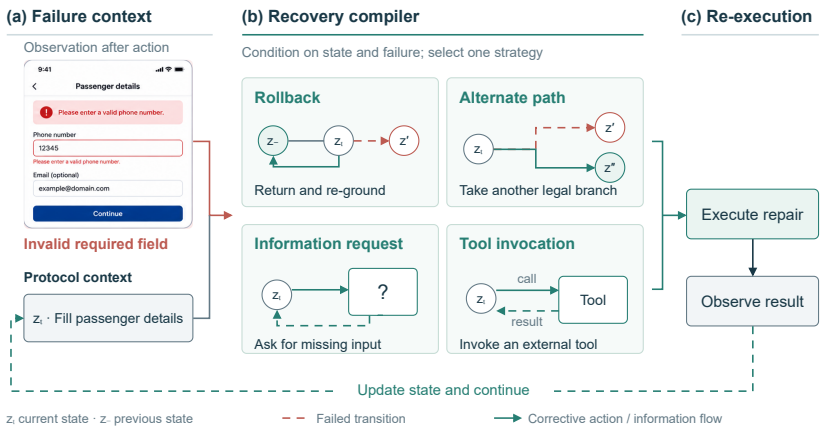

Figure 5: Protocol-level recovery. The current state and failure signal determine a choice among rollback, an alternate legal successor, an information request, and an external tool call.

4 Experiments

4.1 Setup and Supervision Controls

We evaluate IPS on SPA-Bench, AndroidWorld, and GUI-KRB (Chen et al., 2024; Rawles et al., 2024; Xie et al., 2025). The default action backbone is GPT-4o (OpenAI, 2024); a Qwen2-VL-7B variant tests backbone dependence (Wang et al., 2024b). The matcher trains for 10 epochs on 5,000 aligned trajectory pairs from 15 application families using four A100 GPUs. Protocol graphs allow up to 32 states per family, with 16–24 selected by validation. We set $\lambda = 1.0$ and use 128 invariant predicates. Appendix C records the splits, hyperparameter selection, and metric definitions.

We construct seen/unseen splits by application family: 10/5 families for SPA-Bench and 8/4 for AndroidWorld. Unseen-family trajectories stay out of matcher and invariant-miner training. We distinguish interface transfer within a known family from evaluation on a held-out family: template reuse addresses the first setting, whereas the second tests transfer under a new family template.

The comparison set includes ReAct, a three-demonstration few-shot agent, SOP-guided planning, GUI-explorer, and hierarchy-only. Table 1 details the controls. SOP-guided receives the same family templates; hierarchy-only also receives the same slot labels and aligned pairs. The hierarchy-only comparison therefore isolates the added cross-environment alignment more closely than the SOP comparison. Backbone-only comparisons serve as end-task references rather than representation-learning controls.

4.2 Task Success and Component Contributions

IPS reaches 58.9% SPA-Bench success and 54.1% AndroidWorld success (Table 2), improving over GUI-explorer by 5.2 and 6.7 percentage points. Against the same-template SOP control, the gains are 5.0 and 5.0 points. Hierarchy-only reaches 51.23% on AndroidWorld while sharing templates, slot labels, and trajectory pairs, placing cross-environment alignment at the center of the difference under this setup.

Removing the matcher produces the largest unseen-success drop among individual component removals: 3.74 points on AndroidWorld, with a 9.58-point alignment drop. Removing the recovery compiler reduces recovery by 7.91 points and task success by 1.43 points. Alignment and recovery therefore address different failure modes. Removing the invariant miner costs 2.08 points in task success. Appendix D.1 reports the per-benchmark breakdown.

On GUI-KRB, prior-knowledge error decreases from 6.8% for GUI-explorer to 5.9%, and dynamic-component error from 6.4% to 5.6%. Recovery success reaches 58.20%, compared with 46.36% for ReAct, a gain of 11.84 points. A recovery-only control reaches 51.34% recovery and 46.82% task

Table 1: Supervision supplied to the controls. SOP means standard operating procedure. The controls isolate procedural supervision from cross-environment matching.

Method	Procedural supervision	Cross-environment component
SOP-guided	Same family templates	Templates supplied as procedural guidance; no state matching
Hierarchy-only	Same templates, LLM slot labels, and trajectory pairs	Hierarchical planner without cross-environment contrastive alignment
IPS	Same templates, LLM slot labels, and trajectory pairs	Signature projections, contrastive matching, invariant checks, and recovery

Table 2: Task success (%). The first block gives the main benchmark comparison. The second reports the unseen-setting SOP control and component ablations; SPA and AW denote SPA-Bench and AndroidWorld. Additional reference systems appear in Appendix D.

Method	SPA	AW	AW recovery	AW state alignment
GUI-explorer	53.7	47.4	–	–
IPS	**58.9**	**54.1**	58.20	71.30
SOP-guided, same templates	53.9	49.1	–	–
Hierarchy-only	56.03	51.23	53.94	63.27
Without state matcher	55.42	50.36	53.58	61.72
Without invariant miner	56.81	52.02	55.03	66.84
Without recovery compiler	57.14	52.67	50.29	68.22

success. The reported protocol-violation reduction is 28.7%. These diagnostics use learned state assignments and complement end-task success.

4.3 Operational Validity and Matching Errors

Same-state pairs agree on the next-subgoal class in 78.4% of cases and on recovery options in 72.6% (Table 3). Joint agreement is 68.1%, compared with 48.7% for hierarchy-only. The representation therefore groups observations by more consistent operational roles than the controls, while leaving room to improve exact signature agreement.

Conditioning on all three signature predictions being correct raises next-subgoal agreement to 91.7%, separating signature-prediction quality from matching quality. A human audit of 150 pairs by three annotators judged 71.2% equivalent; the original evaluation reports an agreement coefficient of 0.71. Appendix E describes the error categories and qualitative cases.

Error propagation. The additional matching analysis reports 61.8% task success for correctly matched states and 34.3% for mismatches, with 0.7 versus 2.6 wasted steps on average (Table 4). Invariant checks or recovery catch 58.4% of mismatches. These conditional statistics connect matching quality with downstream execution; the gap is descriptive because task difficulty can affect both quantities.

Cross-site variation and failure interpretation. Table 11 instantiates the signature criterion with the flight-booking examples from Section 1. Different layouts can support the same next step, while visually similar forms can require different information. The matcher must therefore use task context alongside visual evidence.

The error analysis attributes disagreements to granularity mismatch (38.4%), context-dependent branching (29.7%), and state-matching error (31.9%). These proportions describe the disagreement subset. Granularity mismatch merges stages that need different actions or splits a reusable stage into separate classes; context-dependent branching changes the valid continuation with a user's choice or site requirement; state-matching error assigns an observation to the wrong class. Appendix E gives the denominators.

Table 3: Operational consistency of same-state pairs (%). Values are mean ± standard deviation over three seeds. These are empirical agreement tests of predicted states.

Metric	IPS	Hierarchy-only	Random
Next-subgoal agreement	78.4 ± 1.3	62.1 ± 1.8	34.8 ± 0.9
Recovery-option agreement	72.6 ± 1.6	55.3 ± 2.1	28.2 ± 1.1
Both agree	68.1 ± 1.9	48.7 ± 2.4	21.4 ± 1.3



Figure 6: Schematic comparison of action-level retry and protocol-level recovery after a grounding failure.

These categories affect execution in different ways. A merged state can make the planner act before its prerequisites hold, while an overly specific state can block transfer to a compatible interface. A missing branch condition can make the recovery compiler repeat a valid action in the wrong context. Invariant checks can flag a visible unmet requirement, but hidden context can mislead both the match and the check. End-task completion therefore remains the primary recovery measure.

4.4 Robustness and Scope

The Qwen2-VL-7B variant reaches 40.7% SPA-Bench success and 37.2% AndroidWorld success, compared with 28.4% and 26.8% for its ReAct control. Gains of 12.3 and 10.4 points show that the improvement extends beyond GPT-4o. This experiment covers two action backbones; the remaining encoder, labeling, and prompt choices are held fixed.

Injecting 10%, 20%, and 30% additional noise into positive-pair labels changes AndroidWorld unseen success from 54.10% to 53.24%, 51.87%, and 49.15%, respectively. At 20% injected noise, success

Table 4: Matching-error propagation on the AndroidWorld unseen setting, as reported in the additional analysis. The final row has mismatches as its denominator and is a detection rate, not task success.

Condition	Task success (%)	Mean wasted steps
Correct state match	61.8	0.7
State mismatch	34.3	2.6
Mismatches caught by invariant checks or recovery		58.4%

Table 5: Action-backbone comparison. Entries show SPA-Bench / AndroidWorld success (%).

Backbone	ReAct	IPS	Gain (percentage points)
GPT-4o	42.0 / 40.5	58.9 / 54.1	16.9 / 13.6
Qwen2-VL-7B	28.4 / 26.8	40.7 / 37.2	12.3 / 10.4

remains above the no-matcher value of 50.36%; at 30%, it falls below that value. Zero injected noise denotes the default labels, which already have an estimated natural error rate of about 12%. This corruption study measures sensitivity to the specified noise process. Tables 5 and 6 give the full results; Appendix F discusses their interpretation.

The granularity sweep reports unseen success of 49.6%, 51.7%, and 52.5% for coarse (8–12), medium (16–24), and fine (28–32) state graphs. Medium granularity balances alignment quality with graph-search cost, while the fine setting raises latency by 41% relative to medium. Appendix G describes the selection criterion. We report the sweep separately from the main AndroidWorld ablation.

Reported task-family gains are 7.8 points for form filling and 2.1 for open-ended browsing. Open-ended tasks expose fewer recurring protocols, so the abstraction has less structure to reuse. The evaluation covers English-language interfaces and the reported labeling and visual encoders; multilingual transfer and prompt variation remain future tests.

4.5 Construction and Execution Cost

Offline construction is a one-time per-family cost; online execution adds 6.7 seconds and 15.4% tokens per episode. Tables 12 and 13 report the measurements, with accounting details in Appendix H.

Table 6: Additional injected label noise and recovery isolation on the AndroidWorld unseen setting. The default labels already contain estimated natural noise.

Variant	Task success	Recovery	State alignment
IPS, 0% injected noise	54.10	58.20	71.30
IPS, 10% injected noise	53.24	57.41	69.18
IPS, 20% injected noise	51.87	55.63	65.72
IPS, 30% injected noise	49.15	52.08	59.41
Recovery-only	46.82	51.34	–
ReAct	40.50	46.36	–

Table 7: Reported granularity sweep on unseen-application success. The original analysis labels this sweep as unseen-application success.

Granularity	Success	State alignment	Recovery
Coarse, 8–12 states	49.6	62.4	51.8
Medium, 16–24 states	51.7	71.3	58.2
Fine, 28–32 states	52.5	73.1	59.4

5 Discussion and Limitations

IPS uses templates and trajectories that expose reusable structure. Recurring stages with compatible requirements offer the clearest fit; hidden state and few repeated stages reduce reuse. Learned signatures approximate the ideal relation, so checks and recovery inherit representation errors. The comparisons support matching under supplied supervision. The learning argument is conditional on capacity assumptions, and agreement rates denote operational identifiability. The open-source result tests one alternative action model; Appendices C, E, and F detail the measurement boundaries.

6 Conclusion

IPS represents GUI stages through preconditions, next subgoals, and recovery options. Matching, invariant checks, planning, and recovery improve benchmark success; supervision controls tie the gains to alignment. Error and cost analyses link representation to execution, while a conditional argument formalizes protocol reuse. Recurring-stage workflows offer the clearest setting for broader transfer.

AI USE STATEMENT

For manuscript editing, we used generative AI tools for grammatical correction and language polishing. The authors reviewed the edited text and take responsibility for the final manuscript, its claims, and its reported results. The LLM-based slot assignment, predicate extraction, and action generation described in Sections 3.2–4.1 are components of the research method.

ETHICS STATEMENT

We acknowledge the ICLR Code of Ethics. Our experiments use established GUI-agent benchmarks and benchmark trajectories; we do not collect, deanonymize, or redistribute new real-user account content. Human audits concern benchmark observations and aggregate judgments. GUI automation can expose private information or execute unwanted actions; deployment should restrict agents to authorized tasks, protect credentials, and require user control over consequential actions. Our English-language evaluation does not establish equitable performance across languages or user groups. Benchmark and model use must respect the providers' terms and licenses.

REPRODUCIBILITY STATEMENT

Section 4.1 specifies the backbones, training scale, and supervision controls. Appendix A gives the execution loop; Appendix B states the assumptions and scope of the learning argument; Appendix C describes splits, tuning, and evaluation metrics. Additional benchmark results and error analyses appear in Appendices D and E; Tables 12 and 13 document the measured costs. An anonymous code package is available at https://anonymous.4open.science/r/GUI_explorer-4E8D. The open-source backbone results in Table 5 document one alternative to proprietary action generation.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: An Open Agentic Framework that Uses Computers Like a Human. *arXiv preprint arXiv:2410.08164*, 2024. doi: 10.48550/arXiv.2410.08164. URL https://arxiv.org/abs/2410.08164.

Bizhe Bai, Jiakang Yuan, Hongming Wu, Xinyue Wang, Jie Ren, Siyao Chen, Yuchen Ya, Fan Bai, Pai Peng, Huafeng Qin, and Tao Chen. Efficient GUI Agents: A Systems Survey of Observation, Memory, Action, and Runtime Optimization. *arXiv preprint arXiv:2609.02309*, 2026. URL https://arxiv.org/abs/2609.02309.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Yixing Li, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye Hao, Jun Wang, and Kun Shao. SPA-Bench: A comprehensive benchmark for SmartPhone agent evaluation. *arXiv preprint arXiv:2410.15164*, 2024. doi: 10.48550/arXiv.2410.15164. URL https://arxiv.org/abs/2410.15164.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 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)*, pp. 21936–21959, 2025. doi: 10.18653/v1/2025.acl-long.1065. URL https://aclanthology.org/2025.acl-long.1065/.

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic Reward Modeling: Verifying GUI Agent via Progressive Trajectory-Grounded Interaction. *arXiv preprint arXiv:2602.00575*, 2026. doi: 10.48550/arXiv.2602.00575. URL https://arxiv.org/abs/2602.00575.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023. doi: 10.52202/075280-1220. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. *arXiv preprint arXiv:2410.05243*, 2024. doi: 10.48550/arXiv.2410.05243. URL https://arxiv.org/abs/2410.05243.

Yuyu Guo, Wenjie Yang, Siyuan Yang, Ziyang Liu, Cheng Chen, Yuan Wei, Yun Hu, Yang Huang, Guoliang Hao, Dongsheng Yuan, Jianming Wang, Xin Chen, Hang Yu, Lei Lei, and Peng Di. OpAgent: Operator Agent for Web Navigation. *arXiv preprint arXiv:2602.13559*, 2026. doi: 10.48550/arXiv.2602.13559. URL https://arxiv.org/abs/2602.13559.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. OpenWebVoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27545–27564, 2025. doi: 10.18653/v1/2025.acl-long.1336. URL https://aclanthology.org/2025.acl-long.1336/.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024. doi: 10.1109/CVPR52733.2024.01354. URL https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, March 2025. URL https://openreview.net/forum?id=5298fKGmv3.

Zhigen Li, Jianxiang Peng, Yanmeng Wang, Yong Cao, Tianhao Shen, Minghui Zhang, Linxi Su, Shang Wu, Yihang Wu, YuQian Wang, Ye Wang, Wei Hu, Jianfeng Li, Shaojun Wang, Jing Xiao, and Deyi Xiong. ChatSOP: An SOP-guided MCTS planning framework for controllable LLM dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17637–17659, 2025. doi: 10.18653/v1/2025.acl-long.863. URL https://aclanthology.org/2025.acl-long.863/.

Zichuan Lin, Feiyu Liu, Yijun Yang, Jiafei Lyu, Yiming Gao, Yicheng Liu, Zhicong Lu, Yangbin Yu, Mingyu Yang, Junyou Li, Deheng Ye, and Jie Jiang. UI-Voyager: A Self-Evolving GUI Agent Learning via Failed Experience. *arXiv preprint arXiv:2603.24533*, 2026. doi: 10.48550/arXiv.2603.24533. URL https://arxiv.org/abs/2603.24533.

Ziwei Liu, Borui Kang, Hangjie Yuan, Zixiang Zhao, Wei Li, Yifan Zhu, and Tao Feng. Continual GUI Agents. *arXiv preprint arXiv:2601.20732*, 2026. doi: 10.48550/arXiv.2601.20732. URL https://arxiv.org/abs/2601.20732.

OpenAI. GPT-4o system card. Technical report, OpenAI, 2024. URL https://openai.com/index/gpt-4o-system-card/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024. doi: 10.48550/arXiv.2405.14573. URL https://arxiv.org/abs/2405.14573.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, Xingran Zhou, Weizhi Chen, Sunhao Dai, Jingya Dou, Yichen Gong, Yuan Guo, Zhenlin Guo, Feng Li, Qian Li, Jinzhen Lin, Yuqi Zhou, Linchao Zhu, Liang Chen, Zhenyu Guo, Changhua Meng, and Weiqiang Wang. UI-Venus-1.5 Technical Report. *arXiv preprint arXiv:2602.09082*, 2026. doi: 10.48550/arXiv.2602.09082. URL https://arxiv.org/abs/2602.09082.

Gaurav Verma, Rachneet Kaur, Nishan Srishankar, Zhen Zeng, Tucker Balch, and Manuela Veloso. AdaptAgent: Adapting multimodal web agents with few-shot learning from human demonstrations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20635–20651, 2025. doi: 10.18653/v1/2025.acl-long.1008. URL https://aclanthology.org/2025.acl-long.1008/.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *Advances in Neural Information Processing Systems*, volume 37, pp. 2686–2710, 2024a. doi: 10.52202/079017-0088. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0520537ba799d375b8ff5523295c337a-Abstract-Conference.html.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b. doi: 10.48550/arXiv.2409.12191. URL https://arxiv.org/abs/2409.12191.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. AutoDroid: LLM-powered task automation in Android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 543–557, 2024. doi: 10.1145/3636534.3649379. URL https://doi.org/10.1145/3636534.3649379.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. GUI-explorer: Autonomous exploration and mining of transition-aware knowledge for GUI agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5650–5667, 2025. doi: 10.18653/v1/2025.acl-long.282. URL https://aclanthology.org/2025.acl-long.282/.

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. *arXiv preprint arXiv:2603.26266*, 2026. doi: 10.48550/arXiv.2603.26266. URL https://arxiv.org/abs/2603.26266.

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. *arXiv preprint arXiv:2404.07972*, 2024. doi: 10.48550/arXiv.2404.07972. URL https://arxiv.org/abs/2404.07972.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. *arXiv preprint arXiv:2412.04454*, 2024. doi: 10.48550/arXiv.2412.04454. URL https://arxiv.org/abs/2412.04454.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-UI: Visual grounding for GUI instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, 2025. doi: 10.18653/v1/2025.findings-acl.1152. URL https://aclanthology.org/2025.findings-acl.1152/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. AppAgent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023. doi: 10.48550/arXiv.2312.13771. URL https://arxiv.org/abs/2312.13771v2. Version 2, December 22, 2023.

Yichun Zhang, Xiangwu Guo, Yauhong Goh, Jessica Hu, Zhiheng Chen, Xin Wang, Difei Gao, and Mike Zheng Shou. ShowUI-Aloha: Human-Taught GUI Agent. *arXiv preprint arXiv:2601.07181*, 2026. doi: 10.48550/arXiv.2601.07181. URL https://arxiv.org/abs/2601.07181.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61349–61385. PMLR, 2024. URL https://proceedings.mlr.press/v235/zheng24e.html.

Yuhao Zheng, Li'an Zhong, Yi Wang, Rui Dai, Kaikui Liu, Xiangxiang Chu, Linyuan Lv, Philip Torr, and Kevin Qinghong Lin. Code2World: A GUI World Model via Renderable Code Generation. *arXiv preprint arXiv:2602.09856*, 2026. doi: 10.48550/arXiv.2602.09856. URL https://arxiv.org/abs/2602.09856.

Sibo Zhu, Wenyi Wu, Kun Zhou, Stephen Wang, and Biwei Huang. Hybrid Self-evolving Structured Memory for GUI Agents. *arXiv preprint arXiv:2603.10291*, 2026. doi: 10.48550/arXiv.2603.10291. URL https://arxiv.org/abs/2603.10291.

A Execution Algorithm

Algorithm 1 specifies the loop for protocol inference, planning, grounding, and recovery. Both the state estimate and the invariant check are predictions, so a passing check alone does not establish the correctness of the abstract state.

Algorithm 1 Execution with Identifiable Protocol States

Require: Goal g, observation history $o_{\leq t}$, protocol graph G

Encode current observation as h_t
Compute signature projections and embedding ν_t; infer z_t
Predict and check the state constraints I_{z_t}
Select a legal successor z^*_{t+1} according to Eq. (6)
Ground the next state: $a_t \sim p_{\theta_a}(a_t \mid z^*_{t+1}, o_t, g)$
Execute a_t and observe the result
if the intended transition fails **then**
    Classify the failure from the protocol state and failure signal
    Apply rollback, alternate path, information request, or tool invocation
end if
Continue from the updated observation until completion or the task budget

B Assumptions and Scope of the Adaptation Argument

The parameter space is $\Theta_z \times \Theta_a$, with θ_z controlling the protocol factor and θ_a controlling the grounding factor. We write $d_z = \dim(\Theta_z)$ and $d_a = \dim(\Theta_a)$ for the corresponding class dimensions. Let $L(\theta_z, \theta_a)$ denote expected target loss. The argument in Section 3.1 rests on the following assumptions.

A1: Transferability and realizability.
A source-learned protocol θ^*_z is fixed independently of the target adaptation sample and satisfies $\inf_{\theta_a} L(\theta^*_z, \theta_a) \leq \epsilon/2$. This assumption concerns the transferred representation and is task dependent.

A2: Sampling and loss.
The loss lies in $[0, 1]$, and target training episodes are independent and identically distributed. Observations within one trajectory can remain correlated and therefore do not by themselves form independent adaptation samples.

A3: Restricted-class capacity.
After fixing θ^*_z, the loss class has Rademacher complexity $\widetilde{O}(\sqrt{d_a/n})$. This scaling describes the function class and does not follow from the neural-network parameter count alone.

Uniform-convergence argument. With the protocol fixed, empirical risk minimization searches only the grounding class. The uniform-convergence argument in the main text combines the restricted-class estimation error with the approximation error in A1. It gives the sufficient sample scaling $\widetilde{O}(d_a/\epsilon^2)$, with logarithmic factors and the confidence parameter suppressed. The comparison with $\widetilde{O}((d_z + d_a)/\epsilon^2)$ also assumes that the joint loss class has capacity scaling with $d_z + d_a$. These upper bounds motivate the capacity comparison; they do not characterize the exact sample complexity of joint learning or of the implemented GUI system.

Relation to the experiments. Next-subgoal and recovery agreement diagnose whether the agents share a useful protocol. They do not test the target-risk optimum in A1. Backbone variation measures one source of implementation dependence, while label corruption probes matcher sensitivity. These experiments target neither A2 nor A3. The smaller gains in open-ended browsing mark a setting where shared task structure contributes less. The theoretical argument therefore remains conditional on A1–A3.

C Detailed Experimental Protocol

Inputs and supervision. Training uses successful task-family trajectories, templates of 4–8 subgoals, automatic LLM slot assignments, and an LLM-derived invariant vocabulary. The matcher uses 5,000 aligned trajectory pairs from 15 families, including form filling, search–filter–confirm, settings navigation, and mixed workflows. Three LLM calls and majority voting produce each slot assignment. Optional steps receive optional-slot tags. Human audits of noise and state quality are separate from template-writing effort.

Splits. SPA-Bench assigns 10 of 15 application families to the seen set and 5 to the unseen set. AndroidWorld uses an 8/4 split. We assign families before training, and exclude unseen-family trajectories from matcher and invariant-miner training. A held-out validation set contains 15% of seen-family trajectories. New-interface transfer within a known family and transfer to a held-out family rely on different assumptions; template reuse addresses the former.

Table 8: Additional task-success comparisons (%). A dash marks an unreported result.

Agent	Input	SPA-Bench	AndroidWorld
Human	Screen	–	80.0
AppAgent	SoM	14.0	14.9
AutoDroid	a11y	12.0	15.7
M3A	SoM	42.0	40.5
MobileAgentV2	SoM	20.0	–
SeeAct	SoM	12.0	15.5
Aguvis	Screen	–	37.1
Aria-UI	Screen	–	44.8
UGround	Screen	–	32.8
GUI-explorer	SoM	53.7	47.4
IPS	SoM	**58.9**	**54.1**

Granularity and hyperparameters. The reported default uses 16–24 states per family. A BIC-like score selects this range by balancing held-out next-subgoal agreement with graph complexity, using seen-family validation data. We search $\tau \in \{0.05, 0.07, 0.10\}$, $\lambda \in \{0.5, 1.0, 2.0\}$, and learning rate in $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$. The reported values are $\tau = 0.07$ and $\lambda = 1.0$. The matcher trains for 10 epochs on four A100 GPUs. We use the same selected hyperparameters across benchmarks without per-benchmark retuning.

Backbones and features. The default agent uses GPT-4o for action generation and frozen CLIP ViT-L/14 for observation encoding. Three projection heads, a two-layer MLP, and layer normalization produce the protocol-state embeddings. The invariant vocabulary contains 128 predicates; the planner uses 256-dimensional goal and state projections. The open-source action-backbone comparison uses Qwen2-VL-7B.

Recovery success. A recovery attempt succeeds when the agent reaches its intended next protocol state within three additional actions after intervention. We divide by the number of detected transition failures. We register a failure when the post-action observation maps to the same state as before or to an unexpected state outside $\mathcal{N}(z_t)$. Because this operational definition uses state prediction, coarse granularity can classify legitimate within-state actions as failures.

Protocol violations. We count a violation when the agent executes a transition to a state outside $\mathcal{N}(z_t)$ or fails to meet the predicted target-state constraints. The rate divides violations by attempted transitions. These model-relative metrics complement benchmark task success and measure legality under the learned protocol.

Variability. The operational-consistency table reports the mean and standard deviation over three seeds. Several other tables report point estimates only. We retain their reported precision and do not infer confidence intervals, statistical significance, or episode counts from those values.

D Additional Benchmark Results

Table 8 retains the additional comparison rows from the original manuscript. The systems use different input representations, so the supervision controls in the main text provide the clearest evidence for the contribution of protocol matching. Set-of-marks (SoM) input augments a screenshot with element markers; accessibility (a11y) input uses the interface accessibility representation.

Table 9: GUI-KRB error rates (%; lower is better).

Model	Prior-knowledge error	Dynamic-component error
Qwen2-VL	22.8	19.8
Qwen2.5-VL	16.6	14.0
Gemini 2.0 Flash	15.2	11.2
GPT-4o	18.2	13.4
GUI-explorer without ranker	9.8	6.8
GUI-explorer	6.8	6.4
IPS	**5.9**	**5.6**

D.1 Per-Benchmark Ablations

Removing the matcher causes the largest task-success drop among the three individual component removals. The hierarchy-only comparison preserves templates, labels, and aligned pairs while omitting cross-environment contrastive matching. This makes it a closer supervision control than the comparison with ReAct.

E Matching Errors and Qualitative Interpretation

Table 10: Per-benchmark component ablations (reported success, %). GUI-KRB retains the original manuscript's reported success column, which measures a separate outcome from AndroidWorld task completion.

Variant	SPA-Bench	AndroidWorld	GUI-KRB
IPS	58.90	54.10	94.10
Without state matcher	55.42	50.36	91.83
Without invariant miner	56.81	52.02	92.64
Without recovery compiler	57.14	52.67	93.01
Hierarchy-only	56.03	51.23	92.18

The error-category proportions discussed in Section 4.3 describe the disagreement subset and sum to 100%. The main agreement table reports 78.4% next-subgoal agreement, while the error analysis reports 21.9% disagreement from its own subset. We keep their denominators separate rather than combine the two estimates.

Table 11 in the main text gives conceptual cases that separate visual similarity, shared task stages, and context-dependent variation.

The propagation analysis in Table 4 reports that invariant checks or recovery catch 58.4% of mismatches. Catching a mismatch and completing the episode measure different outcomes. The reported recovery-category frequencies are rollback 73.4%, alternate path 18.2%, information request 6.1%, and tool invocation 2.3%; these values describe category frequency, not category-specific success rates. A per-family gallery of audited failures is not included in the supplied evidence.

Table 11: Conceptual examples explaining the matching criterion. These are illustrative workflow cases, not additional audited trajectories or experimental outcomes.

Case	Relevant signature distinction	Interpretation
Different passenger-detail layouts	Same required details and admissible confirmation step	Candidate positive match if recovery options also agree
Passenger versus billing form	Similar visual fields, different prerequisites and next subgoal	A visually similar pair need not share a protocol state
Optional seat selection	Optional continuation can depend on fare or user choice	A slot label alone can hide conditional compatibility
Site requiring extra verification	A prerequisite exists on only one site	Split states at this stage instead of forcing equivalence
Dynamic content or user-specific branch	The visible screen may omit a condition governing the next step	A match can fail despite visual similarity; checks may also miss hidden context

F Backbone and Label-Noise Analyses

Label corruption reassigns a fraction of observations to incorrect subgoal slots before training, and the experiment measures sensitivity to this process. The analysis covers this corruption setting; tests with another labeling LLM, prompt paraphrases, a different visual encoder, or systematically biased annotations remain separate questions. The recovery-only control adds the compiler to ReAct without protocol matching, isolating part of the recovery gain from the rest of IPS.

G Protocol Granularity

The validation criterion balances alignment quality against graph size. The reported BIC-like score favors about 18–22 states for form-filling families and 20–26 for mixed workflows. Fine graphs improve the reported metrics but raise graph-search latency by 41% relative to medium; coarse graphs lose 6.4 recovery points relative to medium. We describe medium granularity as the selected trade-off because it balances these outcomes.

H Offline and Online Cost Profiles

Table 12: One-time offline construction per task family, averaged over 15 families. The measurements are approximate.

Item	Per-family value
Trajectories	~ 90
Aligned pairs	~ 333
LLM labeling calls, three-vote assignment	~ 5.4K
LLM input / output tokens	~ 4.3M / ~ 0.4M
Wall-clock time	~ 1.1 hours
Estimated API cost at run-time pricing	$\sim$USD 14
Human template writing	~ 10 minutes

Template-writing time excludes other research labor, including successful-trajectory collection and human audits. API cost depends on model pricing and reflects a historical estimate. At the reported average size, constructing more families increases the offline workload roughly with the number of families. The experiment does not measure routing cost as the family collection grows.

We precompute the graph for the selected family, which bounds one source of online construction cost. Graph size, routing, and interface-grounding difficulty can still affect latency. The 13.8% latency increase and 15.4% token increase measure different resources.

Table 13: Online cost per episode. Component percentages use ReAct's total episode latency as the denominator; rounded times need not reproduce each reported percentage exactly.

Item	ReAct	IPS	Change
LLM tokens per episode	$\sim$ 5.2K	$\sim$ 6.0K	+15.4%
Episode latency (seconds)	$\sim$ 48.6	$\sim$ 55.3	+13.8%
Matching contribution (seconds)	–	+2.5	+5.2%
Invariant checks (seconds)	–	+1.7	+3.4%
Planning and grounding (seconds)	–	+2.5	+5.2%

Offline construction averages about 90 trajectories, 333 aligned pairs, and 5,400 LLM labeling calls per family. The reported input/output token counts are 4.3M/0.4M, with approximately 1.1 hours of wall-clock time, USD 14 of API cost, and 10 minutes of template writing per family. The resulting graphs can be reused across episodes in the same family.

Online execution uses about 6.0K LLM tokens per episode, compared with 5.2K for ReAct. Latency rises from 48.6 to 55.3 seconds; the added 6.7 seconds comprise approximately 2.5 seconds for matching, 1.7 for invariant checks, and 2.5 for planning and grounding. With family size and data volume fixed, total preprocessing grows with the number of families. Template reuse removes repeated template writing for a new interface within a known family, while trajectory collection, labeling, and grounding remain part of extending coverage.

I Related Evaluation, Verification, and Training Directions

Evaluation scope and reusable experience. Mind2Web studies generalization across websites and domains (Deng et al., 2023). VisualWebArena emphasizes visually grounded web tasks (Koh et al., 2024), while OSWorld evaluates open-ended tasks in real computer environments (Xie et al., 2024). BrowserGym unifies observation and action interfaces across web benchmarks to support consistent evaluation (Le Sellier de Chezelles et al., 2025). These settings extend beyond our smartphone evaluation and motivate tests of protocol-signature transfer across operating systems and richer visual tasks. Agent S combines hierarchical planning with experience retrieval (Agashe et al., 2024); it offers a relevant comparison for separating explicit state matching from retrieved procedural knowledge.

Verification and online learning. VAGEN combines trajectory evidence with active environment probing to verify outcomes under partial observability (Cui et al., 2026). Such verification could complement IPS invariant checks, which may share errors with the state matcher. UI-Venus-1.5 combines GUI mid-training, full-trajectory online reinforcement learning, and model merging (Venus Team et al., 2026). OpAgent combines online reinforcement learning with modular planning, grounding, reflection, and summarization (Guo et al., 2026). These works study training and control mechanisms beyond our template-supervised matcher. Their published success rates use different models and evaluation protocols, so the IPS tables do not provide a direct comparison.

End-to-end efficiency. The systems survey by Bai et al. (2026) distinguishes observation, memory, action, and runtime costs. This perspective supports reporting verification and retrieval overhead alongside execution success. Our offline and online cost tables cover the measured IPS configuration. They leave the costs of adding tutorial retrieval, structured memory, world-model prediction, or active verification for future measurement.