PGGA: A Plan-Grounded GUI Agent for Automated Device Support

Lei Hsiung*[1,2], Zhiyu Chen[2], Seonhoon Kim[2], Qun Liu[2]

[1]Dartmouth College, [2]Amazon.com, Inc.

Abstract

Current GUI agents struggle with multi-step digital device support. We demonstrate that this failure stems from a procedural knowledge deficit, as agents rely on zero-shot visual guessing instead of executing verified instructions. To address this, we introduce the Plan-Grounded GUI Agent (PGGA), framing interface navigation as a knowledge-execution problem by conditioning low-level actions on step-by-step text plans. Evaluated on our novel Device-Support Interaction Benchmark (DSIB), results reveal a severe capability gap: state-of-the-art models like GTA1-7B show perfect operational understanding but are poor at visual grounding. Conversely, our fine-tuned 2B-parameter PGGA achieves a 54.39% Task Success Rate when guided by expert plans, proving that explicit procedural grounding is essential for robust, parameter-efficient GUI automation.

1 Introduction

Autonomous Graphical User Interface (GUI) agents powered by Large Multimodal Models (LMMs) have demonstrated significant potential for automating digital workflows (Zhang et al., 2025; Nguyen et al., 2025; Hong et al., 2024). These agents aim to execute multi-step, long-horizon tasks by using the model's reasoning and decision-making abilities to translate visual interface states into actionable commands. However, deploying these agents in real-world environments, particularly for complex device support, remains a persistent challenge. In this context, device support refers to the multi-step, highly specific procedures required to configure, maintain, or adapt digital devices—such as navigating deep-nested system menus to alter localization inputs or managing granular security and privacy configurations.

A fundamental challenge in this domain is planning. While LMMs exhibit strong zero-shot reasoning capabilities, their efficacy in situated GUI navigation is often bottlenecked by the semantic gap between high-level user instructions and the low-level visual control required for execution. Existing approaches predominantly rely on zero-shot visual planning, implicitly expecting the model to infer complex, multi-step procedural logic purely from visual observations (Wu et al., 2025; Gou et al., 2025; Yan et al., 2023a). We argue that the frequent failure of GUI agents in these scenarios stems not merely from a deficit in visual grounding, but from an acute lack of domain-specific procedural knowledge. When confronted with unfamiliar interfaces, agents lacking structural priors default to inefficient exploration, leading to compounding errors and ultimate task failure.

This paper targets the domain of device support: tasks that are highly deterministic given a standard operating procedure but remain opaque to an unguided agent. To address this, we introduce the Plan-Grounded GUI Agent (PGGA), as illustrated in Figure 1. Instead of relying on the Vision-Language Model's (VLM) internal, implicit knowledge to guess the next action, PGGA grounds its execution in authoritative, step-by-step action plans retrieved from an external knowledge base (Hayashi et al., 2025). By formulating the task as the execution of a verified procedure rather than zero-shot exploration, we demonstrate a significant improvement in grounding accuracy and task completion. Through rigorous evaluation across varied instruction formats (task-only, retrieved action plans, and expert-annotated plans), we empirically validate that injecting domain knowledge directly into the observation space substantially alleviates the planning bottleneck for GUI agents.

To evaluate this device support capability, we curate the Device-Support Interaction Benchmark (DSIB). DSIB features 57 high-intent device config-

*Work done during an internship at Amazon.

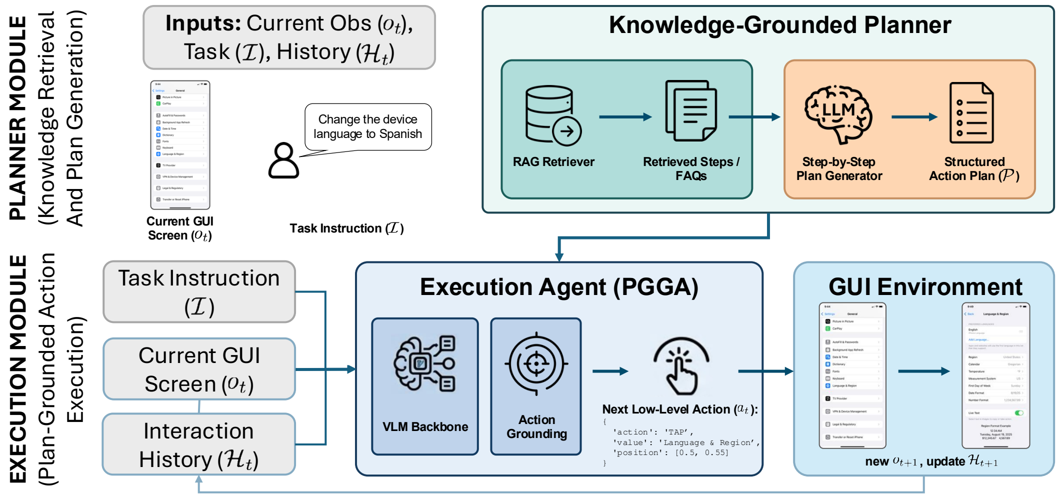

Figure 1: Plan-Grounded GUI Agent (PGGA) Framework. Given a natural language task and the current visual state, an intermediate Planner module queries an external Knowledge Base to retrieve relevant documentation and synthesizes it into a step-by-step Action Plan. The model then predicts the next action by heavily conditioning on this explicit procedural plan, alongside the original task, current screenshot, and interaction history.

uration tasks, comprising 241 individual steps collected directly on a mobile device. The benchmark focuses explicitly on deep-nested system configurations and complex visual disambiguation, requiring navigation depths of up to five hierarchical menu levels across diverse functional domains.

We empirically validate PGGA by evaluating performance across varying degrees of plan quality: 1) Task Only, 2) Task with a Retrieved Action Plan, and 3) Task with an Expert-Annotated Action Plan. Our experimental results confirm our core hypothesis: domain knowledge is the primary bottleneck. For instance, relying solely on task instructions yields a mere 7.02% Task Success Rate (SR) for PGGA-2B. However, when grounded in expert-annotated plans, PGGA-2B achieves a state-of-the-art 54.39% Task SR. This substantial performance gap demonstrates that equipping GUI agents with authoritative, step-by-step priors drastically improves their robustness and next-action prediction accuracy in complex digital environments. We summarize our main contributions as follows:

- We introduce the Device-Support Interaction Benchmark (DSIB), a novel dataset of complex, multi-step device support tasks designed to evaluate the procedural execution capabilities and visual disambiguation of GUI agents.
- We propose the Plan-Grounded GUI Agent (PGGA), a framework that reformulates interface navigation from a zero-shot exploration problem into a knowledge-execution problem by conditioning low-level actions on step-by-step text plans.
- We empirically demonstrate that explicit procedural grounding is essential for robust GUI automation. Our fine-tuned, parameter-efficient 2B model significantly outperforms larger zero-shot baselines when guided by structural priors, isolating domain knowledge as the current fundamental bottleneck.

2 Methodology

We model the environment as a Partially Observable Markov Decision Process (POMDP). At each timestep t, the agent receives an observation $o_t \in \mathcal{O}$ (the current GUI screenshot) and must predict an action $a_t \in \mathcal{A}$ based on a natural language task description $\mathcal{I}$ and the interaction history $h_t = \{(o_0, a_0), \ldots, (o_{t-1}, a_{t-1})\}$. Standard GUI agents learn a policy $\pi(a_t | o_t, h_t, \mathcal{I})$.

We hypothesize that learning this direct mapping is sub-optimal for complex device support, as it forces the agent to rely on ungrounded visual guessing when encountering unfamiliar interfaces. Inspired by the natural human cognitive workflow of consulting external manuals to resolve device issues (detailed in Appendix B), the PGGA framework introduces an intermediate planning module. Given the task $\mathcal{I}$ and current state o_t, the Planner queries an external Knowledge Base to retrieve a relevant document D. The Planner synthesizes D into a structured Action Plan $P = \{p_1, p_2, \ldots, p_n\}$, where each p_i represents a discrete operational step. The PGGA policy is thus reformulated to condition heavily on this explicit plan: $\pi_\theta(a_t | o_t, h_t, \mathcal{I}, P)$.

2.1 Model Architecture and Training

As illustrated in our system architecture, PGGA integrates the generated Action Plan alongside the visual state and action history. To optimize the agent for plan-following, we fine-tune a parameter-efficient base model, ShowUI-2B (Lin et al., 2025). The fine-tuning process utilizes the Mind2Web dataset (Deng et al., 2023), where we programmatically construct the instruction set by converting ground-truth interaction traces into readable Action Plans using an advanced LLM. This phase trains the PGGA-2B model to strictly ground its visual feature extraction and action generation in the provided text plan.

2.2 Device-Support Interaction Benchmark (DSIB)

We curate the DSIB, building up 57 high-intent device support and configuration tasks, comprising a total of 241 individual steps, designed to mimic real-world mobile setup scenarios (Liu et al., 2025). All task trajectories within the dataset were collected directly on an Apple iPhone XR and manually annotated by the authors. DSIB explicitly focuses on deep-nested system configurations and complex visual disambiguation, requiring navigation depths of 3 to 5 hierarchical menu levels (see DSIB task example in Appendix C).

The dataset evaluates breadth and stability across six functional domains: Localization & Input (13.5%), System Customization (22.5%), Connectivity & Networking (17.5%), Security & Privacy (12.5%), Accessibility & Vision (19%), and Battery & Power Management (15%). Tasks frequently feature near-neighbor distractors (e.g., distinguishing “DD/MM/YYYY” from “DD-MM-YYYY”), demanding high-fidelity visual grounding. Instructions vary in semantic diversity, ranging from technical directives to colloquial user queries.

3 Experiments

3.1 Experimental Setup

Models. To rigorously evaluate the impact of plan grounding, we benchmark PGGA-2B against two representative baselines:

- **ShowUI-2B (Lin et al., 2025):** The model serving as our base architecture, evaluated zero-shot to demonstrate the delta achieved by our plan-grounding fine-tuning.
- **GTA1-7B (Yang et al., 2026):** A larger, state-of-the-art 7B parameter VLM specialized for GUI tasks, serving as a high-capacity baseline.

We evaluate these models across three distinct instruction formats (see Appendix D for examples):

1. **Task Only** (zero-shot exploration): Only provide the task (e.g., Change the device language to Spanish),
2. **Task + Action Plan via Retrieval**: The action plan is constructed using a search-enabled large language model, and
3. **Task + Expert-Annotated Action Plan** (the theoretical upper bound for plan quality): The action plan is constructed from all the ground-truth steps.

Evaluation Metrics. Agents are evaluated on four key axes to decouple reasoning from visual grounding:

- **Task Success Rate (Task SR):** A strict, binary metric indicating if the final target state was successfully reached and the correct ultimate action was executed.
- **Element Accuracy (Elem. Acc.):** The accuracy of correctly predicting the spatial coordinates or bounding box of the target UI element.
- **Operation Accuracy (Op. Acc.):** The accuracy of selecting the correct interaction type (e.g., TAP, TYPE, SCROLL) independent of element localization.
- **Step Success Rate (Step SR):** The percentage of individual steps within a task completed correctly, providing partial credit for trajectory progress.

3.2 Results

Table 1 presents the comparative performance of our finetuned PGGA-2B against the baselines across the three instruction formats.

Zero-Shot Performance (Task Only). Without procedural guidance, all models struggle significantly to complete multi-step GUI navigation. PGGA-2B achieves the highest Task Success Rate (Task SR) at 7.02% and Step Success Rate (Step SR) at 51.45%. Notably, the GUI-specific GTA1-7B completely fails to complete any full trajectories (0.00% Task SR), highlighting the severe limitations of purely visual zero-shot exploration.

Retrieval-Augmented Performance. Augmenting the instruction with a retrieved action plan yields noticeable improvements. GTA1-7B exhibits

Instruction Format	Models	Task SR.	Elem. Acc.	Op. Acc.	Step SR.
Task Only	ShowUI-2B	1.75%	46.47%	65.97%	35.27%
	GTA1-7B	0.00%	40.66%	**98.76%**	40.66%
	PGGA-2B	**7.02%**	**57.26%**	90.87%	**51.45%**
Task Only + Action Plan (Retrieval)	ShowUI-2B	3.51%	46.47%	70.95%	40.25%
	GTA1-7B	**29.82%**	65.15%	**97.10%**	65.15%
	PGGA-2B	19.30%	**76.76%**	87.55%	**68.05%**
Task + Action Plan (Expert-Annotated)	ShowUI-2B	3.51%	50.21%	88.38%	48.96%
	GTA1-7B	45.61%	82.99%	**99.59%**	82.57%
	PGGA-2B	**54.39%**	**91.28%**	95.02%	**88.38%**

Table 1: Performance comparison of GUI navigation agents on the DSIB dataset.

the strongest task-level performance in this setting, jumping to a 29.82% Task SR, outperforming PGGA-2B (19.30% Task SR). However, PGGA-2B maintains a higher Element Accuracy (76.76%) and Step SR (68.05%), suggesting it follows individual retrieved steps more accurately, even if the overall retrieved plan ultimately fails to reach the final target state.

Expert-Annotated Performance. Grounding the models in perfect, expert-annotated plans unlocks their latent capabilities. PGGA-2B achieves state-of-the-art results in this setting, dominating with a 54.39% Task SR, 91.28% Element Accuracy, and 88.38% Step SR. GTA1-7B also shows massive gains (45.61% Task SR), demonstrating the efficacy of using a high-quality action plan in bridging the procedural knowledge gap for high-capacity visual models.

4 Discussion

The Knowledge Bottleneck. When models are restricted to the Task-Only format, performance collapses across all architectures. Notably, the GUI-specific GTA1-7B fails entirely, achieving a 0.00% Task SR. However, when provided with an Expert-Annotated Action Plan, PGGA-2B's Task SR surges to 54.39%. This massive improvement demonstrates that models possess the requisite visual acuity to interact with interfaces (as evidenced by PGGA-2B's 91.28% Element Accuracy under expert guidance), but fundamentally lack the intrinsic procedural knowledge required to sequence these interactions autonomously.

Parameter Efficiency via Grounding. Our fine-tuned PGGA-2B consistently outperforms the much larger GTA1-7B model in Element Accuracy and Step SR when provided with structured plans. The behavior of the 7B-parameter baseline is revealing: while it maintains near-perfect Operation Accuracy (peaking at 99.59%, indicating a flawless semantic understanding of action types), its lower Element Accuracy (82.99%) compared to PGGA-2B suggests that scaling parameters alone does not guarantee precise pixel-level grounding in dense GUIs without specialized fine-tuning. PGGA proves that a highly efficient 2B-parameter model, when properly conditioned to follow textual plans, can achieve state-of-the-art visual grounding.

The Retrieval Gap. An important nuance in our findings is the performance degradation between Expert-Annotated plans and Retrieved plans. While PGGA-2B achieves a 54.39% Task SR with expert plans, it drops to 19.30% with retrieved plans. Interestingly, the larger GTA1-7B model demonstrates superior robustness to noisy or imperfect retrieved plans, achieving a 29.82% Task SR in this setting. This highlights a critical frontier for future work: while PGGA-2B excels at executing perfect instructions (achieving the highest Step SR at 68.05%), its lower capacity makes it less resilient to recovering from the cascading errors induced by sub-optimal retrieval compared to the 7B-parameter architecture.

5 Conclusion

This paper introduces the Plan-Grounded GUI Agent (PGGA), a framework that reframes automated device support from a problem of zero-shot visual exploration to one of grounded procedural execution. By formalizing the integration of retrieved action plans into the VLM observation space, we demonstrate that a parameter-efficient 2B model can achieve profound improvements in visual grounding and task success. Our findings isolate long-horizon planning and domain knowledge as the current fundamental bottlenecks in GUI agent capabilities. While PGGA establishes a new standard for plan execution, our analysis of the *retrieval gap* underscores the necessity for future research focused on improving the robustness of the knowledge retrieval mechanisms and error-recovery policies.

Limitations

While PGGA demonstrates the efficacy of plan-grounded GUI navigation, the stark performance contrast between expert-annotated and retrieved plans, the "Retrieval Gap", highlights a key limitation. Our 2B-parameter model, though highly precise when provided with perfect instructions, lacks the capacity to robustly recover from noisy or suboptimal retrieved plans compared to larger models like GTA1-7B. Furthermore, relying on human-annotated expert plans presents a significant scalability bottleneck for real-world deployment. Future work will focus on bridging this gap through two primary avenues: 1) scaling the PGGA architecture to larger parameter counts to improve inherent error-recovery capabilities when faced with imperfect retrieval, and 2) developing automated self-reflection and re-planning pipelines. By utilizing reinforcement learning or prompting strategies to help models detect execution failures and iteratively refine their action plans during navigation, we aim to eliminate the reliance on human annotation and achieve robust, fully autonomous device support.

References

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*.

Hiroaki Hayashi, Bo Pang, Wenting Zhao, Ye Liu, Akash Gokul, Srijan Bansal, Caiming Xiong, Semih Yavuz, and Yingbo Zhou. 2025. Self-abstraction from grounded experience for plan-guided policy refinement. *arXiv preprint arXiv:2511.05931*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2024. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*.

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*.

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

Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, Xiaoyu Liang, WenHao Wang, Tianze Wu, Zhengxi Lu, Siheng Chen, LiLinghao, Hao Wang, Guanjing Xiong, and 2 others. 2025. LLM-powered GUI agents in phone automation: Surveying progress and prospects. *Transactions on Machine Learning Research*.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, and 1 others. 2025. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, and 1 others. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer:

Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. Computer Vision Foundation / IEEE.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *CoRR*, abs/2212.04088.

Hang Wu, Hongkai Chen, Yujun Cai, Chang Liu, Qingwen Ye, Ming-Hsuan Yang, and Yiwei Wang. 2025. Dimo-gui: Advancing test-time scaling in gui grounding via modality-aware visual reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26257–26267.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, and 1 others. 2023a. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, and 1 others. 2023b. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *Preprint*, arXiv:2310.11441.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, JUNZHE HUANG, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. 2026. GTA1: GUI test-time scaling agent. In *The Fourteenth International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2024. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *arXiv preprint arXiv:2408.15978*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*.

Appendix

A Related Work

Graphical User Interface (GUI) Agents. Recent advancements in Large Multimodal Models (LMMs) have catalyzed the development of autonomous agents capable of navigating digital environments (Zhang et al., 2025; Zheng et al., 2024; Hong et al., 2024). These systems generally fall into two paradigms: training-free and training-based models. Training-free approaches rely on proprietary models like GPT-4 to process external UI representations, such as accessibility trees or OCR-derived bounding boxes, via zero-shot prompting (Yang et al., 2023; Yan et al., 2023b). Conversely, training-based models fine-tune open-weight architectures on massive datasets of web and mobile interactions to directly predict actions from raw pixels (Hong et al., 2024; Li et al., 2024). While these models have significantly closed the visual perception gap, they predominantly focus on generalized navigation. In contrast, PGGA explicitly targets the domain of automated device support. We recognize that visual acuity alone is insufficient; without the domain-specific procedural knowledge required for complex system configurations, models default to inefficient and error-prone guessing.

Planning and Reasoning in Autonomous Agents. To tackle long-horizon tasks, prior work has extensively explored hierarchical planning and tool use (Yao et al., 2023; Schick et al., 2023). In the digital agent domain, systems like WebPilot (Zhang et al., 2024), AgentOccam (Yang et al., 2024), and ADaPT (Prasad et al., 2023) employ complex, multi-agent prompting frameworks using massive closed-source models to iteratively decompose tasks and replan upon failure. While effective in unconstrained web browsing, these zero-shot exploratory methods are highly inefficient—and potentially unsafe—for deterministic device support. Operating deep within system settings demands a high degree of operational robustness to avoid unintended configurations. PGGA addresses this by replacing open-ended exploration with authoritative plan execution. By grounding a parameter-efficient 2B model in retrieved procedural steps, we ensure a more reliable, trustworthy, and computationally efficient trajectory than complex multi-agent prompting cascades.

Grounded Language Understanding and Synthetic Data. Mapping high-level natural language instructions to executable environment actions is a core challenge in grounded language understanding (Shridhar et al., 2020; Song et al., 2022). For digital agents, datasets like Mind2Web (Deng et al., 2023) and AndroidInTheWild (Rawles et al., 2024) have provided critical interaction traces. Recently, synthetic data generation has been widely adopted to augment these datasets, utilizing powerful LLMs to annotate trajectories, filter failures, or synthesize new task queries (Qi et al., 2024; Bai et al., 2024; Lee et al., 2024). Our methodology builds upon this paradigm but shifts the fundamental focus from purely visual-action mapping to procedural conditioning. We utilize LLMs to programmatically convert ground-truth interaction traces into structured step-by-step action plans, enabling our model to learn explicit plan-following behaviors. This data synthesis approach facilitates our parameter-efficient fine-tuning and directly motivates the creation of our Device-Support Interaction Benchmark (DSIB) to rigorously evaluate these specific capabilities.

B Motivation from Human Problem-Solving

To design a more robust autonomous agent, we draw direct inspiration from how human users naturally resolve complex device support tasks (Figure 2). When a user encounters an unfamiliar interface or does not intuitively know how to configure a specific setting, they rarely resort to blind, trial-and-error clicking. Instead, they implicitly acknowledge their knowledge deficit and consult external authoritative sources—such as searching online tutorials or reading user manuals. Once this procedural knowledge is acquired, the user systematically follows the suggested step-by-step instructions, directly translating text-based guidance into visual actions until the problem is solved. This human baseline is highly efficient and minimizes the risk of altering unintended system configurations.

In stark contrast, current state-of-the-art GUI agents handle knowledge deficits fundamentally differently. As illustrated in the lower trajectory of Figure 2, when an agent lacks the requisite domain knowledge

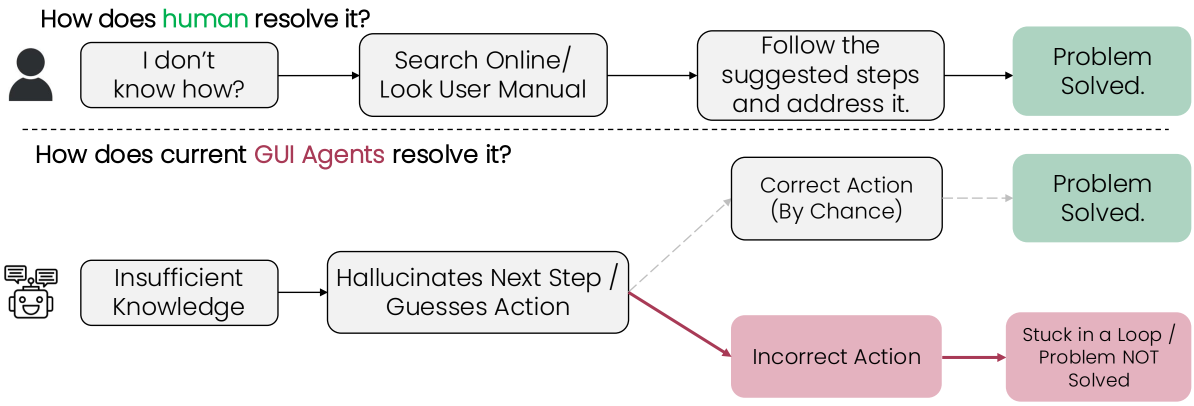

Figure 2: Task Resolving Trajectory

to navigate a deep-nested menu, it relies heavily on zero-shot visual guessing. Rather than retrieving a verified plan, the agent hallucinates the next logical step based solely on the immediately visible UI elements. While this ungrounded exploration might occasionally result in the correct action by sheer chance, it predominantly leads to incorrect operations. In complex device support scenarios, a single incorrect action can trap the agent in an unrecoverable state, causing it to become stuck in a loop and ultimately fail the task.

This dichotomy reveals that the primary failure mode of current agents is not a lack of visual perception, but an architectural absence of procedural grounding. PGGA is directly motivated by this human-centric problem-solving trajectory. By equipping the agent with an intermediate planner to retrieve and strictly follow verified action plans, we shift the paradigm from brittle, hallucination-prone guessing to robust, knowledge-grounded execution.

C DSIB Task Example

To illustrate the complexity and depth required by the DSIB dataset, Figure 3 provides a concrete example of a "Localization & Input" task: changing the primary device language to Spanish. This sequence demonstrates the precise hierarchical navigation and visual disambiguation challenges agents face.

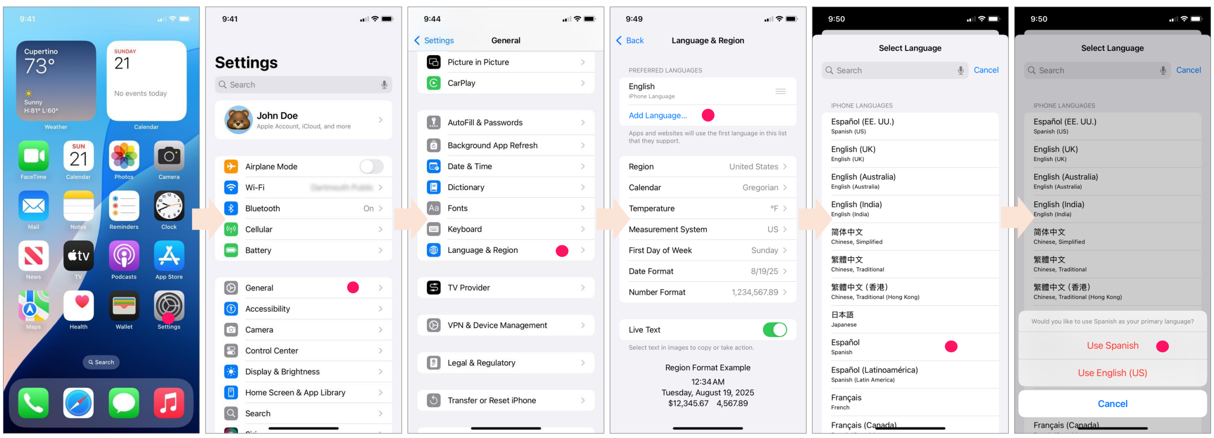

Figure 3: Example DSIB trajectory for a Localization & Input task. The visual sequence illustrates the necessary steps to change the device language to Spanish.

D Task-format Examples

Task-Only

You are an assistant trained to navigate the iPhone.
Given a task instruction, a screen observation, and an action history sequence, output the next action and wait for the next observation.
Here is the action space:
1. 'TAP': Tap on an element, value is the element to tap and the position [x,y] is required.
2. 'TYPE': Type a string into an element, value is the string to type and the position [x,y] is required.
3. 'SCROLL': Scroll the screen, value is the direction to scroll and the position [x,y] is not applicable.

Format the action as a dictionary with the following keys:
{'action': 'ACTION_TYPE', 'value': 'element', 'position': [x,y]}

Position represents the relative coordinates on the screenshot and should be scaled to a range of 0-1. Only respond in dictionary format.

Task: Change the device language to Spanish
<IMAGE - Screenshot Observation>

What is the next action? (Response should only contain the json dictionary.)

Task + Action Plan (Retrieval)

You are an assistant trained to navigate the iPhone.
Given a task instruction, action plan, a screen observation, and an action history sequence, output the next action and wait for the next observation.
Here is the action space:
1. 'TAP': Tap on an element, value is the element to tap and the position [x,y] is required.
2. 'TYPE': Type a string into an element, value is the string to type and the position [x,y] is required.
3. 'SCROLL': Scroll the screen, value is the direction to scroll and the position [x,y] is not applicable.

Format the action as a dictionary with the following keys:
{'action': 'ACTION_TYPE', 'value': 'element', 'position': [x,y]}

Position represents the relative coordinates on the screenshot and should be scaled to a range of 0-1. Only respond in dictionary format.

Task: Change the device language to Spanish
Action Plan: 1. Tap Settings on the iPhone Home Screen.
2. Scroll if needed and tap General.
3. Tap Language & Region.
4. Tap Add Language... if Spanish is not already listed, or tap iPhone Language if that option is visible.
5. In the language list or search field, select Spanish.
6. If multiple Spanish variants appear, tap the desired option, such as Español (España) or Español (Latinoamérica).
7. Tap Use Spanish or confirm the change when prompted.
8. Wait for the iPhone interface to refresh and display in Spanish.

<IMAGE - Screenshot Observation>

What is the next action? (Response should only contain the json dictionary.)

Task + Action Plan (Expert-Annotated)

You are an assistant trained to navigate the iPhone.
Given a task instruction, action plan, a screen observation, and an action history sequence, output the next action and wait for the next observation.
Here is the action space:
1. ‘TAP’: Tap on an element, value is the element to tap and the position [x,y] is required.
2. ‘TYPE’: Type a string into an element, value is the string to type and the position [x,y] is required.
3. ‘SCROLL’: Scroll the screen, value is the direction to scroll and the position [x,y] is not applicable.

Format the action as a dictionary with the following keys:
{‘action’: ‘ACTION_TYPE’, ‘value’: ‘element’, ‘position’: [x,y]}

Position represents the relative coordinates on the screenshot and should be scaled to a range of 0-1. Only respond in dictionary format.

Task: Change the device language to Spanish
Action Plan: 1. Tap Settings
2. Tap General
3. Tap Language & Region
4. Tap Add Language ...
5. Tap Español
6. Tap Use Spanish

<IMAGE - Screenshot Observation>

What is the next action? (Response should only contain the json dictionary.)