Click, Type, Repeat: A Comprehensive Survey on GUI Agents

Anonymous ACL submission

Abstract

Graphical User Interface (GUI) agents, powered by Large Foundation Models, have emerged as a transformative approach to automating human-computer interaction. These agents autonomously interact with digital systems via GUIs, emulating human actions such as clicking, typing, and navigating visual elements across diverse platforms. Motivated by the growing interest and fundamental importance of GUI agents, we provide a comprehensive survey that categorizes their benchmarks, evaluation metrics, architectures, and training methods. We propose a unified framework that delineates their perception, reasoning, planning, and acting capabilities. Furthermore, we identify important open challenges and discuss key future directions. Finally, this work serves as a basis for practitioners and researchers to gain an intuitive understanding of current progress, techniques, benchmarks, and critical open problems that remain to be addressed.

1 Introduction

Large Foundation Models (LFMs) are among the most transformative technologies that have recently changed the entire research landscape of AI as well as our everyday lives (Naveed et al., 2023; Wang et al., 2024d). Recently, we have witnessed a paradigm shift from using LFMs purely as conversational chatbots (Touvron et al., 2023; Chiang et al., 2023; Dam et al., 2024) to employing them for performing actions and automating useful tasks (Wang et al., 2024b; Zhao et al., 2023; Yao et al., 2023; Shinn et al., 2023; Shen et al., 2024b; Cheng et al., 2024c). In this direction, one approach stands out: leveraging LFMs to interact with digital systems, such as desktops, mobile phones, or web browsers, through Graphical User Interfaces (GUIs) in the same way humans do—for example, by controlling the mouse and keyboard to interact with visual elements displayed on a device's monitor (Iong et al., 2024; Hong et al., 2023; Lu et al., 2024; Shen et al., 2024a).

This approach holds great potential, as GUIs are ubiquitous across almost all computer devices that humans interact with in their work and daily lives. However, deploying LFMs in such environments poses unique challenges, such as dynamic layouts, diverse graphical designs across different platforms, and grounding issues—for instance, fine-grained recognition of elements within a page that are often small, numerous, and scattered (Liu et al., 2024b). Despite these challenges, many early efforts have shown significant promise (Lin et al., 2024; Cheng et al., 2024a), and growing interest from major players in the field is becoming evident[1].

Given the immense potential and rapid progress in this field, we propose a unified and systematic framework to categorize the various types of contributions within this space.

Organization of this Survey. We begin our survey by clearly defining the term "GUI Agent," followed by a traditional RL formalism of GUI Agent tasks in Section 2. We then summarize different datasets and environments in Section 3 to provide readers a clearer picture of the kinds of problem settings currently available. We summarize various GUI Agent architectural designs in Section 4, followed by different ways of training them in Section 5. Lastly, we discuss open problems and future prospects of GUI Agent research in Section 6.

2 Preliminaries

Definition 1 (GUI AGENT)**.** *An intelligent autonomous agent that interacts with digital platforms, such as desktops, or mobile phones, through their Graphical User Interface. It identifies and observes interactable visual elements displayed on the device's screen and engages with them by clicking, typing, or tapping, mimicking the interaction patterns of a human user.*

[1] Anthropic, Google DeepMind, OpenAI

3 Benchmarks

GUI agents are developed and evaluated on various platforms, including desktops, mobile phones, and web browser environments. This section summarizes benchmarks for all of these platform types.

When evaluating GUI Agents, it is crucial to distinguish between an **environment** and a **dataset**. A **dataset** is a static collection of data point, where each consists of several input features (e.g., a question, a screenshot of the environment, or the current state of the environment) and some output features (e.g., correct answers or actions to be taken). A dataset remains unchanged throughout the evaluation process. In contrast, an **environment** is an interactive simulation that represents a real-world scenario of interest. A GUI environment includes the GUI interface of a mobile phone or a desktop. Unlike datasets, environments are dynamic—actions taken within the environment can alter its state, hence, allowing modeling the problem as Markov Decision Processes (MDPs) or Partially Observable MDPs (POMDPs), with defined action, state, and observation spaces, and a state transition function.

Another critical dimension of the existing benchmarks for GUI Agentsis the distinction between the open-world and closed-world assumptions. Closed-world datasets or environments presume that all necessary knowledge for solving a task is contained within the benchmark itself. In contrast, open-world benchmarks relax this constraint, allowing relevant information required to complete a task to exist outside the benchmark.

3.1 Static Datasets

3.1.1 Closed-World Datasets

RUSS dataset introduces real-world instructions mapped to a domain-specific language (DSL) that enables agents to execute web-based tasks with high precision (Xu et al., 2021). Similarly, Mind2Web expands the task set to 2000 diverse tasks (Deng et al., 2023), and MT-Mind2Web adapts into conversational settings with multi-turn interactions (Deng et al., 2024). In contrast, TURKINGBENCH focuses on common micro tasks in crowdsourcing platforms, featuring a rich mix of textual instructions, multi-modal elements, and complex layouts (Xu et al., 2024). Focusing on visual and textual interplay, VisualWebBench includes OCR, element grounding, and action prediction tasks, which require fine-grained multimodal understanding (Liu et al., 2024b). Similarly, ScreenSpot focuses on GUI grounding for clicking and typing directly from screenshots (Cheng et al., 2024b). Complementing this, WONDERBREAD extends evaluation to business process management tasks, emphasizing workflow documentation and improvement rather than automation alone (Wornow et al., 2024). EnvDistraction dataset explores agent susceptibility to distractions in GUI environments, offering insights into faithfulness and resilience under cluttered and misleading contexts (Ma et al., 2024). NaviQAte introduces functionality-guided web application navigation, where tasks are framed as QA problems, pushing agents to extract actionable elements from multimodal inputs (Shahbandeh et al., 2024).

Evaluating on static closed-world datasets is particularly convenient, thanks to their lightweight and ease in setting up compared to environments. They are also especially valuable for fine-grained evaluation, reproducibility, and comparing models under identical conditions. However, they lack the dynamism of real-world applications, as models are tested on fixed data rather than adapting to new inputs or changing scenarios.

3.1.2 Open-World Datasets.

While most existing datasets are designed under the closed-world assumption, several datasets do not follow this paradigm. GAIA dataset tests agent integration diverse modalities and tools to answer real-world questions, often requiring web browsing or interaction with external APIs (Mialon et al., 2023). WebLINX emphasizes multi-turn dialogue for interactive web navigation on real-world sites, enhancing agents' adaptability and conversational skills (Lù et al., 2024).

Evaluation on static open-world datasets balances the ease of setting up an evaluation setting with realism since the agents interact with real-world websites. However, due to the nature of real-world websites, they are often unpredictable and prone to changes, which makes it more challenging to reproduce and compare with prior methods.

3.2 Interactive Environments

3.2.1 Closed-World Environments.

Closed-world interactive environments provide controlled and reproducible settings for evaluating agent capabilities. MiniWoB offers synthetic web tasks requiring interactions with webpages using mouse and keyboard inputs (Shi et al., 2017). It focuses on fundamental skills like button clicking

and form filling, providing a baseline for evaluating low-level interaction. CompWoB extends MiniWoB with compositional tasks, requiring agents to handle multi-step workflows and generalize across task sequences (Furuta et al., 2023). This introduces dynamic dependencies that reflect real-world complexity. WebShop simulates e-shopping tasks that challenge agents to navigate websites, process instructions, and make strategic decisions (Yao et al., 2022). WebArena advances realism with self-hosted environments across domains like e-commerce and collaborative tools, requiring agents to manage long-horizon tasks (Zhou et al., 2023b). VisualWebArena adds multimodal challenges, integrating visual and textual inputs for tasks like navigation and object recognition (Koh et al., 2024a). Shifting to enterprise settings, WorkArena evaluates agent performance in complex UI environments, focusing on knowledge work tasks in ServiceNow platform (Drouin et al., 2024). ST-WebAgentBench incorporates safety and trustworthiness metrics, assessing policy adherence and minimizing risky actions, critical for business deployment (Levy et al., 2024). Lastly, VideoWebArena introduces long-context video-based tasks, requiring agents to understand instructional videos and integrate them with textual and visual data to complete tasks. It emphasizes memory retention and multimodal reasoning (Jang et al., 2024).

Closed-world environments serve as evaluation platforms that mimic the dynamism of real-world environments while offering stability and reproducibility. However, setting up such benchmarks is often challenging, as they typically require considerable storage space and engineering skills.

3.2.2 Open-World Environments.

Open-world interactive environments challenge agents to navigate dynamic, real-world websites with evolving content and interfaces. WebVLN introduces a novel benchmark for vision-and-language navigation on websites, requiring agents to interpret visual and textual instructions to complete tasks such as answering user queries (Chen et al., 2024). It emphasizes multimodal reasoning by integrating HTML structure with rendered webpages, setting a foundation for realistic web navigation. WebVoyager leverages LLM to perform end-to-end navigation on 15 real websites with diverse tasks (He et al., 2024b). Its multimodal approach integrates screenshots and HTML content, enabling robust decision-making in dynamic online settings. AutoWebGLM optimizes web navigation through HTML simplification and reinforcement learning (Lai et al., 2024). This framework tackles the challenges of diverse action spaces and complex web structures, demonstrating significant improvement in real-world tasks with its AutoWebBench benchmark. MMInA evaluates agents on multihop, multimodal tasks across evolving real-world websites (Zhang et al., 2024e). The benchmark includes 1,050 tasks requiring sequential reasoning and multimodal integration to complete compositional objectives, such as comparing products across platforms. WebCanvas pioneers a dynamic evaluation framework to assess agents in live web environments (Pan et al., 2024). Its Mind2Web-Live dataset captures the adaptability of agents to interface changes and includes metrics like key-node-based intermediate evaluation, fostering progress in online web agent research.

Open-world environments are ideal for achieving both realism and dynamism. However, getting consistent evaluation and reproducibility is difficult as they evaluate agents on live websites that are subject to frequent changes.

3.3 Evaluation Metrics

Task Completion Metrics. The majority of benchmarks use task completion rate as the primary metric to measure GUI Agents' performance. However, different papers define task completion differently. Success can be defined as whether an agent successfully stops at a goal state (Chen et al., 2024; Zhou et al., 2023b), with Zhou et al. (2023b) programmatically checking if the intended outcome has been achieved (e.g., a comment has been posted, or a form has been completed), or whether the returned results exactly match the ground truth labels (Shi et al., 2017; Yao et al., 2022; Koh et al., 2024a; Drouin et al., 2024; Levy et al., 2024; Mialon et al., 2023). Another approach is to measure success based on whether an agent completes all required subtasks (Lai et al., 2024; Zhang et al., 2024e; Pan et al., 2024; Furuta et al., 2023; Jang et al., 2024; Cheng et al., 2024b). This approach can be further extended to measure partial success, as shown in Zhang et al. (2024e). WebVoyager uses GPT-4V to automatically determine success based on the agent's trajectory, reporting a high agreement rate of 85.3% with human judgments (He et al., 2024b). Instead of using a single final-state success metric, WebLINX measures an

overall success rate based on aggregated turn-level success rates across tasks (Lù et al., 2024). The turn-level success rates are computed depending on the type of actions, e.g., Intersection Over Union (IoU) for `click` or `submit` actions, and F1 for `say` or `textinput` actions. Lastly, there are task-specific metrics to measure success, e.g., using ROUGE-L, F1 for open-ended generation (Liu et al., 2024b; Xu et al., 2024; Wornow et al., 2024), accuracy for multiple choice question tasks (Liu et al., 2024b), Precision and Recall for Standard Operating Procedure (SOP) validation (Wornow et al., 2024), and so on.

Intermediate Step Metrics. While the task completion rate is a straightforward single-numeric metric that simplifies comparing the overall performance of agents, it fails to provide clear insights into their specific behaviors. Although some fine-grained metrics measure step-wise performance, their scope remains limited. WebCanvas evaluates step scores using three distinct targets: URL Matching, which verifies whether the agent navigated to the correct webpage; Element Path Matching, which checks if the agent interacted with the appropriate UI element, such as a button or text box; and Element Value Matching, which ensures the agent inputted or extracted the correct values, such as filling a form or reading text. WebLINX uses an intent match metric to assess whether the predicted action's intent aligns with the reference intent. Similarly, Mind2Web and MT-Mind2Web evaluate Element Accuracy by measuring the rate at which the agent selects the correct elements. These systems also measure the precision, recall, and F1 score for token-level operations, such as clicking or typing, and calculate the Step Success Rate, which reflects the proportion of individual task steps completed correctly. While step-wise evaluations provide more fine-grained insight into the agent's performance, it is often challenging to collect reference labels at the step level while also providing enough flexibility to consider different paths to achieve the original tasks.

Efficiency, Generalization, Safety and Robustness Metrics. Lastly, we summarize additional metrics that evaluate various aspects of GUI agents beyond their raw performance. Existing benchmarks include metrics for efficiency (Shahbandeh et al., 2024; Chen et al., 2024; Shahbandeh et al., 2024), generalization across diverse or compositional task settings (Furuta et al., 2023), adherence to safety policies (Levy et al., 2024), and robustness to environmental distractions (Ma et al., 2024).

4 GUI Agent Architectures

This section focuses on various architectural designs of a GUI Agent agent, which we categorize into four main types: (1) **Perception**: designs that enable the GUI Agent agent to perceive and interpret observations from its environment; (2) **Reasoning**: designs related to the cognitive processes of a GUI Agent agent, such as using an external knowledge base for long-term memory access or a world model of the environment to support other modules like planning; (3) **Planning**: designs related to decomposing a task into subtasks and creating a plan for their execution; and (4) **Acting**: mechanisms that allow the GUI Agent agent to interact with the environment, including representing actions in natural language using specific templates, JSON, or programming languages as action representations.

4.1 Perception

Unlike API-based agents that process structured, program-readable data, GUI agents must perceive and understand the on-screen environment that is designed for human consumption. This requires carefully chosen interfaces that allow agents to discover the location, identity, and properties of the interactive elements. Broadly, these perception interfaces can be categorized into four types: accessibility-based, HTML/DOM-based, screen-visual-based, and hybrid ones, with each offering different capabilities and posing distinct privacy and implementation considerations.

4.1.1 Accessibility-Based Interfaces

Modern mobile and desktop operating systems usually provide accessibility APIs[2] that expose a semantic hierarchy of UI components, including their roles, labels, and states[3][4][5]. GUI agents can utilize accessibility APIs to identify actionable elements and derive semantic cues without relying solely on pixel-based detection. These interfaces are resilient

[2] https://en.wikipedia.org/wiki/Computer_accessibility

[3] https://developer.apple.com/library/archive/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXmodel.html

[4] https://developer.apple.com/design/human-interface-guidelines/accessibility

[5] https://learn.microsoft.com/en-us/windows/apps/design/accessibility/accessibility

to minor layout changes or styling updates; however, their effectiveness depends on proper implementation by developers. Accessibility APIs may also be limited when dealing with highly dynamic elements (e.g., custom drawing canvases or gaming environments) and may not natively expose visual content. Although these APIs help reduce the complexity of visually parsing the screen, the agent may need additional perception methods for full functionality. On the positive side, accessibility-based interfaces typically require minimal sensitive user data, thereby reducing privacy concerns.

4.1.2 HTML/DOM-Based Interfaces

For web GUIs, agents frequently utilize the Document Object Model (DOM) to interpret the structural layout of a page. The DOM provides a hierarchical representation of elements, allowing agents to locate targets like buttons or input fields based on tags, attributes, or text content. However, raw HTML data or DOM tree usually has redundant and noisy structure. Various methods are proposed to handle this. Mind2Web (Deng et al., 2023) utilizes a fine-tuned small LM to rank the elements in a page before the final prediction of action with a large LM, and WebAgent (Gur et al., 2023) uses a specialized model HTML-T5 to generate task-specific HTML snippets. AutoWebGLM (Lai et al., 2024) designs an algorithm to simplify HTML content. While HTML/DOM-based interfaces provide rich structural data, they require careful preprocessing and, in some cases, additional heuristics or trained models to locate and interpret key UI components accurately.

4.1.3 Screen-visual-based Interfaces

With advances in computer vision and multimodal LLM, agents can utilize screen-visual information, like screenshots, to perceive on-screen environment. OmniParser (Lu et al., 2024) utilizes an existing multimodal LLM (e.g., GPT-4V) to parse a screenshot into a structured representation of the UI elements. However, screen-visual-based perception introduces privacy concerns since entire screenshots may contain sensitive information. Additionally, computational overhead increases as models must handle high-dimensional image inputs. Despite these challenges, such interfaces are crucial for agents operating in environments where high-quality accessibility interfaces and DOM information are unavailable, or environments where dynamic or visual information is crucial, like image or video editing software.

4.1.4 Hybrid Interfaces

To achieve robust and flexible performance across diverse environments, many GUI agents employ a hybrid approach. These systems combine accessibility APIs, DOM data, and screen-visual information to form a more comprehensive understanding of the interface. Leading methods in GUI agent tasks, such as OS-Atlas(Wu et al., 2024b) and UGround (Gou et al., 2024), demonstrates that hybrid interfaces that combine visual and textual inputs can enhance performance. Hybrid interfaces based approaches also facilitate error recovery—when accessibility or DOM data are incomplete or misleading, the agent can fall back on screen parsing, and vice versa.

4.2 Reasoning

WebPilot employs a dual optimization strategy for reasoning (Zhang et al., 2024d). WebOccam improves reasoning by refining the observation and action space of LLM agents (Yang et al., 2024). OSCAR introduces a general-purpose agent to generate Python code from human instructions (Wang and Liu, 2024). LAST leverages LLMs for reasoning, acting, and planning (Zhou et al., 2023a).

4.3 Planning

Planning involves decomposing a global task into multiple subtasks that progressively approach the goal state starting from an initial state (Huang et al., 2024). Traditional planning methods, such as symbolic approaches and reinforcement learning, have significant limitations: symbolic methods require extensive human expertise to define rigid system rules and lack error tolerance (Belta et al., 2007; Pallagani et al., 2022), while reinforcement learning demands impractical volumes of training data, often derived from costly environmental interactions (Acharya et al., 2023). Recent advancements in LLM-powered agents offer a transformative alternative by positioning LLM-powered agents as the cognitive core for planning agents (Huang et al., 2024). When equipping agents with GUIs as the medium, LLM-powered agents can directly interact with nearly all application domains and resources to enhance planning strategies. Based on what application domains/resources agents use for planning, we divide existing works into planning with internal and external knowledge.

4.3.1 Planning with Internal Knowledge

Planning with internal knowledge of GUI agents is to leverage the inherent knowledge to reason and think about the potential plans to fulfill the global task goals (Schraagen et al., 2000). WebDreamer (Gu et al., 2024) uses LLMs to simulate the outcomes of the actions of each agent and then evaluate the result to determine the optimal plan at each step. MobA (Zhu et al., 2024) devises a two-level architecture to power the mobile phone management, with a high level for understanding user commands, tracking history memories and planning tasks, and a low level to act the planned module. Agent S (Agashe et al., 2024) introduces an experience-augmented hierarchical planning to perform complex computer tasks.

4.3.2 Planning with External Knowledge

Enabling LLM-powered agents to interact with diverse applications and resources through GUIs allows them to leverage external data sources, thereby enhancing their planning capabilities. For example, Search-Agent (Koh et al., 2024b) combines LLM inference with A* search to explore and backtrack to alternative paths explicitly, AgentQ (Putta et al., 2024) combines LLM with MCTS. Toolchain (Zhuang et al.) models tool planning as a tree search algorithm and incorporates A* search to adaptively retrieve the most promising tool for subsequent use based on accumulated and anticipated costs. SGC (Wu et al., 2024a) decomposes the query and performs embedding similarity match between the concatenated subquery with the current retrieved task API and each of the existing APIs, and then selects the top one from the existing neighboring APIs. Thought Propagation Retrieval (Yu et al., 2023) prompts LLMs to propose a set of analogous problems and then applies established prompting techniques, like Chain-of-Thought, to derive solutions. The aggregation module subsequently consolidates solutions from these analogous problems, enhancing the problem-solving process for the original input. WebShop, Mind2Web, and WebArena (Zhou et al., 2023c; Deng et al., 2023) allow agents to interact with webs to plan for web browsing for search. WMA (Chae et al., 2024) utilizes world models to address the mistakes made by LLMs for long-horizon tasks.

4.4 Acting

Acting in GUI agents involves translating the agent's reasoning and planning outputs into executable steps within the GUI environment. Unlike purely text-based or API-driven agents, GUI agents must articulate their actions at a finer granularity—often down to pixel-level coordinates—while also handling higher-level semantic actions such as typing text, scrolling, or clicking on specific elements. Several directions of approaches have emerged:

Those utilizing textual interfaces may only rely on text-based metadata (HTML, accessibility trees) to identify UI elements. For example, WebAgent (Gur et al., 2023) and Mind2Web (Deng et al., 2023) use DOM or HTML representations to locate interactive elements. Similarly, AppAgent(Zhang et al., 2023) and MobileAgent (Wang et al., 2024a) leverage accessibility APIs to identify GUI components on mobile platforms.

However, as highlighted in UGround (Gou et al., 2024), such metadata can be noisy, incomplete, and computationally expensive to parse at every step. To overcome these limitations, recent research emphasizes visual-only grounding—mapping textual referring expressions or instructions directly to pixel-level coordinates on a screenshot. UGround trains large action models using only screen-level visual inputs. OmniParser (Lu et al., 2024) also demonstrates how vision-only approaches can parse GUIs without HTML or accessibility data. Similarly, OS-Atlas (Wu et al., 2024b) leverages large-scale multi-platform training data to achieve universal GUI grounding that generalizes across web, mobile, and desktop platforms. By unifying data sources and action schemas, OS-Atlas showcases the feasibility of a universal approach to action grounding.

5 GUI Agent Training Methods

This section summarizes different strategies to elicit the ability to solve agentic tasks in a GUI Agent agent. We broadly categorize these strategies into two types: (1) **Prompt-based Methods** and (2) **Training-based Methods**. Prompt-based methods do not involve the training of parameters; they elicit the ability to solve agentic tasks by providing detailed instructions or demonstrations within the prompt. Training-based methods, on the other hand, involve optimizing the agent's parameters to maximize an objective, such as pretraining, fine-tuning, or reinforcement learning.

5.1 Prompt-based Methods

Prompt-based methods enable GUI agents to exhibit learning and adaptation during inference through carefully designed prompts and interaction mechanisms, without modifying model parameters. This learning and adaptation occur as the agent's state evolves by incorporating context from past actions or stored knowledge.

One key approach is the use of dynamic action generation and accumulation. DynaSaur (Nguyen et al., 2024) enables agents to dynamically create and compose actions by generating and executing Python code via prompting. Given task instructions, the agent outputs code snippets defining new actions or reusing existing ones, effectively learning new skills and improving performance over time. Agent Q (Putta et al., 2024) and OSCAR (Wang and Liu, 2024) incorporate self-reflection and self-critique mechanisms via prompts, enabling agents to iteratively improve decision-making by identifying and rectifying errors. Auto-Intent (Kim et al., 2024) focuses on unsupervised intent discovery and utilization, extracting intents from interaction histories and incorporating them into future prompts. Other techniques include state-space exploration in LASER (Ma et al., 2023), state machine in OSCAR (Wang and Liu, 2024), expert development and multi-agent collaboration in MobileExperts (Zhang et al., 2024b), and app memory in AutoDroid (Wen et al., 2024).

Despite the potential of prompt-based methods, the limited context size of LLMs and the difficulty of designing effective prompts that elicit the desired behavior remain.

5.2 Training-based Methods

5.2.1 Pre-training

Earlier models for GUI tasks relied on assembling smaller encoder-decoder architectures to address visual understanding challenges due to its ability to learn unified representations from diverse visual and textual data, enhance transfer learning capabilities, and integrate multiple modalities deeply. For example, PIX2STRUCT (Lee et al., 2023) is pre-trained on a screenshot parsing task, which involves predicting simplified HTML representations from screenshots with visually masked regions. It employs a ViT (Dosovitskiy, 2020) as the image encoder, T5 (Raffel et al., 2020) as the text encoder, and a Transformer-based decoder.

Training of recent GUI agent models often involve the continual pre-training of existing vision large language models on additional large-scale datasets. This step refines the model's general knowledge and modifies or assembles new neural network modules into the backbone, providing a stronger foundation before fine-tuning on smaller, curated datasets for GUI tasks. VisionLLM (Wang et al., 2023) utilizes public datasets to integrate BERT (Devlin, 2018) and Deformable DETR (Zhu et al., 2020) into large language models, focusing on visual question answering tasks centered on grounding and detection. SeeClick (Cheng et al., 2024a) is built using continual pre-training on Qwen-VL (Bai et al., 2023) with datasets incorporating OCR-based layout annotation to predict click actions. UGround (Gou et al., 2024) use continual pre-training on the LLaVA-NEXT (Liu et al., 2024a) model without its low-resolution image fusion module on a large dataset and synthetic data to align visual elements with HTML metadata for planning and grounding tasks.

Pre-training is also used to adapt new designs for improved computational efficiency in GUI-related tasks. CogAgent (Hong et al., 2023) employs a high-resolution cross-module to process small icons and text, enhancing its efficiency for GUI tasks such as DOM element generation and action prediction. ShowUI (Lin et al., 2024) built on Qwen2-VL (Wang et al., 2024c) with a visual-token selection module to improve the computational efficiency for interleaved high-resolution grounding.

5.2.2 Fine-tuning

Fine-tuning has emerged as a key strategy to adapt large vision-language models (VLMs) and large language models (LLMs) to the specialized domain of GUI interaction. Unlike zero-shot or prompt-only approaches, fine-tuning can enhance both the model's grounding in GUI elements and its ability to execute instructions reliably.

Recent work highlights reducing hallucinations and improving grounding. Falcon-UI (Shen et al., 2024a) fine-tunes on large-scale instruction-free GUI data and then fine-tunes on Android and Web tasks, achieving high accuracy with fewer parameters. VGA (Ziyang et al., 2024), through image-centric fine-tuning, reduces hallucinations by tightly coupling visual inputs with GUI elements, thus improving action reliability. Similarly, UI-Pro (Li et al., 2024) identifies a hidden recipe for systematic fine-tuning of VLMs, scaling

down model size while maintaining state-of-the-art grounding accuracy.

Other methods leverage fine-tuning to incorporate domain-specific reasoning and functionalities such as functionality-aware fine-tuning for generating human-like interactions (Liu et al., 2024d), alignment strategies to handle multilingual, variable-resolution GUI inputs (Nong et al., 2024). Some methods emphasize autonomous adaptation, such as learning to execute arbitrary voice commands through trial-and-error exploration (Pan et al., 2023) and learning for cross-platform GUI grounding without structured text (Cheng et al., 2024a). Additionally, fine-tuning can specialize models for context-sensitive actions. Techniques proposed by Liu et al. (2023) enable context-aware text input generation, improving coverage in GUI testing scenarios. Taken together, these fine-tuning methods demonstrate how careful parameter adaptation, data scaling and multimodal alignment can collectively advance the reliability, interpretability, and performance of GUI agents.

5.2.3 Reinforcement Learning

Reinforcement learning (RL) was used in the early text-based agent WebGPT to improve information retrieval of the GPT-3 based model (Nakano et al., 2021). Liu et al. (2018) use human demonstrations to constrain the search space for RL, though using *workflows* as a high-level process for the model to complete without specifying the specific details. An example from Liu et al. (2018) is for the specific process of forwarding a given email, the *workflow* would involve clicking forward, typing in the address, and clicking send. Deng et al. (2023) uses RL based on human demonstrations as the reward signal. While early agents constrained the input and action spaces to only text, recent work has extended to GUI agents.

WebRL framework uses RL to generate new tasks based on previously unsuccessful attempts as a mitigation for sparse rewards (Qi et al., 2024). Task success is evaluated by an LLM-based outcome reward model (ORM) and KL-divergence is used to prevent significant shifts in policies during the curriculum. AutoGLM apply online, curriculum learning, in particular to address error recovery during real-world use and to correct for stochasticity not present in simulators (Liu et al., 2024c). DigiRL uses a modified advantage-weighted regression (AWR) algorithm for offline learning (Peng et al., 2019), but modifies AWR for more stochastic environments by using a simple value function and curriculum learning.

6 Open Problems & Challenges

Graphical User Interface (GUI) agents face critical challenges in understanding user intent, ensuring security and privacy, optimizing inference latency, and achieving personalization. Current systems often struggle to infer goals accurately, reaching only around 51.1% accuracy on unseen websites (Kim et al., 2024), and robust generalization across diverse tasks remains a priority (Stefanidi et al., 2022; Gao et al., 2024). Security and privacy concerns become prominent as agents handle sensitive information, potentially exposing users to risks (He et al., 2024a; Zhang et al., 2024a), particularly when relying on cloud-based processing and raising issues of unauthorized access (Zhang et al., 2024c). Inference latency poses additional hurdles, as real-time responsiveness is essential for seamless user interactions, especially in resource-constrained scenarios, demanding efficiency without compromising accuracy. Future efforts should focus on lightweight modeling, adaptive methods, and hardware acceleration to reduce computational overhead. Meanwhile, personalization aims to refine user experiences by predicting intentions and tailoring interactions (Berkovitch et al., 2024), potentially guided by explicit feedback. Addressing these interconnected challenges will foster more secure, responsive, and user-centric GUI agents that adapt to evolving requirements and environments. Ultimately, advancing these areas will elevate the abilities of GUI agents in real-world deployments.

7 Conclusion

In this survey, we have thoroughly explored GUI Agents, examining various benchmarks, agent architectures, and training methods. Although considerable strides have been made, problems such as intent understanding, security, latency, and personalization remain critical challenges. We hope this survey will act as a valuable resource for researchers, offering structure and practical guidance in this rapidly growing and exciting field, and inspiring further inquiry into GUI Agents. We are confident that the progress in this area will mark an important milestone, benefiting humankind, significantly enhancing our daily productivity, and transforming the way we interact with computers.

Limitations

We recognize that some studies have explored interactions between LFM-based agents and digital systems through interfaces other than GUIs, such as Command Line Interfaces (CLI) or Application Programming Interfaces (API). However, these approaches are relatively limited in scope compared to GUI-based methods. To maintain a focused scope for our survey, we have chosen not to include them in our discussion.

References

Kamal Acharya, Waleed Raza, Carlos Dourado, Alvaro Velasquez, and Houbing Herbert Song. 2023. Neurosymbolic reinforcement learning and planning: A survey. *IEEE Transactions on Artificial Intelligence*.

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. 2024. Agent s: An open agentic framework that uses computers like a human.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George J Pappas. 2007. Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):61–70.

Omri Berkovitch, Sapir Caduri, Noam Kahlon, Anatoly Efros, Avi Caciularu, and Ido Dagan. 2024. Identifying user goals from ui trajectories. *ArXiv preprint*, abs/2406.14314.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2024. Web agents with world models: Learning and leveraging environment dynamics in web navigation.

Qi Chen, Dileepa Pitawela, Chongyang Zhao, Gengze Zhou, Hsiang-Ting Chen, and Qi Wu. 2024. Webvln: Vision-and-language navigation on websites. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 1165–1173. AAAI Press.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024a. Seeclick: Harnessing gui grounding for advanced visual gui agents. *ArXiv preprint*, abs/2401.10935.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024b. Seeclick: Harnessing gui grounding for advanced visual gui agents.

Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and Xiuqiang He. 2024c. Exploring large language model based intelligent agents: Definitions, methods, and prospects.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A complete survey on llm-based ai chatbots.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024. On the multi-turn instruction following for conversational web agents.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. Workarena: How capable are web agents at solving common knowledge work tasks?

Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. 2023. Language model agents suffer from compositional generalization in web automation. *ArXiv preprint*, abs/2311.18751.

Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. 2024. Assistgui: Task-oriented pc graphical user interface automation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13289–13298.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents.

Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis.

Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. 2024a. The emerged security and privacy of llm agent: A survey with case studies. *ArXiv preprint*, abs/2407.19354.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024b. Webvoyager: Building an end-to-end web agent with large multimodal models.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. Cogagent: A visual language model for gui agents.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *ArXiv preprint*, abs/2402.02716.

Iat Long Iong, Xiao Liu, Yuxuan Chen, Hanyu Lai, Shuntian Yao, Pengbo Shen, Hao Yu, Yuxiao Dong, and Jie Tang. 2024. Openwebagent: An open toolkit to enable web agents on large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 72–81.

Lawrence Jang, Yinheng Li, Charles Ding, Justin Lin, Paul Pu Liang, Dan Zhao, Rogerio Bonatti, and Kazuhito Koishida. 2024. Videowebarena: Evaluating long context multimodal agents with video understanding web tasks.

Jaekyeom Kim, Dong-Ki Kim, Lajanugen Logeswaran, Sungryull Sohn, and Honglak Lee. 2024. Auto-intent: Automated intent discovery and self-exploration for large language model web agents. *ArXiv preprint*, abs/2410.22552.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024a. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ArXiv preprint*, abs/2401.13649.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024b. Tree search for language model agents.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 18893–18912. PMLR.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2024. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents.

Hongxin Li, Jingran Su, Jingfan CHEN, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2024. UI-pro: A hidden recipe for building vision-language models for GUI grounding.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. Llava-next: Improved reasoning, ocr, and world knowledge.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024b. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *ArXiv preprint*, abs/2404.05955.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024c. Autoglm: Autonomous foundation agents for guis. *ArXiv preprint*, abs/2411.00820.

Zhe Liu, Chunyang Chen, Junjie Wang, Xing Che, Yuekai Huang, Jun Hu, and Qing Wang. 2023. Fill in the blank: Context-aware automated text input generation for mobile gui testing. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1355–1367. IEEE.

Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. 2024d. Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

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

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024. Caution for the environment: Multimodal agents are susceptible to environmental distractions.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback, 2021. *URL https://arxiv. org/abs/2112.09332*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models.

Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A. Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, Franck Dernoncourt, and Tianyi Zhou. 2024. Dynasaur: Large language agents beyond predefined actions. *ArXiv preprint*, abs/2411.01747.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. *ArXiv preprint*, abs/2407.04346.

Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Lior Horesh, Biplav Srivastava, Francesco Fabiano, and Andrea Loreggia. 2022. Plansformer: Generating symbolic plans using transformers. *ArXiv preprint*, abs/2212.08681.

Lihang Pan, Bowen Wang, Chun Yu, Yuxuan Chen, Xiangyu Zhang, and Yuanchun Shi. 2023. Autotask: Executing arbitrary voice commands by exploring and learning from mobile gui. *ArXiv preprint*, abs/2312.16062.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. Webcanvas: Benchmarking web agents in online environments.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Jan Maarten Schraagen, Susan F Chipman, and Valerie L Shalin. 2000. *Cognitive task analysis*. Psychology Press.

Mobina Shahbandeh, Parsa Alian, Noor Nashid, and Ali Mesbah. 2024. Naviqate: Functionality-guided web application navigation. *ArXiv preprint*, abs/2409.10741.

Huawen Shen, Chang Liu, Gengluo Li, Xinlong Wang, Yu Zhou, Can Ma, and Xiangyang Ji. 2024a. Falcon-ui: Understanding gui before following user instructions. *ArXiv preprint*, abs/2412.09362.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024b. Taskbench: Benchmarking large language models for task automation.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Zinovia Stefanidi, George Margetis, Stavroula Ntoa, and George Papagiannakis. 2022. Real-time adaptation of context-aware intelligent user interfaces, for enhanced situational awareness. *IEEE Access*, 10:23367–23393.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan

Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024c. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xiaoqiang Wang and Bang Liu. 2024. Oscar: Operating system control via state-aware reasoning and re-planning. *ArXiv preprint*, abs/2410.18963.

Zichong Wang, Zhibo Chu, Thang Viet Doan, Shiwen Ni, Min Yang, and Wenbin Zhang. 2024d. History, development, and principles of large language models-an introductory survey.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 543–557.

Michael Wornow, Avanika Narayan, Ben Viggiano, Ishan S Khare, Tathagat Verma, Tibor Thompson, Miguel Angel Fuentes Hernandez, Sudharsan Sundar, Chloe Trujillo, Krrish Chawla, et al. 2024. Do multimodal foundation models understand enterprise workflows? a benchmark for business process management tasks. *ArXiv preprint*, abs/2406.13264.

Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. 2024a. Can graph learning improve task planning? *ArXiv preprint*, abs/2405.19119.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. 2024b. Os-atlas: A foundation action model for generalist gui agents. *ArXiv preprint*, abs/2410.23218.

Kevin Xu, Yeganeh Kordi, Tanay Nayak, Ado Asija, Yizhong Wang, Kate Sanders, Adam Byerly, Jingyu Zhang, Benjamin Van Durme, and Daniel Khashabi. 2024. Tur [k] ingbench: A challenge benchmark for web agents. *ArXiv preprint*, abs/2403.11905.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica Lam. 2021. Grounding open-domain instructions to automate web support tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1022–1032, Online. Association for Computational Linguistics.

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.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Junchi Yu, Ran He, and Rex Ying. 2023. Thought propagation: An analogical approach to complex reasoning with large language models. *ArXiv preprint*, abs/2310.03965.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, et al. 2024a. Large language model-brained gui agents: A survey. *ArXiv preprint*, abs/2411.18279.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users.

Jiayi Zhang, Chuang Zhao, Yihan Zhao, Zhaoyang Yu, Ming He, and Jianping Fan. 2024b. Mobileexperts: A dynamic tool-enabled agent team in mobile devices.

Xinyu Zhang, Huiyu Xu, Zhongjie Ba, Zhibo Wang, Yuan Hong, Jian Liu, Zhan Qin, and Kui Ren. 2024c. Privacyasst: Safeguarding user privacy in tool-using large language model agents. *IEEE Transactions on Dependable and Secure Computing*.

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

Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. 2024e. Mmina: Benchmarking multihop multimodal internet agents.

Pengyu Zhao, Zijian Jin, and Ning Cheng. 2023. An in-depth survey of large language model-based artificial intelligence agents.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023a. Language agent tree search unifies reasoning acting and planning in language models. *ArXiv preprint*, abs/2310.04406.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023b. Webarena: A realistic web environment for building autonomous agents.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023c. Webarena: A realistic web environment for building autonomous agents. *ArXiv preprint*, abs/2307.13854.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

Zichen Zhu, Hao Tang, Yansi Li, Kunyao Lan, Yixuan Jiang, Hao Zhou, Yixiao Wang, Situo Zhang, Liangtai Sun, Lu Chen, and Kai Yu. 2024. Moba: A two-level agent system for efficient mobile task automation.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*.

Meng Ziyang, Yu Dai, Zezheng Gong, Shaoxiong Guo, Minglong Tang, and Tongquan Wei. 2024. Vga: Vision gui assistant-minimizing hallucinations through image-centric fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1261–1279.