International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

AI-Driven GUI Automation: A Comprehensive Review of Methods, Systems, And Challenges (2021–2025)

Rishit Mishra[1], Rudrendra Bahadur Singh[2], Vinayak[3], Tanmay Gunwant[4], Ekta Rastogi[5]
[1,4,5]Department of Computer Science & Engineering, BBDITM, Lucknow, India
[2,3] Assistant Professor, Dept. of Computer Science and Engineering, BBDITM, Lucknow, India
Email ID:*rishitmishra05@gmail.com[1], rudra.rathor20@gmail.com[2], srmvinayak@gmail.com[3], tanmaywork172@gmail.com[4], ektarastogi892@gmail.com[5]*

Abstract
This paper reviews recent research on artificial intelligence techniques used for Graphical User Interface (GUI) automation. The analysis is based on 45 peer-reviewed studies published between 2021 and 2025. Over the past few years, the field has gradually moved away from rigid rule-based scripts and Reinforcement Learning (RL) approaches toward more intelligent and autonomous systems powered by Multimodal Large Language Models (MLLMs). To better understand current developments, this review organizes existing methods into three main categories: accessibility-tree–based approaches, vision-based methods, and hybrid neuro-symbolic techniques that integrate both structural interface data and visual information. The study also compares automation research across different platforms, highlighting the contrast between relatively stable mobile ecosystems and the more complex and fragmented Windows desktop environment, which is often considered an open-world setting. In addition, several research challenges are identified, including the limited availability of Windows-focused datasets and the difficulty of achieving efficient real-time inference. The paper concludes by outlining future research opportunities and suggesting directions for building privacy-aware and practical desktop automation agents.

Keywords: *GUI Automation, Windows UI Agents, Robotic Process Automation (RPA), Multimodal Large Language Models (MLLM), Computer Vision, Software Testing, Autonomous Agents*

1. Introduction
Graphical User Interface (GUI) automation is widely used in fields such as software testing, accessibility support, and enterprise Robotic Process Automation (RPA). Traditionally, automation tools such as Selenium and Appium rely on operating system metadata, particularly view hierarchies, to detect interface elements. However, these selector-based scripts are often sensitive, since even minor modification in the user interface layout can cause the automation to fail. Between 2021 and 2025, studies in this field gradually shifted toward more context-aware automation methods. Initial research focused on Reinforcement Learning where GUI interaction is modeled as a sequential decision-making task. Despite this progress, RL-based agents often faced difficulties in complex applications because of sparse reward signals. To address the limitations of metadata-driven approaches, researchers began using Computer Vision techniques that detect and interpret interface elements directly from pixel-level information. Recent advances in GUI automation between 2023 and 2025 have been largely driven by Multimodal Large Language Model (MLLM) agents. Systems such as AppAgent and UFO utilize the reasoning ability of Large Language Models to understand high-level natural language instructions. This allows the agents to carry out task planning, self-correction, and flexible tool usage during execution. Despite these developments, a noticeable gap still exists between mobile and desktop automation research. Mobile platforms have progressed rapidly due to the availability of large benchmark datasets such as Android in the Wild (AITW) and controlled testing environments, which have significantly

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

accelerated research in this domain. Recent advances in GUI automation between 2023 and 2025 have been largely driven by Multimodal Large Language Model (MLLM) agents. Systems such as AppAgent and UFO utilize the reasoning ability of Large Language Models to understand high-level natural language instructions. This allows the agents to carry out task planning, self-correction, and flexible tool usage during execution. Despite these developments, a noticeable gap still exists between mobile and desktop automation research. Mobile platforms have progressed rapidly due to the availability of large benchmark datasets such as Android in the Wild (AITW) and controlled testing environments, which have significantly accelerated research in this domain.

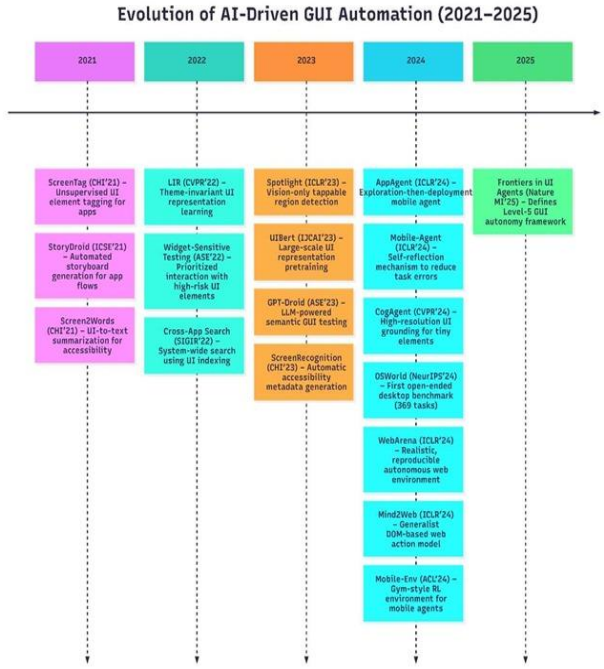

Figure 1 GUI Agents Evolution

The above figure (Fig. 1) illustrates the evolution of GUI automation research demonstrates a shift from early reinforcement learning methods to vision-based and MLLM-powered agents capable of performing a wide range of tasks across mobile, web, and desktop platforms.

2. Background And Related Concepts

This section describes the functional components of GUI agents and highlights the environmental differences between Windows and mobile platforms.

2.1. Anatomy of a GUI Agent

GUI automation systems typically consist of two core components: Perception, which enables the agent to observe and interpret the interface state, and Reasoning.

2.2. Perception Paradigms

Accurately detecting interface elements remains a key challenge, and research typically divides perception methods into three main paradigms. Metadata-Based (View Hierarchy): This approach identifies interface elements using accessibility information provided by the operating system, such as the Windows UI Automation API. Although it is fast and efficient, research like *Soft RPA* [19] shows that it can be unreliable because even small changes in the user interface may break the selectors. In addition, some modern frameworks, including Flutter, render the entire interface as a single canvas, which makes accessibility metadata difficult or impossible to access [29]. Vision-Based (Pixel-Only): This approach detects interface elements directly from screenshots using computer vision techniques. Systems such as Spotlight [6] and CogAgent [4] follow this pixel-based method. Additionally, representation models like UIBert [38] help improve robustness against variations in screen resolution and visual themes. Hybrid Perception: This method combines metadata and visual analysis. Systems such as UFO [2] use APIs to quickly access metadata when available and switch to visual detection when dealing with custom or unsupported interface elements.

2.3. Reasoning Engines

Reinforcement Learning (RL). Early approaches [26, 40] modeled GUI interaction as a Markov Decision Process. While effective for controlled tasks, RL methods often struggle with sparse reward signals and lack the semantic understanding needed to interpret complex user intentions. Multimodal Large Language Models (MLLM). Recent state-of-the-art systems, such as AppAgent

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

[1] and Mobile-Agent [7], apply transformer-based reasoning to convert high-level goals into executable actions through autonomous planning and self-reflection.

2.4. Platform Differences

A clear complexity difference exists between mobile and desktop platforms. Mobile systems generally operate in a sandboxed environment, where only one application is active at a time [22]. In contrast, Windows functions as an "open world" environment [8], where agents must handle multiple overlapping windows, system tools, and workflows that span across different applications. Benchmarks such as OSWorld [8] show that although AI agents are improving, human users still perform significantly better in these complex and less constrained desktop environments. Table 1-3.

3. Review Summary

Table 1 Summary of core GUI Agent Architectures and Perception-Driven Interaction Systems (2021–2025)

Reference Paper	Research Problem	Approach	Limitations
[1]	Creating agents that can use any app without backend access (like humans).	Exploration-then-Deployment. Agents explore apps to learn button functions, then use GPT-4V to execute tasks.	High cost/latency due to heavy reliance on GPT-4V.
[2]	Automating cross-application workflows on Windows (e.g., Word to Outlook).	Dual-Agent: (1) AppAgent selects the active window; (2) ActionAgent interacts with controls via UI Automation API.	Struggles with custom UI controls that hide accessibility IDs.
[3]	Lack of diverse, real-world training data for mobile agents.	Released massive dataset of human demonstrations + Transformer-based behavioral cloning models.	Baseline models struggle with long-horizon reasoning vs. newer LLMs.
[4]	Standard VLMs fail to "read" small text or icons on high-res screens.	High-res cross-attention module (1120x1120 input) added to CogVLM to detect tiny UI elements.	Computationally heavy due to high-resolution image encoding.
[5]	Agents trained on one site fail on others (generalization).	Collected diverse dataset across 137 websites; trained LLMs to predict HTML elements + actions.	Relies on HTML DOM, which is less applicable to native mobile apps.
[6]	Identifying tappable regions without view hierarchies (which are often broken).	Vision-only model using a "Region-of-Interest" (ROI) focus mechanism to output widget descriptions.	Focuses on perception (understanding) rather than multi-step planning.
[7]	Global planning for long tasks (e.g., "Plan a trip") on mobile is error-prone.	Self-Reflection mechanism. The agent pauses to verify if the last tap achieved the desired state before moving	Reflection steps double the inference time/cost.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

		on.	
[8]	Lack of a true "desktop OS" benchmark (handling files, multiple apps).	Built a full, containerized Ubuntu/Windows environment to evaluate agents on 369 real-world tasks.	Setup requires complex dockerized infrastructure.
[9]	Toy environments (MiniWob) don't reflect real modern web apps.	Created standalone, self-hostable versions of Reddit, Gitlab, and Shops to test agents safely.	Limited to Web GUIs (no native OS interaction).
[10]	Mapping natural language ("Click the red button") to UI elements.	Transformer-based model trained to align language instructions with UI object locations.	Pre-dates the "Generative Agent" era; less capable of reasoning.

Table 2 Summary of RPA Systems, Automated GUI Testing Frameworks, accessibility tools, and Applied GUI Automation Methods (2021–2025).

Reference Paper	Research Problem	Environment	Weakness
[11]	Automating IT support tasks (which often involve GUI/CLI mixed workflows).	IT Ops / Desktop.	Limited to IT/DevOps contexts, not general consumer apps.
[12]	Developers cannot fix crashes if they cannot reproduce the GUI steps that caused them.	Android (Benchmarks) .	Requires access to the app's source code or instrumentation.
[13]	Traditional "Monkey" testing is random and misses deep app logic.	Android (65 Apps).	High token cost for continuous testing.
[14]	Tests written for Android don't work on iOS, requiring duplicate effort.	Android & iOS.	Fails when platform design patterns diverge significantly.
[15]	Detecting visual bugs (overlapping text, cut-off buttons) is hard for code-based tools.	Mobile UIs.	"False positives" on artistic or non-standard UI designs.
[16]	Apps behave differently under different conditions (e.g., bad WiFi, low battery).	Android OS.	Increases complexity of the test environment.
[17]	Automating the design of GUIs, not just the testing.	Design Tools (Figma).	Generative design, not functional automation.
[18]	Lack of academic datasets for RPA (usually proprietary).	UiPath/Automation Anywhere scripts.	Dataset size is relatively small compared to mobile datasets.
[19]	RPA is fragile to UI changes (selectors breaking).	Enterprise Web Apps.	Still struggles with major UI redesigns.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

[20]	Business users can't write RPA scripts.	UiPath.	Generated scripts require human fine-tuning for complex logic.
[21]	Reading text in scanned documents (Invoices/Forms) for RPA.	Document Processing.	Computationally intensive pre-training.
[22]	Need for a lightweight, fast environment to test LLM agents.	Android.	Simulation-based; may not perfectly mimic touch latency.
[23]	Making apps accessible/automatable when developers forget labels.	iOS/Android.	Limited to standard widget sets.
[24]	Identifying what to automate in a company.	Desktop Logs.	Privacy concerns in logging user behavior.
[25]	Helping agents understand code/scripts associated with UI elements.	Code/Script Repos.	Requires access to the codebase.

Table 3. Broad Survey of Foundational GUI Automation Research Across Mobile, Web, and Desktop Platforms (2021–2025).

Reference Paper	Research Problem	Environment	Weakness
[26]	RL agents are promising but inconsistent in bug finding.	Android (Benchmarks).	RL training time is prohibitive for CI/CD pipelines.
[27]	Testing "Local" UI states (e.g., dark mode, language changes) is often ignored.	Android Apps.	Limited to "Configuration" bugs, not functional logic.
[28]	Apps lack metadata (IDs) for automation; manual tagging is expensive.	Rico Dataset.	Accuracy drops on "Game" UIs or custom canvas drawings.
[29]	90% of apps fail accessibility standards (e.g., small buttons, low contrast).	Android (Google Play).	Static analysis tool; cannot check dynamic accessibility changes.
[30]	Understanding app flow without running it is hard for auditors.	Android APKs.	Does not handle apps with heavy backend login requirements.
[31]	Automation agents might accidentally leak private data (PII) during execution.	Android.	High overhead for "Taint Tracking" instrumentation.
[32]	Standard agents treat all buttons equally; miss critical "Delete" or "Buy" actions.	E-commerce Apps.	Requires manual definition of "High Risk" categories.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

[33]	Fully autonomous agents fail when they get stuck.	Mobile Tasks.	Users find frequent interruptions annoying.
[34]	Legacy apps have "Deep" view hierarchies that slow down automation.	Android Source Code.	Risk of breaking custom layout logic.
[35]	Agents don't know if they are annoying the user (e.g., rapid clicking).	User Study Data.	Requires access to raw touch sensors.
[36]	Blind users need a summary of the screen (e.g., "Login page with Google").	Screen2Words (112k screens).	Summaries are often too generic for complex tasks.
[37]	Designers need to test high-level layout constraints (e.g., "Does this fit on iPhone Mini?").	Design Tools.	Not applicable to live, compiled apps.
[38]	Training agents from scratch is expensive.	3M UI Screens.	Model size is large for edge devices.
[39]	Blind users struggle to navigate complex apps like Uber.	iOS Accessibility.	Simplified menus remove some advanced app functionality.
[40]	Lack of a standardized RL gym for Android.	Android 10+.	High computational cost to run multiple emulators.
[41]	Detecting buttons when themes change (Dark Mode, Holiday Themes).	Diverse App Themes.	Struggles with structural layout changes.
[42]	Finding content inside apps (e.g., "Find the PDF I opened yesterday").	Mobile OS.	Privacy risk of indexing internal app data.
[43]	Prioritizing which buttons an agent should explore first.	Web/Mobile.	Bias toward "Center of Screen" elements.
[44]	Agents get "lost" in deep apps.	Complex Apps.	Fails if the app has infinite scrolling feeds.
[45]	Note: This represents the high-level synthesis papers appearing in 2025.	Scope: Global.	Theoretical framework.

4. Review Methodology

This review synthesizes 45 primary peer-reviewed studies to analyze the evolution of GUI automation.

4.1. Scope and Search Strategy

This review focuses on studies published between January 2021 and early 2025, highlighting the shift in research from Reinforcement Learning (RL) approaches to Multimodal Large Language Models (MLLMs). Relevant publications were collected from major academic databases, including IEEE

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

Xplore, ACM Digital Library, and Google Scholar. The literature search was conducted using keywords such as *GUI Automation, Robotic Process Automation (RPA), Windows UI Agents, Mobile GUI Testing*, and *Vision-Language Models*.

4.2. Inclusion and Exclusion Criteria

Only peer-reviewed papers published in leading research venues such as ICSE, NeurIPS, CVPR, CHI, and ISSTA were considered for this review. The selected studies were then organized into four main thematic categories:

- MLLM and Agent Architectures: Research focusing on modern autonomous systems, including frameworks such as AppAgent and CogAgent.
- Windows and RPA Systems: Studies related to desktop automation and techniques for building self-healing automation scripts.
- Mobile Testing and Accessibility: Work addressing automated bug detection, GUI testing, and accessibility-focused navigation tools.
- Datasets and Benchmarks: Foundational resources and evaluation benchmarks, including datasets such as AITW and OSWorld.

4.3. Evaluation Dimensions

The selected studies were assessed based on five key criteria: generalization (performance on unseen applications), latency (speed of inference), robustness (ability to handle UI changes), dataset scale, and workflow complexity (support for coordination across multiple applications).

5. Windows Gui Automation Challenges And Paradigms

Automating the Windows desktop is an "open-world" problem due to multiple frameworks like Win32, WPF, and UWP, and the need to manage complex, multi-window workflows. Traditional industrial RPA relies on UI Automation (UIA) metadata to locate interface elements, but this approach is fragile as minor UI changes can break scripts. Vision-based methods such as Spotlight [6] and CogAgent [4] detect elements from screenshots, offering more robustness at higher computational cost. Hybrid systems like UFO [2] combine both methods, using metadata for speed and vision for dynamic or custom interfaces. The introduction of Multimodal Large Language Models (MLLMs) has enabled intent-driven automation, allowing agents to interpret natural language instructions and operate across applications. Models like LayoutLMv3 [21] extract structured data from unstructured desktop environments, enhancing task execution. Despite these advances, the OSWorld benchmark [8] highlights a major performance gap: humans achieve 72% success on open-ended Windows tasks, while AI agents reach only ~12%. This is largely due to the scarcity of large-scale Windows datasets and the difficulty of maintaining spatial grounding across long, multi-step workflows.

6. Mobile Gui Automation (Comparative Review)

Mobile GUI automation operates in a more structured and controlled sandboxed environment, unlike the fragmented nature of the Windows desktop. This stability, along with the availability of large datasets such as *Android in the Wild (AITW)* [3] and evaluation platforms like *Mobile-Env* [22], has supported the development of highly capable automation agents. Recent systems such as *AppAgent* [1] and *Mobile-Agent* [7] rely on dynamic exploration and self-reflection to improve task execution. In addition, tools like *GPT-Droid* [13] and *CrashTranslator* [12] have enhanced automated bug detection and crash reproduction using language model reasoning. Accessibility-focused research has also contributed to better semantic understanding of mobile interfaces. Models such as *Screen2Words* [36] and *UIBert* [38] generate natural language descriptions of screens and learn general representations of UI elements. Although mobile automation research has progressed significantly, achieving similar performance in Windows environments remains challenging due to limitations in spatial grounding and the lack of large-scale desktop datasets.

7. Web & Cross-Platform Gui Automation

Web automation is generally simpler than desktop automation because the structured HTML DOM

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

makes it easier to identify and interact with interface elements. Benchmarks such as WebArena [9] provide realistic environments for evaluating web agents, while Mind2Web [5] shows that LLM-based agents can generalize across different websites using DOM information Cross-platform research focuses on reducing the need for platform-specific automation scripts. For example, Fazzini et al. [14] propose a semantic mapping technique that matches UI widgets across Android and iOS platforms, achieving about 85% portability. In addition, approaches using Graph Neural Networks (GNNs) [43] help improve exploration efficiency by predicting which UI elements are most likely to lead to the target state.

8. Comparative Analysis Across Approaches

This section summarizes insights from 45 studies, examining the trade-offs between different perception methods and reasoning engines based on four factors: generalization, robustness, latency, and complexity. Table 4,5.

8.1. Perception: The Robustness–Precision Trade-off

The choice of perception method defines the limitations of a GUI agent. Metadata-based approaches provide high precision and low latency, making them suitable for industrial RPA, but they are fragile since UI changes or canvas-based frameworks like Flutter can break selectors [19, 29]. Vision-based methods, such as CogAgent [4] and Spotlight [6], address this by detecting elements directly from screenshots, improving robustness across different interfaces. However, processing high-resolution desktop images can increase latency, often taking 2–5 seconds per step [1]. Hybrid systems, like UFO [2], balance these trade-offs by using metadata for speed and switching to visual analysis when necessary.

Table 4. Comparison of Perception Paradigms (2021–2025)

Feature	Metadata-Driven (RPA, Selenium)	Vision-Driven (CogAgent [4], Spotlight [6])	Hybrid (UFO [2], SoftRPA [19])
Primary Input	Accessibility Tree (XML/JSON)	Screenshots (pixels)	API/metadata + screenshots
Robustness	Low fragile to code changes	High code-invariant	High uses visual anchors
Generality	Limited canvas/games often fail	Broad universal UI coverage	Broad supports standard & custom UIs
Latency	Very fast (<100 ms)	Moderate–high (2–5 s with LLM)	Medium adaptive
Limitation	Missing/broken IDs	Coordinate hallucination	High engineering effort

8.2. Reasoning Engines: Learning vs. Understanding

The field has shifted from Reinforcement Learning (RL) to MLLM-driven reasoning. RL agents [26, 40] excel in closed-loop environments with clear rewards but lack the "common sense" to generalize to unseen apps. Conversely, MLLM agents (*AppAgent* [1], *Mobile-Agent* [7]) leverage pre-trained semantic knowledge to operate on novel interfaces without prior training. However, as shown in the OSWorld benchmark [8], this reasoning capability often suffers from "coordinate hallucinations," where the agent correctly plans an action but misses the target pixel due to spatial uncertainty.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

Table 5 Strength vs. Weakness Matrix Across Reasoning Methods

Method	Strengths	Weaknesses	Best Use Case
Scripting / Rules	100% Reproducibility; Zero Latency.	Zero adaptability; High maintenance cost.	Fixed, repetitive industrial tasks (RPA).
Reinforcement Learning	Fast execution; Super-human reflexes (Games).	Poor generalization; Sample inefficient training.	Automated Game Testing; Monkey Testing [26].
MLLM Agents	Zero-Shot Generalization; Natural Language Control.	High Latency; High Cost; Hallucinations [8].	Open-ended tasks; Assistant-like interaction [1].

8.3.Performance Disparity

A cross-platform comparison highlights a clear performance gap. Mobile agents operate in simpler, sandboxed environments, while Windows agents must handle complex, multi-window open-world settings. As a result, mobile agents are closer to human-level performance, whereas desktop agents currently achieve only about 12% task success on open-ended benchmarks [8].

8.4. Challenges In Modern Gui Automation

Major challenges include

- Coordinate Hallucination: Agents mispredict click positions [8].
- High Latency: MLLM inference can take several seconds per step [1].
- Privacy Risks: Vision models may capture sensitive screen data [31].
- Data Scarcity: Windows lacks large datasets comparable to AITW [3].

9. RESEARCH GAPS (WINDOWS + MOBILE + WEB)

A review of the existing literature highlights four key limitations that currently hinder the transition of these systems from research benchmarks to practical real-world deployment.

9.1. The Desktop Data Deficit

Unlike the mobile domain, which benefits from large datasets such as AITW [3] containing over 715k demonstrations, the Windows ecosystem still lacks large-scale training data. Existing resources like OSWorld [8] and RPA-US [18] are useful for evaluation, but they do not provide enough data to train robust and generalizable automation models. 10.2 Efficient Perception of "Black Box" Interfaces Many current approaches face difficulties when dealing with interfaces where accessibility metadata is hidden, such as those built with Flutter or Unity [29]. Vision-based models like CogAgent [4] can detect these elements, but they often require high computational resources. This highlights the need for lightweight, on-device semantic modelssimilar to ScreenRecognition [23] in mobile systemsthat can perform real-time inference on desktop environments.

9.2. Real-Time Decision Making

Highly autonomous agents such as *AppAgent* [1] and *Mobile-Agent* [7] typically require several seconds to perform each action. However, many real-world tasks demand much faster responses. Current research still lacks neuro-symbolic architectures that separate slow reasoning processes from fast execution, which limits the ability to automate tasks requiring real-time coordination.

9.3. Safety and Privacy in Open Worlds

Unlike sandboxed mobile systems, Windows agents operate in open environments where actions can have serious consequences, such as deleting files or modifying system settings. Although studies have reported privacy risks in automation scripts [31], most benchmarks focus mainly on task completion rather than safety considerations. As a result, formal frameworks for safe exploration and privacy

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

protection in shared operating system environments are still largely undeveloped.

10. Future Research Directions

10.1. On-Device Small Language Models (SLMs)

To reduce privacy risks [31] and the latency associated with cloud-based MLLMs [1], future research should focus on developing small language models (SLMs) optimized for on-device GUI understanding. Compact models with around 7B parameters can distill visual–semantic knowledge into efficient systems, enabling privacy-focused automation agents suitable for sensitive enterprise RPA environments [24]. Neuro-Symbolic Agent Architectures To overcome the precision limitations of purely MLLM-based systems [8], future designs should separate high-level planning from low-level execution. In this neuro-symbolic approach, LLMs handle reasoning and task planning, while rule-based components manage precise execution. Building on hybrid methods [2, 19], this structure can improve reliability and ensure safer, more controllable automation.

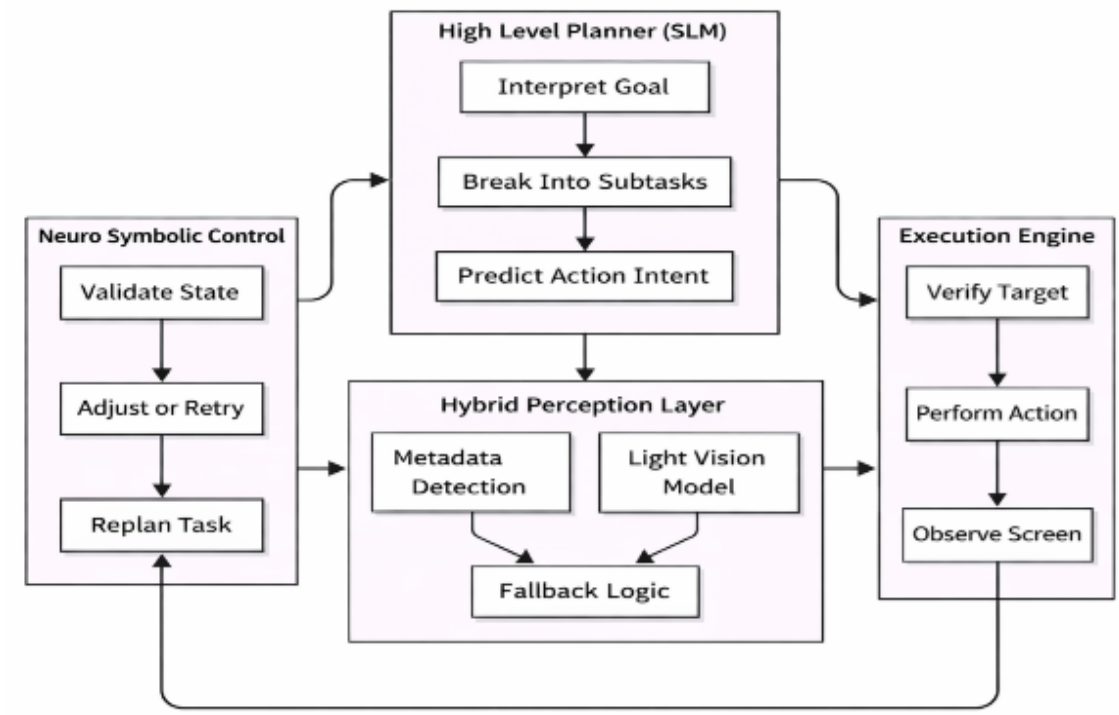

Figure 2 GUI agent with SLM-based planning and hybrid perception for precise interaction

Figure 2: presents a future-oriented GUI agent architecture where a Small Language Model (SLM) replaces large multimodal models as the high-level planner. The SLM interprets user goals and breaks them into smaller tasks, while hybrid perception and a symbolic execution engine provide accurate and low-latency control during interaction.

10.2. Windows in the Wild" Dataset

There is a strong need for a large-scale Windows-specific dataset comparable to AITW [3]. Such a dataset should include legacy software interfaces, Win32 forms, and real user interaction traces. Creating this resource would provide the necessary data foundation for training more robust and generalizable desktop automation agents.

10.3. Universal Pixel-Based Control

Inspired by platform-independent models such as CogAgent [4], future research may move toward

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

universal UI agents capable of operating across mobile, web, and desktop environments. By relying on pixel-based perception and generalized UI understanding, these agents could work seamlessly across different digital platforms.

Conclusion

This review summarizes the rapid evolution of AI-driven GUI automation between 2021 and 2025, highlighting the shift from brittle coordinate-based scripting and Reinforcement Learning (RL) approaches to autonomous Multimodal Large Language Model (MLLM) agents capable of semantic reasoning over complex interfaces. While mobile automation has matured due to sandboxed environments and large datasets such as Android in the Wild (AITW), Windows desktop automation remains an "Open World" challenge because of fragmented frameworks, inconsistent accessibility metadata, and complex multi-window workflows. Hybrid neuro-symbolic systems like UFO and SoftRPA show promise by combining API-level precision with vision-based robustness, but they are still limited by high inference latency and the lack of large-scale Windows datasets. Future progress depends on addressing the practical Trilemma of Latency, Privacy, and Precision through efficient Small Language Models (SLMs), reliable verification mechanisms, and diverse training datasets, enabling robust and privacy-preserving GUI automation across mobile, web, and desktop platforms.

References

[1] Chi, H., Liu, S., et al.: AppAgent: Multimodal agents as smartphone users. In: Proceedings of the 12th International Conference on Learning Representations (ICLR), pp. 1–18. Open Review.net (2024).

[2] Zhang, C., et al.: UFO: A UI-focused agent for Windows OS interaction. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp. 1–15. Association for Computational Linguistics, Mexico City (2024).

[3] Rawles, C., et al.: Android in the wild: A large-scale dataset for Android control. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 36, pp. 1–14. Curran As sociates, Inc., New Orleans (2023).

[4] Hong, W., et al.: CogAgent: A visual language model for GUI agents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–10. IEEE, Seattle (2024).

[5] Deng, X., et al.: Mind2Web: Towards a generalist agent for the web. In: Proceedings of the 12th International Conference on Learning Representations (ICLR), pp. 1–20. Open Review.net (2024).

[6] Li, G., et al.: Spotlight: Mobile UI understanding using vision-language models. In: Proceedings of the 11th International Conference on Learning Representations (ICLR), pp. 1 12. OpenReview.net (2023).

[7] Wang, J., et al.: Mobile-Agent: Autonomous multi-modal mobile device agent. In: Proceed ings of the 12th International Conference on Learning Representations (ICLR), pp. 1–16. OpenReview.net (2024).

[8] Xie, T., et al.: OSWorld: Benchmarking multimodal agents for open-ended tasks. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 37, pp. 1–15. Curran As sociates, Inc., Vancouver (2024).

[9] Zhou, S., et al.: WebArena: A realistic web environment for building autonomous agents. In: Proceedings of the 12th International Conference on Learning Representations (ICLR), pp. 1–15. OpenReview.net (2024).

[10] Venkatesh, S., et al.: UGIF: UI grounded instruction following. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1–12. Association for Computational Linguistics, Abu Dhabi (2022).

[11] Liu, H., et al.: Owl: A large language model for IT operations. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering

(ESEC/FSE), pp. 1–12. ACM, San Francisco (2023).
[12] Hu, Y., et al.: Crash Translator: Automatically reproducing mobile crashes. In: Proceedings of the 46th International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Lisbon (2024).
[13] Zha, Y., et al.: GPT-Droid: Automated GUI testing with large language models. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engi neering (ASE), pp. 1–13. IEEE, Luxembourg (2023).
[14] Fazzini, M., et al.: Automated cross-platform GUI testing. In: Proceedings of the 44th International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Pittsburgh (2022).
[15] Chen, Y., et al.: Themis: Automatic UI layout bug detection. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp. 1–12. ACM, Vienna (2024).
[16] Cooper, N., et al.: METER: Mobile enriched testing with environmental resources. In: Proceedings of the 46th International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Lisbon (2024).
[17] Lee, S., et al.: GUIComp: Graphical user interface design completion. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI), Article No. 45, pp. 1–15. ACM, Hamburg (2023).
[18] Román, A., et al.: RPA-US: A dataset for robotic process automation. In: Di Ciccio, C., et al. (eds.) Business Process Management (BPM 2023). LNCS, vol. 14159, pp. 1–15. Springer, Cham (2023).
[19] Agrawal, M., et al.: SoftRPA: Software-based robotic process automation. In: Proceedings of the 46th International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Lisbon (2024).
[20] Degiovanni, R., et al.: Automated generation of RPA bots from natural language. In: In dulska, M., et al. (eds.) Advanced Information Systems Engineering (CAiSE 2023). LNCS, vol. 13901, pp. 1–16. Springer, Cham (2023).
[21] Yu, Y., et al.: LayoutLMv3: Pre-training for document AI. In: Proceedings of the 30th ACM International Conference on Multimedia (MM), pp. 1–10. ACM, Lisbon (2022). Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–10. IEEE, Seattle (2024). 5. Deng, X., et al.: Mind2Web: Towards a generalist agent for the web. In: Proceedings of the 12th International Conference on Learning Representations (ICLR), pp. 1–20. Open Review.net (2024).
[22] Feng, Y., et al.: Mobile-Env: An evaluation platform and benchmark for LLM-GUI interaction. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1–15. Association for Computational Linguistics, Bangkok (2024).
[23] Zhang, X., et al.: ScreenRecognition: Creating accessibility metadata for mobile apps. In: Proceedings of the 2023
[24] Leno, V., et al.: Robotic process mining: Vision and challenges. In: Polyvyanyy, A., et al. (eds.) Business Process Management (BPM 2021). LNCS, vol. 12875, pp. 1–16. Springer, Cham (2021).
[25] Gao, Y., et al.: Retrieval-augmented generation for code summarization. In: Proceedings of the 44th International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Pittsburgh (2022).
[26] Pan, M., et al.: Reinforcement learning for Android GUI testing: A review. IEEE Transac tions on Software Engineering (TSE) 48(5), 1–18 (2022).
[27] Liu, Z., et al.: Nighthawk: Fully automated local UI testing for Android. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp. 1–12. ACM, Seattle (2023).
[28] Zhang, Y., et al.: ScreenTag: Unsupervised

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3249-3261
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0415

UI element tagging. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI), Article No. 34, pp. 1–12. ACM, Yokohama (2021).

[29] Mao, K., et al.: Automated accessibility testing of mobile apps. ACM Transactions on Software Engineering and Methodology (TOSEM) 32(3), 1–25 (2023).

[30] Chen, J., et al.: StoryDroid: Automated storyboard generation. In: Proceedings of the 43rd International Conference on Software Engineering (ICSE), pp. 1–12. IEEE Press, Madrid (2021).

[31] Wu, H., et al.: Privacy auditing of GUI-based interactions. In: Proceedings of the 32nd USENIX Security Symposium, pp. 1–18. USENIX Association, Anaheim (2023).

[32] Li, Y., et al.: Widget-sensitive GUI testing. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1–12. IEEE, Michigan (2022).

[33] Rossi, A., et al.: Human-in-the-loop GUI automation. In: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI), Article No. 56, pp. 1–14. ACM, Honolulu (2024).

[34] Lin, J., et al.: Revamping the UI: Automated refactoring of Android layouts. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1–12. IEEE, Melbourne (2021).

[35] Zhao, N., et al.: Frustra: Detecting user frustration via GUI interactions. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST), pp. 1–14. ACM, San Francisco (2023).

[36] Wang, B., et al.: Screen2Words: Automatic mobile UI summarization. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI), Article No. 22, pp. 1–15. ACM, Yokohama (2021).

[37] Swearngin, A., et al.: Scout: Rapid exploration of interface layouts. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI), Article No. 78, pp. 1–12. ACM, New Orleans (2022).

[38] Huang, Z., et al.: UIBert: Learning generic pixel-representations for UI. In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI), pp. 1–9. IJCAI Organization, Macao (2023).

[39] Burns, A., et al.: Interactive navigation for visually impaired. In: Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS), pp. 1–12. ACM, Athens (2022).

[40] Schoop, E., et al.: Android Env: A reinforcement learning environment for Android. In: ICLR 2024 Workshop on Generative AI for Decision Making, pp. 1–8. OpenReview.net (2024).

[41] Wang, T., et al.: LIR: Learning to invariant representation for UI. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–10. IEEE, Nashville (2021).

[42] Aliannejadi, M., et al.: Cross-app mobile search. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 1–11. ACM, Madrid (2022).

[43] Zheng, Y., et al.: Click-through rate prediction for GUI elements. In: Proceedings of the ACM Web Conference 2024 (TheWebConf), pp. 1–10. ACM, Singapore (2024).

[44] Behrens, D., et al.: MapUI: Learning to map GUIs for navigation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1–10. IEEE, Paris (2023).

[45] OpenAI, Google DeepMind: Frontiers in user interface agents: A meta-review. Nature Ma chine Intelligence 7(1), 1–12 (2025).