UINavBench: A Framework for Comprehensive Evaluation of Interactive Digital Agents

Harsh Agrawal*, Eldon Schoop*, Xinlei Pan°, Anuj Mahajan°, Ari Seff°, Di Feng°, Ruijia Cheng,
Andres Romero Mier Y Teran, Esteban Gomez, Abhishek Sundararajan, Forrest Huang,
Amanda Swearngin, Mohana Prasad Sathya Moorthy, Jeff Nichols, Alexander Toshev†
Apple
{hagrawal2, eldon, toshev}@apple.com
*First Author °Core Contributors †Project Lead

Abstract

We build a comprehensive online evaluation benchmark for language-conditioned multi-step task execution on mobile interfaces. Our benchmark strives to evaluate the multi-step planning, reasoning, and visual grounding capabilities of agents, using mobile user interfaces as a concrete testbed. To build diverse, challenging tasks that reflect real-world use cases, we propose an exhaustive taxonomy that allows us to measure progress along multiple decision-making abilities including multi-step planning, visual perception, action grounding, and using memory or external knowledge. We also highlight important factors such as statefulness, safety, and evaluation complexity that are key to design tasks that can be reliably evaluated. Using this taxonomy, we design 116 tasks across 36 unique apps. Through an automatic framework, we stage and evaluate several natural baselines with different input representations and planning strategies. We show that the best-performing agent achieves 40% success on our benchmark. We further measure agents' abilities to plan, ground, and utilize world knowledge highlighting areas of improvement.

1. Intro

Building autonomous agents has been a long-standing goal in Artificial Intelligence (AI) [1–4]. With recent advances in Large Language Models (LLMs), and Vision-Language Models (VLMs), there has been a surge in the development of interactive digital agents that can automate tasks on mobile phones [5–13]. These agents are designed to automate everyday activities such as shopping for groceries, planning trips, and organizing calendars.

Several benchmarks have been introduced to evaluate these agents' ability to understand and navigate Web [14–18], Android [19–25], and Desktop environments [26–28]. Benchmarks for mobile agents have typically been offline, consisting of static sets of images or ground truth trajectories against which an agent is evaluated [20–22, 25]. While offline benchmarks are performant, they do not reflect the real-world stochasticity of mobile environments, and they have limited ability to measure how agents recover from mistakes or re-plan.

In contrast, online benchmarks require agents to interact with realistic dynamic environments. However, building online benchmarks for mobile devices presents the unique challenges of staging, managing state, and designing evaluations that can account for dynamic content. Recent efforts avoid these challenges by collecting tasks for small sets of apps which have limited effects on external databases [24, 25]. More importantly, many of these benchmarks suffer from limited task coverage due to sourcing tasks from LLMs (which risks sampling tasks the LLM already has knowledge of) or are open-ended, making robust evaluation harder [25]. Other benchmarks [23] use tasks from datasets known to have limited diversity [19, 20].

Toward building a more diverse and realistic benchmark, we introduce **UINavBench**, a platform that can stage, watch, and evaluate agents on mobile user interfaces. A key contribution of our work is a task taxonomy that allows us to express a variety of axes along which our system can measure different aspects of agent performance (Fig. 1). We involved domain experts in HCI, digital agents, robotics, and AI safety to co-create this taxonomy and parameterize it with **116** carefully designed tasks across **36** apps that measure agents' abilities to plan and reason, ground, and integrate memory and knowledge. Our taxonomy categorizes tasks along several dimensions—such as perceptual complexity, memory requirements, planning, fine-grained interactions, action diversity (clicks, typing, gestures), the number of interactions, unique screens, and multi-app usage—to

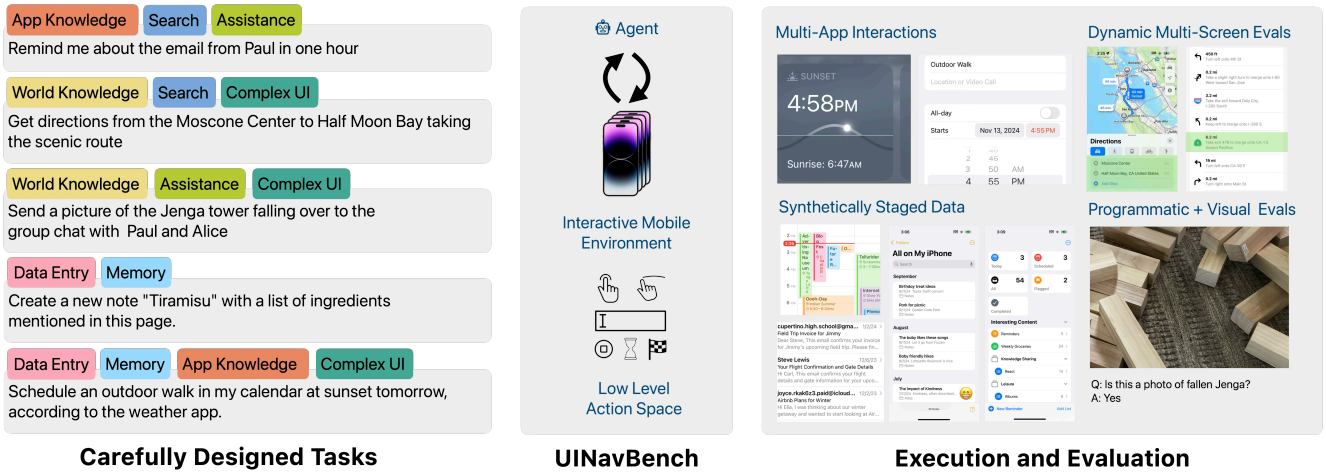

Figure 1. **UINavBench**: A framework for evaluating interactive digital agents on mobile devices. Our benchmark provides a diverse set of tasks across different apps, with varying levels of complexity within planning and reasoning, visual grounding, and memory and knowledge. The framework includes robust evaluation mechanisms, and handles staging and cleanup steps for tasks requiring setup.

ensure balanced difficulty and robust benchmarking.

Another key contribution of our work is implementing carefully designed staging and post-completion steps for tasks that require setup, e.g., user accounts and staged data. These setup and teardown routines are written by experts in iOS and used to reset the environment before interaction and minimize the impact on the external state. To evaluate task success, we design an expressive set of criteria to programmatically check for success by inspecting the text and UI elements present on the last screen or the interaction history. We use this approach in conjunction with a VLM-as-judge evaluation [29] which we verify to align strongly with expected ground-truth answers. We report both mean episode success as well as path efficiency.

A unique aspect of our benchmark is that it is the first to evaluate agents on the iOS platform. While training data exists for UI navigation on Android, Web and Desktop [19, 21, 30–32], public UI navigation data is scarce for iOS. This enables the community to study whether today's agents generalize to new platforms.

Finally, we experiment with five baselines consisting of both open-weights end-to-end UI navigation methods as well as modular methods using a commercial multi-modal language model, GPT-4o [33], as a planner. We experiment with different representations of the input screen, different grounding methods, as well as popular reasoning techniques like ReACT [34], and Reflexion [35]. The best-performing method achieves 40.5% episode success on our full benchmark. We further break down the performance along key axes of decision-making—planning and reasoning, grounding, and memory and knowledge—to highlight areas of improvement in existing methods. We also plan to make **UINavBench** accessible to the research community.

2. Related Works

Static Benchmarks for Digital Agents Static approaches of benchmarking digital agents compare predictions against pre-recorded, *static* demonstrations by human users, and measure step-wise or episode-wise error rates. Prior works have built benchmarks for mobile UI navigation tasks [19, 21, 30, 32, 36], web navigation tasks [15, 37, 38], and cross-platform tasks covering web, desktop and mobile devices [39–42]. While these approaches are easy to scale and are relatively stable across multiple execution runs, they often do not reflect real-world challenges that agents might face, such as having to replan their task completion approach after a non-deterministic UI pops up in the app. As such, our work focuses on building a dynamic benchmark.

Interactive Digital Benchmarks In contrast, dynamic approaches measure digital agents' task completion performance in live environments. These simulate the real environments the agents are deployed in [17, 43–45], such as in a controlled web [14, 46] or mobile [23] environments. While these benchmarks are more realistic, their tasks were designed in an ad-hoc fashion, or were sourced from the internet and/or LLM. In contrast, our curated tasks based on the taxonomy ensure balanced category representation and enable deeper analysis beyond success rate. Further, instead of relying on access to the device state, our automatic evaluation uses screen observations to validate task success allowing verification to be agnostic of app implementation. We are also the first benchmark designed on the iOS platform enabling the community to test the generalization capability of agents.

Mobile Agents Prior work in digital mobile agents can be further categorized into multi-agent systems and end-to-end agent models. Multi-agent systems (or agentic work-

Group	Category	Definitions	Count	Example Task(s)
Planning & Reasoning	Task Length	Simple (1-3 actions required)	18	Simple: Sort my [stocks] watchlist by price change
		Medium (4-6 actions required)	35	Medium: Add the Physics [podcasts[category to favorites
		Complex (7+ actions required)	61	Complex: What is the moonrise time during next new moon day
	Navigation Depth	Surface (Must navigate 1 unique view, or 2 if the second is reachable in one step)	26	Surface: Change the ringtone of the existing timer to Daybreak
		Shallow (Must navigate 2-3 unique views)	52	Shallow: Add Apple Park Visitor Center to my pinned locations
		Deep (Must navigate 4+ unique views)	38	Complex: Change the clock settings to only allow alerts on the lock screen and banners
	Multiple Applications	Requires interacting with multiple applications	29	Multi App: Search for the first hike in "Baby Friendly hikes" note in Apple Maps
	Goal Specification	Direct (desired state is described in the task)	93	Direct: Draft a text to Alice saying if I have more events on the 12th or 13th between 2 and 5 pm
		Under-constrained (multiple plausible goal states are feasible)	23	Underconstrained: In Youtube, play a video about how to cook Tamales
	Determinism	Deterministic (taking an action will result in the same state change with the same initialization)	50	Deterministic: Schedule a time block this afternoon from 2 pm to 3 pm for dental visit
		Non-Deterministic (same actions may lead to different results, or content is dynamic)	66	Non-Deterministic: I want to see the top books in Libby, sorted by popularity
Grounding	Perceptual Difficulty	Text Only (only requires perceiving text)	30	Text: In google maps, find the driving time between madrid and paris.
		Text+UI (requires text and common UI elements, e.g., tabs, toggles, common icons)	47	Text+UI: Add a cup of Lemonade for kids size to my starbucks order.
		Text+Visual (requires text and uncommon UI or graphics, e.g., charts, images, diagrams)	39	Text+Visual: What is the moonrise time during next new moon day
	Action Complexity	Tapping the screen	108	Enable Voiceover in Settings
		Typing Text	74	Search for the note about curiosity
		Swiping (to manipulate a UI element or scroll a view)	38	Go to the references at the end of this document
		Drag-and-drop: Gesture to finely arrange elements on the screen	6	Gesture: Draw a palm tree on top of a rabbit in Freeform
		Double Tap: Tapping the screen at a location twice in quick succession	5	Double Tap: Create a new slide that says “Orcas are the best” with a picture of an orca from wikipedia
Memory & Knowledge	Task Memory	Requires recalling specific details from earlier in an interaction	30	Add all the ingredients from the quesadillas recipe I downloaded to my Whole Foods cart
	Env. Knowledge	Understand uncommon or app-specific navigation design patterns	40	Make the spoken directions volume in maps louder
	World Knowledge	Requires knowledge from outside the environment	15	Get directions from the Moscone Center to Half Moon Bay taking the scenic route

Table 1. Task Taxonomy for **`UINavBench`**. We consider a diverse set of axes along which we want to measure agent performance. The axes include task length, navigation depth, perceptual difficulty, multi-app tasks, action complexity, among others.

flows) [37, 47–49] commonly include several modules such as a screen perception component (e.g. via UI detectors, application trees), a planner agent (which typically leverages LLMs [50–53]), an action agent, and an evaluator [37, 54–57]. In contrast, end-to-end agent models are unified models that jointly perform mobile UI perception, task planning, and action prediction. The models directly translate raw input images into low-level actions [19, 58–61]. So far, agentic workflows implemented using commercial models have outperformed end-to-end models. We conduct experiments using both approaches.

3. Benchmark

`UINavBench` provides a testbed for evaluation and development of Digital Agents. A design goal of our system was to evaluate agents on *meaningful* tasks along general challenges in agent design, to be useful beyond the concrete instantiations in mobile interfaces.

1. **Planning and Reasoning.** Performing tasks specified in language on mobile apps requires breaking down a high-level description into concrete interactions, often with a large number of steps. When deployed in online settings, agents will need to adapt plans to changes in the environment and external state.
2. **Visual Grounding.** To execute plans, digital agents must perceive the semantics and affordances of mobile UIs and ground various actions into screen elements. This can be challenging in a environment with animations, dynamic content, and frequent UI updates.
3. **Memory and Knowledge.** Solving tasks can require remembering past interactions, user preferences, searching external databases, the web, and account history. Digital agents must also contend with tasks that are not fully specified, and use reasoning abilities to infer intent or draw upon world knowledge.

As a central contribution of this work, we introduce a *comprehensive taxonomy* in the following sections that categorizes tasks along these fundamental axes of multi-step decision-making (Sec. 3.1). We expand the taxonomy with criteria important for reliable evaluations (Sec. 3.2), reflecting several practical challenges that exist in staging data, evaluating agents in dynamic environments, and ensuring the safety of repeating tasks at scale without side effects.

3.1. Multi-Step Decision Making Challenges

In this section, we describe our taxonomy’s axes within the challenges in multi-step decision-making described above. A full description of these axes is listed in Tab. 1.

3.1.1. Multi Step Planning / Reasoning

Task Length. A simple yet incomplete measure of planning difficulty is counting the actions required to complete the task. We divide this into 3 buckets: *simple* (1-3 interactions); *medium* (4-6), and *complex* (7 or more).

Navigation Depth. Navigating many views requires a memory of app structures or having a strong prior for common affordances across apps. This measure is not always related to task length. For example, scrolling through search results can involve many swipe actions within the same view, but updating some settings within apps can require traversing a new view with almost every action.

Multiple Applications. Many real-world scenarios require multiple apps, testing planning and memory when transferring information between apps.

Goal Specification. *Underconstrained* tasks pose challenges when agents struggle to determine if the task is complete. Tasks such as “play a video of how to cook tamales” require agents to decide if a video satisfies the task.

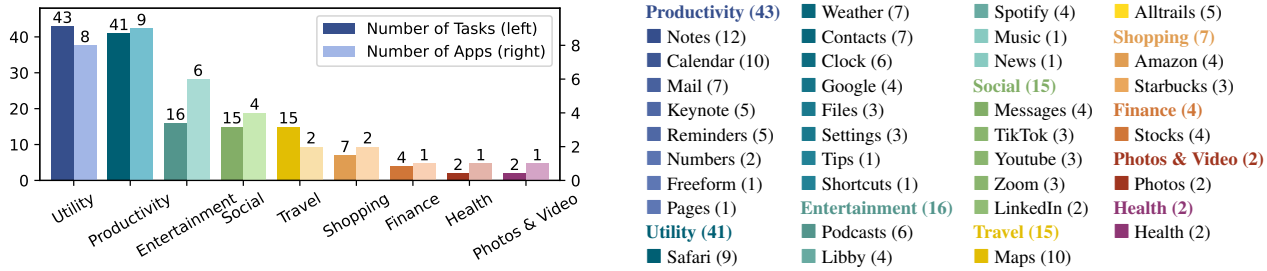

Figure 2. Domains used in **`UINavBench`**. **Left:** count of total tasks (left bars) and apps (right bars) for each domain. **Right:** A detailed breakdown of apps in each domain. Note that some tasks require interacting with multiple apps.

Task Intent. To understand how user intent can affect planning, we used open coding [62] to categorize our final set of tasks into a set of common intents. We categorize tasks as *nav*, traversing an app to a specific view; *search*, retrieving information through text or faceted filters [63]; *assistance*, manipulating content or controls; and *data_entry*, entering text from the goal or memory.

3.1.2. Visual Grounding

Perceptual Difficulty. Mobile UIs have different design characteristics than desktop applications and often require a deeper semantic understanding of UI components to effectively take actions [64]. Tasks require understanding a broad set of common UI components [65] and icons [66]. Other tasks require understanding more complex graphics, such as maps, diagrams, and image content.

Action Complexity. Unlike robotics tasks which typically have a fixed action space, most digital tasks require a combination of *click*, *swipe*, and *type* actions. Difficult tasks require *gestures*, e.g., to position or resize a graphic.

3.1.3. Memory and Knowledge

We measure three ways in which agent can recall relevant information required to complete a task. *Task memory* requires recalling information from an earlier observation in the trajectory. Other tasks may require *environment knowledge* to navigate apps with idiosyncratic design patterns. Some tasks require *world knowledge*, or commonsense knowledge from outside the environment.

3.2. Implementation Challenges

There are many practical challenges to maintaining consistent state in online mobile environments. In this section, we detail some challenges with staging and evaluation.

3.2.1. Repeatable staging of the environment.

Setup Criteria. Restoring the state of apps to be consistent across multiple runs may involve setting up *user accounts* to access app capabilities and *staged data* to make the task feasible ("delete my note about concerts" requires the note to exist). While creating tasks, we constructed repeatable staging setups through OS automation APIs. Tasks that require *payment*, *sensors* such as a camera or accelerometer, or *physical hardware*, like a smart vacuum are excluded.

Post Execution Cleanup. Executing realistic tasks often requires making persistent changes to state. We categorize all potential changes our tasks may result in. Tasks can have *no effect*, i.e., when purely navigating or searching (impacts to ads or recommendations are not considered), and require no cleanup upon episode termination. Other tasks affect *local* app state, such as setting alarms, which are handled automatically when the device is reset across runs. In some cases, state changes are persistent in a remote database that cannot be easily reset. *External private* changes are not visible beyond the user and the app they are interacting with, e.g., creating a note in a cloud account. *External public* state changes are generally visible to, or affect, the outside world, e.g., following a podcast channel. For these, we programmatically take actions on the UI post-execution using OS automations to ensure all changes are undone.

Safety Considerations Importantly, the risks of leaving lasting impacts on the real world are compounded in the large-scale online benchmark setting. Running even simple tasks can have unintended consequences at scale, e.g., playing a song on a streaming service can artificially increase its popularity. We carefully design tasks to have minimal side effects. We also exclude tasks with reversibility or impact concerns as defined by a recent work introducing a taxonomy of safety impacts for UI agents [67].

3.2.2. Reliable evaluation.

Different agents can complete tasks through different trajectories. Validating these different trajectories against ground truth without inspecting system state is complex. In the simplest case, a task can be designed so that its success can be determined entirely by the contents of the *last screen* only, e.g., adding a product to a shopping cart. Other tasks require inspecting *multiple screens* in the trajectory if the last screen is not enough, e.g., adding a calendar event with many attendees. Evaluations can consider *text* (e.g., matching the product name), or *visual* information (e.g., the product's color). Supported evaluations are discussed in Sec. 4.4.

3.3. Benchmark Details

Overall, we believe that the task taxonomy not only helps measure key agentic abilities required for multi-step decision-making but also gives the community a *"data card"* to help design new tasks.

4. Environment

The mobile **UINavBench** environment is formalized as a partially observable Markov decision process (POMDP). At each time step, the agent receives an observation (Sec. 4.2) that partially describes the current state of the environment. The observation primarily consists of a screenshot of the current screen on the device, and a textual description of the goal. Given the current observation, the agent takes an action (Sec. 4.3) in the environment, and receives the next observation. Actions include common mobile interactions such as tapping, swiping, and text input. The interaction between the agent and the environment continues until the agent terminates the episode, or a maximum number of steps is reached. We run evaluation on task-completion (Sec. 6.1) to measure the performance of the agent.

4.1. Simulation

For each episode, **UINavBench** provisions a remote physical device running iOS 18.1. The device performs all task pre-staging steps like setting up user accounts, installing necessary apps, and staging any required data. We support a number of apps, including pre-installed apps such as Notes, Clocks, and Weather, as well as apps available in the App Store. The agent uses a VNC server on the device to observe the screen state and control the device. To ensure repeatability across evaluations, we silence several system notifications and dialogs using operating system APIs. Using the task taxonomy, we design **116 task** tempaltes across **36 apps**. These apps cover a range of app categories like utility, productivity, entertainment, social and more. The apps and corresponding task distributions are shown in Fig. 2.

4.2. Observation Space

The observation includes the current screenshot and the natural language goal description. Inspired by AXNav [54], the environment also provides a list of UI elements obtained from a detection model [65, 66]. Each element in the list contains an integer id, detected label (e.g., Icon, Label, Button, etc), contained text (if any), bounding box coordinates, and predicted clickability. We further filter the detection outputs by removing spurious detections of single character keys on the virtual keyboard, preserving special keys like "enter", "go", etc., since they are useful for navigation.

4.3. Action Space

Our environment supports all mobile input types required to complete our tasks. Supported actions are summarized in Table 1. Click based interactions like `CLICK`, `LONG_PRESS`, and `DOUBLE_CLICK` take (x, y) coordinates as input which specify the location on the screen to interact with. We also support `SWIPE` actions which take start (x_1, y_1) and end (x_2, y_2) coordinates as input. The swipe action is implemented by starting a touch down at the start coordinates, moving to the end coordinate in small steps, and lifting up at the end coordinate. For entering text, the agent can use the `TYPE` action which takes a string as input. The environment supports additional actions like `WAIT` which takes a number of seconds as input, a special `NAVIGATE_HOME` action, and sending special keys like `ENTER`, `BACKSPACE`, and `SHIFT` using the `SEND_KEYS` action.

4.4. Task Validator

Manually verifying the success of a given task execution by an agent can be tedious. Episodes can contain dozens of steps, and there are often multiple correct ways of completing tasks. An accurate evaluation needs to be robust to variations in successful trajectories. We develop an automated task validator (Fig. 3) which can account for such variations in successful trajectories without requiring a human in the loop. While this relates to the general problem of reward modeling, in our setting the set of tasks is fixed and known in advance. We do not require our validator to generalize to unseen tasks, only requiring it to generalize across different agent trajectories, for it to perform this task successfully.

We take the simple approach of evaluating episodes based on a set of predefined criteria which must all be satisfied for the trajectory to be labeled as successful. The validation criteria utilized here are either based on handcrafted, programmatic rules, or based on inference outputs from a vision-language model (VLM). For each task, criteria are iterated until 100% alignment with human judgment is observed on 5 successful and failed *human-collected* trajectories.

Rule-based. Our rule-based criteria check for expected markers encountered during an episode from the environment state. This is similar to rule-based evaluation frameworks employed by some open-ended reasoning benchmarks like MMMU [68].

- `TextContains`: The screen must contain certain key strings, with an optional UI element type also matching with the ground truth type. The UI element type options include text, checkbox, tab, toggle, icon, and button.
- `TextClose`: In cases where it is necessary to check for proximity between two text strings, we evaluate whether one text string is the closest to another target text string, in either the horizontal direction, vertical direction, or both.
- `TimeRangeMatch`. In tasks involving constraints on an input time (e.g., a calendar event), this criterion implements a corresponding check.

Multi-screen evalation. For some tasks, success criteria

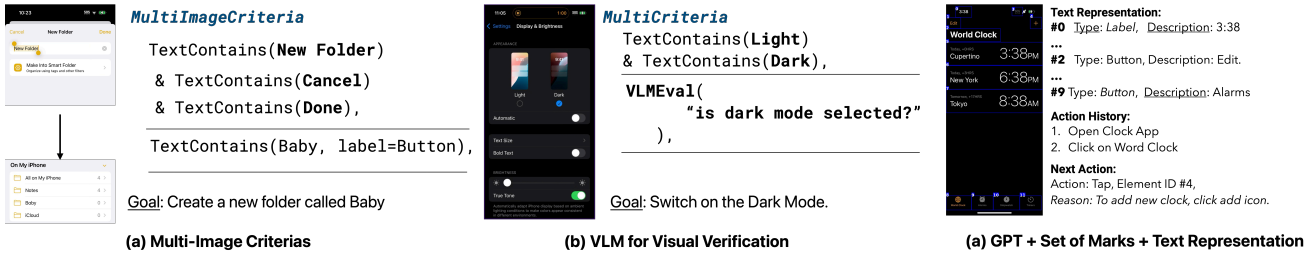

Figure 3. **(a) Multi-Image Criteria**: We support testing criterias jointly across multiple frames when last-screen evaluation cannot completely verify success. **(b) VLM Eval**: We also employ vision-language models to check criteria that cannot be evaluated using text (e.g., the correct color of a product is added to cart). **(c) Input representation of our model**: We employ set-of-marks prompting along with a compact text representation. At each step, the agent also uses action history to reason about the next action.

need to be specified jointly across multiple frames of an episode (Fig. 3(a)). Our system also supports these scenarios by composing rule-based criteria.

VLM-based. We also employ vision-language models (VLM), such as GPT-4o, to evaluate device screens using task-specific prompts (Fig. 3(b)) This naturally complements the stricter criteria above with more flexible evaluation that can account for underconstrained tasks.

Logical compositions. The above criteria may be arbitrarily composed using AND or OR logic to form new validation criteria. AND logic is used in cases where multiple text strings must be present to validate a screen, or multiple criterias must be satisfied at different parts of the trajectory. OR is used when the presence of two or more alternative strings may equally satisfy a criterion.

We debated using database/file based evaluation. We found combining rule-based and VLM-based criteria removed the need for database queries and API calls specific to apps, making it easier to generalize our validators to new tasks, especially *dynamic tasks that cannot be evaluated using databases or accessibility metadata (e.g. Maps)*. This allowed us to scale to 36 apps (vs 12 in OSWorld and 20 in AndroidWorld). For each task, we evaluated against multiple failed and successful trajectories to refine the criteria. We share examples in the supplementary material.

5. Baselines

We experiment with five approaches to building interactive agents. All baselines have a common base architecture, consisting of an off-the-shelf VLM, GPT4o [33], as the planner. It takes the natural language goal, screenshot at the current step, and action history as input. Since GPT4o cannot output pixel-level actions directly, we use set-of-marks (SoM) [47] prompting by drawing bounding boxes around all detected UI elements on the screen and annotate each with a unique ID. To ground actions into the screen, the planner outputs the relevant action (e.g., click, type) and SoM ID on which the action will be performed. The action type and the SoM ID are heuristically converted into a pixel-level action (using the center of the SoM bounding box) to be executed on the mobile device. We also experiment with UGround [69], a state-of-the-art grounding model on the ScreenSpot UI grounding benchmark [70]. When using UGround, the GPT4o based planner outputs a language description of the element on which the action will be performed. A VLM based grounding model then converts this description into a pixel-location on the screen. Finally, we experiment with an end-to-end baseline. Overall we have:

1. **ReACT:** Given a goal, current screenshot and action history, we use the ReACT framework [34] to first reason about the goal, UI state, and history to come up with the next action that will lead to the goal. After the reasoning chain, the planner outputs a structured action, which is converted into a pixel-level action by our grounding heuristic. At the next time step, the previous action and reasoning chain is appended to the action history.
2. **ReACT + Text:** This is the same as ReACT, but with added simplified text representations of the detected UI elements as context (Fig. 3(c)). Similar to findings in Omniparser [49], we noticed that this helps the model interpret SoM more easily. We follow AXNav [54] to generate text representations consisting of a list of SoM IDs for the detected UI elements, their labels (e.g., button, progress bar), and corresponding text if present.
3. **Reflexion + Text:** To improve the reasoning proces, we experiment with reflexion [35]. In this setup, the agent uses a VLM critic that looks at the screen before and after taking an action, and critiques the action. It reasons about whether the action changed the state of the UI, and whether any progress was made towards the goal. This feedback is appended to the action history, and used by the planner to come up with an improved action. We use GPT4o as the critic.
4. **Reflexion + UGround:** This is the same as Reflexion + Text, but we use the UGround UI grounding model instead of heuristics to ground the actions into the screen.
5. **UI-TARS:** Finally, we experiment with a state-of-the-art open-weights end-to-end model UI-TARS [71] specifically trained for UI interaction. We use the 72B version of UI-TARS with the QwenVL2 [72] backbone.

6. Experiments

We evaluate the performance of the five baselines discussed in Sec. 5 in **UINavBench**.

6.1. Metrics

- **Episode Success (ES):** This is a binary indicator evaluating if the agent was able to successfully execute the task.
- **Success Weighted by Path Length (SPL):** Popular in embodied navigation tasks [73], we use SPL to compute path efficiency. SPL weighs Episodic Success by the ratio of Agent Trajectory Length and GT Trajectory Length. Intuitively, this ratio will be closer to one if the agent's trajectory is as short as the optimal expert trajectory.

Please note that episodic success for all baselines are manually *verified by multiple humans*.

6.2. Results

#	Model	Episode Success (ES)	SPL
1	ReACT	25.8%	0.230
2	ReACT + Text	31.8%	0.260
3	Reflexion + Text	40.5%	0.345
4	Reflexion + UGround	33.5%	0.235
5	UI-Tars	11%	0.085

Table 2. Performance of the three baselines on **UINavBench**

Augmenting Set-of-Marks with Text-Based Representations and Reflexion improves performance. In Tab. 2, we report the performance of the three baselines on our benchmark. There are two key observations. First, consistent with Omniparser [49], we show that augmenting the set-of-marks (#1) with a text-based representation (#2) improves the performance on both episodic success (+6%) and SPL (+0.03). Second, adding Reflexion (#3) further improves the performance by episodic success (+9.7%) and SPL (+0.085). This shows that models that self-critique their actions and use this to replan can improve over a model that simply uses per-action reasoning. Fig. 4(a) shows how agent can succesfully replan using Reflexion.

End-to-end agents still perform poorly. We observed that end-to-end methods still perform poorly on our benchmark. UI-Tars [71] (#5) achieves 11% success compared to 40.5% for Reflexion + Text (#3). On the other hand, on Android World and OSWorld, UI-Tars achieves 46.6% and 24.6% success respectively. This gap in task success can be attributed to the lack of environment specific training data for UI-Tars. Compared to other environments like Web, Desktop and Android, which have public datasets [19, 21, 30–32], iOS lacks public UI navigation datasets. *This result underscores the importance of evaluating agents against diverse environments to test generalization.*

Grounding models generalize better but still underperform. In Tab. 2, we show that grounding models (UGround) generalize better than end-to-end models (UI-Tars). Since UGround was largely trained on synthetic data, this result is quite promising. However, grounding models still underperform compared to set-of-marks based models (#3), despite strong grounding performance on grounding benchmarks like ScreetSpot Pro [74].

Length \ Unique Views	Surface	Shallow	Deep
Simple	11/14 (79%)	3/4 (75%)	0/0
Medium	6/9 (67%)	17/24 (71%)	2/2 (100%)
Complex	1/3 (33%)	3/22 (14%)	4/36 (11%)

Table 3. Success of the **Reflexion + Text** baseline grouped by task depth and unique views

Agents do well on short-horizon tasks. In Tab. 3, we show that agents do reasonably well on tasks that are short-horizon achieving 79% success. As shown in Fig. 4(b), this is especially true when interactions are with common UI elements (search field, add icon)

Agents perform poorly on long-horizon tasks In Tab. 3, we show both the number of actions required to complete a task (rows) and the number of unique views to be navigated (columns) have a bearing on task difficulty. It is difficult for agents to integrate the memory and knowledge required of deep tasks into planning long sequences (Fig. 4(f)), leading to poor performance on complex, deep tasks.

# Model	Pure Nav. (14)	Search (41)	Assistance (60)	Data Entry (47)
1 ReACT	42.8%	26.8%	25%	12.8%
2 ReACT + Text	50%	43.9%	28.3%	10.6%
3 Reflexion + Text	64.3%	63.4%	33.3%	8.5%
4 Reflexion + UGround	50%	51.2%	36.7%	10.6%
5 UI-Tars	21.4%	22%	8.3%	0%

Table 4. Success of baselines split by task intent. The number of tasks for each intent is shown in the brackets.

Agents show poor performance on Assistance and Data Entry tasks. In Tab. 4, we show that agents exhibit poor performance on tasks requiring editing text or changing the state of the app (e.g., setting a timer, adding a note). We hypothesize this is due to many of these tasks requiring planning several steps and drawing on grounding ability to interact with fine-grained UI controls.

# Model	Text Only (30)	Text+UI (47)	Text+Visual (39)
1 ReACT	50%	29.8%	2.6%
2 ReACT + Text	56.7%	34%	10.3%
3 Reflexion + Text	80%	42.6%	7.7%
4 Reflexion + UGround	50%	44.7%	12.8%
5 UI-Tars	23.3%	10.6%	2.6%

Table 5. Success of baselines split by perceptual difficulty

Agents struggle with visual perception and fine-grained

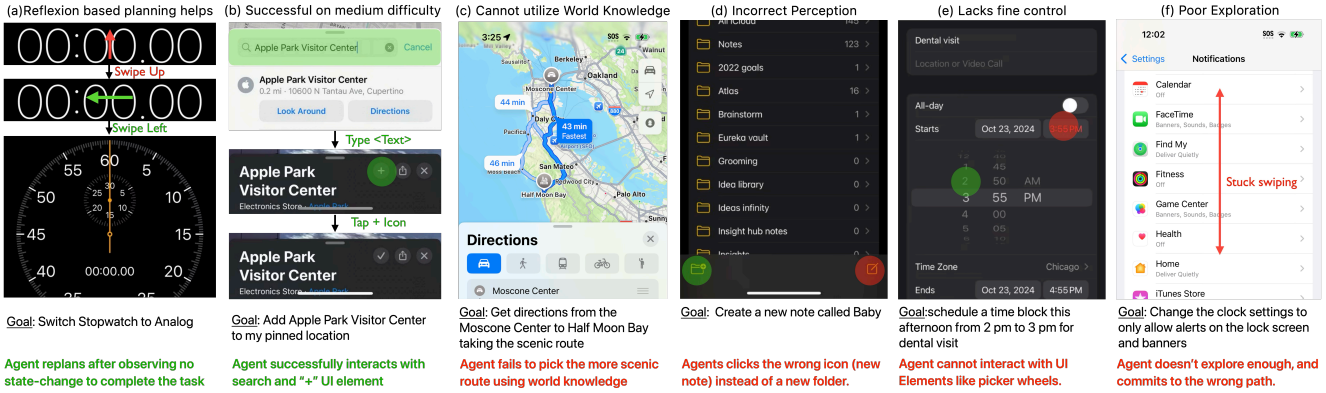

Figure 4. Examples of successful (green) and failure (red) cases. We find that reflexion helps with planning (a), and agents are generally successful on medium length tasks (b). Grounding (d, e), and exploration (f), and integrating world knowledge (d) remain challenging.

interaction with UI elements. In Tab. 5, we show that all our baselines struggle with visual understanding and grounding of UI Elements. Tasks that require text-only understanding achieve 80% completion (#3), but this drops to 7.7% when visual understanding is required. We hypothesize this is due to fine-grained UI details being hard to capture with set-of-marks and the deep knowledge required to understand uncommon graphical UI elements [66]. We show examples in Fig. 4(d), e.g., where an agent cannot distinguish between the "New Folder" and "New Note" icon.

# Model	T (24)	T+Y (46)	T+S (10)	T+Y+S (28)
1 ReACT	41.7%	28.3%	40%	10.7%
2 ReACT + Text	45.8%	36.9%	30%	17.9%
3 Reflexion + Text	54.2%	56.5%	40%	14.3%
4 Reflexion + UGround	54.2%	41.3%	40%	14.3%
5 UI-Tars	20.8%	15.2%	0%	3.6%

Table 6. Performance by action complexity (**T**ap, T**Y**pe, **S**wipe)

Agents fail to complete tasks even with simple actions (TAP only). Performance drops sharply with more complex actions. We find agents perform relatively poorly (54.2%) even when tasks only require tap interactions (Tab. 6). Performance drops drastically when the task requires navigating views using swipe actions (Fig. 4(e)). We noticed that swipe actions are especially tricky—agents often overshoot target elements and get stuck in a loop of swiping up or down when the target is not in view.

Despite access to world knowledge, agents fail to utilize it to help solve tasks. Despite our baselines using GPT-4o, which has access to necessary world knowledge, the agents perform poorly on tasks that require it. Our best performing agent achieves 13.3% success on such tasks (Tab. 7, Col. 3). We believe that agents cannot generally emit actions conditioned on world knowledge even when apparent in the context of tasks and past observations (Fig. 4(c)).

# Model	No (65)	Env. (40)	World (15)
1 ReACT	32.3%	20%	6.7%
2 ReACT + Text	41.5%	25%	0%
3 Reflexion + Text	47.7%	35%	13.3%
4 Reflexion + UGround	40%	32.5%	13.3%
5 UI-Tars	12.3%	10%	6.6%

Table 7. Success of baselines split by knowledge required

7. Discussion and Conclusion

In this work, we build an online benchmark to measure digital agents' abilities to accomplish natural language goals by executing multi-step tasks. We create a taxonomy identifying axes of decision-making including planning and reasoning, grounding, and memory and knowledge, and use it to curate a set of diverse mobile UI tasks. Our benchmark supports device staging and cleanup for consistent evaluations across runs. We evaluate several baselines and report their performance on task completion rate and planning efficiency. We observe that while agents are good at short-horizon tasks, challenges remain in visual understanding, low-level control, exploration, and integrating world knowledge. Because our benchmark is the first of its kind for iOS, we were able to study the generalization of agents to new platforms. We found that modular methods based on commercial VLM models as planners generalize better to iOS than end-to-end methods trained on Android, Web and Desktop data. In future work, we will show results with open-weight multimodal language models fine-tuned on this task. There are also opportunities for Reinforcement Learning methods to improve task performance. Finally, our hand-crafted validators can be iterated further. We plan to make **`UINavBench`** accessible to the research community to encourage progress in these areas.

Acknowledgement

We thank Zhe Gan, Keen You, Haotian Zhang for their valuable guidance, feedback and discussions.

References

[1] Scott McGregor. Prescient agents: A Radar O'Reilly for your desktop. *The X Resource*, 1991. 1

[2] Pattie Maes. Agents that reduce work and information overload. 1994.

[3] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 1995.

[4] Michael Wooldridge and Nicholas R. Jennings. Agent theories, architectures, and languages: A survey. In *ECAI Workshop on Agent Theories, Architectures, and Languages*, 1995. 1

[5] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *ICML*, 2024. 1

[6] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *ArXiv*, abs/2307.12856, 2023.

[7] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents. *ArXiv*, abs/2312.08914, 2023.

[8] China. Xiaoyan Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *ArXiv*, abs/2312.13771, 2023.

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

[10] Junyang Wang, Haiyang Xu, Jiabo Ye, Mingshi Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *ArXiv*, abs/2401.16158, 2024.

[11] Paloma Sodhi, S. R. K. Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked llm policies for web actions. 2023.

[12] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *ACM MobiCom*, 2024.

[13] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[14] Peter C Humphreys, David Raposo, Toby Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Alex Goldin, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers, 2022. 1, 2

[15] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[16] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *ArXiv*, preprint.

[17] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. 2

[18] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ArXiv*, abs/2401.13649, 2024. 1

[19] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023. 1, 2, 3, 7

[20] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *ArXiv*, abs/2403.02713, 2024. 1

[21] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024. 2, 7

[22] Wei Li, Will Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents. *ArXiv*, abs/2406.03679, 2024. 1

[23] Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. Llamatouch: A faithful and scalable testbed for mobile ui task automation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. 1, 2

[24] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, Will Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. *ArXiv*, abs/2405.14573, 2024. 1

[25] Danyang Zhang, Hongshen Xu, Zihan Zhao, Lu Chen, Ruisheng Cao, and Kai Yu. Mobile-env: an evaluation platform and benchmark for llm-gui interaction. *arXiv preprint arXiv:2305.08144*, 2023. 1

[26] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. *ArXiv*, abs/2402.07456, 2024. 1

[27] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

[28] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Fender C. Bucker, Lawrence Jang, and Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale. *ArXiv*, abs/2409.08264, 2024. 1

[29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. 2

[30] Wei Li, Wiliam Bishop, Christopher Rawles, Folawiyo Campbell-Ajala, and Divya Tyamagundlu. On the effects of data scale on computer control agents. *ArXiv*, abs/2406.03679, 2024. 2, 7

[31] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.

[32] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *ArXiv*, abs/2407.17490, 2024. 2, 7

[33] OpenAI. Gpt-4 technical report, 2024. 2, 6

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. 2, 6

[35] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 6

[36] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In *European Conference on Computer Vision*, pages 312–328. Springer, 2022. 2

[37] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024. 2, 3

[38] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. GUICourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024. 2

[39] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *arXiv preprint arXiv:2406.10819*, 2024. 2

[40] Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*, 2024.

[41] Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, Anjie Yang, Zhaoxuan Jin, Jianbo Deng, Philip Torr, Bernard Ghanem, and Guohao Li. Crab: Cross-environment agent benchmark for multimodal language model agents, 2025.

[42] Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. AgentStudio: A toolkit for building general virtual agents. In *The Thirteenth International Conference on Learning Representations*, 2024. 2

[43] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale, 2024. *URL https://arxiv. org/abs/2409.08264*, 2024. 2

[44] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

[45] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2024. 2

[46] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. 2

[47] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023. 3, 6

[48] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. GPT-4V in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

[49] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024. 3, 6, 7

[50] Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024. 3

[51] Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.

[52] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024.

[53] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi

Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. OS-ATLAS: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024. 3

[54] Maryam Taeb, Amanda Swearngin, Eldon Schoop, Ruijia Cheng, Yue Jiang, and Jeffrey Nichols. Axnav: Replaying accessibility tests from natural language. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2024. 3, 5, 6

[55] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024.

[56] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. OS-Copilot: towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*, 2024.

[57] Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024. 3

[58] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*, 2023. 3

[59] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[60] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023.

[61] Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, et al. Mobilevlm v2: Faster and stronger baseline for vision language model. *arXiv preprint arXiv:2402.03766*, 2024. 3

[62] R.S. Weiss. *Learning From Strangers: The Art and Method of Qualitative Interview Studies*. Free Press, 1995. 4

[63] Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, 2009. 4

[64] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 845–854, New York, NY, USA, 2017. Association for Computing Machinery. 4

[65] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021. 4, 5

[66] Jieshan Chen, Amanda Swearngin, Jason Wu, Titus Barik, Jeffrey Nichols, and Xiaoyi Zhang. Towards complete icon labeling in mobile applications. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. 4, 5, 8

[67] Zhuohao Jerry Zhang, Eldon Schoop, Jeffrey Nichols, Anuj Mahajan, and Amanda Swearngin. From interaction to impact: Towards safer ai agents through understanding and evaluating ui operation impacts, 2024. 4

[68] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024. 5

[69] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. 6

[70] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. 6

[71] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. 6, 7

[72] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, abs/2409.12191, 2024. 6

[73] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9338–9346, 2019. 7

[74] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use, 2025. Preprint. 7