

Dynamics-Aware Navigation for Autonomous UI Agents on Directional-Input Devices

Ishitwa Viranchi, Nagesh Kashyap Akshintala and Mrunal Shah

Amazon

{ivviranc, nageshk, smrunal}@amazon.com

Abstract

Autonomous GUI agents have achieved near-perfect task completion on touchscreen benchmarks by tapping directly at target coordinates. However, smart televisions, game consoles, and set-top boxes require users to navigate with a directional pad (D-PAD), turning element selection into a planning problem over an unknown focus-transition graph. We study autonomous task completion on streaming TV devices, where an agent navigates 68 streaming applications using only D-PAD inputs to perform four tasks: login, reaching subscription plans, content search, and settings navigation. We formalize this setting as a partially observable Markov decision process and compare two architectures using three open-weight vision-language models (Google Gemma 3 at 4B and 27B parameters, and Alibaba Qwen3-VL at 235B): (i) a VLM that maps screenshots directly to D-PAD actions, following the paradigm that dominates current benchmarks, and (ii) a system that maintains an explicit model of focus transition dynamics via Bayesian belief updates and graph-based path planning. The dynamics-aware system improves login success for every model tested, by a mean of 8.8 percentage points, and for the Gemma models its relative advantage widens on tasks requiring extended navigation. The benefit is inverse to model scale, ranging from 16.2 points for the 4B model to 2.9 points for the 235B model. A 4B model with dynamics awareness (32.4%) outperforms a 27B model without it (26.5%), indicating that transition knowledge can outweigh a $6.75\times$ increase in model parameters.

1 Introduction

When a person picks up a television remote and encounters an unfamiliar streaming application, they adapt within seconds. They press the down arrow, observe the highlight moving to the element below, and immediately form a prediction: pressing down

again will likely continue in the same direction. If the prediction is violated (the focus jumps to a sidebar, or wraps around to the top), they update their mental model. Within a minute, they have internalized enough of the interface’s *transition dynamics* to navigate purposefully.

What exactly is this person learning? Not what a login button looks like. Not the goal of their task. They are learning how this specific application responds to directional inputs: which elements connect to which, and where focus travels for each button press.

The dominant paradigm for GUI agents treats this as a visual question-answering problem: given a screenshot, output the action that leads toward the goal. On touchscreen devices this paradigm works well, because reaching any visible element is a single tap at its coordinates. Recent systems achieve 94–100% on AndroidWorld (Rawles et al., 2025): askui’s VisionAgent reaches 94.8% with a single model and zero fine-tuning (askui, 2025), and Minitap achieves 100% through multi-agent decomposition (Favreau et al., 2026). On the desktop, OSWorld (Xie et al., 2024) has seen similarly rapid progress, with leading systems now exceeding the reported human baseline. The challenge on these platforms is largely perceptual: identifying *which* element to interact with.

Television interfaces are fundamentally different. A typical directional-input remote has seven buttons: four directional arrows, select, back, and home. Reaching a visible element requires traversing an unknown graph of focus transitions through a sequence of directional presses. The agent can see the target element on screen, but seeing it is insufficient. It must predict or discover the sequence of inputs that moves focus from its current position to the target. This transforms the problem from classification into sequential planning under uncertainty, and planning requires a model of the environment’s dynamics.

This problem arises concretely in streaming application certification, where applications must pass functional tests before publication to users. Manual testing with physical remotes is expensive and slow; application-specific scripts break whenever an interface changes. Automating certification requires an agent that navigates *arbitrary* applications it has never seen before, with unknown layouts, unknown focus behavior, and unknown login flows.

Contributions. (1) We formalize D-PAD navigation as a POMDP with three layers of partial observability and show that touchscreen agent architectures fail in this setting (§3). (2) We describe a deployed system combining VLM reasoning with explicit dynamics modeling: an AOSP-derived transition prior, Beta-Binomial belief updates, and A* path planning (§4). (3) We compare this system against a vision-only baseline using three open-weight VLMs from two model families across 68 applications and four tasks on physical devices (§5). (4) We show that dynamics knowledge provides consistent improvements regardless of model family or scale, and that a 4B model with dynamics outperforms a 27B model without.

2 Related Work

The study of autonomous GUI agents sits at the intersection of several research threads that have evolved substantially over the past decade.

From reinforcement learning to language model agents. Early work on GUI automation framed the problem as reinforcement learning. AndroidEnv (Toyama et al., 2021) provided an RL environment for Android, and Humphreys et al. (2022) demonstrated data-driven computer control using behavioral cloning from human demonstrations. These approaches required extensive interaction data and struggled to generalize across applications. The advent of large language models shifted the paradigm: rather than learning policies from scratch, agents leverage the broad world knowledge of pretrained models. SeeAct (Zheng et al., 2024) demonstrated that GPT-4V could navigate real websites in a zero-shot setting when properly grounded, and Mind2Web (Deng et al., 2023) established benchmarks for web navigation with cached environments. This transition from learned policies to prompted reasoning is now the dominant approach, with top systems relying on frontier VLMs

with no task-specific training.

Benchmarks and their saturation. OSWorld (Xie et al., 2024) introduced a scalable desktop environment supporting 369 tasks across Ubuntu, Windows, and macOS with execution-based evaluation. AndroidWorld (Rawles et al., 2025) provided a dynamic mobile benchmark with 116 parameterized tasks across 20 real applications, where reward signals derive from system state rather than text matching. Both have been rapidly saturated: on AndroidWorld, multiple systems exceed 97%, and Minitap achieves 100% (Favreau et al., 2026). OSWorld has seen comparably rapid progress, with successive leaderboard updates showing large jumps in reported success rates. Critically, every top-performing system uses touch or mouse interaction, where transition dynamics are trivial. Until recently, no benchmark evaluated agents on directional-input interfaces where transition dynamics are nontrivial.

Directional-input and TV agents. Concurrent work addresses this setting: TVWorld (Ma et al., 2026) builds an offline graph abstraction of TV navigation with benchmarks and a specialised model, and TVAgent (Liu et al., 2026) a lightweight vision-language model for TV interfaces. Agent+P (Ma et al., 2025) is the closest touchscreen analogue of our planner, modelling transitions as a UI Transition Graph for symbolic planning. These systems plan over a graph built offline or during training; we construct it from the current screen and correct it online, which is what allows previously unseen applications without a per-application model.

Architectural paradigms. Our survey of top AndroidWorld agents reveals two dominant paradigms. The first comprises single VLM agentic loops: askui’s VisionAgent achieves 94.8% with a single model, screenshot-only input, and zero fine-tuning (askui, 2025). The second comprises multi-agent frameworks with specialized roles: Minitap orchestrates six agents via task decomposition, with ablation studies showing that decomposition alone contributes +21 points over a single-agent baseline (Favreau et al., 2026). Trained models such as UI-TARS-2 (ByteDance Seed, 2025) use reinforcement learning with proximal policy optimization for end-to-end agent training. A notable finding is that most top agents use zero training or fine-tuning: architecture and prompt design dominate over model training for touch interfaces. None

of these systems models transition dynamics, because on touch interfaces the mapping from an action to its target is direct.

Modular design and visual grounding.

UGround (Gou et al., 2025) advocates modular GUI agent design: a monolithic model is unlikely to capture all environmental complexities, so planning should be separated from grounding. They train a universal visual grounding model on 10M web elements that generalizes across platforms. SeeClick (Cheng et al., 2024) demonstrates that grounding can be trained from synthetic data. Our architecture follows this modular philosophy, but the specialized module is temporal (predicting focus transitions) rather than spatial (mapping descriptions to coordinates).

World models for GUI navigation. The WMA Web Agent (Chae et al., 2025) trains a world model to predict next observations for web navigation and plans against it. We share that premise, and like it we consume structured observations: the AOSP focus algorithm we build on is defined over view-hierarchy geometry. What differs is the role the model plays. In a browser any visible element can be clicked directly, so predicting transitions refines an action that is already available. Under directional input an element cannot be addressed directly at all and must be reached by a traversal, which makes a transition model a precondition for acting rather than a refinement of one.

Open-weight vision-language models. We evaluate Google’s Gemma 3 (Gemma Team, 2025), providing efficient multimodal models from 4B to 27B parameters designed to run on consumer hardware, and Alibaba’s Qwen3-VL (Qwen Team, 2025), offering strong spatial grounding and GUI understanding at 235B parameters. Using open-weight models enables full reproducibility of our experimental results.

3 Problem Formulation

3.1 D-PAD Navigation as a POMDP

We model the problem as a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \gamma \rangle$. The state $\mathcal{S}$ captures the full application configuration (screen, focused element, internal state). The action space consists of D-PAD directions plus text input:

$$\mathcal{A} = \{\text{UP, DOWN, LEFT, RIGHT, CENTER, BACK, TEXT}(v)\}$$

Observations combine a screenshot $\mathcal{O}_v$ and an accessibility tree $\mathcal{O}_s$ (possibly empty). The transition function T is unknown, reward is sparse (1 on task completion), and $\gamma = 1$.

3.2 Three Layers of Partial Observability

Sensory. Screenshots show everything but encode no structure. Accessibility trees provide metadata (element types, bounds, focus state) but are incomplete for WebView or Canvas applications, which in the limiting case expose a single opaque container node with no focusable children.

Transitional. Android defines a default Focus-Finder algorithm scoring candidates by:

$$\text{score}(e_s, e_c, d) = 13 \cdot \Delta_{\text{major}}^2 + \Delta_{\text{minor}}^2 \quad (1)$$

where Δ_{major} is edge-to-edge distance along the navigation direction and Δ_{minor} is perpendicular center distance. The $13\times$ weight favors elements directly ahead, with beam-overlap providing priority. This prior is correct for $\sim 80\%$ of transitions but apps routinely override it with custom focus handling.

Task-structural. The instruction names a terminal condition (“log in”) rather than a procedure. It does not enumerate the intermediate screens, specify which field to complete on each, or indicate how many steps remain, so the agent must infer this latent subgoal structure online and revise it as new screens appear. This layer is not specific to directional input. High-level instructions that must be decomposed into multi-screen action sequences are the standard formulation in web and desktop GUI agent benchmarks (Deng et al., 2023; Zheng et al., 2024; Xie et al., 2024), and hierarchical planning failures are a recognised source of long-horizon agent error. We list it not as a novel observation but because it compounds with the two layers above: an agent that misjudges which subgoal it is pursuing will also misjudge which element to move toward, and under directional input that error costs an entire multi-step traversal rather than one misplaced tap.

3.3 Why Touchscreen Solutions Fail

On touchscreens, reaching element e_j is one action: $\text{TAP}(x_j, y_j)$. On D-PAD interfaces, it requires solving a shortest-path problem on the unknown focus graph $G = (V, E)$. The agent must either explore to discover G , predict it from a prior, or navigate without any model.

4 System Design

We compare two systems using the same VLM for task reasoning but differing in within-screen navigation.

4.1 Vision-Only Baseline

A VLM in an agentic loop: capture screenshot, send to VLM with task description and conversation history (last 10 pairs), receive a single D-PAD action, execute. Limited to 40 steps per attempt. Representative of systems achieving 94–100% on AndroidWorld.

4.2 Dynamics-Aware System

Our proposed system separates high-level task reasoning from low-level navigation planning. The VLM receives structured element representations (identifier, type, text content, bounding box, and focus state) rather than raw screenshots, and selects a target element along with an action type such as focus or text entry. This structured input reduces token consumption by approximately $3\times$ compared to screenshot-based conversations.

Once a target element is selected, the POMDP engine plans and executes the D-PAD sequence to reach it (Algorithm 1). The engine maintains a directed graph where each edge carries a Beta-Binomial belief distribution over the probability that pressing a given direction from one element moves focus to another. Edges are initialized from the AOSP FocusFinder prior ($\alpha=2, \beta=1$ for predicted transitions, $\alpha=1, \beta=2$ otherwise), giving a starting probability of 0.67 for predicted edges. The engine then finds the shortest path via A* search with edge costs equal to $1/P_{\text{edge}}$. This cost is a heuristic ordering rather than an expected number of attempts: an unexpected transition moves focus elsewhere instead of leaving the agent able to retry the same edge, so $1/P_{\text{edge}}$ should be read as penalising low-confidence edges, not as a calibrated retry count. After each executed action, the system observes which element actually received focus and updates the relevant belief parameters: if pressing direction d from e_i lands on the predicted e_j , increment α_{ij}^d ; otherwise increment β_{ij}^d and increment α_{ik}^d for the transition $e_i \xrightarrow{d} e_k$ that was actually observed. This conjugate update converges quickly, typically requiring only 2–3 observations to correct

Algorithm 1 POMDP Navigation Engine

Require: Elements $\{e_1, \dots, e_n\}$, focused e_f , target e^*

- 1: Build graph G using AOSP FocusFinder (Eq. 1)
- 2: Init beliefs: $\alpha=2, \beta=1$ (predicted); $\alpha=1, \beta=2$ (others)
- 3: $\text{path} \leftarrow A^*(G, e_f, e^*, \text{cost}=1/P_{\text{edge}})$
- 4: **if** path found and $\min P_{\text{edge}} \geq \tau$ **then**
- 5: Execute path; update beliefs from each observation
- 6: **else**
- 7: Execute least-attempted direction; update beliefs
- 8: **end if**
- 9: **return** current focus element

an incorrect prior.

$$P(e_i \xrightarrow{d} e_j \mid \text{obs}) = \frac{\alpha_{ij}^d}{\alpha_{ij}^d + \beta_{ij}^d} \quad (2)$$

When path confidence falls below threshold $\tau = 0.27$, the engine switches to exploration mode, executing the least-attempted direction to gather new observations. When the accessibility tree is degenerate (fewer than 2 focusable elements), the system falls back to vision-only mode identical to the baseline.

5 Experiments

5.1 Setup

We evaluate on 68 streaming applications on physical streaming TV devices with confirmed email-and-password login paths and usable test credentials, selected from 136 candidates. Exclusions include applications offering device pairing only (10), OTP authentication (5), geo-restricted content (3), infrastructure failures (6), and those for which valid test credentials could not be provisioned. We test four tasks of increasing navigational complexity: login (5–8 steps), plans (login followed by 4–8 additional steps to reach the subscription plans page), search (login followed by 15–25 steps including on-screen keyboard entry), and settings (login followed by 8–12 steps through sidebar menu navigation).

We evaluate three open-weight multimodal VLMs: Qwen3-VL at 235B parameters (Qwen Team, 2025), and Gemma 3 at both 27B and 4B parameters (Gemma Team, 2025). All models are

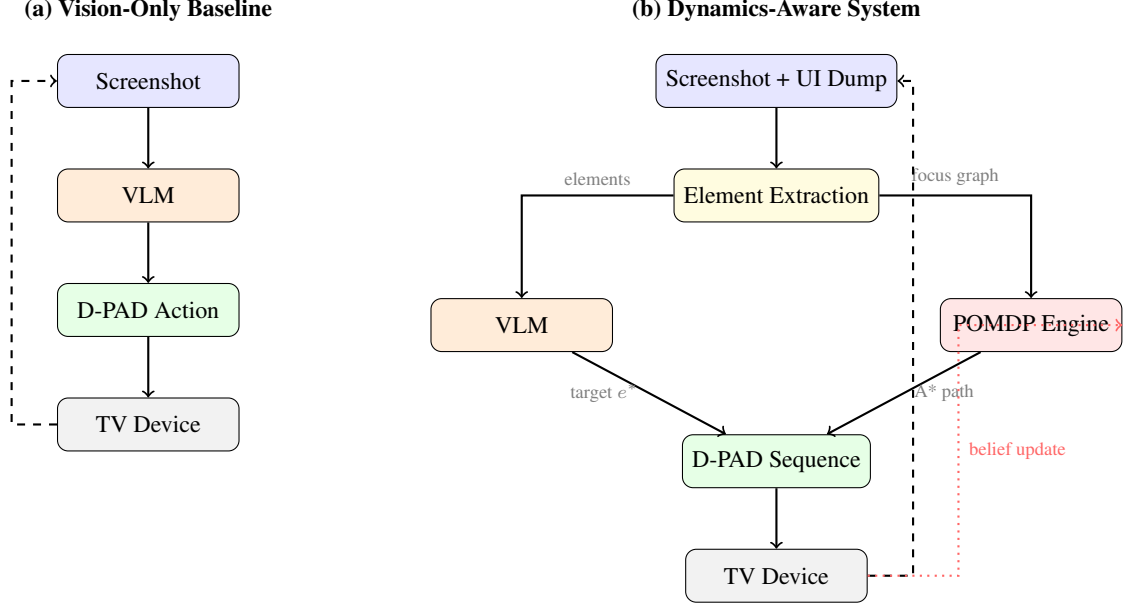


Figure 1: (a) The vision-only baseline sends screenshots to a VLM, which outputs one D-PAD action per step. (b) The dynamics-aware system extracts structured elements from the accessibility tree, uses a VLM to select the target element, and employs a POMDP engine with a learned focus graph to plan the optimal D-PAD sequence. Belief updates from observed transitions (dotted red) continuously refine the dynamics model.

accessed through a managed inference API that provides a uniform request interface across model families. Experiments run on 9 physical streaming TV devices distributed across 3 host machines. The vision-only baseline is allowed 40 steps per attempt; the dynamics-aware system receives 15 reasoning calls of approximately 3 actions each.

Success is scored from the recorded trajectory rather than from in-process signals: every run persists a screenshot, a UI hierarchy dump, and the model input and output at each step. For login the trajectory must show credentials being entered, since several applications render browsable content without a session, so a post-login-looking screen alone is not evidence of authentication. We report pass@1 task success rate across a total of 1,632 test runs (3 models $\times$ 2 systems $\times$ 68 applications $\times$ 4 tasks).

5.2 Login Results

Table 1 shows that dynamics awareness improves login success for all three models, though the magnitude varies more than fivefold. The largest gap (+16.2 pp) occurs with Gemma 3 4B and the smallest (+2.9 pp) with Qwen3-VL 235B. This ordering is inverse to parameter count: the benefit of an explicit dynamics model shrinks as the base model grows. We read the small Qwen3-VL gain as evidence that large-scale spatial grounding training

Table 1: Login pass@1 on 68 streaming applications.

Model / System	Pass@1	Δ (pp)
Qwen3-VL 235B / POMDP	38.2%	+2.9
Qwen3-VL 235B / Vision	35.3%	
Gemma 3 27B / POMDP	33.8%	+7.4
Gemma 3 27B / Vision	26.5%	
Gemma 3 4B / POMDP	32.4%	+16.2
Gemma 3 4B / Vision	16.2%	

partially, though not wholly, internalizes the transition patterns that smaller models lack.

We hypothesize that Qwen3-VL’s GUI-focused pretraining (Qwen Team, 2025), which included large-scale element localization and spatial reasoning tasks, allows it to implicitly predict where focus will move based on element layout alone. In effect, the model has learned an approximate dynamics prior from its training data, reducing the marginal benefit of our explicit AOSP-derived prior. However, this capability has not transferred to smaller models (Gemma 3 at 4B and 27B), suggesting it requires both scale and targeted training data.

5.3 Multi-Task Results

The dynamics gap widens with task complexity for the Gemma models (Table 2); for Qwen3-VL

Table 2: Pass@1 (%) across four tasks for the model with the largest dynamics gap (Gemma 3 4B) and the largest model (Qwen3-VL 235B).

Task	Gemma 3 4B		Qwen3-VL 235B	
	POMDP	Vision	POMDP	Vision
Login	32.4	16.2	38.2	35.3
Plans	26.5	10.3	23.5	22.1
Search	22.1	8.8	19.1	20.6
Settings	19.1	7.4	16.2	14.7

235B it remains small and reverses on search. For Gemma 3 4B, the relative advantage grows from $2.0\times$ on login to $2.6\times$ on settings. Both harder tasks place more focus transitions between the initial state and the goal: Search involves an on-screen keyboard, and Settings requires traversing 4–7 screens with ambiguous labels. Longer traversals give an incorrect transition prior more opportunities to send the agent off path.

5.4 Dynamics Knowledge vs. Model Scale

Gemma 3 4B with dynamics (32.4%) outperforms Gemma 3 27B without it (26.5%) despite a $6.75\times$ parameter gap, and this holds across all four tasks. The comparison should not be overstated. Within the dynamics-aware system, login results order with scale (38.2%, 33.8%, and 32.4% for 235B, 27B, and 4B respectively), though on the other three tasks the 4B counts exceed the 27B counts (Table 3), so single-run differences of a few applications can invert adjacent rankings. A transition model does not substitute for capacity outright. What it does is close most of the distance: supplying a 4B model with dynamics recovers more ground than a $6.75\times$ parameter increase does without one.

6 Discussion and Conclusion

We presented a controlled study of autonomous navigation on directional-input devices, comparing vision-only and dynamics-aware architectures using three open-weight VLMs across 68 applications and four tasks on physical hardware.

Central finding. Transition dynamics knowledge, largely independently of visual understanding, is a primary bottleneck for cross-application D-PAD navigation. A 4B model with an explicit dynamics model outperforms a 27B model relying on vision alone, and every model we tested

improved when given one. The benefit is largest for the smallest model and smallest for the largest, which suggests that scale partially substitutes for an explicit transition model without replacing it. The relative gap widens with task complexity, confirming that structured environment models grow in value with navigational depth.

What the comparison isolates. The two systems differ in representation as well as planning, necessarily so: a transition model needs stable element identities as graph vertices, which a screenshot-to-keypress loop does not maintain. The comparison is therefore architectural, and we cannot attribute the gap to transition modelling alone. Three observations still point at dynamics: the baseline’s prompt already encodes geometric navigation toward an identified target; its characteristic failures are dynamical rather than perceptual, pressing the spatially intuitive direction and having focus move elsewhere; and for the smaller models the gap grows with the number of transitions to the goal, which a representation advantage would not predict. Isolating the components requires an intermediate condition, structured input with geometric rather than belief-based movement, which we did not run.

Industry applications. Automated navigation on constrained-input devices has broad applicability beyond application certification. Accessibility testing tools could verify that applications are navigable by users who rely on switch controls or keyboard-only interaction. Quality assurance pipelines for game consoles, automotive infotainment systems, and medical device interfaces face the same challenge of validating navigation flows across diverse UIs without application-specific scripts. The modular architecture (general-purpose VLM plus domain-specific dynamics model) provides a template for these applications: the VLM handles task understanding while the dynamics model adapts to each new interface through online learning.

Future directions. A complementary direction to explicit dynamics modeling is learning a transferable dynamics prior from interaction data, replacing the hand-crafted AOSP FocusFinder with a learned model that generalizes across applications. Our results with Qwen3-VL suggest that sufficiently capable VLMs may internalize some transition patterns through training, though this has

not emerged at the 27B scale.

Limitations

Our transition prior (AOSP FocusFinder) is correct for approximately 80% of focus transitions. The remaining 20% require online Bayesian adaptation that operates within a single session and does not transfer across applications. Applications producing degenerate accessibility trees (WebView, Canvas, Jetpack Compose) force fallback to vision-only mode, eliminating the dynamics advantage. Our evaluation uses a single device family of streaming TV devices operated by a directional-input remote; results may differ on other form factors. The 68-app test set represents a convenience sample of applications with valid test credentials rather than a statistically representative sample.

Ethics Statement

All experiments were conducted on physical devices in a controlled laboratory environment using dedicated test accounts created for this research. No personal or customer data was accessed, collected, or processed at any point. Login credentials used in the experiments belong to test accounts provisioned specifically for automated certification testing and are not associated with real users.

The intended application of this work is automated quality assurance: verifying that applications function correctly before release. We note a dual-use consideration: automated navigation of application interfaces could in principle be applied to circumvent usage controls or automate interactions against a service operator’s terms of use. We regard automated testing as the intended use, and we do not release the credentials, application-specific configurations, or device provisioning required to reproduce the deployment at scale.

Our focus-transition implementation is derived from the Android Open Source Project, used under the Apache License 2.0. We evaluate only open-weight models, used in accordance with their respective licenses. We report results for every model and configuration evaluated, including those for which the improvement was marginal.

References

askui. 2025. askui vision agent achieves 2nd place on AndroidWorld benchmark. [https://www.askui.com/blog-posts/askui-vision-agent](https://www.askui.com/blog-posts/askui-vision-agent-achieves-2nd-place-on-androidworld-benchmark)

[t-achieves-2nd-place-on-androidworld-benchmark](https://arxiv.org/abs/2509.02544).

ByteDance Seed. 2025. UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*.

Hyunjoo Chae, Namyoun Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2025. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations (ICLR)*. ArXiv:2410.13232.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. ArXiv:2401.10935.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*.

Pierre-Louis Favreau, Jean-Pierre Lo, Clement Guiguet, Charles Simon-Meunier, Nicolas Dehandschoewercker, Allen G. Roush, Judah Goldfeder, and Ravid Shwartz-Ziv. 2026. Do multi-agents dream of electric screens? achieving perfect accuracy on AndroidWorld through task decomposition. *arXiv preprint arXiv:2602.07787*.

Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *International Conference on Learning Representations (ICLR)*. ArXiv:2410.05243.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhapparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. 2022. A data-driven approach for learning to control computers. In *International Conference on Machine Learning (ICML)*.

Jinyang Liu et al. 2026. TVAgent: A lightweight vision-language model for TV GUI agent. OpenReview preprint.

Shang Ma, Xusheng Xiao, and Yanfang Ye. 2025. Agent+P: Guiding UI agents via symbolic planning. *arXiv preprint arXiv:2510.06042*.

Zhantao Ma, Quanfeng Lu, Shuai Zhong, Dahai Yu, Ping Luo, and Michael K. Ng. 2026. TVWorld: Foundations for remote-control TV agents. *arXiv preprint arXiv:2601.13142*.

Qwen Team. 2025. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations (ICLR)*. ArXiv:2405.14573.

Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. 2021. AndroidEnv: A reinforcement learning platform for Android. *arXiv preprint arXiv:2105.13231*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a generalist web agent, if grounded. In *International Conference on Machine Learning (ICML)*. ArXiv:2401.01614.

A AOSP FocusFinder Algorithm Detail

Our implementation is a direct port of Android’s `FocusFinder.java` from `frameworks/base/core/java/android/view/`. Given source element e_s and direction d , the algorithm proceeds as follows: (1) filter candidates to elements at least partially in direction d ; (2) compute beam overlap, which tests whether the candidate’s perpendicular projection overlaps with the source; (3) score candidates using Eq. 1; (4) select the lowest-scoring candidate, with beam-overlap candidates always preferred regardless of distance score. The $13\times$ major-axis weight is the exact constant from the Android source code.

For DOWN: a candidate e_c is valid if $e_s.\text{bottom} < e_c.\text{bottom}$ and ($e_s.\text{top} < e_c.\text{top}$ or $e_s.\text{bottom} \leq e_c.\text{top}$). Beam overlap holds when $e_c.\text{right} > e_s.\text{left}$ and $e_c.\text{left} < e_s.\text{right}$. Analogous conditions apply to other directions with appropriate axis transposition.

For RIGHT: a candidate is valid if $e_s.\text{right} < e_c.\text{right}$ and ($e_s.\text{left} < e_c.\text{left}$ or $e_s.\text{right} \leq e_c.\text{left}$). Beam overlap requires $e_c.\text{bottom} > e_s.\text{top}$ and $e_c.\text{top} < e_s.\text{bottom}$. The major axis distance is $e_c.\text{left} - e_s.\text{right}$ and the minor axis distance is the absolute center-to-center offset on the perpendicular axis.

B On Modular Design for GUI Agents

Our architectural choice to separate task reasoning from dynamics modeling follows the argument by Gou et al. (2025): a monolithic model, regardless of future scale, is unlikely to fully encapsulate the diverse complexities across all environments. The digital world comprises billions of applications, each with dynamic states and idiosyncratic semantics (uncommon icons, custom focus behavior, counter-intuitive navigation patterns). A generalist agent therefore requires a modular system, orchestrating a foundation model with specialized modules.

In UGround, the specialized module is a spatial grounder mapping text descriptions to pixel coordinates. In our framework, it is a temporal dynamics model predicting focus transitions given state and action. Both serve the same architectural purpose: providing domain-specific knowledge to a general-purpose reasoning model, enabling effective action without requiring the foundation model to internalize every environmental detail.

The empirical evidence from both our work and the broader GUI agent landscape supports this philosophy. On AndroidWorld, the top-performing systems are modular frameworks rather than monolithic fine-tuned models. On our D-PAD benchmark, the modular dynamics-aware system outperforms the monolithic vision-only approach across model scales. The question of *what* to modularize depends on the bottleneck: for touchscreen interfaces, it is spatial grounding; for constrained-input interfaces, it is transition dynamics.

C Full Multi-Task Results

Table 3: Pass@1 success counts for all model, system, and task configurations. Each cell is out of $n = 68$ applications; the 24 cells together comprise all 1,632 runs, of which 332 succeeded (20.3% overall).

Model	System	Login	Plans	Search	Settings
Qwen3-VL 235B	POMDP	26	16	13	11
Qwen3-VL 235B	Vision	24	15	14	10
Gemma 3 27B	POMDP	23	15	12	10
Gemma 3 27B	Vision	18	11	9	8
Gemma 3 4B	POMDP	22	18	15	13
Gemma 3 4B	Vision	11	7	6	5