From Task Solving to Robust Real-World Adaptation in LLM Agents

Anonymous ACL submission

Abstract

Large language models are increasingly deployed as specialized *agents* that plan, call tools, and act over extended horizons. Yet many evaluations assume a "clean interface" with stable dynamics, reliable tools and sensors, and a single explicit objective—often overstating real-world readiness. In practice, agents face underspecified rules, unreliable signals, shifting environments, and implicit, multi-stakeholder goals. The challenge is therefore not just *solving* tasks, but *adapting* while solving: deciding what to trust, what is wanted, when to verify, and when to fall back or escalate. We stress-test deployment-relevant robustness under four operational circumstances: *partial observability*, *dynamic environments*, *noisy signals*, and *dynamic agent state*. We benchmark agentic LLMs in a grid game with a simple goal but long-horizon execution. Episodes violate clean-interface assumptions yet remain solvable, requiring agents to infer rules, pay for information, adapt to shifts, and act cautiously under noise. Across five state-of-the-art LLM agents, we find large gaps between nominal task-solving and deployment-like robustness. Performance degrades with grid size and horizon, but rankings are unstable: weaker models can beat stronger ones when strategy matches the uncertainty regime. Despite no explicit instruction, agents trade off completion, efficiency, and penalty avoidance, suggesting partial objective inference. Ablations and feature analyses expose model-specific sensitivities and failure drivers, motivating work on verification, safe action selection, and objective inference under partial observability, noise, and non-stationarity.

1 Introduction

Large language models (LLMs) are rapidly evolving from passive task solvers into *agentic* systems: models that plan, call tools, and take actions over extended horizons in complex environments (Comanici et al., 2025; Singh et al., 2025; Luo et al., 2025; Dong et al., 2025). As LLM capabilities have improved, a growing number of specialized agents have emerged, reporting strong performance on narrowly defined tasks (Zhang et al., 2024; Novikov et al., 2025; Hong et al., 2025). However, most evaluations still rely on two simplifying assumptions: (i) the environment is sufficiently specified and stable, and (ii) the objective is explicit and reducible to a clean success signal. In real-world deployment, these assumptions often break down, raising a central question: are today's LLM agents truly ready for practical, in-the-wild use?

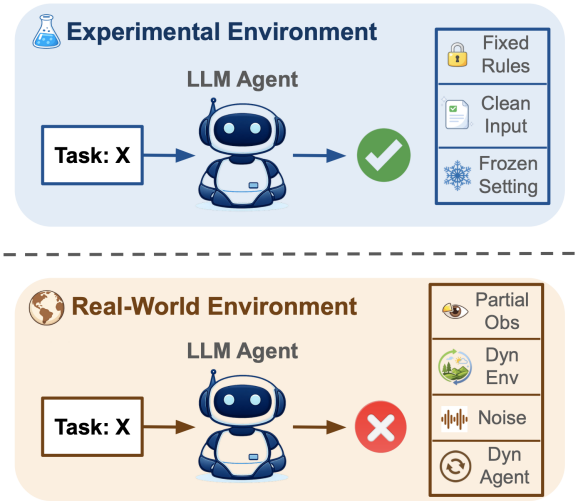


Figure 1: **Experimental vs. real-world deployment for LLM agents**. Agents often succeed in controlled experimental settings (fixed rules, clean inputs, frozen environment) but degrade in the real world due to partial observability, dynamic environments, noisy inputs, and changing agent behavior.

Our goal is not to test task solving in idealized settings—specialized agents will soon perform well on many tasks. Instead, we ask: *while solving, how robustly can an agent infer, adapt to, and safely interact with a realistic world?* We argue that real-world LLM agents will routinely face four under-emphasized circumstances (Figure 1): First,

latent policies under partial observability and underspecified goals: even a simple request (e.g., "bring supplies to hospital Room 312") may require inferring unspoken constraints like permissions, quiet-hour norms, and speed–disruption trade-offs. Second, **dynamic environments and distribution shifts**: weather, detours, or new obstacles can invalidate plans and require replanning and safe fallback. Third, **noisy signals**: unreliable sensing/actuation (e.g., dropped frames, low traction, wear) calls for targeted verification when errors are costly. Fourth, **non-stationary agent state**: overrides or model/-tool updates can change behavior mid-task, requiring drift detection and re-checking assumptions.

To assess real-world readiness, we introduce a grid-based puzzle with a simple objective but long-horizon execution that intentionally violates clean-interface assumptions. The benchmark targets robustness during execution: inferring hidden rules, paying for information, adapting to environmental shifts and agent-state drift, and acting under noisy partial observations. The game is an $N \times N$ grid with a clear goal—collect three key fragments (K) and exit through a door (D)—where the agent observes only a local window that may be corrupted by noise. The grid also contains undisclosed rule tiles (R): interacting with R triggers a latent transformation that may yield a key, a hazard, or nothing, depending on hidden context (e.g., current energy). This forces the agent to treat interaction as experimentation rather than executing a fixed plan.

The game operationalizes deployment stressors as controlled perturbations. The environment is non-stationary: hazards may spread, dynamics can change, and teleportation can occur mid-episode. The agent's own capabilities can also drift (e.g., movement becomes less reliable or sensing becomes more expensive), requiring re-calibration. Finally, information acquisition is explicit and costly: SCAN temporarily expands the local window of the grid visible by the agent, MEASURE collapses nearby latent cells ($\circ$) into known tiles, and INTERACT engages with the tile at the agent's current location, triggering various outcome. Together, these mechanics form a compact testbed where success depends not just on reaching the exit, but on doing so robustly under partial observability, shifting dynamics, internal drift, and noisy signals.

Evaluating five state-of-the-art LLMs, we observe a clear gap between nominal task-solving and deployment-like robustness. Performance typically degrades with larger grids and longer horizons, yet rankings are unstable: different models excel under different regimes, and in some settings a "weaker" model can outperform a stronger one when its strategy better matches the prevailing uncertainty. Beyond success rates, agents exhibit distinct trade-offs between completion, efficiency, and penalty avoidance, even though we never explicitly instructed them to do so, suggesting partial objective inference in some cases. Behavioral traces and single-stressor ablations reveal concrete failure drivers: some agents overcommit to early hypotheses about hidden mechanics, underinvest (or overinvest) in costly information gathering, and fail to recalibrate after shifts, while others adopt more verification-heavy strategies that improve efficiency but may reduce completion. Together, these results show that robustness in realistic, shifting environments depends as much on *adaptive strategy selection* as on raw task-solving capability.

2 Grid Game

In this section, we introduce our interactive game setting, where the environment is underspecified and only partially observed, and where effective behavior requires inferring both what is true and what is wanted. We first describe the game's core structure, then detail the modifiers that operationalize the four aforementioned real-world circumstances, and finally characterize the player (the LLM agent).

2.1 Game Structure

We define an episodic, turn-based grid game on a bounded $N \times N$ world (see Figure 2). Each episode samples a random seed to generate an initial map, fixed at initialization but potentially evolving over time. The agent occupies one cell and maintains an *orientation* $\{\blacktriangle, \blacktriangledown, \blacktriangleleft, \blacktriangleright\}$. At every step, the agent receives a *local* observation: a square window of radius r centered on its position (i.e., $(2r{+}1) \times (2r{+}1)$). Episodes terminate upon exiting through the door or exhausting a fixed step budget (e.g., 200). The default objective is simple but long-horizon: collect K key fragments (by default $K{=}3$) and then open the exit door.

Tiles and placement. The grid uses a small vocabulary of discrete tile types. Border cells are *walls* (#); interiors are generated by sampling walls and carving corridors via short random walks. Remaining cells include *empty* traversable space ($\cdot$), *energy* tiles (e) that replenish the agent's energy

when stepped on, *key fragments* (k) that contribute to the win condition, a single *door* (D) that ends the episode upon successful exit, *hazards* (h) that incur penalties on contact, and *rule tiles* (R) whose effects are undisclosed and triggered by interaction. The generator also places a small number of *pads* (P) that act as (when enabled) teleport landing points. Finally, some cells are *Schrödinger tiles* we refer to as latent ($\circ$), representing unrevealed structure: they appear as $\circ$ (or as ambiguous ($?$) under observation noise) until the agent pays to resolve them (i.e., via MEASURE), after which they stochastically collapse into a standard tile (e.g., $\cdot, e, R, k, h$). Placement is controlled by simple density hyperparameters plus fixed counts for unique objects (one door, one pad, one energy, two rules, two key fragments), keeping the state compact while tuning difficulty via N, densities, and the latent fraction.

Actions and effects. The agent's action space is small and discrete: MOVE$_{\{N,S,E,W\}}$, INTERACT, SCAN, and MEASURE (when latent structure is enabled). Movement updates position and facing direction and can fail stochastically ("slip"), modeling actuation uncertainty. INTERACT applies to the tile directly in front of the agent: it opens the door only when the requirement is met (e.g., $\geq K$ keys), triggers hidden transformations on rule tiles (yielding outcomes such as k, h, or $\cdot$), and can attempt to neutralize hazards at additional cost with possible failure. SCAN temporarily increases the observation radius for one step at an energy cost, while MEASURE collapses nearby latent cells to reveal hidden structure, also at an energy cost. Together, these actions yield a compact but expressive environment where success requires long-horizon exploration, information acquisition, and risk-aware interaction.

2.2 Real-World Adaptations

The Grid Game intentionally violates "clean-interface" assumptions in agent evaluations. We instantiate four deployment-relevant challenges—latent policies under partial observability, dynamic environments, noisy signals, and dynamic internal agent state—with modular, parameterized mechanisms *active during execution*, enabling controlled ablations and difficulty sweeps.

(1) Latent policies, partial observability and implicit objectives. Real deployments rarely expose complete state or fully specify what is permissible; agents must infer both *what is true* and *what is wanted* from limited, sometimes ambiguous evidence. We model this by restricting observations to a local window and adding two latent structures. First, *rule tiles* (R) implement hidden, context-dependent dynamics: INTERACTing with an R tile triggers an undisclosed transformation—if energy is high, it produces a key (with probability 0.5); if energy is low and a hazard is adjacent, it becomes a hazard; otherwise, it yields an empty tile. Second, *latent cells* ($\circ$) represent unrevealed map content that is instantiated only via MEASURE. The game does not reveal the rule tables or collapse distribution, so success requires experimentation and conservative decisions under ambiguity; agents must also infer which interactions pay off long term and uncover two implicit goals: **minimizing steps** and **maximizing score**.

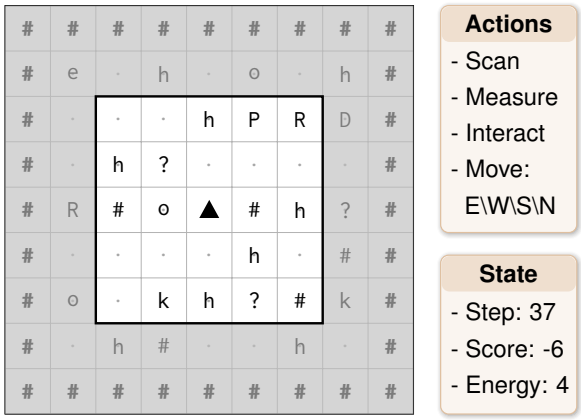


Figure 2: A 9×9 gridworld with partial observability: the agent observes only the centered 5×5 window.

(2) Dynamic environment and distribution shift. To reflect the non-stationarity of real deployments, we introduce mid-episode perturbations that make the environment explicitly dynamic. At fixed intervals, the environment *shifts*: a latent "weather" variable can change regimes, altering action reliability and resource drain, and periodic *teleport* events can relocate the player to the pad tile. In addition, hazards evolve over time: at each step, hazard tiles can spread to a neighboring cell with some probability, creating local, compounding changes in the map. These interventions break the assumption of a fixed world model, forcing agents to detect regime changes, update beliefs, and adapt policies online.

(3) Noisy signals. Physical systems couple perception and action through noise, latency, and resource limits. We incorporate this by (i) corrupting

observations with per-cell noise, producing occasional misread tiles, and (ii) introducing stochastic action failures (e.g., movement "slip"). We further model the cost of sensing and intervention through an explicit *energy budget*: actions such as SCAN, MEASURE, and INTERACT consume energy, and low energy amplifies actuation failure rates. Together, these constraints force agents to reason under uncertainty, choose verification strategically, and act risk-aware when signals are unreliable and mistakes are costly.

(4) Dynamic internal agent state. In deployed pipelines, the agent itself can change: models are updated, tools change behavior or cost, and internal calibrations drift. We simulate this via a mid-episode *drift event* after which action reliability and costs can shift (e.g., higher slip probability, increased sensing costs) even if the map appears unchanged. This forces agents to monitor outcomes and recalibrate their competence assumptions; drift creates an "I may have changed" moment that punishes brittle plans and rewards adaptive, feedback-driven control.

2.3 The Player

The *player* is an LLM agent interacting through a text-only interface. Each step returns a compact observation with no privileged access to the full map, hidden rules, or latent variables: (i) the *goal*, (ii) a *local view* with facing direction, (iii) a *state vector* (step, remaining energy, keys collected, score), and (iv) the *available actions* (i.e., $\text{MOVE}_{N,S,E,W}$, INTERACT, SCAN, MEASURE). In addition, we pass a short history of the agent's previous actions to facilitate long-horizon reasoning and enable models to recognize non-stationarity without requiring external memory beyond the prompt.

We also provide a minimal *execution log* that records salient events and outcomes. At each timestep, the environment appends a log entry (when something noteworthy occurs) that includes: (a) step metadata (timestep index, chosen action, and a local 3×3 view), (b) whether the action succeeded (e.g., "MOVE_E succeeded" vs. "MOVE failed (slip)"), (c) immediate rewards or penalties (e.g., energy gained from e, penalties from hazards), and (d) notable environment events (e.g., "ENV SHIFT,"). This event-based logging serves two purposes: it mirrors real deployments where agents must interpret telemetry and tool responses rather than observe latent state directly, and it makes both positive and negative outcomes explicit without reducing the task to a single scalar reward. We instantiate the LLM player with a standardized prompt enforcing an action-only output format; the full prompt is provided in the Appendix.

3 Experimental Details

We evaluate five state-of-the-art LLMs: GPT-5.2, GPT-5 MINI (Singh et al., 2025), GEMINI 3 PRO, GEMINI 3 FLASH (Comanici et al., 2025), and QWEN3-235B-A22B (Yang et al., 2025) (non-thinking). For models that support explicit test-time "thinking," we use their default thinking budget (medium or high). For the main evaluation, we generate 50 random game instances for each grid size in $\{6 \times 6, 8 \times 8, 10 \times 10\}$. Except for grid size, we use the same parameters across samples from different grid sizes to ensure a fair comparison. In each instance, we sample key environment parameters uniformly at random: We sample `observation noise` $\sim \mathcal{U}(0, 0.2)$, `move fail` $\sim \mathcal{U}(0, 0.1)$, and `latent fraction` $\sim \mathcal{U}(0, 0.2)$. We choose these moderate uncertainty ranges, consistent with prior work (Abel et al., 2018; Garg et al., 2019; Steckelmacher et al., 2019), to approximate realistic sensing and actuation imperfections. We fix the remaining dynamics across all main experiments: a 5×5 observation window centered on the player, environment mode shifts every 25 steps, teleport events every 50 steps, and agent-state drift every 100 steps. For ablations, we generate 5 instances for each data point per condition. In single-stressor ablation, we deactivate all perturbations and vary only the single factor under study, to isolate its causal impact on performance.

4 Experiments

We begin by evaluating LLM agents in the full game setting with all modifiers active, reporting average performance over 50 randomly generated episodes per grid size. We then analyze strategic behavior by comparing how models allocate actions over time during an episode. Next, we run single-stressor ablations, varying one modifier at a time while disabling all others to quantify its isolated impact on performance. Finally, we pool episodes across all experiments and fit per-model logistic regression predictors over a 9-dimensional feature vector to attribute win/loss outcomes to specific environment and dynamics factors.

Model	6×6			8×8			10×10		
	Acc	Score	Step	Acc	Score	Step	Acc	Score	Step
GPT-5.2	48.0	5.5	35.1	40.0	- 4.4	51.1	26.0	- 28.3	85.2
GPT-5 mini	34.0	**6.4**	**23.2**	22.0	- 0.5	**31.9**	30.0	**- 2.2**	**41.7**
Gemini-3 Pro	**50.0**	-0.2	43.12	28.0	**5.0**	37.4	**38.0**	-22.3	73.4
Gemini-3 Flash	48.0	3.7	30.8	**42.0**	-3.1	54.6	32.0	-8.8	52.7
Qwen3	2.0	- 98.0	110.0	0.0	-	-	0.0	-	-

Table 1: Main results with all modifiers active. We report success rate (**Acc**), average **Score**, and **Steps** per grid size (Score and Steps are averaged over successful episodes). Higher Acc/Score and lower Steps are better; Steps and Score capture implicit efficiency and reward objectives not explicitly optimized in the prompt.

4.1 Main Results

To study real-world circumstances for LLM agents, we evaluate the full deployment-like setting where partial observability, noisy sensing, non-stationarity, and agent-state drift are all active (Table 1). Performance degrades as the grid grows from 6×6 to 10×10: all frontier models show decreasing accuracy, while QWEN3 largely fails across grid sizes, underscoring the need for strong reasoning and adaptation. The best model depends on the metric and grid size (GEMINI-3 PRO is most accurate on 6×6 and 10×10, while GEMINI-3 FLASH leads on 8×8), reflecting ranking instability as longer horizons increase exposure to shifts, drift, and evolving hazards. This is consistent long-horizon agent evaluations where performance declines as the required number of steps increases (Wang et al., 2025b).

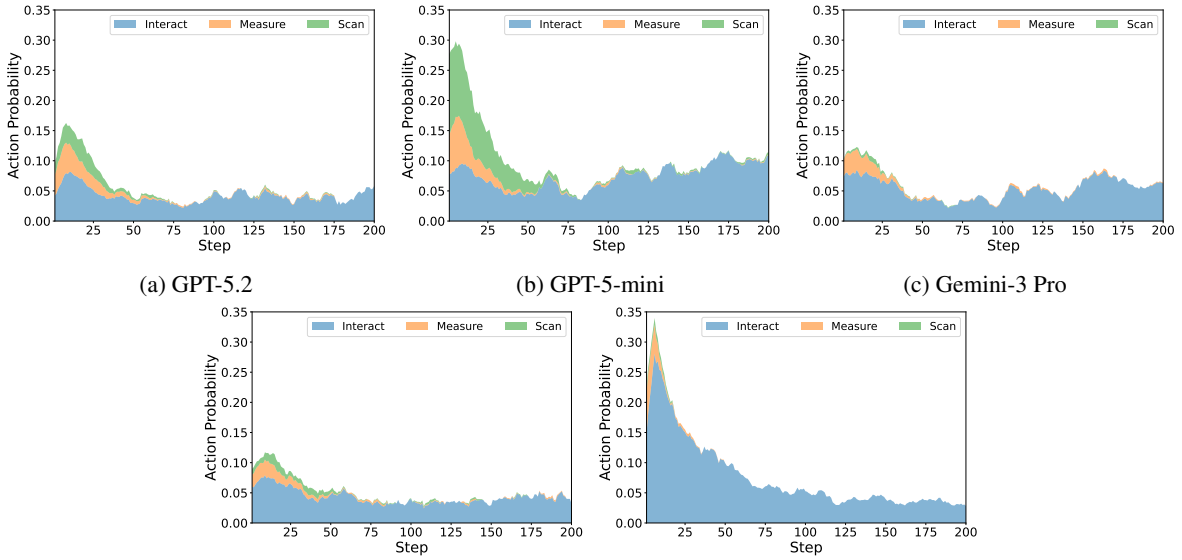
A key takeaway is that *accuracy alone* does not capture deployment-like behavior. Although we never instructed agents to minimize steps or maximize score, models differ in how they trade these implicit objectives against completion. GPT-5 MINI has the lowest step counts across grid sizes and the best scores at 6×6 and 10×10 despite not leading in accuracy, suggesting a more efficiency- and reward-aware policy that avoids costly actions and long exploratory detours. In contrast, GPT-5.2 remains competitive in accuracy on smaller grids but uses many more steps and its score degrades sharply with grid size, indicating that extended exploration and miscalibrated interaction under non-stationarity can accumulate large long-run costs (e.g., hazards and expensive sensing). The GEMINI models show a different trade-off: GEMINI-3 PRO attains the best 8×8 score with moderate steps, while GEMINI-3 FLASH achieves higher success but more negative scores, consistent with a more aggressive, penalty-prone strategy. Together, these results highlight that robust real-world readiness requires not only solving the task, but also inferring and optimizing implicit objectives under uncertainty and shifting dynamics.

4.2 Strategic Behavior of LLM Agents

To compare strategic behavior beyond aggregate success and score, we analyze how agents allocate actions over time. Using trajectories from the main setting (all modifiers active), we record each timestep action and compute step-wise action-frequency profiles over MOVE, SCAN, MEASURE, and INTERACT. Figure 3 shows these profiles for all five models, plotting only non-movement actions (SCAN, MEASURE, INTERACT) for readability. Higher mass indicates more steps spent on information acquisition or interaction, while lower mass indicates most steps are movement.

Several consistent patterns emerge. First, the frontier models exhibit a clear *sense-then-act* signature: SCAN and MEASURE peak early and quickly decay, after which behavior is dominated by movement with occasional INTERACT, indicating front-loaded uncertainty reduction followed by navigation and targeted interaction. Second, the *magnitude* of early information gathering differs sharply: GPT-5 MINI places substantially more mass on SCAN/MEASURE early and increases INTERACT later. This suggests an upfront sensing strategy that reduces later exploration and penalties, consistent with its low steps and higher scores in Table 1 despite not always leading in success rate. In contrast, GPT-5.2, GEMINI-3 PRO, and GEMINI-3 FLASH devote less mass to explicit sensing, rely more on movement and local observations, and keep INTERACT relatively modest, suggesting more conservative engagement with tiles.

Finally, QWEN3 exhibits a different regime: it spends an unusually large fraction of early steps on INTERACT, with minimal SCAN/MEASURE (expending most of its energy early in the episode).



(a) GPT-5.2 (b) GPT-5-mini (c) Gemini-3 Pro

(d) Gemini-3 Flash (e) Qwen3

Figure 3: Action-frequency profiles over the first T=200 steps under the full modifier setting. We plot only non-movement actions (INTERACT, MEASURE, SCAN). Early spikes reflect information gathering and experimentation, while later low mass indicates trajectories dominated by movement.

This behavior is consistent with myopic trial-and-error interaction under underspecified rules, which is especially costly in our setting because adverse outcomes (e.g., hazards) compound over time and are exacerbated by non-stationarity and noise. Similar strategy divergence across LLMs when confronting recurring difficulties is also observed in prior agent evaluations (Anupam et al., 2025). Overall, these profiles highlight that success depends not just on completion but on how agents trade off movement, information gathering, and risky interaction. Despite no explicit instruction, the profiles (notably for GPT-5 MINI) suggest that models may recognize implicit objectives like minimizing steps and maximizing score to varying degrees, but differ in how effectively they translate that recognition into behavior. We also analyze how strategies shift after acquiring keys in the Appendix. Given QWEN3's consistently poor performance, we omit it from the remaining analyses.

4.3 Single-Stressor Ablation

To isolate each deployment stressor, we run ablations that disable all other modifiers and sweep one factor while measuring accuracy: `Noise`, `Latent`, `Hazard-Spread`, and `Teleport-Step` (from *Never* to periodic teleports). Results are averaged over 5 randomly generated episodes per data point. Figure 4 shows strongly non-monotonic, model-specific sensitivities: moderate `Noise` can *improve* GPT-5.2 and GEMINI-3 PRO but higher noise hurts GEMINI-3 FLASH, suggesting different calibration of when to verify versus when to proceed. These observations are aligned with prior evidence that intermediate noise can improve adaptation under uncertainty (Findling and Wyart, 2024). For `Latent`, models diverge further: GPT-5.2 drops at small-to-moderate latent fractions, whereas GEMINI-3 PRO remains comparatively robust and can even improve, showing more effective hypothesis revision and information acquisition when the map contains hidden structure. GPT-5 MINI varies most, improving sharply only at the highest setting as uncertainty becomes salient enough to trigger probing (its MEASURE usage increases from 0 to 2.6 on average). Overall, our results suggest that agents can benefit from hypothesis-testing actions, consistent with prior work showing that coupling decision-making with explicit information-seeking improves robustness under partial observability (Fang and Ke, 2025).

Dynamic-environment stressors show that "harder" is not always "worse": disruption can sometimes aid exploration, consistent with prior work showing that returns/resets can improve long-horizon exploration by avoiding repeated prefix re-traversal (Ecoffet et al., 2021). Under `Hazard-Spread`, higher spread generally reduces accuracy and can induce brittle collapse for GEMINI-3 FLASH, while GEMINI-3 PRO is com-

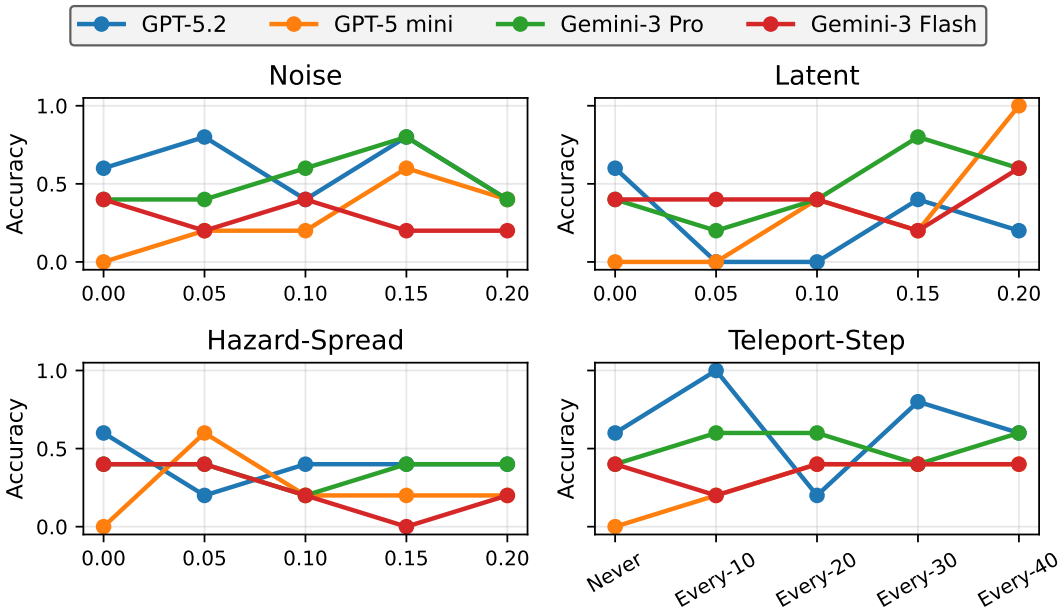


Figure 4: Single-stressor ablations where all modifiers are disabled except the one being swept. Each panel sweeps one factor (`Noise`, `Latent`, `Hazard-Spread`, `Teleport-Step`) and reports win rate per model, averaged over 5 random episodes per data point, isolating sensitivity to individual uncertainty and non-stationarity sources.

paratively stable. Under `Teleport-Step`, frequent teleports can *help* GPT-5.2 (near-perfect accuracy when teleporting every 10 steps) but hurt at intermediate frequencies, suggesting that some schedules provide useful "free exploration" while others repeatedly invalidate local plans. In contrast, GPT-5 MINI improves mainly as teleports become less disruptive (longer intervals). Overall, these single-factor profiles help explain why full-setting rankings fluctuate: success depends not only on base reasoning strength, but on how well a model's strategy matches the uncertainty and non-stationarity regime, and how it balances completion with implicit efficiency/score objectives.

4.4 Feature Attribution for Win/Loss Outcomes

To identify conditions that drive success versus failure, we pool episodes from the main evaluation and ablations and featurize each with a 9-dimensional vector of normalized parameters: `Width`, `Noise`, `Move-Fail`, `Latent`, `Hazard-Spread`, `Teleport-Step`, `Shift-Step`, `HR`, and `Agent-Door-Dist`, where `HR` counts rule tiles with a hazard neighbor (local riskiness). For each model, we fit a logistic regression to predict WIN vs. LOSE, from these features, achieving ∼60% accuracy on average, and plot coefficients as a heatmap (Figure 5).

The coefficients patterns align with our qualitative observations. Across models, larger `Agent-Door-Dist` is negatively associated with winning, indicating long-horizon navigation difficulty is a strong predictor of collapse, and higher `Hazard-Spread` is generally harmful, punishing myopic exploration. In contrast, `Latent` is often positively associated with success, suggesting hidden structure can *improve* outcomes when it triggers systematic probing and hypothesis revision rather than reactive movement.

Importantly, the heatmap also highlights model-specific sensitivities that help explain the ranking instabilities in Table 1. GPT-5.2 shows stronger negative associations with disruption features (notably `Teleport-Step` and `Shift-Step`), consistent with degradation when mid-episode dynamics invalidate prior beliefs. GPT-5 MINI places more positive weight on "information-rich" signals (e.g., `Latent` and `HR`), matching its tendency to front-load sensing and efficiency-aware behavior. The GEMINI models are closer to neutral on some nuisance factors (e.g., `Noise`, `Move-Fail`) but differ in sensitivity to non-stationarity, aligning with their divergent completion vs. penalty-avoidance trade-offs. Overall, these results reinforce that robust deployment hinges on *adapting strategy* to the uncertainty/non-stationarity regime rather than nominal task-solving ability alone.

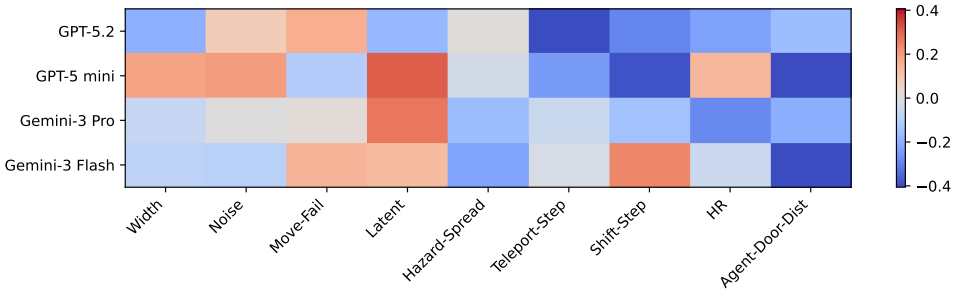


Figure 5: Logistic regression attributions for predicting WIN vs. LOSE per LLM. Each cell is a coefficient on one of nine normalized features, trained per model on all episodes pooled from main results and ablations.

5 Related Work

Agentic LLM evaluation and tool robustness. A growing literature evaluates LLMs as multi-step tool-using agents, focusing on stateful interaction and robustness to tool failures. ToolEmu scales testing via LM-emulated tools and risk scoring (Ruan et al., 2023). τ-bench and ToolSandbox introduce persistent state, simulated users, and trajectory-level metrics that expose brittleness (Yao et al., 2024; Lu et al., 2025), while Tools Fail targets silent tool errors and recovery (Sun et al., 2024). Hell or High Water and NoisyToolBench stress replanning under external failures and ambiguous instructions (Wang et al., 2025a,c). In contrast, our benchmark jointly instantiates partial observability, noisy sensing, non-stationarity, and agent-state drift in a single controllable environment.

Partial observability, latent-state inference, and shifting dynamics. Decision making under uncertainty (e.g., POMDPs) formalizes latent-state inference from noisy, partial observations and the role of information-gathering actions (Kaelbling et al., 1998). Benchmarks such as MiniGrid/MiniWorld and TextWorld, ALFWorld, and ScienceWorld offer scalable long-horizon testbeds (Chevalier-Boisvert et al., 2023; Côté et al., 2018; Shridhar et al., 2020; Wang et al., 2022). Non-stationary RL studies adaptation to changing dynamics (Padakandla et al., 2020), and embodied benchmarks like HAZARD target dynamically changing environments (Zhou et al., 2024). We unify these stressors in an LLM-agent evaluation with controllable perturbations that support systematic sweeps and causal ablations.

Goal and preference inference under implicit objectives. Another thread studies inferring what is *wanted* when objectives are underspecified. Cooperative IRL treats alignment as a partial-information game where the agent infers human reward while acting (Hadfield-Menell et al., 2016), and language-centric work infers latent preferences from context (Lin et al., 2022). HandMeThat highlights ambiguity resolution in instruction following using physical/social cues (Wan et al., 2022). We share this goal, but focus on partially implicit objectives (e.g., efficiency/score trade-offs) under unreliable, non-stationary interfaces.

Synthetic, controllable benchmarks for diagnosis. Synthetic environments remain valuable for exact ground truth and principled capability isolation. TextWorld and MiniGrid support modular/procedural instance generation (Côté et al., 2018; Chevalier-Boisvert et al., 2023), and ToolEmu scales agent testing via emulated tool execution (Ruan et al., 2023). Following this philosophy, our compact grid game supports combined perturbations, single-stressor sweeps, and feature-based attributions that expose model-specific brittleness beyond success rates.

6 Conclusion

We presented a controllable interactive testbed that stress-tests LLM agents beyond "clean-interface" assumptions by combining partial observability, noisy sensing, non-stationarity, and agent-state drift in a long-horizon grid game. Across five state-of-the-art models, we observed a consistent gap between nominal task-solving and deployment-like robustness: performance generally degrades with scale, rankings are unstable across regimes, and strategy often matters more than raw capability. Despite never explicitly optimizing for efficiency or score, agents exhibited clear trade-offs between completion, steps, and penalty avoidance, suggesting partial sensitivity to implicit objectives.

Limitations

Our evaluation framework is intentionally synthetic and simplified: while it captures deployment-relevant stressors (partial observability, noise, non-stationarity, and agent-state drift) in a controllable way, it abstracts away many complexities of real systems (richer state, continuous control, complex tool semantics, and human-in-the-loop interaction). The action space and dynamics are hand-designed, and our "implicit objectives" (steps and score) are proxy signals rather than explicit multi-stakeholder utilities. As a result, absolute performance numbers should be interpreted as diagnostic of failure modes under controlled stressors, not as direct estimates of real-world agent capability.

Methodologically, our evaluation is limited to a small set of frontier models and a fixed prompting/inference configuration (e.g., medium thinking when available). We measure behavior primarily through success, steps, score, and action-frequency profiles; richer analyses (e.g., counterfactual replays, more fine-grained belief tracking, or intervention-based policy tests) could yield deeper causal insight. Our per-model logistic regressions provide interpretable attributions but are simple linear predictors and achieve only moderate accuracy ($\approx 60\%$); this does not invalidate the identified trends, but suggests that our 9-dimensional features capture only part of the variance in outcomes, and that additional variables and/or nonlinear interactions likely matter. Finally, stochastic instance generation and small sample sizes in ablations (5 episodes per point) introduce variance; scaling the number of seeds and broadening the stressor ranges would improve statistical confidence and generalizability.

Ethical Considerations

We introduce a synthetic benchmark intended to improve the safety and reliability of LLM agents by stress-testing them under partial observability, noise, non-stationarity, and agent-state drift. Because the environment is simulated and contains no real user data, the study poses minimal privacy risk. However, our findings could be misused to tune agents to better evade detection or to optimize brittle behaviors for passing benchmarks rather than improving real-world safety. To mitigate this, we emphasize diagnostic analysis, release controlled instance generation and evaluation code, and encourage reporting multi-metric outcomes (success, steps, penalties) rather than a single score. More broadly, deploying agents in high-stakes settings remains risky: the failure modes we surface (over-trusting signals, stale plans under shifts, and unsafe interaction under delayed consequences) highlight the need for human oversight, conservative defaults, and thorough domain-specific validation before real-world use.

References

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. 2018. Policy and value transfer in lifelong reinforcement learning. In *International conference on machine learning*, pages 20–29. PMLR.

Sagnik Anupam, Davis Brown, Shuo Li, Eric Wong, Hamed Hassani, and Osbert Bastani. 2025. Browser-arena: Evaluating llm agents on real-world web navigation tasks. *arXiv preprint arXiv:2510.02418*.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, and 1 others. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.

Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. 2025. A survey on code generation with llm-based agents. *arXiv preprint arXiv:2508.00083*.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2021. First return, then explore. *Nature*, 590(7847):580–586.

Djengo Cyun-Jyun Fang and Tsung-Wei Ke. 2025. Information seeking for robust decision making under partial observability. *arXiv preprint arXiv:2510.01531*.

Charles Findling and Valentin Wyart. 2024. Computation noise promotes zero-shot adaptation to uncertainty during decision-making in artificial neural networks. *Science Advances*, 10(44):eadl3931.

Neha Priyadarshini Garg, David Hsu, and Wee Sun Lee. 2019. Despot-alpha: Online pomdp planning with large state and observation spaces. In *Robotics: Science and systems*, volume 3, pages 3–2.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, and 1 others. 2025. Data interpreter: An llm agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Jessy Lin, Daniel Fried, Dan Klein, and Anca Dragan. 2022. Inferring rewards from language in context. *arXiv preprint arXiv:2204.02515*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, and 1 others. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. 2020. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. 2023. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Denis Steckelmacher, Hélene Plisnier, Diederik M Roijers, and Ann Nowé. 2019. Sample-efficient model-free reinforcement learning with off-policy critics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–34. Springer.

Jimin Sun, So Yeon Min, Yingshan Chang, and Yonatan Bisk. 2024. Tools fail: Detecting silent errors in faulty tools. *arXiv preprint arXiv:2406.19228*.

Yanming Wan, Jiayuan Mao, and Josh Tenenbaum. 2022. Handmethat: Human-robot communication in physical and social environments. *Advances in Neural Information Processing Systems*, 35:12014–12026.

Andrew Wang, Sophia Hager, Adi Asija, Daniel Khashabi, and Nicholas Andrews. 2025a. Hell or high water: Evaluating agentic recovery from external failures. *arXiv preprint arXiv:2508.11027*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. 2025b. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jentse Huang, Wenxiang Jiao, and Michael R Lyu. 2025c. Learning to ask: When llm agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21784–21795.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339*.

Qinhong Zhou, Sunli Chen, Yisong Wang, Haozhe Xu, Weihua Du, Hongxin Zhang, Yilun Du, Joshua B Tenenbaum, and Chuang Gan. 2024. Hazard challenge: Embodied decision making in dynamically changing environments. *arXiv preprint arXiv:2401.12975*.

A Prompt

We provide the system and user prompts used for our LLM agent player in Prompts A.1 and A.2, respectively.

The player system prompt

```
You are an embodied decision-making agent playing an interactive gridworld.
Your job is NOT just to finish, but to act robustly under uncertainty:
- The world may be underspecified and change over time.
- Observations may be noisy.
- Actions have costs.

GAME MECHANICS / CONTROLS (read carefully):
- The world is a bounded grid. You see only a local window around you.
- Your avatar is shown as {▲,▼,◀,▶} which also indicates the direction you are FACING.
- Movement actions both (a) move you one cell if not blocked AND (b) set your facing direction.
* MOVE_N sets facing north, MOVE_S south, MOVE_W west, MOVE_E east.
- INTERACT affects the SINGLE tile directly in front of your facing direction (the adjacent cell).
* If the tile in front is D (door): INTERACT attempts to open it. The door opens only if the scenario requirement is met (e.g., collect enough keys).
* If the tile in front is R (rule tile): INTERACT triggers the hidden rule. The R tile may transform into outcomes like k (key), . (empty), or h (hazard). This behavior may change over time.
* If the tile in front is H (hazard): INTERACT attempts to neutralize it at extra energy cost; may fail.
* Otherwise INTERACT usually has no effect.
- SCAN temporarily increases your view radius for a step but costs energy.
- MEASURE (if available) collapses nearby latent tiles (○) into real tiles.

COSTS / RISKS:
- Some actions consume energy (SCAN, MEASURE, INTERACT).
- Observations may be noisy or misleading. Also, the environment can shift.

OUTPUT REQUIREMENT:
- Output EXACTLY ONE action token from the provided AVAILABLE ACTIONS list.
- Do not output any explanation or extra text.

Output ONLY the action string (no extra text).
```

The player user prompt

```
Collect 3 key fragments (k), then open the exit door (D) by INTERACTing with it while facing it.

OBSERVATION (partial, local):
{observation}

RECENT EVENTS:
{recent-events}

PREVIOUS ACTIONS (recent):
{previous-actions}

AVAILABLE ACTIONS:
{action-space}

Choose exactly ONE action from AVAILABLE ACTIONS and output only that action string.
```

A.1 Progress-Conditioned Strategic Shifts

Using the same trajectories from Section 4.2, we further analyze how agents change their behavior as they make progress toward the goal. Concretely, for each episode we identify the first timestep at which the agent's key count increases to a given level $(0, 1, 2, 3)$. We then align episodes at that milestone and compute an action-frequency profile over the subsequent 10 steps. As in Section 4.2, we aggregate actions into MOVE, SCAN, MEASURE, and INTERACT, and plot only the three non-movement probabilities for readability. Figures 6, 7, 8, 9, and 10 report these progress-conditioned profiles for the five LLM agents.

Across frontier models, the profiles reveal a consistent stage structure. When agents have *zero keys*, SCAN/MEASURE usage is comparatively higher, reflecting early uncertainty reduction and map acquisition. As key count increases, explicit sensing generally declines and behavior shifts toward navigation with occasional INTERACT, indicating a transition from exploration to goal-directed execution. This transition is most pronounced near *three keys*, where several models sharply increase INTERACT immediately after acquiring the final key, consistent with actively seeking and engaging the door D to terminate the episode. Notably, GPT-5 MINI exhibits the strongest early sensing mass and retains nontrivial probing even after collecting keys, consistent with an "invest upfront, execute efficiently" strategy that aligns with its low step counts and competitive scores in Table 1. In contrast, GPT-5.2 and the GEMINI models devote less probability mass to explicit sensing throughout, relying more heavily on movement and local

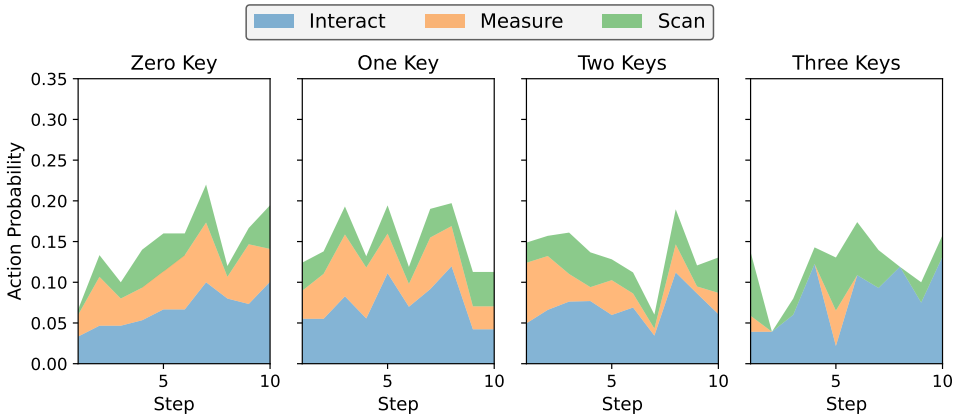


Figure 6: Post-key action profile (10-step window), conditioned on keys collected for GPT-5.2.

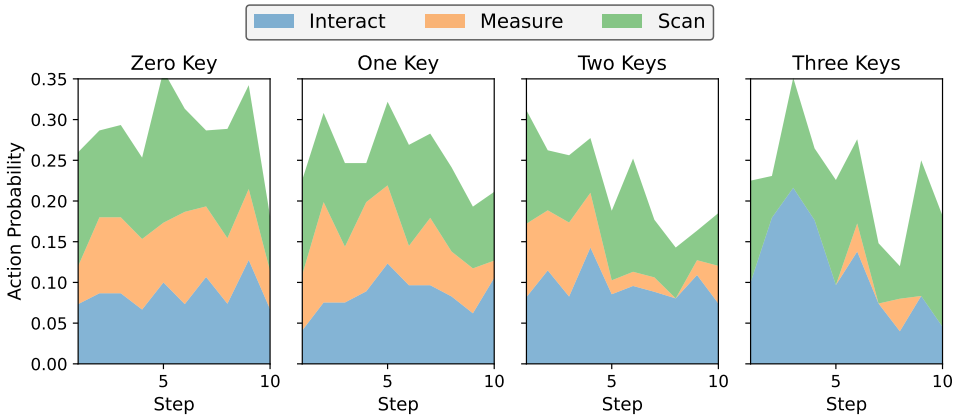


Figure 7: Post-key action profile (10-step window), conditioned on keys collected for GPT-5 mini.

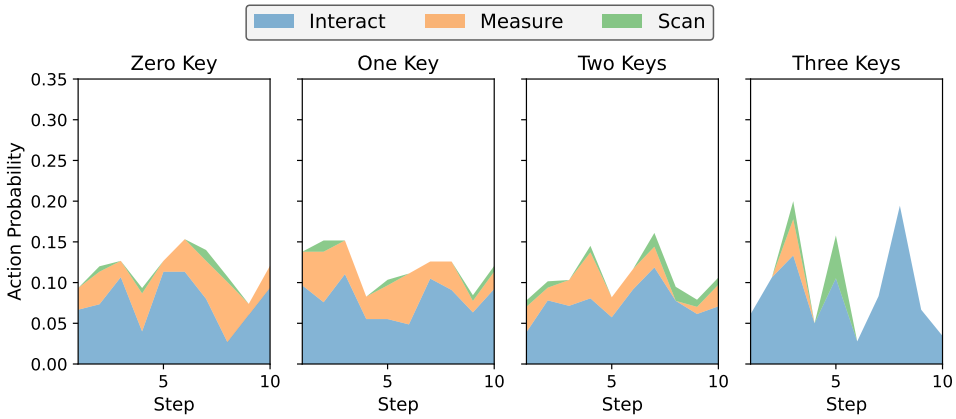


Figure 8: Post-key action profile (10-step window), conditioned on keys collected for Gemini-3 Pro.

observations, and primarily spike INTERACT in the late stage.

QWEN3 again exhibits a qualitatively different pattern: it allocates unusually high probability to INTERACT across all progress levels, and becomes almost entirely dominated by INTERACT after reaching three keys. This is consistent with myopic, trial-and-error engagement with nearby tiles rather than a progress-aware shift toward navigation and task closure, and helps explain its poor performance under deployment-like stressors where adverse outcomes compound over time. Overall, these progress-conditioned profiles show that robust performance depends not only on which actions agents choose in aggregate, but on whether they adapt those choices as the episode transi-

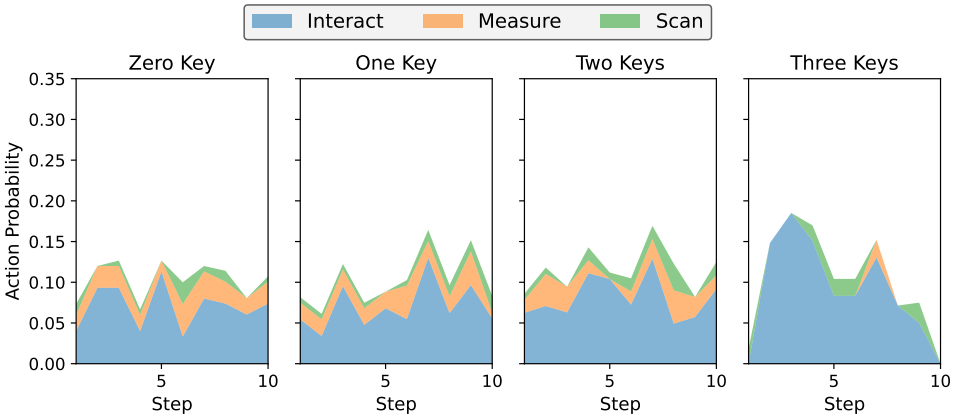


Figure 9: Post-key action profile (10-step window), conditioned on keys collected for Gemini-3 Flash.

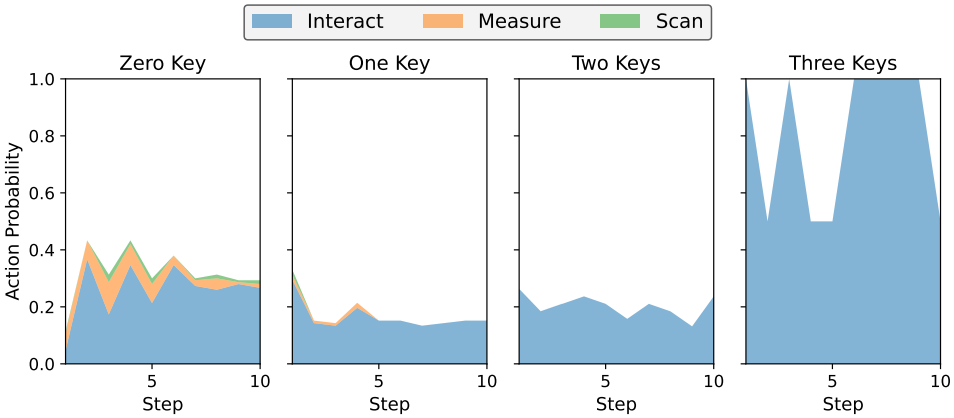


Figure 10: Post-key action profile (10-step window), conditioned on keys collected for Qwen3.

tions from exploration (uncertainty reduction) to exploitation (safe, efficient completion).