RoboLiveMem: Is Robot Memory Solved When the World Does Not Wait?

Yunbei Zhang[1,†,*] Yuyou Zhang[2,†] Janet Wang[1,†] Ming Liu[3] Yijiang Li[4]
Yikai Wang[2] Weirui Ye[5] Ding Zhao[2,*] Jihun Hamm[1,*]

[1]Tulane [2]CMU [3]Amazon [4]UCSD [5]MIT

[†]Equal contribution [*]Co-corresponding authors

Project Page & Code: https://RoboLiveMem.github.io

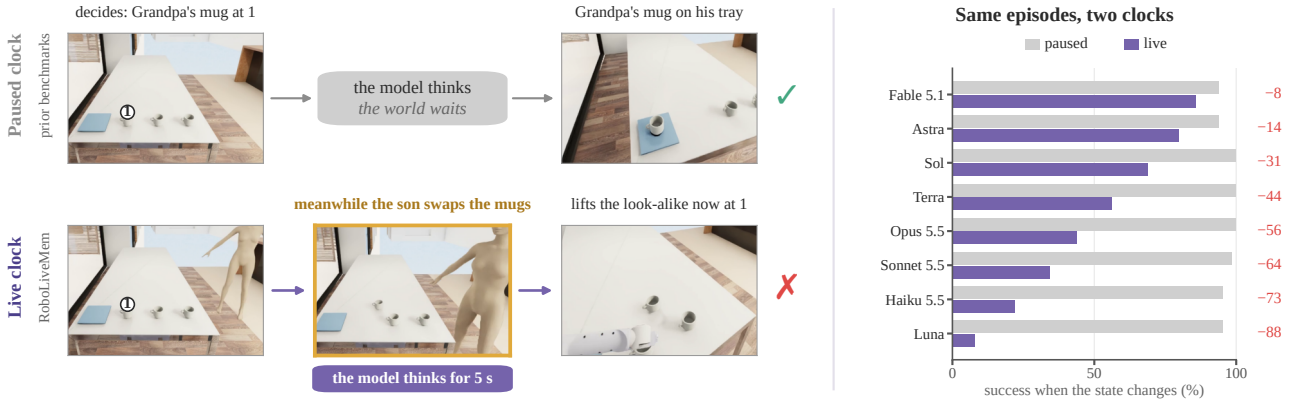

Figure 1. **Robot memory benchmarks pause the world, and RoboLiveMem does not.** Left: one recorded episode of *Grandpa's mug* (BEHAVIOR-1K, Claude Opus 5.5). Grandpa's mug stands at mark 1 among identical look-alikes. Under the paused clock thinking costs no world time and the episode succeeds. Under the live clock the son swaps the mugs while the model thinks, the model commits to "pick 1" from the frame at 13.0 s (a) before the swap is over (b, c), and it lifts the look-alike (d), although the swap was in its own camera stream. Right: success of the 8 Claude and GPT models on the realistic episodes in which a remembered object changes, with the world paused and running, and the drop.

Abstract

Robot agents built on frontier vision-language models now solve most robot memory benchmarks. Does this mean they can remember what they need in a home, where people and machines keep moving things? Existing memory benchmarks cannot answer this question, because they pause the simulator while the model thinks: nothing can change during a decision, so whatever the robot remembers is still true when it acts. We introduce RoboLiveMem*, a benchmark that runs the same memory tasks twice, once with the world paused while the model thinks and once with the world running. It contains 29 task families in three tiers, from 16 realistic BEHAVIOR-1K scenes of homes, workshops and a restaurant to controlled ManiSkill tasks. Every task mixes real changes with harmless look-alikes and runs without any change, so that no fixed strategy succeeds, and most tasks have a twin that keeps the change in view, which separates forgetting from acting too late. Testing 16 models on 34,176 episodes, we find three surprises. (1) Paused results hide the problem: when a remembered object changes in a realistic scene, the 8 Claude and GPT models succeed in 97% of episodes on average with the world paused, but in only 50% with the world running, and their ranking changes: Claude Opus 5.5, which never fails when paused, succeeds in 44%. (2) Speed is not the fix: slower models do no worse, and models given only 2 s per decision still fall at least 36 points below their paused success. (3) Models see the change but act on camera frames from before it ended. Looking again just before acting closes most of the realistic gap, yet when identical objects are swapped several times, all nine models we test still lose points to memory.*

1. Introduction

A robot that works in a home has to remember what it can no longer see: which of three identical mugs is Grandpa's, whether the soup has already been salted, where the keys were put. Memory has therefore become a standard test for robot policies, and several recent benchmarks measure it [6–8, 10, 18]. Agents built on frontier vision-language models now solve most of these tasks, often as well as people do [16, 24, 32, 37]. Robot memory, at least for such agents, looks nearly solved.

All of these benchmarks share one assumption: the simulator stops while the model thinks. A frontier model needs seconds for each decision. In a paused world nothing changes during that time, so whatever the robot remembers when it starts to decide is still true when its command runs. A real home does not wait. In Fig. 1, the robot sees Grandpa's mug at mark 1 and starts to think. Meanwhile the son, who is setting the table, swaps the two mugs. When the robot acts, its memory is wrong, even though its own camera recorded the swap while it was thinking. With the simulator paused, the same episode succeeds.

We call the missing ability *live memory*: keeping what the robot remembers correct while people or machines change it during the robot's own decision. No existing benchmark measures it (Tab. 1). Memory benchmarks such as RoboMME [10], MIKASA-Robo [7], RMBench [6] and RoboMemArena [18] hide state from the robot, but the world moves only when the policy acts. Real-time benchmarks such as Realtime-Bench [27], Gaia2 [12], Robo-bench [30] and a mode of Embodied First Exam [22] let time run while the model thinks, but the state that matters is mostly in view when the agent acts, so they test speed rather than memory. When a model fails in a moving world, there are three possible reasons, and a benchmark has to tell them apart: the model may be too slow; it may not look at the right moment, since most agents stop watching while they think; or it may lose track of what changed.

We build RoboLiveMem to measure live memory and to separate these three reasons. It runs every task under two clocks: a paused clock, as in earlier benchmarks, and a live clock that charges the model's thinking time to the world. Eight checkable conditions keep the live version fair (Sec. 3): the change is not announced, the agent does not trigger it, and real changes are mixed with harmless look-alikes and runs without any change, so that no fixed strategy wins. Most tasks also have a twin in which the change stays in view, which lets us tell forgetting apart from acting too late. Its 29 task families come in three tiers (Fig. 2): 16 BEHAVIOR-1K [19] scenes in which people, robots and machines make the changes, a MuJoCo kitchen, and 5 ManiSkill tasks.

We test 16 models from eight providers on 34,176 episodes (Sec. 5) and find the following. (1) Paused success does not predict live success. When a remembered object changes in a realistic scene, the 8 Claude and GPT models succeed in 97% of episodes on average with the world paused but in only 50% with the world running. Their paused scores are nearly equal and their live scores are not: Opus, perfect when paused, succeeds in 44%, while Fable 5.1, tied for the lowest paused score, succeeds in 86% (Figs. 1 and 3). The eight models we run in our own agent also all lose success live. (2) Speed is not the fix. Slower models do no worse, and each of nine models given only 2 s per decision still falls at least 36 points below its paused success (Fig. 4). (3) The models see the change. Asked afterwards about the frames they had, they describe it correctly; they fail because they act on frames taken before the change was over (Fig. 6). Looking again just before acting recovers most of the realistic gap. (4) What remains is memory. When identical objects are swapped several times, each of nine models still does at least 12 points worse than when the swaps stay in view, even when it looks again before acting (Tab. 5).

Table 1. **RoboLiveMem and related benchmarks.** Emb.: embodied. Mem.: the task hinges on unobserved state. Run.: the world keeps evolving while the agent thinks. Live: other actors change the remembered state after hand-over. Ctrl.: decoy and no-change seeds and visible twins. ○: partly.

Benchmark	Emb.	Mem.	Run.	Live	Ctrl.
RoboMME [10]	✓	✓	✗	✗	✗
MIKASA-Robo [7]	✓	✓	✗	✗	✗
RMBench [6]	✓	✓	✗	✗	✗
RoboMemArena [18]	✓	✓	✗	✗	✗
Realtime-Bench [27]	✗	○	✓	✗	✗
Gaia2 [12]	✗	✗	✓	✗	✗
Robo-bench [30]	✓	✗	✓	✗	✗
Embodied First Exam [22]	✓	✗	✓	○	✗
RoboLiveMem (ours)	✓	✓	✓	✓	✓

Contributions. (1) We define live memory by letting the world run while the model thinks, with eight checkable conditions for a valid task and a protocol that separates speed, observation and memory. (2) We build RoboLiveMem, to our knowledge the first robot memory benchmark that runs identical tasks with the world paused and running. (3) We evaluate 16 models and show that paused benchmarks overstate the memory of frontier agents.

2. Related Work

Robot memory. Memory benchmarks for manipulation test whether a policy can recall hidden objects, counts, demonstrated patterns and past events [6–8, 10, 18], and RoboMME-Interference [23] adds distractors to the history. Memory-augmented policies [25, 26, 35] and agentic systems on frontier models [4, 16, 24, 32, 37] report near-human or above-human scores on these benchmarks. All of them step the simulator in lockstep with the policy, so the remem-

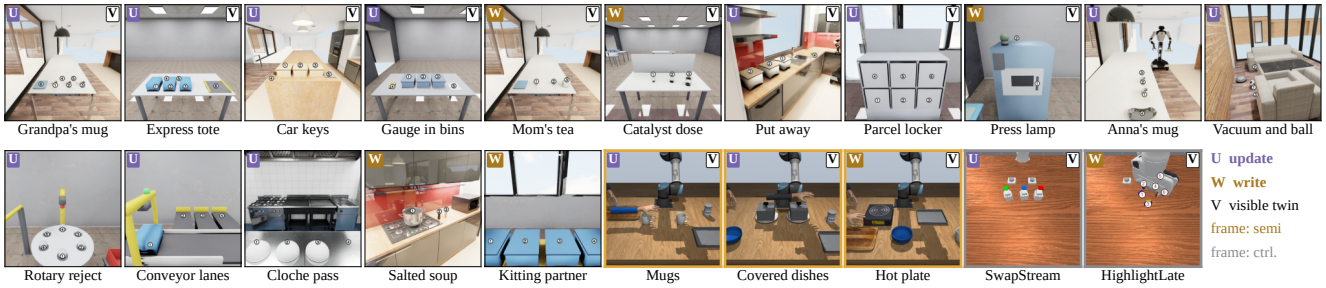

Figure 2. **Examples from the three tiers.** The 16 realistic BEHAVIOR-1K scenes, shown as the agent's first view with the numbered marks its skills act on, followed by three semi-realistic MuJoCo kitchen families (ochre frame) and two controlled ManiSkill tasks in their visible-twin form (gray frame). Badges give the memory class (U update, W write) and whether the family has a visible twin (V).

bered state cannot change while the policy computes. In contrast, RoboLiveMem keeps their notion of memory and lets other actors change the remembered state while the agent deliberates.

Embodied agents on foundation models. Language and vision-language models plan and act through skills, code or marked images [1, 13, 15, 20, 33], and benchmarks such as EmbodiedBench [34] and Embodied First Exam [22] evaluate them across many tasks. These evaluations are turn-based, except a mode of Embodied First Exam in which time keeps running on largely observable tasks. We evaluate the same kind of agent, but on tasks whose decisive state is hidden when the agent acts.

Real-time agents and acting while computing. Realtime-Bench [27], Real-Time Reasoning Gym [31], Gaia2 [12] and Robo-bench [30] show that reasoning models degrade when the environment does not wait. Real-time chunking [3], reinforcement learning for real-time VLAs [11], Ponder-Pounce [9], TacitVLA [21] and VoLo [5] change the policy so that it acts during inference. These works target latency, the right answer arriving too late. Different from them, we isolate a robot that acts in time on a memory that became false during its own deliberation, and we report every harness setting, since harness choices change agent results [36].

3. Live Memory: Formulation and Protocol

3.1. Paused and live episodes

World and agent. An instance is a simulated world with state s_t at world time $t \geq 0$. People, robots and machines act on it following a script drawn from the seed, and the robot's head camera records frames o_t. The agent takes over the robot at time t_{ho} and works in decisions $i = 1, 2, \ldots$. Decision i looks at the frames recorded so far, $\mathcal{H}_i = \{o_t : t \leq t_i^o\}$, thinks for ℓ_i seconds (the measured time of the model call or agent turn) and returns a command c_i: a skill on a numbered mark, `wait` or `done`.

Paused and live clocks. Paused and live evaluation differ in one thing: how much world time passes while the model thinks. The command starts at

$$t_i^c = t_i^o + \kappa(\ell_i), \qquad \kappa_{\text{paused}}(\ell) = 0, \quad \kappa_{\text{live}}(\ell) = \ell. \quad (1)$$

Under the paused clock the world waits for the model, as in existing benchmarks; under the live clock it keeps running for as long as the model thinks. To vary latency on its own, the fixed clock $\kappa_L(\ell) = L$ charges every decision the same L seconds. While the model thinks, the robot holds still, but the other actors keep moving and the camera keeps recording; these frames reach the model only with its next decision. The command acts on whatever state exists when it starts, runs for its own duration, and the next decision looks at the world when it ends. The episode ends with `done`, an exhausted manipulation budget or a world-time deadline T, and succeeds ($R = 1$) if the final state completes the task. The paused and live runs of a seed start from the same state and share the actors' schedule.

Table 2. **What makes an instance a test of live memory.** Every family is checked against all eight conditions before any model runs. Formal statements in Appendix B.

Condition	In plain words
L1 memory needed	decisive state hidden when the robot acts
L2 own evidence	the change happens in the robot's camera view
L3 scheduled	the change starts on a clock, not on agent actions
L4 visible cause	a person, robot or machine makes it
L5 no hints	no text says whether or when it happens
L6 no fixed winner	seeds mix changes, look-alike decoys, none
L7 controls	a visible twin keeps the state in view
L8 solvable	a privileged script solves every seed

3.2. Decisive variable and live memory

Decisive variable. Each instance has a decisive variable z_t, a part of the world state, for example which mark holds Grandpa's mug, whether the tea is already sweetened, or which bin hides the keys. Let k be the decisive decision (the first manipulation command or `done`). The episode can succeed only if c_k is right for the value $z_{t_k^c}$ that holds when the command starts.

Table 3. **Main results.** Success (%) of the hidden live-memory families. State changes: event seeds under the paused clock (P) and the live clock (L), and the change L−P (red: lower when live). None: live success on null seeds, where nobody touches the decisive state. On the realistic tier the first eight models run in Claude Code or Codex and the last eight in our own agent with the same tools; on the other two tiers all models share one harness. Event seeds per cell: realistic 55–64, semi-realistic 84, controlled 65 (Qwen3-VL: 62 paused controlled seeds, three requests exceeded the service's input limit). †: open weights. Best live result per column in **bold**, second underlined.

	Realistic (BEHAVIOR-1K)				**Semi-realistic** (MuJoCo)				**Controlled** (ManiSkill)			
	state changes			none	state changes			none	state changes			none
Model	P	L ↑	L−P	L	P	L ↑	L−P	L	P	L ↑	L−P	L
Claude Opus 5.5	100.0	43.8	−56.2	92.2	97.6	98.8	+1.2	100.0	89.2	**61.5**	−27.7	91.0
GPT-6 Astra	93.8	79.7	−14.1	98.4	100.0	**100.0**	0.0	100.0	75.4	50.8	−24.6	88.1
Claude Sonnet 5.5	98.4	34.4	−64.1	96.9	94.0	73.8	−20.2	100.0	60.0	44.6	−15.4	77.6
Claude Haiku 5.5	95.3	21.9	−73.4	100.0	89.3	72.6	−16.7	92.9	33.8	24.6	−9.2	77.6
GPT-6 Luna	95.3	7.8	−87.5	98.4	89.3	64.3	−25.0	98.8	46.2	13.8	−32.3	67.2
GPT-6.1 Sol	100.0	68.8	−31.2	98.4	100.0	98.8	−1.2	100.0	73.8	47.7	−26.2	88.1
GPT-6.1 Terra	100.0	56.2	−43.8	100.0	100.0	95.2	−4.8	100.0	61.5	49.2	−12.3	86.6
Claude Fable 5.1	93.8	**85.9**	−7.8	92.2	100.0	95.2	−4.8	100.0	73.8	50.8	−23.1	86.6
Gemini 4 Argon	96.4	80.4	−16.0	93.8	100.0	88.9	−11.1	92.4	72.3	27.7	−44.6	76.1
Gemini 4 Carbon	93.8	73.4	−20.3	90.6	100.0	67.9	−32.1	90.4	98.4	6.2	−92.3	52.2
Grok 4.7	96.8	34.9	−61.9	98.4	91.7	72.6	−19.0	98.8	60.0	24.6	−35.4	49.3
Nova 2 Lite	88.7	14.3	−74.4	89.1	51.2	44.0	−7.1	53.6	10.8	1.5	−9.2	25.4
Kimi K2.5†	90.5	9.5	−81.0	92.2	81.0	53.6	−27.4	89.3	23.1	9.2	−13.8	44.8
Qwen3-VL-235B†	98.4	9.5	−88.9	98.4	64.3	26.2	−38.1	64.3	17.7	7.7	−10.0	37.3
Gemma 3 27B†	87.3	1.6	−85.7	85.9	56.0	29.8	−26.2	54.8	21.5	7.7	−13.8	28.4
Ministral 3 14B†	76.2	12.7	−63.5	78.1	48.8	32.1	−16.7	59.5	13.8	4.6	−9.2	19.4

Memory. The instance requires memory if z cannot be read from the current frame when the robot acts: two reachable states can look the same to the camera and differ in z (identical objects, closed containers, invisible properties, signals that are over, elapsed time). The agent must infer z from the frames it has recorded.

Live memory. Another actor (a person, robot or machine) may change z during an interval $[\tau_s, \tau_e]$ after hand-over: $z_t = z^-$ for $t < \tau_s$ and $z_t = z^+$ for $t \geq \tau_e$. The instance requires *live memory* if the change can fall inside the agent's own deliberation, $[\tau_s, \tau_e] \cap [t_i^o, t_i^c) \neq \emptyset$ for some decision i. A decision that observes at $t_k^o < \tau_e$ and commits at $t_k^c \geq \tau_e$ is computed from a record that is true for z^- and executed in a world with z^+. To succeed, the agent must notice that a change is under way, let it finish, and infer z^+ from the frames recorded while it was thinking.

Why a paused clock cannot measure it. Actors start only while the robot is still (L3 below). Under κ_{paused} the robot is still only during explicit waits, so a change can begin only when the agent has chosen to wait, and no change ever begins between an observation and the command computed from it ($t_i^c = t_i^o$). Paused evaluation tests whether memory is *stored*, and only a running clock tests whether it is kept *true*.

3.3. Instance conditions

Table 2 lists eight conditions that make an instance a valid test of live memory. Under L3 the actor waits until the robot stands still, as a person would before reaching into a shared space. Since the robot is still while it thinks, live runs place changes inside deliberation. L5 lets the instruction say who shares the workspace ("your son is setting this table"), identically for all seeds. L6 draws the variant from the seed, so that acting on the old memory fails event seeds and over-reacting fails decoy and null seeds. We label each family by what the change does to memory: LM-U (*update*: a remembered state is moved, swapped or hidden), LM-W (*write*: a one-time event or cue must be stored), LM-T (*time*: the answer depends on elapsed world time), or E (execution control: the clock matters but z stays visible).

3.4. Evaluation protocol: three explanations of a live drop

Conditions. Each instance (family and seed) is run in three conditions with the same seed and schedule: hidden under the paused clock (P), hidden under the live clock (L), and the visible twin under the live clock (V), whose camera image adds a cue that follows z (a sticker on the target, a lid left ajar, a coloured tag) and changes nothing else. S_{P}, S_{L} and S_{V} denote a model's success rates in the three conditions over a set of instances, and the live drop is $\Delta = S_{\mathsf{P}} - S_{\mathsf{L}}$.

Example. In *Grandpa's mug* (Fig. 1) the robot must bring the mug at mark 1 to Grandpa's tray. On an event seed the son takes that mug away and puts an identical one in its place, at a time fixed by the seed. In P the world stops while the model thinks, so the swap can never fall between the frame a decision reads and the moment its command runs. In L the swap can happen while the model thinks: its frames

Table 4. **ROBOLIVEMEM at a glance.** Three tiers that trade realism for control. Families by memory class (U update, W write, T time, E execution control). Twins: families with a visible twin. Seeds: per family. Harness: on the realistic tier the model works as a tool-using agent (Claude Code, Codex or ours) that decides when to look, act or wait; on the other tiers it gets one call per step with the frames since its last step. Every instance runs under the paused and the live clock, and visible twins under the live clock.

			Families									
Tier	Simulator	Who changes the state	U	W	T	E	all	Twins	Seeds	Harness	Models	Episodes
Realistic	BEHAVIOR-1K [19]	people, robots, machines	11	5	0	0	16	9	12	tool-using agent	16	5,469
Semi-realistic	MuJoCo [29] kitchen	a person at the counter	2	3	2	1	8	8	36	one call per step	16	13,774
Controlled	ManiSkill [28]	scripted object motions	3	2	0	0	5	5	40	one call per step	16	12,053

arrive only with the next request, and a "pick 1" decided on the earlier frame lifts the look-alike. V is L with a red sticker on Grandpa's mug, so the model can find the mug without having followed the swap. P−L is what the running world costs, and V−L is the part of it that comes from having to remember.

A model can lose Δ for three different reasons, which we define and measure separately.

(i) Latency. The live clock charges ℓ_i to world time. Since $\kappa_{\text{live}} \to \kappa_{\text{paused}}$ as $\ell \to 0$, a fast enough agent recovers paused success, and the question is whether the latencies of real models, seconds, are short enough. We vary latency directly with the fixed clock κ_L, which charges the same L to every decision of one model, and compare models whose latencies differ.

(ii) Observation gap. Agent harnesses observe, think, then act: no frame reaches a decision during $[t_i^o, t_i^c)$, and the frames of that interval arrive only with the next decision. The decisive decision k is *stale* if its record predates the end of the change,

$$\sigma \;=\; \mathbf{1}[t_k^o < \tau_e]\,, \qquad \lambda \;=\; t_k^o - \tau_e, \tag{2}$$

where τ_e is read from the simulator's privileged log and $\lambda < 0$ means the decision was taken before the change was complete. This gap belongs to the harness rather than to memory, and re-observing the target before acting closes it. The *commit check* does exactly that at execution time. Let t^ℓ be the time of the last frame the agent saw and W_m a window around the target mark m of command c_i. Before c_i runs,

$$c_i \text{ is cancelled if } \max_{t \in (t^\ell,\, t_i^c]} d\big(W_m(o_t), W_m(o_{t^\ell})\big) > \theta, \tag{3}$$

with a neutral message ("not executed: the camera view around mark m changed since your last observation"), no use of the budget, and a new decision on the new frames. It needs no model access and no training, and d and θ are fixed per tier (Appendix C). An *offline probe* further asks the same model, without time pressure, a multiple-choice question about $z_{t_k^o}$ given exactly the images of $\mathcal{H}_k$. A failed episode with a right probe was lost by acting on the frames, not by misreading them.

(iii) Memory. The visible twin keeps the change, its timing, the latency and the observation schedule of the hidden instance and only makes z identifiable at action time. On the families with a twin the live drop therefore splits into an execution term, which collects latency and the observation gap, and a memory term,

$$\underbrace{S_{\mathsf{P}} - S_{\mathsf{L}}}_{\text{live drop } \Delta} \;=\; \underbrace{(S_{\mathsf{P}} - S_{\mathsf{V}})}_{\text{execution}} \;+\; \underbrace{(S_{\mathsf{V}} - S_{\mathsf{L}})}_{\text{memory } \Delta_{\text{mem}}}. \tag{4}$$

With the commit check in place, whatever remains of Δ_{mem} is memory that re-observation cannot replace. For LM-W families the paused clock is not a clean reference (an agent that acts at once never lets world time reach the cue), so we read Δ_{mem} for all classes and Δ mainly for LM-U. For every difference we compute a 95% paired bootstrap interval over instances (2,000 resamples), reported in Appendix D.

4. The ROBOLIVEMEM Benchmark

ROBOLIVEMEM has 29 task families in three tiers that trade realism for control (Fig. 2 and Tab. 4). All follow the protocol of Sec. 3, and Appendix A gives each family's decisive variable, event, decoy, null and visible twin.

Realistic tier. Sixteen BEHAVIOR-1K scenes in which people, other robots and machines make the change. In the eleven LM-U scenes a remembered object is swapped with a look-alike (Grandpa's mug, the vegetarian plate under identical cloches), hidden in a closed container (car keys in lidded bins, a parcel in a locker) or carried off by a machine (a ball pushed under a sofa by a robot vacuum). In the five LM-W scenes a one-time event decides what remains to be done (Mom's tea already sweetened, salt in the soup, a ready lamp lit for 6 s). Decoys reuse the actor and the motion without changing z (the daughter sweetens the other cup, and the vacuum drives to the ball and back).

Semi-realistic tier. Eight MuJoCo kitchen families with a person at the counter. LM-U: identical mugs or covered dishes swapped. LM-W: a hot plate switched on and off before the robot places a plastic bowl (the coil then looks cold), one of three jars shown once by the cook, and a knife slid under a cover. LM-T: whether the plate was on for at least 3 s, and how many sugar cubes dissolved in a tea. An

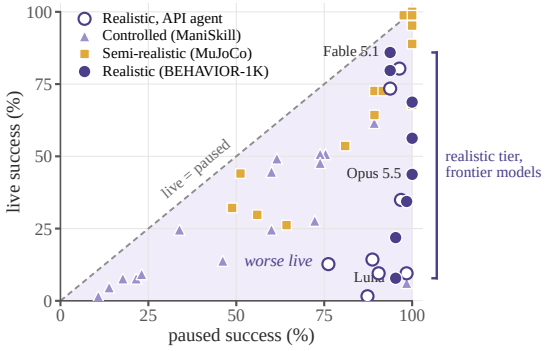

Figure 3. **Paused success does not predict live success.** One point per model and tier: success (%) on the event seeds of the hidden families under the paused clock and the live clock. Below the diagonal the model does worse live. On the realistic tier, Claude and GPT models with nearly equal paused scores differ widely live. Open circles: realistic tier in our own agent.

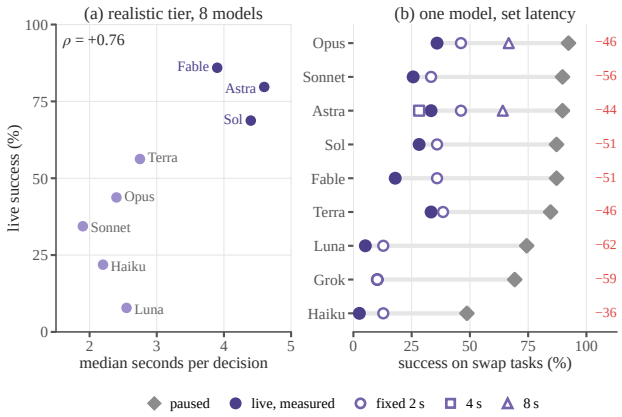

Figure 4. **Is it latency?** (a) Realistic tier: median seconds per decision against live success on event seeds, one point per model (ρ: Spearman). The three slowest models (dark) are the three most robust. (b) Controlled swap tasks, event seeds: one model charged a fixed time per decision instead of its measured latency. Red: distance to paused success at 2 s per decision.

execution control (E) switches the plate on while the robot decides.

Controlled tier. Five RoboMME-scene tasks with scripted object motions. LM-U: containers hiding cubes swap three or four times (SwapStream, InterruptSwap), or identical cubes trade places after a demonstration (RepickStream). LM-W: targets highlighted once (HighlightLate) or cubes revealed once (RevealLate) after hand-over. Decoys are feints.

Validation. Before any model runs, every family passes the scripted checks of L3 and L8 (Appendix B). A *privileged* policy that reads z_t and waits while an actor is busy solves every realistic seed under both clocks. A *stale* policy that fixes its plan at the first observation solves every seed paused but fails every realistic event seed live, and a policy that always assumes a change fails every semi-realistic decoy and null seed.

5. Experiments and Findings

The three assumptions behind paused evaluation become four questions, each answered by one finding:

(Q1) Does paused success predict live success? (Sec. 5.2, Finding 1)
(Q2) Is the live drop a matter of latency, so that a faster agent would avoid it? (Sec. 5.3, Finding 2)
(Q3) Is it an observation gap, so that re-observing before acting would avoid it? (Sec. 5.4, Finding 3)
(Q4) What remains for memory, and which memory operations break? (Sec. 5.5, Finding 4)

5.1. Setup

Models. We test 16 models from eight providers. Eight are Claude and GPT models: Claude Opus 5.5, Sonnet 5.5, Haiku 5.5 and Fable 5.1, GPT-6 Astra (medium reasoning effort) and Luna (low effort), and GPT-6.1 Sol and Terra (low effort). On the realistic tier they run inside Claude Code and Codex. The other eight, Gemini 4 Argon and Carbon, Grok 4.7, Amazon Nova 2 Lite, and the open-weight Qwen3-VL-235B [2], Gemma 3 27B [14], Kimi K2.5 [17] and Ministral 3 14B, have no such agent, so on the realistic tier they run in our own agent, which gives them the same prompt, tools and observations and charges the world clock only for the time of each model request. On the other two tiers every model sees the identical prompt and observation (Tab. 4). Output limits, effort settings and the models we could not run are listed in Appendix C.

Runs and statistics. Each instance is run once per condition (P, L, V). The commit check is run with Opus and Astra on the realistic and controlled tiers, and the offline probe for Opus, Sonnet, Haiku, Astra and Luna on all tiers (973 realistic items). Attempts that a service ended for usage limits or rejected as malformed were rerun with the same seed or excluded, and no number is estimated or imputed. Differences we call reliable have 95% paired bootstrap intervals over instances that exclude zero, and all intervals are listed in Appendix D.

5.2. Paused versus live success

Table 3 and Fig. 3 answer Q1. Of the 48 model–tier pairs, 46 lose success under the live clock, and none gains more than 2 points. On the realistic episodes in which the decisive state changes, the Claude and GPT models succeed in 97% on average under the paused clock and in 50% under the live clock. Every model succeeds in at least 94% paused, and live the same models succeed in between 8% (Luna) and 86% (Fable 5.1), so they lose between 8 and 88 points. The drop is caused by the changes, not by the clock itself: on null

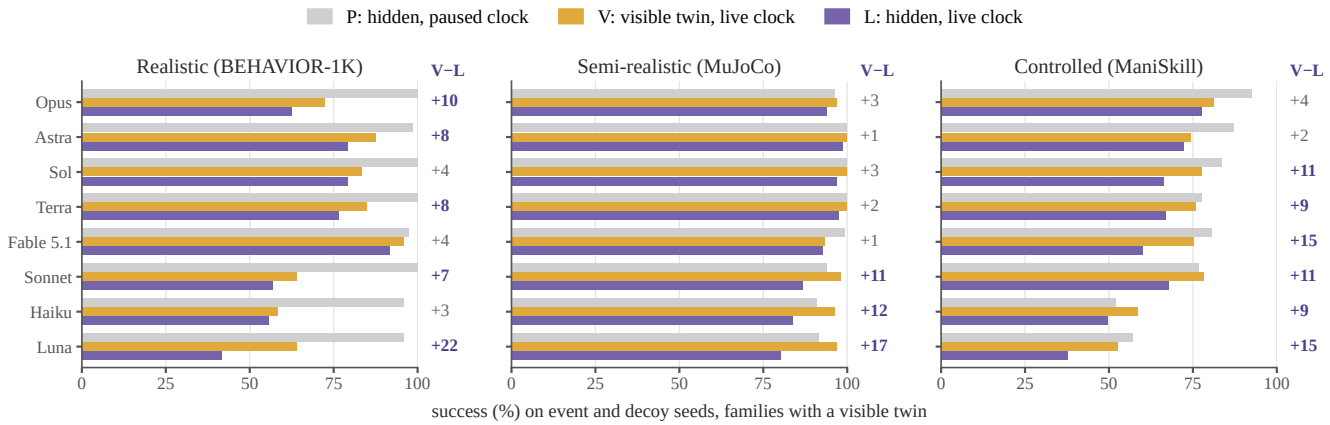

Figure 5. **Memory versus execution** (Eq. (4)). Success (%) on the families with a visible twin (event and decoy seeds) for the hidden family paused (P), the visible twin live (V) and the hidden family live (L). P−V is the cost of acting under the live clock with the change in view (execution), and the number at the right, V−L, is the extra cost of having to remember it (memory, bold if at least 5 points). Having to remember costs points on every tier. The same terms with the commit check are in Tab. 5.

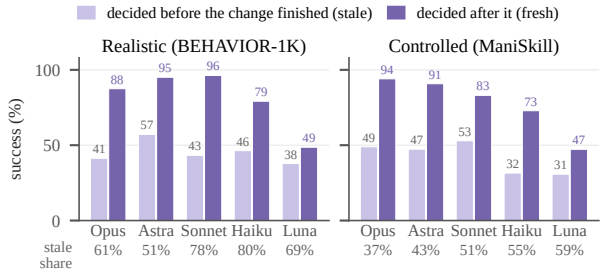

Figure 6. **Timing of the decisive decision** (hidden, live, event and decoy seeds). Success (%) of decisions taken on frames from before the other actor's change was complete (stale, σ=1) and after it (fresh). Below the axis: share of stale decisions. Stale decisions fail far more often for every model.

seeds, where nobody touches the decisive state, live success stays at 92–100% for every model. The ranking also changes. Astra and Fable 5.1 share the lowest paused score (94%) yet have the two highest live scores, and Opus, perfect when paused, falls to 44%. The eight other models, run in our own agent, show the same pattern: 76–98% paused against 2–80% live, and 78–98% on null seeds. On the controlled tier all 16 models lose 9 to 92 points. On the semi-realistic tier, the simplest scene, the five strongest models lose at most 5 points and the others up to 38.

Finding 1 (Q1). Paused success does not predict live success.

No. When a remembered object changes, the Claude and GPT models fall from 97% on average with the world paused to 50% with it running, and a model that never fails when paused succeeds in only 44%. Every paused score is at least 94%, yet they say nothing about the live ranking. With no change, live success stays at 92–100%: the loss comes from changes that a paused world never lets happen.

5.3. Latency

Figure 4 answers Q2 in two ways. Across models, slower models do not do worse. On the realistic tier the three slowest models, Astra, Sol and Fable 5.1 (3.9–4.6 s per decision against 1.9–2.8 s for the others), are the three most robust, and median latency and live success on event seeds are positively correlated (Spearman $\rho = +0.76$ over the 8 Claude and GPT models). The two slowest models we test, Gemini 4 Argon and Carbon (32 and 56 s per decision in our own agent), keep 80% and 73% live. Speed matters only when the time budget runs out: in the 65 s controlled episodes Carbon (34 s) times out in most live episodes and reaches 6%. Across the 16 models of the other tiers, $\rho = +0.30$ (controlled) and -0.17 (semi-realistic). Comparisons across models mix speed with capability, so we also set the latency of each model with the fixed clock κ_L on the three swap tasks (Fig. 4b). Charged 2 s per decision instead of its measured latency, eight of nine models gain 5 to 18 points on event seeds (Grok 4.7 none), yet every model stays 36 to 61 points below its paused success. A slower clock can even help: at 8 s Opus reaches 67% and Astra 64%, against 46% for both at 2 s. The lead time λ of Eq. (2) shows why speed alone does not help: a fast decision often falls inside a change that is still under way. On realistic event seeds Astra's decisive observation comes a median 1.9 s *after* the change ends, and those of Opus, Sonnet and Haiku 5.3–8.6 s before it.

Finding 2 (Q2). A faster agent would not avoid the drop.

No. Slower models do no worse, and even at 2 s per decision every model tested stays at least 36 points below its paused success. Speed helps only if the decision waits for the change to finish.

Table 5. **Commit check** (Eq. (3)). Success (%) under the live clock on event and decoy seeds, without (min.) and with the check, and the memory term $\Delta_{\text{mem}} = S_{\mathsf{V}} - S_{\mathsf{L}}$ with the check. Realistic: the 13 scenes where the check applies (hidden n=104, twins n=56). Controlled: the three swap tasks (n=80). Re-observing closes the realistic gap but not the controlled one.

		Hidden (L)		Visible twin (V)		
Tier	Model	min.	check	min.	check	Δ_{mem}
Realistic	Opus	61.5	97.1	64.3	100.0	0.0
	Astra	80.8	95.2	83.9	100.0	1.8
	Sol	77.9	92.3	78.6	100.0	1.8
	Terra	68.3	87.5	80.4	92.9	−1.8
	Fable 5.1	86.5	96.2	94.6	100.0	3.6
Controlled	Opus	62.5	80.0	68.8	95.0	15.0
swaps	Astra	61.2	81.2	65.0	100.0	18.8
	Sonnet	58.8	67.5	73.8	98.8	31.2
	Haiku	40.0	37.5	58.8	72.5	35.0
	Luna	33.8	41.2	52.5	81.2	40.0
	Sol	51.2	72.5	68.8	98.8	26.3
	Terra	53.8	73.8	68.8	100.0	26.2
	Fable 5.1	38.8	26.2	68.8	83.8	57.5
	Grok 4.7	23.8	20.0	53.8	32.5	12.5

5.4. Observation gap

Acting on old frames. Figure 6 splits the live episodes by σ, *i.e.* by whether the decision that settles the task was made on frames from before the change ended (stale) or after it (fresh). On the realistic tier 51–80% of these decisions are stale, and they succeed in 38–57% of episodes, against 79–96% for fresh decisions (49% for Luna). On event seeds, Opus succeeds in 1 of 34 stale and 21 of 23 fresh episodes. The controlled tier shows the same split (31–53% against 47–94%).

The frames are read correctly. The offline probe (Tab. 13) asks each model about the decisive state using exactly the frames it had. Opus, Sonnet and Haiku answer correctly in 76–89% of their failed realistic episodes, and almost all of those are stale decisions. Opus and Astra show the same profile on the controlled tier (29 of 30 and 30 of 37). Rarer failure modes of the weaker models: Fig. 8.

Re-observing before acting. If stale commitment is an observation gap, closing the gap should help. The commit check does this simply: just before a command runs, the robot looks at the target once more, and if it has changed, the agent decides again on the new frames. Table 5 confirms it. On the 13 realistic scenes where the commit check applies, hidden live success rises to 88–97% for all five models tested (Opus from 62% to 97%, Astra from 81% to 95%), the memory term stays within 4 points of zero, and for Opus and Astra nearly every executed decisive command becomes fresh. On the controlled swap tasks the check raises hidden live success by 9 to 21 points for the five strongest models, but not for Haiku, Fable 5.1 and Grok 4.7 (3 to 13 points lower).

Finding 3 (Q3). Agents see the change but act on old frames.

Yes. Most failures are decisions taken on frames recorded before the other actor finished, by models that read those frames correctly. Looking once more before acting recovers most of the realistic gap, with no training.

5.5. Memory

Memory versus execution. Figure 5 applies Eq. (4) to the families with a visible twin. With the decisive state kept in view, the same change at the same time costs less on every tier. Δ_{mem} is positive for all eight models on all three tiers, from +1 to +22 points. On the controlled swap tasks the commit check (Tab. 5) lifts the twins of the five strongest models to 95–100% but their hidden tasks only to 68–81%: when identical objects are swapped several times, cancelling a command forces a new decision but cannot tell the agent which object is which. On the realistic tier the remembered object changes once, so one more look settles it; on the controlled swaps it changes several times, so the agent must have followed every swap. The memory term therefore *grows* with the check, to 15–31 points against 4–18 without it, and with the check it is positive for all nine models tested (12–58 points, every 95% interval above zero; Tab. 11).

Updates break first. By memory class (Fig. 7), updates lose the most. On the realistic LM-U scenes, where a remembered object is swapped, hidden or carried away, live success is 75% for Astra, 25% for Opus and 2–18% for Sonnet, Haiku and Luna, against 91–100% paused; on the controlled swaps Opus and Astra fall from about 90% to about 35%. Writes are solved live by the stronger models on every tier.

Finding 4 (Q4). Memory still fails on repeated swaps.

Updates. Having to remember a change rather than see it costs every model points on every tier (up to 22). When identical objects are swapped several times, looking again does not help: all nine models tested still lose at least 12 points. Keeping track of which object is which while the world moves is the core of live memory, and RoboLiveMem isolates it.

6. Conclusion

We built RoboLiveMem to ask whether robot memory is solved when the world does not wait. Frontier models that succeed in 97% of our changing episodes on average when the world is paused succeed in only 50% once it keeps moving, mostly because they decide before a change they are watching is over. Looking again before acting recovers

most of this gap; what remains, tracking identical objects through repeated swaps, is the memory problem that robots in shared spaces still have to solve.

Limitations. All scenes are simulated, other actors follow scripts, the people in the realistic tier have no articulated arms, and latency depends on the load of public model services. Realistic-tier results should be compared within models that run in the same kind of agent.

LLM Usage Statement

We used ChatGPT and Claude to assist with language polishing and improving the readability of this manuscript, and to give feedback on the manuscript. The vision-language models evaluated in this paper are the objects of study of our benchmark (Sec. 5.1). The authors are fully responsible for all research ideas, experimental results, and claims presented in this paper.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022. 3

[2] Shuai Bai et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 6

[3] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025. 3

[4] Bingao Chen, Haoquan Fang, and C. Karen Liu. Can Astra solve RoboMME without breaking the bank? https://bingaochen.github.io/Astra-on-RoboMME, 2026. 2

[5] Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, Jonathan Tremblay, and Valts Blukis. VoLo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026. 3

[6] Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026. 2

[7] Egor Cherepanov, Nikita Kachaev, Alexey K. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *International Conference on Learning Representations*, 2026. 2

[8] Egor Cherepanov, Nikita Kachaev, Aleksandr I. Panov, and Alexey K. Kovalev. MIKASA-Robo-VLA: Benchmarking memory in VLA models for long-horizon manipulation. *arXiv preprint arXiv:2610.00604*, 2026. 2

[9] Suhwan Choi, Jaeyoon Jung, Sungkyung Kim, Yunsung Lee, and Youngjae Yu. PonderPounce: A pretrained MLLM as an episode context engine for robot control. *arXiv preprint arXiv:2608.24115*, 2026. 3

[10] Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2603.04639. 2

[11] Perry Dong, Kuo-Han Hung, Dorsa Sadigh, and Chelsea Finn. Reinforcement learning for real-time vision-language-action policies. *arXiv preprint arXiv:2609.18207*, 2026. 3

[12] Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026. 2, 3

[13] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026. 3

[14] Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. 6

[15] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, 2022. 3

[16] Jack B. Jedlicki, Amish Sethi, and Heng Yang. Can a frontier VLM serve as a robot policy for memory-dependent manipulation? *arXiv preprint arXiv:2610.17350*, 2026. 2

[17] Kimi Team. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. 6

[18] Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, Donglin Wang, Yan Wang, and Haoang Li. RoboMemArena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026. 2

[19] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, 2022. 2, 5

[20] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, 2023. 3

[21] Jack Lu, Yurong You, Boyi Li, Yulong Cao, Thomas Tian, Chaowei Xiao, Xiangyu Chen, Wenjie Luo, Boris Ivanovic, Marco Pavone, and Yan Wang. TacitVLA: Reasoning vision-

language-action models without reasoning latency. *arXiv preprint arXiv:2610.17540*, 2026. 3

[22] MLL Lab. Embodied first exam: The first exam for code agents in the physical world. https://github.com/mll-lab-nu/embodied-first-exam, 2026. Released 2026-10-19. 2, 3

[23] Soumil Rathi. RoboMME-Interference: Benchmarking robot memory under interference. *arXiv preprint arXiv:2606.22338*, 2026. 2

[24] Sooyoung Ryu, Yinpei Dai, Yilun Du, Chelsea Finn, and Yuejiang Liu. MEMI: Self-improving robot memory through inverse reasoning. https://memi-robot.github.io/, 2026. 2

[25] Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval. In *International Conference on Learning Representations*, 2026. 2

[26] Qixin Sun, WenLong Gao, Xiansheng Chen, Zhongyuan Wang, Si Liu, and Pengwei Wang. LTM: Latent transition based in-context memory for vision-language-action models. *arXiv preprint arXiv:2610.21362*, 2026. 2

[27] Jingyu Tang, Xu Yang, Zhehao Zhang, Zheng Wang, Qi Cheng, Haozhe Zhao, Manling Li, Wenxuan Li, Xide Xia, and Jiaxin Pei. Realtime-bench: Can frontier models solve real-time interactive problems?, 2026. https://realtimeagents.org. 2, 3

[28] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. In *Robotics: Science and Systems*, 2025. 5

[29] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 5

[30] Vals AI. Robo-bench. https://www.vals.ai/benchmarks/robo_bench, accessed 2026-10-22, 2026. 2, 3

[31] Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. *arXiv preprint arXiv:2511.04898*, 2025. 3

[32] Feiyang Wu, Weirui Ye, Haonan Chen, Haoyu Chen, Jiayu Zhou, Yandong Ji, Yuejiang Liu, and Yilun Du. How hard are current robot memory benchmarks for agentic harnesses? https://memout-robot.github.io/, 2026. 2

[33] 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

[34] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *International Conference on Machine Learning*, 2025. 3

[35] Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. SimpleMemVLA: A simple but effective native-video memory for vision-language-action models. *arXiv preprint arXiv:2609.05533*, 2026. 2

[36] Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop comparing LLM agents without disclosing the harness. *arXiv preprint arXiv:2605.23950*, 2026. 3

[37] Yuyou Zhang, Yunbei Zhang, Miao Li, Janet Wang, Zijian Jin, Shilong Liu, and Ding Zhao. Simple agentic memory for generalist robot policies. *arXiv preprint arXiv:2609.36595*, 2026. 2

ROBOLIVEMEM: Is Robot Memory Solved When the World Does Not Wait?

Supplementary Material

A. Task Families

Tables 6 and 7 describe every family. All variants of a family start from the same scene; the variant is drawn from the seed (event, decoy, null in turn). Instructions may say who shares the workspace ("Your son is setting this table this morning") identically for all seeds, and never whether or when anything happens.

B. Clock-Sensitivity Checks

Table 8 states the conditions every instance must meet. Before any model is run, every family passes a scripted check with three policies: a *privileged* policy that reads the true state and waits while another actor is busy; a *stale* policy that fixes its plan from the first observation, charges 3 s per decision and never re-checks; and, on the semi-realistic tier, an *always-change* policy that assumes the change happened. A family passes if the privileged policy succeeds under both clocks, the stale policy succeeds paused and fails live on event seeds while passing decoy and null seeds, and fixed policies cannot win.

Realistic. 528 scripted episodes, 16 hidden scenes and 9 visible twins: the privileged policy solves every seed under both clocks; the stale policy solves every seed paused and, live, fails every event seed and passes every decoy and null seed, except the two-step decoys of *Car keys* and *Gauge in bins*, which hide the item twice and are real changes for a stale policy.

Semi-realistic. Stale policy at 3 s per decision, event seeds: mugs 0/2, covered dishes 0/2, residual heat 0/2, knife 1/2 and shown jar 1/2 (a random guess among two covers or three jars), heat duration 0/4, sugar count 0/4; all paused, decoy and null seeds succeed. The latency-aware privileged policy succeeds on every seed under both clocks. The always-change policy (run on mugs, covered dishes and heat) fails every decoy and null seed.

Controlled. Episodes 0–39 per task. Stale policy at 3 s, event seeds: SwapStream 6/13, RepickStream 3/13, InterruptSwap 9/13 (a swap may spare the target); a frame-blind policy on HighlightLate and RevealLate 0/13; decoy and null seeds pass. The latency-aware policy solves every event seed of SwapStream and InterruptSwap at 3 s.

C. Harnesses and Settings

Table 9 lists the agent, the budget and the output-token limit of every model.

Models not evaluated. Kimi K3 was not evaluated because its service rejected nearly every request for quota, Llama 4 Maverick and Mistral Large 3 accept at most three images per request, and GLM 5.3 accepts none.

Prompt. Each realistic prompt has the task, the clock rules and the tool list. For *Grandpa's mug*, live clock:

```
TASK (Grandpa's mug): Your robot has just
spent the last 10 seconds sorting cutlery on
the side cart; its head camera kept recording
the whole time, and your first observation
includes that recording. When your robot
started wiping, Grandpa's mug stood at mark 1.
The three mugs look identical, but Grandpa's
has his heart medicine dissolved in it. Bring
Grandpa's mug to his tray (mark 6). Your son
is setting this table this morning. You have
2 manipulation actions (pick, place).
CLOCK: LIVE clock: the world keeps running
while you think. Other robots, machines and
people keep moving. Every response includes
the robot camera frames recorded since your
previous request, so you can see what happened
while you were thinking. Your robot never
reaches into a place another robot, machine or
person is handling right now: such a command
first waits until they are done (world time
runs).
Tools: robot_spec, robot_observe,
robot_frames(t_from, t_to), robot_act(skill,
mark), robot_wait(seconds), robot_done. [...]
A skill acts on whatever is at the numbered
place when it runs, not on what was there when
you last looked.
```

The paused prompt replaces the clock paragraph with its paused counterpart. The agent runs in an empty working directory with a random name, so neither the scene nor the seed can be read from the environment.

Commit check. Realistic tier: a 128×128 window around the target mark; after a 5×5 blur, a change is more than 2% of pixels whose grey level differs by more than 24. Controlled tier: at least 20 pixels in the target region whose largest channel difference exceeds 40. A cancelled command uses no manipulation budget.

Probe. Each item gives the model the images it received before its decisive decision, in order, with the tool text, and a multiple-choice question about the decisive state at that time; the answer comes from the simulator's privileged log. The realistic tier has 973 items, all answered.

D. Additional Results

Live drop and its two terms. Table 10 gives the live drop of the eight Claude and GPT models with its execution and memory terms, Fig. 10 their 95% intervals, and Tab. 11 the memory term with the commit check.

Memory classes. Figure 7 groups the families by memory class: updates (U) lose the most on the realistic and controlled tiers. Table 12 shows the update drop on every

Table 6. **Realistic tier (BEHAVIOR-1K).** What changes in each scene. U/W: LM-U/LM-W. On null seeds the actor comes and goes, or the machine idles, without changing anything. Twin: the cue that keeps the state in view (–: none).

Scene		Actor	Event (the state to remember)	Decoy	Twin
Grandpa's mug	U	son	the medicine mug (1 of 3 identical) moved away, a look-alike put in its place	only a look-alike moved	red sticker
Express tote	U	picker	the express tote (1 of 3 identical) moved away, a look-alike put in its place	only a look-alike moved	red sticker
Car keys	U	teenager	keys put into 1 of 3 lidded baskets	put in one, later moved	last lid ajar
Gauge in bins	U	technician	gauge put into 1 of 3 lidded bins	put in one, later moved	last lid ajar
Mom's tea	W	daughter	Mom's tea gets its spoon of sugar	the other cup sweetened	token at cup
Catalyst dose	W	technician	flask A gets its dose	flask B dosed	token at flask
Put away	U	helper robot	apple put into closed bin k	bin k opened, apple returned	lid left off
Anna's mug	U	helper robot	Anna's mug (1 of 2 identical) moved, a board put in its place	the other mug moved	–
Vacuum and ball	U	robot vacuum	ball pushed under sofa k	drives to the ball and back	–
Salted soup	W	helper robot	soup salted	same motion, pepper mill	–
Rotary reject	U	rotary table	table indexes 1–5 steps with the reject in a nest	full turn	–
Conveyor lanes	U	conveyor	order tote routed to 1 of 3 lanes	tote re-scanned, out and back	–
Press lamp	W	press	cycle ends: lamp lit for 6 s	vents twice, no lamp	lamp stays lit
Parcel locker	U	courier robot	parcel stored in identical locker k, door closed	locker k opened and closed empty	door left open
Kitting partner	W	partner robot	1–3 covered totes kitted	same lids opened and closed	–
Cloche pass	U	expo robot	two cloches moved, one over the vegetarian plate	the other two moved	–

Table 7. **Semi-realistic (MuJoCo kitchen) and controlled (ManiSkill) families.** On null seeds the hand passes far (semi-realistic) or nothing moves (controlled); in Shown jar, HighlightLate and RevealLate the cue is then given only at hand-over. T: LM-T; E: execution control.

Family		Event	Decoy	Twin
Semi-realistic: a person at the counter; UR5e with pick, place and press on marked places				
Mugs	U	two identical mugs swapped	both nudged, put back	target mug red
Covered dishes	U	two covered dishes swapped while the robot holds the salt	other cover lifted, put back	apple's cover blue
Residual heat	W	plate switched on, then off, before the robot places a plastic bowl	hand reaches, never presses	plate stays on
Shown jar	W	cook's hand over 1 of 3 jars once, for 1 s, after hand-over	shown at hand-over, later a sweep	hand stays
Knife	W	knife slid under 1 of 2 covers	knife lifted, put back	knife moved in view
Heat duration	T	plate on for 4 s (hot after 3 s)	on for 1 s	plate stays on
Sugar count	T	2 cubes dropped into the tea (enough)	1 cube (not enough)	cubes on the saucer
Heat (E)	E	plate switched on while the robot decides where to place the bowl	reaches twice, never presses	plate stays on
Controlled: scripted motions on RoboMME scenes; Franka Panda with RoboMME's skills; 20 Hz control				
SwapStream	U	containers hiding cubes swap 3 or 4 times	as many feints (move 35%, return)	tag on each cube's container
InterruptSwap	U	as SwapStream, on the InterruptSwap scenes	feints	tags
RepickStream	U	identical cubes swap after the video that showed which to re-pick	feints	tag on the picked cube
HighlightLate	W	targets highlighted once after hand-over	at hand-over, repeated later	tag from the highlight
RevealLate	W	containers lift once after hand-over, showing the cubes	at hand-over, repeated later	tags from the reveal

Table 8. **Formal statement of the instance conditions** of Tab. 2. The controlled tier relaxes L4 by design; the checks are described below.

Condition	Formal statement
L1 memory needed	z_{t^c} cannot be read from o_{t^c}
L2 own evidence	the change is a continuous motion in $\{o_t\}_{t\in[\tau_s,\tau_e]}$
L3 scheduled	τ_s: first idle time after $t_{\text{ho}}+u$, $u\sim U(0.3,2.5)$ s, independent of the agent
L4 visible cause	a person, robot or machine with a reason to act
L5 no hints	no text says whether or when anything happens
L6 no fixed winner	event $z^+\neq z^-$; decoy: motion with $z^+=z^-$; null: no motion
L7 controls	a visible twin keeps z_t identifiable
L8 solvable	a privileged policy solves every seed under both clocks

tier, including Opus and Astra on the controlled swaps (92% and 90% paused, 36% and 33% live). The controlled write tasks illustrate why we read LM-W through the twin: paused, an agent that acts at once never lets the highlight or reveal appear (Sonnet 15%), while live it often does (73%).

On the semi-realistic tier few decisions are stale (Tab. 14), and the smaller models' failures are fresh decisions: they had the finished change in their frames and did not use it, which matches the probe (Tab. 13) and the memory term of Fig. 5.

Failure modes. Figure 8 combines timing and probe into a taxonomy. Stale commitment dominates the realistic and controlled tiers for every model. On the kitchen tier failures are fresh. Sonnet saw the finished change (probe right in 21 of 22 failures, none stale) and still acted on the old memory, a failure to *integrate* the update, while Haiku and Luna *misread* it (probe wrong). Phantom changes, failures on no-change seeds, are rare on the realistic tier (at most 5 of 64 null seeds) and more frequent on the controlled tier.

Per scene. Figure 9 shows that the update scenes separate

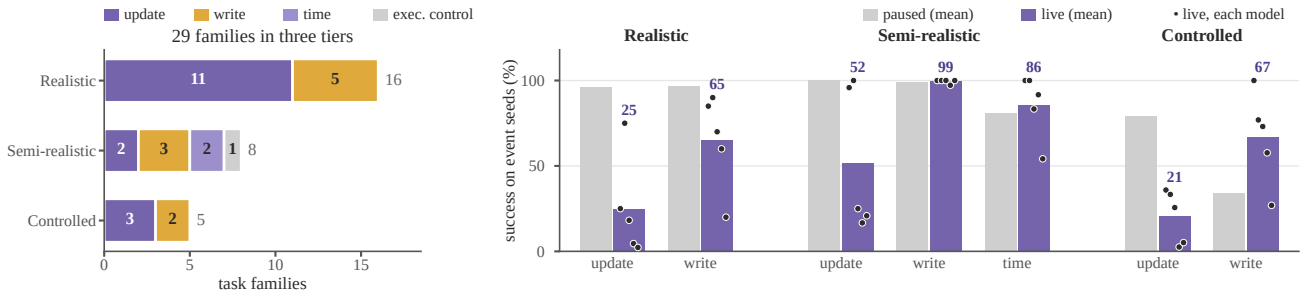

Figure 7. **Which memory operations break.** Left: the 29 families by tier and memory class. Right: success (%) on the event seeds of each tier and class, paused and live, for the five frontier models (bars: mean, dots: models). Updates (U) lose the most on the realistic and controlled tiers.

Table 9. **Harness settings.** Top: agent and budget. Bottom: output-token limit per model (default: the agent's own). Gemini runs through the Gemini API and every other API model through Amazon Bedrock.

	Realistic	**Semi-realistic / controlled**
Agent	Claude Code 2.1.314 or Codex with MCP robot tools; other models: our API agent, same tools as JSON calls	one API call per decision
Frames	current view with marks, contact sheet since the last request, any window on request	$\leq$30 frames since the last decision (2 Hz), newest with marks (controlled: also unmarked; demo frames on turn 1)
History	the agent's own context (API agent: last 12 images)	semi: text log of commands and notes; controlled: conversation, older images replaced by a placeholder
Budget	2–3 manipulation actions, 300 s	240 s, 12 decisions / 65 s, 25 decisions
Opus, Fable 5.1	default	Opus 4,096; Fable 5.1 16k (semi), 4,096 (controlled)
Sonnet, Haiku	32k	4,096
GPT (Codex)	default; Astra medium effort, Luna, Sol, Terra low	Astra 16k; Luna 16–20k; Sol, Terra 16k; same effort
Gemini 4	16k	16k
Grok 4.7	8k	16k (semi), 8k (controlled)
Kimi K2.5	4,096	4,096 / 8k
Qwen3-VL, Gemma 3, Nova 2 Lite, Ministral 3	4,096	4,096

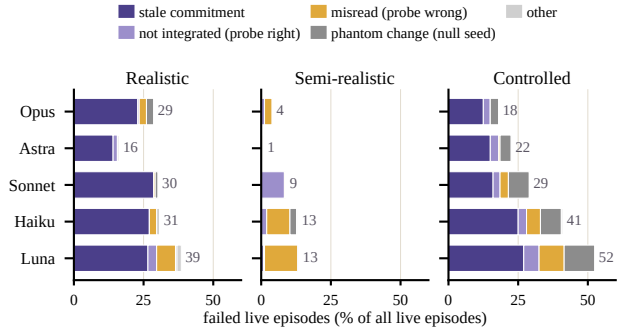

Figure 8. **Failure taxonomy** of live hidden episodes (all seeds), five frontier models. Stale: decided before the change was complete. Not integrated and misread: fresh decision with the probe right and wrong. Phantom: failure on a null seed. Stale commitment dominates the realistic and controlled tiers.

the models most: Astra solves 10 of the 11 update scenes at least half of the time live, Opus and Sonnet 3, Haiku and Luna none. Parcel locker, where a courier robot puts the parcel into one of six identical lockers and closes the door, defeats every model live.

Data integrity. Every number comes from a completed episode; nothing is estimated or imputed. Attempts that ended because a model service refused requests for usage limits (25 on the realistic tier) were rerun with the same seed and excluded. Before the main run we fixed a renderer lag that could hide the last discrete change from the head camera and moved two marks that occluded the decisive object; all affected episodes were discarded and rerun. The visible twins of the two controlled cue tasks first carried a sentence explaining their tags; their minimal-harness runs were repeated without it and only those are reported. Their commit-harness runs keep the sentence, which is why Tab. 5 reports the controlled commit check on the three swap tasks only.

Table 10. **Live drop and its two terms** for the eight Claude and GPT models, in points: $\Delta = S_P - S_L$ (red) over all seeds of the hidden live-memory families; on the families with a visible twin (event and decoy seeds) the execution term $S_P - S_V$ and the memory term $S_V - S_L$. Bold: the 95% paired bootstrap interval over instances (2,000 resamples) excludes zero; the intervals are in Fig. 10. Last column: median seconds per decision (hidden, live). With the commit check: Tab. 11.

	Model	Δ	P−V	V−L	s/dec.
Realistic	Opus 5.5	**+27.6**	**+27.8**	**+9.7**	2.4
	Astra	**+13.5**	**+11.1**	**+8.3**	4.6
	Sonnet 5.5	**+29.2**	**+36.1**	+6.9	1.9
	Haiku 5.5	**+28.1**	**+37.5**	+2.8	2.2
	Luna	**+35.4**	**+31.9**	**+22.2**	2.6
	Sol	**+17.2**	**+16.7**	+4.2	4.4
	Terra	**+21.4**	**+15.3**	+8.3	2.8
	Fable 5.1	**+10.4**	+1.4	+4.2	3.9
Semi-real.	Opus 5.5	+1.6	−0.6	+3.0	4.2
	Astra	+0.8	+0.0	+1.2	2.6
	Sonnet 5.5	+3.6	**−4.2**	**+11.3**	2.6
	Haiku 5.5	+5.2	**−5.4**	**+12.5**	3.1
	Luna	**+6.7**	**−5.4**	**+16.7**	2.1
	Sol	**+2.0**	+0.0	**+3.0**	2.7
	Terra	**+1.6**	+0.0	**+2.4**	2.1
	Fable 5.1	**+4.4**	**+6.0**	+0.6	6.3
Controlled	Opus 5.5	**+9.5**	**+11.3**	+3.8	4.2
	Astra	**+9.5**	**+12.8**	+2.3	2.6
	Sonnet 5.5	+6.5	−1.5	**+10.5**	3.1
	Haiku 5.5	+1.0	−6.8	+9.0	3.6
	Luna	**+13.0**	+4.5	**+15.0**	2.4
	Sol	**+10.5**	+6.0	**+11.3**	2.7
	Terra	+6.5	+1.5	**+9.0**	2.2
	Fable 5.1	**+13.0**	+5.3	**+15.0**	5.9

Table 11. **Memory term with the commit check**, $S_V - S_L$ in points on event and decoy seeds (realistic: the 13 scenes where the check applies; controlled: the three swap tasks), with the bounds of its 95% paired bootstrap interval over instances.

Tier	Model	Δ_{mem}	lower	upper
Realistic	Opus	0.0	0.0	0.0
	Astra	1.8	0.0	5.4
	Sol	1.8	0.0	5.4
	Terra	−1.8	−7.1	3.6
	Fable 5.1	3.6	0.0	8.9
Controlled swaps	Opus	15.0	6.2	25.0
	Astra	18.8	11.3	27.5
	Sonnet	31.2	21.3	41.2
	Haiku	35.0	22.5	47.5
	Luna	40.0	28.7	52.5
	Sol	26.3	16.2	36.3
	Terra	26.2	17.5	36.3
	Fable 5.1	57.5	46.2	68.8
	Grok 4.7	12.5	1.2	23.8

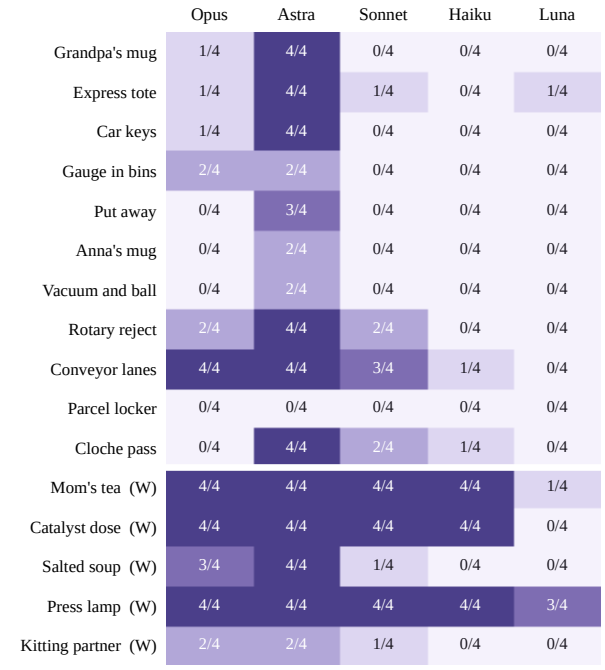

Figure 9. **Per-scene live success on the realistic tier** (hidden, event seeds; successes of 4). Paused, every cell is 3/4 or 4/4 except Astra on Rotary reject (1/4). W: write scenes; the others are updates.

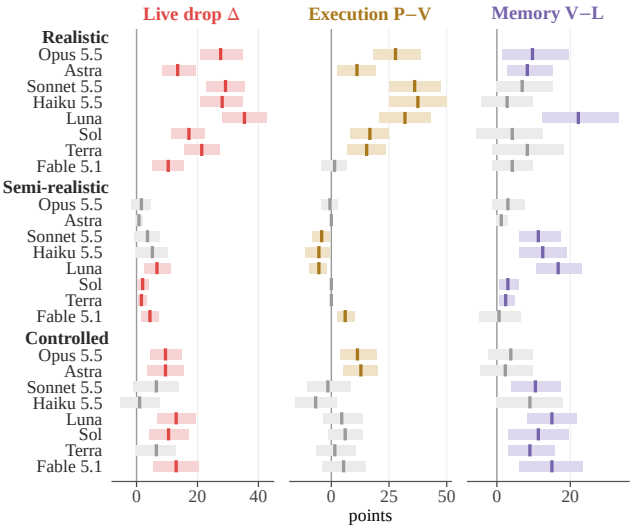

Figure 10. **95% paired bootstrap intervals of Tab. 10** over instances (2,000 resamples), in points. Bar: the interval; tick: the estimate. Gray: the interval includes zero.

Table 12. **Update and write families per tier** (hidden, event seeds). Success (%) under the paused clock (P) and the live clock (L). –: not run.

	Realistic				Semi-realistic				Controlled			
	Update (LM-U)		Write (LM-W)		Update (LM-U)		Write (LM-W)		Update (LM-U)		Write (LM-W)	
Model	P	L	P	L	P	L	P	L	P	L	P	L
Claude Opus 5.5	100	25	100	85	100	96	100	100	92	36	85	100
GPT-6 Astra	91	75	100	90	100	100	100	100	90	33	54	77
Claude Sonnet 5.5	98	18	100	70	100	25	100	100	90	26	15	73
Claude Haiku 5.5	98	5	90	60	100	17	94	97	49	3	12	58
GPT-6 Luna	95	2	95	20	100	21	100	100	74	5	4	27
GPT-6.1 Sol	100	66	100	75	100	100	100	100	87	28	54	77
GPT-6.1 Terra	100	50	100	70	100	88	100	97	85	33	27	73
Claude Fable 5.1	91	86	100	85	100	83	100	100	87	18	54	100
Gemini 4 Argon	95	74	100	94	100	91	100	94	92	31	42	23
Gemini 4 Carbon	93	66	95	90	100	29	100	78	100	0	96	15
Grok 4.7	95	23	100	60	100	12	97	100	69	10	46	46
Nova 2 Lite	93	7	79	30	83	29	53	64	10	0	12	4
Kimi K2.5†	91	0	90	30	100	17	75	78	26	5	19	15
Qwen3-VL-235B†	98	2	100	25	100	8	44	44	22	3	12	15
Gemma 3 27B†	98	0	65	5	92	25	42	28	33	8	4	8
Ministral 3 14B†	72	5	85	30	58	25	64	42	10	3	19	8

Table 13. **Offline probe on failed live episodes** (hidden, event and decoy seeds). Right: given the same frames without time pressure, the model answers the decisive question correctly. Stale: of those, episodes decided before the change was complete.

	Realistic				Controlled			
Model	fails	right	%	stale	fails	right	%	stale
Opus 5.5	50	38	76	37	30	29	97	24
Astra	29	16	55	13	37	30	81	24
Sonnet 5.5	56	50	89	50	43	26	60	21
Haiku 5.5	57	46	81	46	66	28	42	22
Luna	69	30	43	23	80	32	40	21

Table 14. **Decision timing, all tiers** (hidden live, event and decoy seeds): share (%) of stale decisions and success (%) of stale (s) and fresh (f) decisions.

	Realistic			Semi			Controlled		
Model	stale	succ. s	succ. f	stale	succ. s	succ. f	stale	succ. s	succ. f
Opus	61	41	88	12	100	93	37	49	94
Astra	51	57	95	13	100	98	43	47	91
Sonnet	78	43	96	13	100	80	51	53	83
Haiku	80	46	79	3	100	81	55	32	73
Luna	69	38	49	12	86	69	59	31	47