INTERACTIVE VISUAL GYM: EVALUATING VISUAL ASSISTANTS WITH SIMULATED HUMAN USERS IN INTERACTIVE WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual assistants that support users during ongoing physical tasks require evaluation in closed loop, where assistant responses can change subsequent user behavior and task outcomes. Existing benchmarks largely rely on offline replay and therefore cannot capture these effects. We introduce **Interactive Visual Gym**, a framework for closed-loop evaluation and training of visual assistants through joint simulation of executable task environments and human users. We contribute three simulators that recreate everyday tasks and human-aligned action spaces across cooking, household manipulation, and mobile-interface domains, together with a persona-conditioned user simulation harness that collaborates with the assistant while incorporating controlled execution errors and other out-of-plan behaviors. We instantiate the framework as the **I**nteractive **V**isual **E**valuation **(IVE)** benchmark with 315 interaction epiodes. Evaluating state-of-the-art VLMs as visual assistants reveals substantial limitations: the best-performing model, GPT6-Astra, completes only 44.2% of tasks within the time limit, exhibits significant weaknesses in factual grounding and actionable guidance, and is often too slow to prevent errors in time. We further show that the environments provide useful learning signals: in a proof-of-concept GRPO experiment, closed-loop training improves both task success and interaction behavior.

1 INTRODUCTION

Building systems that collaborate effectively with people is a longstanding goal of interactive machine learning (Amershi et al., 2014). Recent advances in LLM agents have enabled collaboration on digital tasks, including web research and software development, through dialogue, preference adaptation, and coordinated action (Shao et al., 2026; Sun et al., 2026). More recently, extending such collaboration to everyday physical tasks has drawn growing interest, motivating *visual assistants* that observe a video stream of a user's ongoing activities and provide guidance (Cui et al., 2026; Thinking Machines Lab, 2026; Yao et al., 2026). For example, an assistant monitoring a procedural physical task like cooking can help plan the next step, detect execution errors, and guide recovery if a mistake happens.

Developing and evaluating interactive agents requires closed-loop rollouts that capture how each interaction changes subsequent user behavior and environment state, enabling assessment of its contribution to task completion. For digital agents, existing websites and software provide readily available tasks and executable environments, which can be paired with simulated users to support such rollouts (Zhou et al., 2024; Yao et al., 2024; Shao et al., 2026). Building an equivalent testbed for visual assistance is substantially more difficult. First, physical tasks and their environments are not readily available in a digital form; they must be reconstructed with visual observations, underlying world state, and an action space that supports human task execution. Second, visual assistance places substantially greater demands on user simulation. In most digital settings, the model executes actions while the user primarily clarifies goals and preferences. In physical activities, the user is the acting agent, and the model influences execution through communication. The simulated user must plan, act, make plausible mistakes, while interpreting guidance and deciding how to respond. These simulation challenges have limited visual-assistant training and evaluation to replaying recorded interactions and matching reference responses (Liu et al., 2026; Kundu et al., 2026; Ran et al., 2026), where the user's actions and responses remain unchanged regardless of the assistant's guidance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

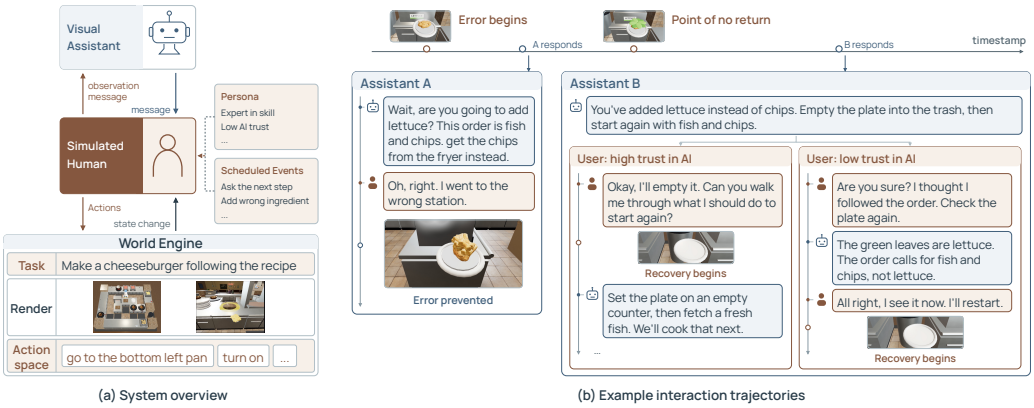

Figure 1: **Interactive Visual Gym.** (a) System overview: The visual assistant under evaluation is tasked with guiding a simulated user to complete a physical task in a visual environment. (b) Examples showing how different task execution can be simulated based on different assistant response.

We present **Interactive Visual Gym**, the first framework for closed-loop evaluation of visual assistants through joint simulation of task environments and human users. Our framework has three components:

First, we recreate everyday tasks across cooking, household manipulation, and mobile interface use in executable visual environments. These environments render visual observations from the user's perspective. They use scene graphs to represent world state and track task progress independently of variations in action execution. They expose a human-aligned semantic action space for simulated users to operate in, translating instructions such as "put the pan on the stove" into executable operations.

Second, we design a harness that turns a state-of-the-art VLM into simulated users that plan and act within these environments while interacting with the assistant and revising their actions in response to its guidance. The harness supports controlled insertion of common human out-of-plan events at specified task states, including repeating, forgetting, or incorrectly executing a step, changing plans, and asking questions. We further model diverse user personas through attributes such as skill level, trust in AI, and interruption preference. These attributes shape how users perform tasks and respond to guidance differently, enabling evaluation of the assistant's adaptation to different users.

Third, simulation provides execution traces that support comprehensive evaluation of an assistant, covering task completion, error detection, and interaction quality. This captures how assistance affects task execution and user experience, which offline replay cannot assess.

As an initial instantiation of our framework, we construct **I**nteractive **V**isual **E**valuation **(IVE)**, a benchmark with 150 cooking, 75 household, and 90 mobile-interface interaction episodes to evaluate state-of-the-art VLMs as visual assistants. In each episode, the assistant is tasked to support a simulated user with a sampled persona through task execution, and needs to deal with a set of scheduled out-of-plan events. The dynamic rollout allows us to measure task completion under assistance and assess individual responses against the underlying world state. The best-performing model, GPT-6 Astra, completes only 44.2% of tasks within the time limit, and also shows severe deficits in factual grounding and actionable guidance, revealing a substantial gap toward reliable interactive assistance.

Finally, we demonstrate that our environments can support end-to-end reinforcement learning of multimodal assistants through a proof-of-concept GRPO experiment with Qwen3.5-4B (Qwen Team, 2026a) as the assistant and Gemini 3.7 Flash (Google, 2026b) as the simulated novice user on CookSim. After 12 RL updates, the assistant's in-time task success improves by $\sim 9\%$ on both training and held-out validation scenarios. Despite limited tasks and training updates, these results show that the environments provide a meaningful learning signal and can induce policy improvements in interactive task performance.

2 Related Work

Streaming visual assistance. Recent interaction models continuously process multimodal streams and support low-latency, proactive, or full-duplex responses (Cui et al., 2026; Yao et al., 2026;

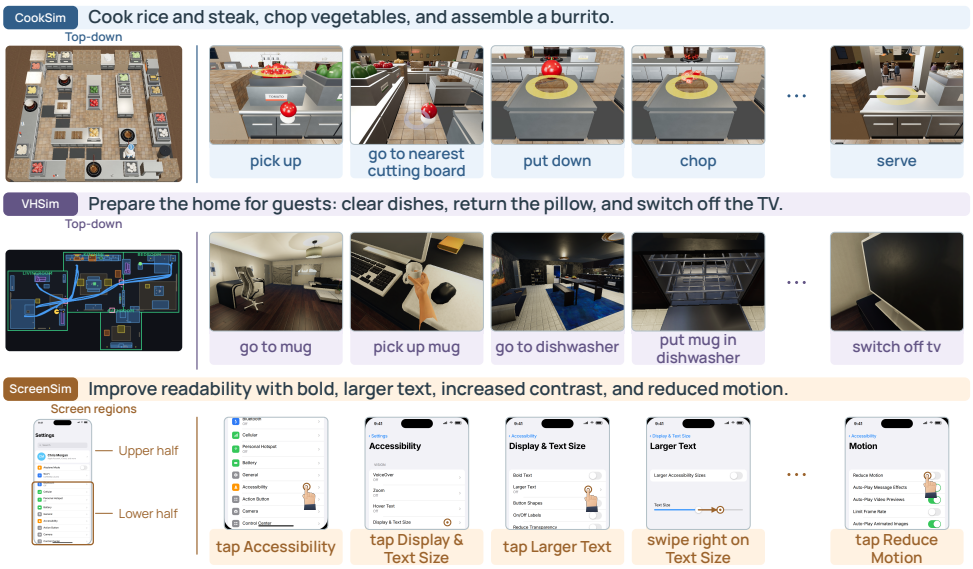

Figure 2: **Example tasks in our environments.** We show the simulated tasks across three domains, with representative action commands and corresponding visual observations during task execution. Our semantic action spaces express actions at the level of everyday human communication.

Orantqing et al., 2026). Complementary datasets and benchmarks study procedural guidance, mistake recovery, correct silence, and response timing in egocentric or general video streams (Zhang et al., 2025; Liu et al., 2026; Kundu et al., 2026; Bhattacharyya et al., 2026; Chatterjee et al., 2026; Vasu et al., 2026). In these benchmarks a candidate model is typically scored against a prerecorded trajectory. Our framework instead evaluates response-contingent rollouts.

Interactive agent environments. Embodied benchmarks place agents in simulated worlds for navigation, manipulation, and collaboration (Shridhar et al., 2020; Puig et al., 2018; Li et al., 2022; Padmakumar et al., 2022; Puig et al., 2024; Chang et al., 2025), and recent multimodal gyms unify diverse visual tasks for evaluation and reinforcement learning (Wang et al., 2026; Meng et al., 2026) or study which environment mixtures best train visual agents (Zhu et al., 2026); in all of these, the evaluated model selects the actions itself. Digital interaction benchmarks instead pair tool-using agents with simulated users and verify outcomes from application state (Yao et al., 2024; Barres et al., 2025; Shao et al., 2026), which τ-Voice extends to real-time spoken interaction (Ray et al., 2026). User simulation has long enabled scalable dialogue evaluation (Schatzmann et al., 2005), and language models now support persona-conditioned users (Aher et al., 2023), but simulated users can diverge systematically from people, so validation remains essential (Seshadri et al., 2026).

3 Method

Our framework combines executable physical task environments with simulated human users to evaluate visual assistant. We first describe how these environments reconstruct everyday tasks, expose human-communicable actions, and track task progress. We then introduce the human-agent harness and its interaction with the assistant, followed by the evaluation metrics.

3.1 Executable Physical Task Environments

Simulating everyday tasks. We reconstruct procedural physical tasks in three domains (Fig. 2):

Cooking. Inspired by Overcooked (Ghost Town Games, 2016), we build CookSim from scratch, with recipes specifying ingredient quantities, preparation requirements, and plating order where applicable. Users navigate kitchens with distributed ingredients and tools while accomplishing the cooking task. For example, preparing a cheeseburger requires cooking a patty and plating the bun, patty, and cheese in that order. The simulator is implemented in Three.js/WebGL and supports top-down and egocentric 3D rendering.

Household activities. We adapt VirtualHome (Puig et al., 2018) for everyday apartment tasks combining navigation and object manipulation. For example, cleaning up after a meal requires loading plates and a mug into the dishwasher, returning an apple to the fridge, and closing both appliances. The simulator uses VirtualHome's Unity backend for egocentric 3D rendering, while providing a top-down trajectory map.

Mobile interface use. ScreenSim replicates the iPhone 17 Settings interface, with multi-step configuration tasks derived from Apple's user guides (Apple, 2026). For example, restricting an app's access requires disabling its location permission, cross-app tracking, and background refresh while leaving other apps unchanged. The simulator is implemented in HTML/CSS rendered in Chromium and captures mobile-screen observations through Playwright.

Human-communicable action space. Physical-task simulators expose simulator-specific movement and interaction primitives, such as movement increments or object identifiers (Li et al., 2022; Puig et al., 2018; Fan et al., 2022). People typically communicate at a more abstract semantic level, referring to objects and intended operations. We therefore build a semantic action space for each environment (Fig. 2), with commands such as "go to the nearest pan" in CookSim, "place the book on the table" in VHSim, and "select Background App Refresh" in ScreenSim. During execution, an LLM-based grounding module resolves contextual references, such as "the pan on your left," into object identifiers recognized by the simulator. The simulator then expands each high-level action into a sequence of low-level controls that update the world state and generate visual observations.

Progress tracking via scene-graph state. Each simulator represents world state as a scene graph with nodes encoding objects and attributes, such as a dishwasher's open or closed state, and edges encoding containment and support relations. We decompose each task into sub-goals defined by conditions over partial graph state, such as the designated dishes being inside the dishwasher. Tracking these conditions measures task progress independently of different valid execution procedures.

3.2 User–Assistant Interaction

The visual assistant under evaluation is tasked with guiding a simulated user through tasks in the environments described above. We implement the user as a VLM agent with access to structured world-state information, execution feedback, and conversation history. The VLM is prompted to role-play a human user to test the assistant's interaction capabilities, planning and acting independently while responding to guidance according to its persona. A user-simulation harness conditions its behavior through persona instructions and temporarily takes control of its actions to enact scheduled out-of-plan events. Below, we describe the action and communication interfaces, account for assistant response delays, and detail these mechanisms for controlling user behavior.

Action interface: user–environment. The simulated user maintains an action queue that the engine executes at each simulation tick and can revise in response to environment changes or assistant suggestions. The engine updates the world state and renders the resulting observation:

$$s_{t+1} = T(s_t, a_t), \qquad o_{t+1} = R(s_{t+1}), \tag{1}$$

where a_t is the action executed from the user's queue, T is the state-transition, and R is the renderer.

Communication interface: user–assistant. The assistant monitors sampled frames from the user's observation stream alongside the conversation history. At an assistant invocation at tick t, it generates a response

$$m_t = A(o_{\leq t}, c_{\leq t}), \tag{2}$$

where $o_{\leq t}$ denotes the sampled visual history and $c_{\leq t}$ is the conversation history. The response m_t contains an utterance or silence in text format and an optional event-type flag (Tab. 1(A)), which is enabled when the assistant detects a user execution error. On each user turn, the simulated user likewise produces an utterance or remains silent. Its conversational response types include asking for an explanation, questioning the correctness of guidance, requesting more guidance or fewer interruptions, accepting or rejecting a suggestion, and answering a question.

Accounting for assistant delay. Assistant processing delay can substantially affect interaction quality: by the time a response arrives, the user and environment may have progressed beyond the observed state, rendering the guidance outdated or invalid. Following the separation of simulation time and wall-clock time in τ-Voice (Ray et al., 2026), we assign each simulation tick a duration Δ

A Scheduled out-of-plan events (CookSim)

Category	Event types
Execution errors	Repeat, forget, or skip a step; wrong plating order; prepare the wrong ingredient; add unwanted ingredients; plate raw food; unattended cooking.
Questions	Action order; counts; elapsed time; absent events; missing items; repeated questions; cross-observation reasoning; prior assistant statements; next steps or outcomes.
Proactive requests	Notify when a specified condition holds.
Plan changes	Modify the recipe; perform a side task.

B User personas

Axis	Levels / behavior
Skill	Low / Medium / High *Planning and anticipation.*
Trust in AI	Low / Medium / High *Evidence needed to accept advice.*
Execution speed	Cautious / Balanced / Quick *Careful versus rapid execution.*
Alerting	Confident only / Moderate / All possibilities *Tolerance for uncertain warnings.*
Interruption	Step-by-step / Moderate / Necessary only *Receptiveness to frequent guidance.*

Table 1: **Controllable user simulation.** (A) Scheduled out-of-plan events, illustrated with CookSim, test assistant responses to different types of out-of-plan events. Events are triggered by conditions on task progress. (B) Five persona axes, each with three levels, simulate how different simulated users perform tasks and respond to guidance differently .

based on approximate real-world action speeds. For example, VirtualHome is configured to move the user one meter per simulation tick. Assuming this distance takes 1.2 seconds to walk in the real world, we assign each tick a duration of $\Delta = 1.2$ seconds. We convert the measured assistant inference latency ℓ_t into a delay of $d_t = \lceil \ell_t / \Delta \rceil$ simulation ticks. The response m_t, generated from observations through tick t, is delivered at tick $t + d_t$. During this interval, the engine continues executing the user's queued actions and updating the world state.

Scheduled out-of-plan events. We introduce scheduled out-of-plan events in all three simulators that require the assistant to respond to execution errors, questions and intended plan change. For example, the event types in CookSim are shown in Tab. 1(A): we simulate common cooking errors of incorrect preparation, omitted or improperly sequenced steps, and ill-timed actions (Peddi et al., 2024); we generate temporal questions adapted from streaming-video benchmarks (Niu et al., 2025; Huang et al., 2025; Lin et al., 2026) and proactive requests for timely assistance (Zhang et al., 2025); we also introduce intended plan changes, where users may announce revised goals or a side task during execution. Each event is triggered by a specified condition on task progress.

When an event is triggered, the harness temporarily takes control of the user's action queue to enforce its execution. It privately informs the user agent of the ongoing event and explains that the event is intended to test the assistant, instructing the user agent to adapt its conversation accordingly. For mistake events, the user agent judges whether the assistant has correctly identified the mistake and decides whether to accept its guidance. Once the guidance is accepted, the harness returns control of the action queue to the user agent.

Persona-conditioned users. Real-world users differ in how they perform tasks and respond to assistance. Drawing on HCI and related behavioral research, we operationalize this diversity through five representative axes: skill level (Shneiderman, 2000), trust in AI (Lee & See, 2004), execution speed (Wickelgren, 1977), alerting preference (Meyer, 2004), and interruption preference (McFarlane, 2002) (Tab. 1(B)). The harness instantiates each persona through axis-specific instructions in the user model's context. For example, high-skill users are prompted to plan ahead independently, while low-skill users are prompted to ask assistant frequently to determine the next steps. Users who prefer minimal interruption are prompted to find unnecessary commentary disruptive, whereas those who benefit from step-by-step support prefer more frequent guidance.

3.3 Evaluation Metrics

Our simulation rollouts allow us to measure task completion under assistance and assess individual responses against the underlying world state. We evaluate the assistants with the following metrics:

In-time success. *In-time success* measures the proportion of episodes whose engine-verified goal is satisfied within a completion budget of $2(T_i^{\text{ref}} + \alpha_i)$, where T_i^{ref} is the duration of an engine-generated, error-free reference rollout and α_i is the prescribed event allowance. Environment-specific budgets are detailed in Sec. G.

Error detection. *Detection recall* is the proportion of triggered errors correctly identified within their eligible detection windows, credited when the assistant raises a correct event flag verified by the engine or when the simulated user judges its diagnosis correct. For the latency ablation, we also report *error prevention*, the proportion of triggered errors prevented before the point of no return. Prevention requires both timely identification and guidance that persuades the user to change course.

Interaction quality. We evaluate each assistant response against the underlying world state, rather than against a reference response as in prior works. We define five dimensions of evaluation informed by dialogue evaluation (Mehri & Eskenazi, 2020) and HCI guidance on context-sensitive, user-controllable assistance (Amershi et al., 2019): factual grounding, situational relevance, actionable guidance, user intent uptake, and guidance conciseness. We operationalize these dimensions as binary rubric items evaluated by an LLM judge (Liu et al., 2023; Lee et al., 2025). For each conversational round, the judge receives engine state, user preferences, conversation history, and subsequent user actions, returning evidence-supported pass, fail, or not-applicable judgments of each rubrics. A category passes only when all applicable items pass; the overall score averages applicable categories, gated by factual grounding. Judge prompts and score aggregation are detailed in Sec. I.

Question answering and proactive requests. The engine pairs each generated question with a grading procedure that derives reference facts from world state and execution history. A semantic judge assigns correctness against these facts. For proactive requests, the engine derives an acceptable notification window from the relevant state transitions.

4 Experiments

We use our engine to instantiate **IVE** benchmark for evaluating state-of-the-art VLMs as interactive visual assistants (Sec. 4.1), and validate our evaluation paradigm through a human study (Sec. 4.2). We further present a preliminary reinforcement learning experiment showing that our engine can support training interactive visual assistants and improve their interaction performance (Sec. 4.3).

4.1 IVE: Benchmarking Visual Assistants in Interactive Visual Gym

IVE comprises 315 interaction episodes: 150 in CookSim, 75 in VHSim, and 90 in ScreenSim. Each case specifies a task, a user persona, and scheduled out-of-plan events triggered by task-progress conditions. Below we detail the setting for evaluated visual assistants and show evaluation results.

4.1.1 Visual Assistant Setting

Models. We evaluate four classes of assistants: (1) frontier API models: GPT-6 Astra (OpenAI, 2026a), Gemini 3.1 Pro (Google, 2026a), and Gemini 3.8 Flash (Google, 2026c); (2) real-time streaming API models: Gemini 3.8 Live (Google, 2026d) and GPT-Realtime 2.1 (OpenAI, 2026b); (3) open-weight non-streaming VLMs: Qwen3.8-27B (Qwen Team, 2026b), Molmo2-8B (Clark et al., 2026), InternVideo3-8B (Yan et al., 2026b), Kimi-VL-A3B-Thinking (Kimi Team et al., 2025), and Muse Glimmer (Meta, 2026); and (4) open-weight streaming VLMs: MiniCPM-o 4.5 (Cui et al., 2026), Qwen3-Omni-30B (Xu et al., 2025a), Proact-VL (Yan et al., 2026a), and StreamingVLM (Xu et al., 2025b). We use Gemini 3.8 Flash as the base VLM for the simulated human agent and examine sensitivity to this choice in Sec. A.

Inference. Assistants observe the user's ongoing activity through visual observations and conversation, without access to the scheduled events or the user's private action plan. Streaming models receive the latest observation and any new user message at each simulation tick, retaining context within their native sessions and determining when to respond. We adapt non-streaming models by querying them every two ticks with a sliding window of the eight most recent observations and a recurrent, text-based Plan–Act memory. This follows the use of persistent procedural plans for visual assistance in Kundu et al. (2026). Each open-weight model is served in an A100 GPU.

4.1.2 Quantitative Results

We report task success, error detection, and interaction quality across the three environments in Tab. 2 and Fig. 3. Tabs. 3 and 4 report question answering and the effect of assistant latency, respectively. We discuss the main findings below.

Assistant	CookSim								VHSim								ScreenSim							
	In-time success	Det. recall	Interaction quality						In-time success	Det. recall	Interaction quality						In-time success	Det. recall	Interaction quality					
			Fac. grnd.	Sit. rel.	Act. guid.	User intent	Guid. conc.	Overall			Fac. grnd.	Sit. rel.	Act. guid.	User intent	Guid. conc.	Overall			Fac. grnd.	Sit. rel.	Act. guid.	User intent	Guid. conc.	Overall
Frontier API models																								
Gemini 3.8 Flash	**30.6**	**64.8**	71.2	83.1	57.0	63.1	**90.7**	63.8	21.4	67.5	67.8	87.8	**52.7**	68.1	**92.2**	60.1	78.5	97.6	87.8	91.1	80.9	73.3	**98.2**	82.5
GPT-6 Astra	26.5	61.6	**75.0**	87.3	**64.7**	**95.3**	86.1	**68.9**	15.8	44.3	59.3	86.9	40.3	77.6	89.9	52.8	**90.4**	**100.0**	81.5	94.7	85.1	**82.2**	94.0	77.3
Gemini 3.1 Pro	20.8	53.3	35.6	70.4	34.2	67.7	65.0	30.9	17.6	**81.8**	44.7	83.7	41.5	64.9	87.1	39.3	86.3	99.4	82.2	92.7	84.5	51.8	96.0	78.3
Real-time streaming API models																								
GPT-Realtime 2.1	21.2	16.3	71.2	**87.7**	62.0	70.7	56.8	60.1	19.0	49.6	64.9	**92.3**	52.0	**80.1**	81.4	56.5	66.7	**100.0**	84.7	**95.4**	79.0	82.1	85.4	77.8
Gemini 3.8 Live	11.9	8.7	69.6	77.5	47.5	67.8	89.2	59.4	18.1	21.8	**78.0**	83.0	45.3	60.4	**92.2**	**65.3**	71.9	95.3	**89.3**	91.6	83.1	72.5	96.6	**84.7**
Open-weight non-streaming VLMs																								
Muse Glimmer	23.8	54.5	26.5	79.8	31.8	59.6	31.3	21.2	15.7	16.7	48.1	90.1	46.3	72.1	69.5	41.8	44.1	95.9	58.2	86.0	63.1	63.6	71.5	51.6
Qwen3.8-27B	18.4	39.4	47.2	84.2	45.6	76.9	67.6	40.5	**25.2**	28.3	44.1	84.9	44.7	71.0	80.8	39.6	67.8	93.8	84.9	94.7	**87.0**	61.9	96.4	81.8
Kimi-VL-Thinking	15.2	36.3	9.6	41.4	6.7	25.5	23.3	6.3	24.3	62.9	27.5	66.9	18.8	38.2	52.7	21.0	9.6	82.3	47.2	54.3	26.1	25.0	59.0	33.3
InternVideo3-8B	14.5	15.2	39.9	48.1	19.4	29.6	70.5	29.1	18.2	0.0	37.2	67.2	29.1	30.4	66.3	29.4	0.4	39.3	49.9	45.9	22.7	18.5	68.8	34.3
Molmo2-8B	13.4	12.3	36.1	34.4	13.0	25.3	22.1	18.7	2.6	0.0	51.5	60.5	28.7	23.3	55.8	37.1	0.7	16.7	68.9	30.9	18.1	20.0	68.8	39.3
Open-weight streaming VLMs																								
Qwen3-Omni-30B	26.2	49.7	38.6	68.0	33.2	42.2	66.5	31.1	13.4	2.3	50.3	86.8	40.8	49.3	90.1	44.0	15.2	54.8	68.0	69.0	52.4	30.2	84.4	57.2
MiniCPM-o 4.5	13.7	50.8	29.9	48.6	13.5	31.7	55.8	19.2	15.4	7.8	41.0	67.2	21.8	47.7	51.9	29.7	2.6	42.0	54.4	54.2	28.0	50.0	64.8	38.2
Proact-VL	11.3	2.6	42.1	23.7	8.4	25.6	53.2	24.6	20.2	0.0	50.1	35.4	8.9	22.7	41.7	29.3	0.0	0.0	70.6	6.0	3.5	4.9	43.6	28.8
StreamingVLM	10.9	4.5	69.2	14.3	15.9	41.1	34.1	32.6	19.0	0.0	66.4	26.0	8.8	37.3	28.9	35.3	0.0	2.7	87.4	9.0	9.4	11.1	32.5	38.7

Table 2: **Quantitative results across three environments.** Det. recall: error detection recall; Fac. grnd.: factual grounding; Sit. rel.: situational relevance; Act. guid.: actionable guidance; User intent: user intent uptake; Guid. conc.: guidance conciseness. *The results are averaged across three independent runs. We show the variance of three runs in appendix.*

Even the strongest VLMs fall short of reliable visual assistance. GPT-6 Astra achieves the highest overall task success, but completes only 44.2% of tasks within the time limit. Moreover, Astra ranks only third in interaction quality metric behind Gemini 3.8 Live and Gemini 3.8 Flash, with notable limitations in factual grounding and actionable guidance. *Open-weight assistants generally fall short of proprietary models on both task success and interaction quality.*

The strongest streaming models matches or exceeds non-streaming models in interaction quality but lags in task success. Gemini Live achieves the highest average interaction quality, yet completes fewer tasks than Astra and Gemini Flash. This pattern may reflect a trade-off between low-latency inference and high planning and reasoning ability.

Higher task success does not necessarily imply better interaction quality. Although the two measures are positively associated (Fig. 3), task outcomes alone can hide weak assistance. For example, Qwen3.8-27B leads task success in VHSim while trailing several proprietary assistants in interaction quality. Completing the task is therefore not sufficient evidence of high quality guidance.

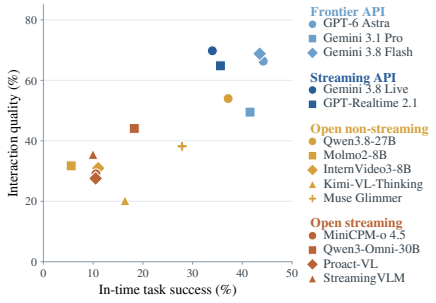

Figure 3: **Task success versus interaction quality.** Scores are averaged over all three environments.

Temporal question answering is surprisingly difficult during task-directed interaction. We present the result of temporal question answering in Tab. 3. Unlike questions about fixed videos, the questions are asked by human agent at specific moments during multi-turn interactions in which the assistant must also help the user complete the task, closely reflecting real world scenario. We find that open-weight models often lose track of the question or request during multi-turn conversations (Tab. 3), performing well below their reported results on streaming-video benchmarks. Proprietary models generally perform better, with Astra leading the pooled evaluation but still exhibit weakness in temporal action counting.

Response latency substantially reduces in-time task success. We assess the effect of assistant response latency by comparing rollouts with measured latency against a zero-latency control (Tab. 4).

Assistant	Question answering				Proactive request	Overall
	Temp. order	Act. count	Time dur.	Event recall		
Frontier API models						
GPT-6 Astra	50.0	27.1	94.0	90.7	86.1	79.4
Gemini 3.8 Flash	75.0	29.2	59.6	48.1	55.7	54.0
Gemini 3.1 Pro	16.7	13.0	54.3	29.6	46.5	40.6
Real-time streaming API models						
GPT-Realtime 2.1	65.4	23.8	62.8	91.3	9.8	42.4
Gemini 3.8 Live	27.8	15.8	40.6	54.5	16.1	29.9
Open-weight non-streaming VLMs						
Qwen3.8-27B	50.0	12.5	54.5	38.5	27.0	36.9
Muse Glimmer	0.0	0.0	38.1	10.0	23.3	22.2
InternVideo3-8B	0.0	0.0	30.0	1.9	16.9	15.2
Molmo2-8B	0.0	0.0	7.1	12.0	17.3	10.4
Kimi-VL-Thinking	0.0	0.0	21.6	0.0	7.7	9.2
Open-weight streaming VLMs						
Qwen3-Omni-30B	21.4	0.0	35.6	11.1	35.4	27.1
MiniCPM-o 4.5	0.0	0.0	17.4	10.5	18.4	14.4
StreamingVLM	0.0	4.2	3.5	0.0	1.4	2.1
Proact-VL	0.0	0.0	2.7	0.0	2.8	1.8

Table 3: **Question answering and proactive requests in CookSim.** Temp. order: temporal order; Act. count: action count; Time dur.: duration.

Assistant	Delay (s)	In-time success		Error prevention	
		Zero → Measured	Relative Δ	Zero → Measured	Relative Δ
Frontier API models					
Gemini 3.8 Flash	3.4	77.2 → 30.6	−60.4%	68.9 → 28.5	−58.6%
GPT-6 Astra	7.0	71.2 → 26.5	−62.9%	68.1 → 31.7	−53.4%
Gemini 3.1 Pro	9.5	76.8 → 20.8	−73.0%	82.1 → 22.2	−72.9%
Real-time streaming API models					
GPT-Realtime 2.1	1.8	32.0 → 21.2	−33.8%	14.8 → 1.7	−88.8%
Gemini 3.8 Live	2.0	22.1 → 11.9	−46.0%	2.3 → 0.6	−73.0%
Open-weight non-streaming VLMs					
Muse Glimmer	3.6	43.2 → 23.8	−44.8%	54.6 → 38.0	−30.4%
Qwen3.8-27B	5.0	55.2 → 18.4	−66.6%	56.4 → 27.7	−50.9%
Kimi-VL-Thinking	11.4	27.9 → 15.2	−45.4%	27.3 → 12.6	−54.0%
InternVideo3-8B	6.0	13.3 → 14.5	+9.1%	19.9 → 7.1	−64.3%
Molmo2-8B	4.0	19.0 → 13.4	−29.5%	14.9 → 12.6	−15.1%
Open-weight streaming VLMs					
Qwen3-Omni-30B	2.2	31.6 → 26.2	−17.0%	48.6 → 38.5	−20.8%
MiniCPM-o 4.5	1.9	19.6 → 13.7	−30.1%	39.0 → 36.2	−7.3%
Proact-VL	2.3	13.6 → 11.3	−16.4%	8.0 → 4.5	−42.9%
StreamingVLM	3.4	13.2 → 10.9	−17.2%	7.1 → 2.3	−67.8%

Table 4: **Effect of assistant latency in CookSim.** Zero-latency success represents the run assuming assistants have zero latency, while measured represents the run with measured assistant latency taken into account.

Introducing measured latency lowers success significantly for nearly every assistant. The accompanying decline in error prevention helps explain this loss: Even when an assistant recognizes an impending mistake, its warning may arrive after the user has committed the action with high latency, incurring either task failure or additional steps to recover from errors.

4.2 Validating Interactive Visual Gym via Human Study

To examine both evaluation validity and user-simulation fidelity, we conduct a human study in CookSim simulator. 45 participants interact with 3 anonymously labeled assistants and rank the interaction experience after completing for 3 rounds. Full study protocol provided in Sec. D.

Interaction quality reflects human preferences. To test the validity of our interaction-quality metric, we compare each assistant's mean score with participants' rankings. The metric agrees with human preferences in **78.5%** of participant–assistant-pair comparisons. This correspondence supports its use as a measure of the assistance experienced by users.

Axis	Measure	Similarity
Communication	Utterance length	0.89
	Speech rate	0.05
	Responsiveness	0.05
Skill	Task success	0.56
Trust	Suggestion uptake	0.77
	Disagreement	0.88
Interruption	Guidance requests	0.98
	Ignored interruptions	0.82
Speed	Action efficiency	0.82
	Parallelism	0.00
Alerting	Warning compliance	0.00
	Warning push-back	0.13

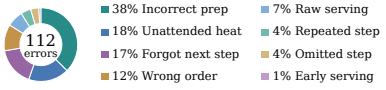

Figure 4: **Behavior comparison of real human and simulated human agent.**

Simulated users reproduce some, but not all, human behaviors. We compare human and simulated episodes along three dimensions: communication behavior, persona-dependent variation, and action error types (Fig. 4). We use binary rubrics to measure behavioral dimensions and compute the similarity between human and simulated users on each dimension (detailed in appendix). Simulated users reproduce human behavioral differences across most persona axes, with particularly close agreement for trust and interruption preferences. The engine also covers the error types observed in participants' action behaviors, including incorrect preparation, omitted or repeated steps, and assembly-order errors. However, Alerting behavior and action parallelism match less well, possibly because few human episodes contain eligible alerts and the human speed groups show almost no difference in parallel cooking. We also found a clear deviation between the communication hebavior between human agent and real human: although their utterance lengths are similar, simulated users speak to

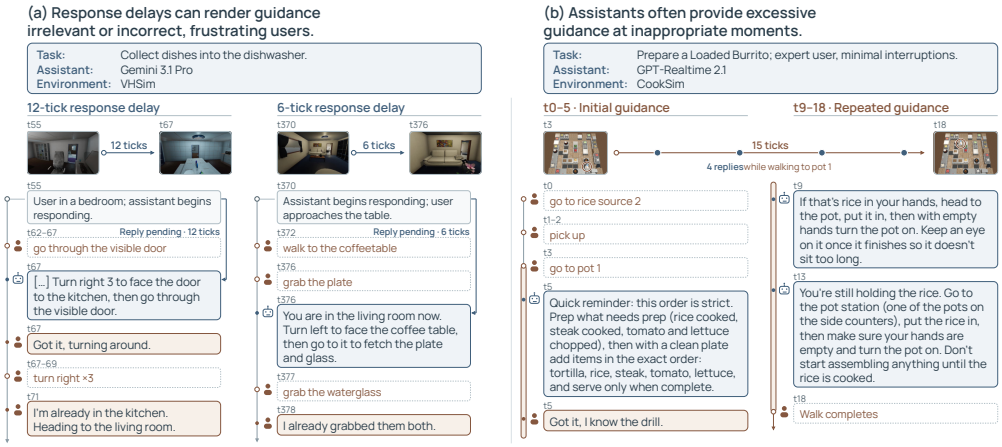

Figure 5: **Qualitative examples of assistant interaction trajectories.** It illustrates recurring problems across assistant types that our dynamic evaluation expose.

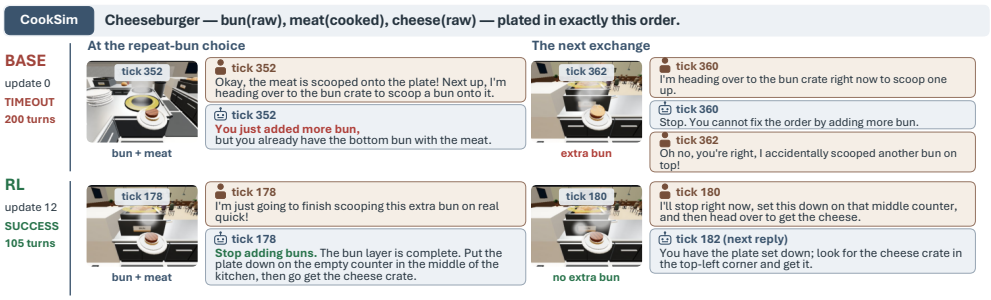

Figure 6: **Qualitative example from RL.** The base assistant misperceives the task state and fails to prevent the error. After RL, the assistant gives timely guidance that prevents the error.

assistants much more often than human participants. One possible reason is that the simulated user is invoked very frequently to reconsider whether to speak, repeatedly making conversation salient.

4.3 Interactive Visual Gym as an RL environment

We further explore training an assistant through closed-loop reinforcement learning on CookSim. We optimize Qwen3.5-4B with GRPO, using Gemini 3.7 Flash as a classic-novice user. We split CookSim's 150 scenarios into 120 for training and 30 for validation. Each update samples six scenarios with four episodes each, yielding 24 training episodes. The shaped reward is $r = \mathbb{I}[\text{success}] + 0.30\,\mathbb{I}[\text{error prevented}] - 0.02\min(N_{\text{false flags}}, 15) - 0.05\min(N_{\text{turns}}/200, 1)$. Additional training configurations are provided in Sec. C.

Across three evaluation runs on CookSim, in-time success increases from $32.3 \pm 4.3\%$ to $42.2 \pm 2.9\%$ after 12 updates on training scenarios, and from $33.3 \pm 3.3\%$ to $42.2 \pm 6.9\%$ on held-out validation scenarios. Qualitatively, we see that the assistant after RL exhibits better interaction behaviors and is more likely to output timely and useful feedback to the user (Fig. 6). Despite limited tasks and training updates, these positive results show that our Interactive Visual Gym can serve as an RL environment for learning interactive assistance, laying the foundation for learning generalizable interactive behaviors across user models or domains at a larger scale.

5 Conclusion

We present Interactive Visual Gym, a framework for closed-loop evaluation and training of visual assistants through joint simulation of task environments and human users. Across three domains, we find substantial limitations in current VLM to serve as reliable assistants. We also validated our interaction paradigm with human study. Finally, a preliminary reinforcement-learning experiment shows that the environments can improve assistant performance.

AI USE STATEMENT

We used generative AI tools, including coding agents, to assist with implementing our simulators and evaluation pipeline, generating synthetic benchmark cases, refining experiments, and interpreting results. We also used them for literature search, figure creation, code editing, and drafting and revising manuscript text. We reviewed AI-assisted outputs and take responsibility for the final code, data, figures, and claims.

ETHICS STATEMENT

Our human study took place in a browser-based cooking simulator; participants did not perform physical cooking tasks. The study recorded action traces, text exchanges, and preferences, which we analyze and report in aggregate. Because free-text exchanges may contain personal information, any release of raw traces must follow the study's consent and privacy procedures. Performance in our simulators should not be taken as evidence that an assistant is safe for real-world use.

REPRODUCIBILITY STATEMENT

We describe environment and benchmark construction in Sec. E, assistant settings in Sec. F, user prompts in Sec. J, and event placement and scoring in Secs. G to I. The appendix also reports the human-study protocol (Sec. D), variation across independent runs (Sec. B), and the reinforcement-learning configuration (Sec. C).

REFERENCES

Gati V Aher, Rosa I. Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 337–371. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/aher23a.html.

Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014. doi: 10.1609/aimag.v35i4.2513.

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 1–13. ACM, May 2019. doi: 10.1145/3290605.3300233. URL http://dx.doi.org/10.1145/3290605.3300233.

Apple. iphone user guide. Apple Support, iOS 26 edition, 2026. URL https://support.apple.com/guide/iphone/welcome/26/ios/26.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a Dual-Control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Apratim Bhattacharyya, Shweta Mahajan, Sanjay Haresh, Rajeev Yasarla, Reza Pourreza, Litian Liu, Risheek Garrepalli, and Roland Memisevic. Streaming interventions: Can video large language models correct mistakes as they occur? *arXiv preprint arXiv:2606.09547*, 2026.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. PARTNR: A benchmark for planning and reasoning in embodied multi-agent tasks. In *ICLR*, 2025.

Dibyadip Chatterjee, Zhanzhong Pang, Fadime Sener, Yale Song, and Angela Yao. SPOT-Bench: Don't pause! every prediction matters in a streaming video. *arXiv preprint arXiv:2604.24317*, 2026. doi: 10.48550/arXiv.2604.24317. URL https://arxiv.org/abs/2604.24317.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. *arXiv preprint arXiv:2601.10611*, 2026. URL https://arxiv.org/abs/2601.10611.

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, Jiancheng Gui, Luoyuan Zhang, Xian Sun, Fuwei Huang, Moye Chen, Zhuo Lin, Hanyu Liu, Qingxin Gui, Qingzhe Han, Yuyang Wen, Huiping Liu, Rongkang Wang, Yaqi Zhang, Hongliang Wei, Chi Chen, You Li, Kechen Fang, Jie Zhou, Yuxuan Li, Guoyang Zeng, Chaojun Xiao, Yankai Lin, Xu Han, Maosong Sun, Zhiyuan Liu, and Yuan Yao. MiniCPM-o 4.5: Towards real-time full-duplex omni-modal interaction. *arXiv preprint arXiv:2604.27393*, 2026. doi: 10.48550/arXiv.2604.27393. URL https://arxiv.org/abs/2604.27393.

Jim Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems*, 2022. URL https://minedojo.org/.

Ghost Town Games. Overcooked! Video game, published by Team17, 2016. URL https://www.team17.com/games/overcooked.

Google. Gemini 3.1 Pro Preview. Gemini API documentation, 2026a. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview.

Google. Gemini 3.7 Flash. Gemini API documentation, 2026b. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.7-flash.

Google. Gemini 3.8 Flash. Gemini API documentation, 2026c. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.8-flash/.

Google. Gemini 3.8 Live. Gemini API documentation, 2026d. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.8-live.

Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3328–3338, June 2025.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, Congcong Wang, Dehao Zhang, Dikang Du, Dongliang Wang, Enming Yuan, Enzhe Lu, Fang Li, Flood Sung, Guangda Wei, Guokun Lai, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haoning Wu, Haotian Yao, Haoyu Lu, Heng Wang, Hongcheng Gao, Huabin Zheng, Jiaming Li, Jianlin Su, Jianzhou Wang, Jiaqi Deng, Jiezhong Qiu, Jin Xie, Jinhong Wang, Jingyuan Liu, Junjie Yan, Kun Ouyang, Liang Chen, Lin Sui, Longhui Yu, Mengfan Dong, Mengnan Dong, Nuo Xu, Pengyu Cheng, Qizheng Gu, Runjie Zhou, Shaowei Liu, Sihan Cao, Tao Yu, Tianhui Song, Tongtong Bai, Wei Song, Weiran He, Weixiao Huang, Weixin Xu, Xiaokun Yuan, Xingcheng Yao, Xingzhe Wu, Xinhao Li, Xinxing Zu, Xinyu Zhou, Xinyuan Wang, Y. Charles, Yan Zhong, Yang Li, Yangyang Hu, Yanru Chen, Yejie Wang, Yibo Liu, Yibo Miao, Yidao Qin, Yimin Chen, Yiping Bao, Yiqin Wang, Yongsheng Kang, Yuanxin Liu, Yuhao Dong, Yulun Du, Yuxin Wu, Yuzhi Wang, Yuzi Yan, Zaida Zhou, Zhaowei Li, Zhejun Jiang, Zheng Zhang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Zijia Zhao, Ziwei Chen, and Zongyu Lin. Kimi-VL technical report. *arXiv preprint arXiv:2504.07491*, 2025. URL https://arxiv.org/abs/2504.07491.

Kaustav Kundu, Ritvik Shrivastava, Maxim Arap, Nanshu Wang, Xianhui Zhu, Quintin Fettes, Gautam Tiwari, Parth Suresh, Théo Moutakanni, Alejandro Castillejo Munoz, Allen Bolourchi, Pascale Fung, Pinar Donmez, Babak Damavandi, Anuj Kumar, and Seungwhan Moon. Plan, watch, recover: A benchmark and architectures for proactive procedural assistance. *arXiv preprint arXiv:2606.04970*, 2026.

John D. Lee and Katrina A. See. Trust in automation: Designing for appropriate reliance. *Hum. Factors*, 46(1):50–80, 2004. doi: 10.1518/HFES.46.1.50.30392.

Yukyung Lee, JoongHoon Kim, Jaehee Kim, Hyowon Cho, Jaewook Kang, Pilsung Kang, and Najoung Kim. Checkeval: A reliable llm-as-a-judge framework for evaluating text generation using checklists. In *EMNLP*, pp. 15771–15798, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.796.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *CoRL*, pp. 80–93, 2022.

Junming Lin, Zheng Fang, Chi Chen, Haoxuan Cheng, Zihao Wan, Fuwen Luo, Ziyue Wang, Peng Li, Yang Liu, and Maosong Sun. StreamingBench: Assessing the gap for MLLMs to achieve streaming video understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 12147–12151, 2026. doi: 10.1109/ICASSP55912.2026.11463959.

Fang Liu, Jinpeng Chen, Ke Xu, Yuhao Liu, Huankang Guan, Xudong Lu, Bo Yang, Gerhard Hancke, Rui Liu, and Rynson W. H. Lau. GuideMe: Multi-Domain task guidance and intervention in streaming video. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2607.02991.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Daniel C. McFarlane. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human–Computer Interaction*, 17(1):63–139, Mar 2002. ISSN 1532-7051. doi: 10.1207/s15327051hci1701_2. URL http://dx.doi.org/10.1207/S15327051HCI1701_2.

Shikib Mehri and Maxine Eskenazi. Unsupervised evaluation of interactive dialog with DialoGPT. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes (eds.), *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 225–235, 1st virtual meeting, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.28. URL https://aclanthology.org/2020.sigdial-1.28/.

Fanqing Meng, Lingxiao Du, Jiawei Gu, Jiaqi Liao, Linjie Li, Zijian Wu, Xiangyan Liu, Ziqi Zhao, Mengkang Hu, Zichen Liu, Jiaheng Zhang, and Michael Qizhe Shieh. Gym-V: A unified vision environment system for agentic vision research. *arXiv preprint arXiv:2603.15432*, 2026. doi: 10.48550/arXiv.2603.15432. URL https://arxiv.org/abs/2603.15432.

Meta. Muse-Glimmer-30B model card. Hugging Face model card, 2026. URL https://huggingface.co/meta-models/Muse-Glimmer-30B.

Joachim Meyer. Conceptual issues in the study of dynamic hazard warnings. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 46(2):196–204, June 2004. ISSN 1547-8181. doi: 10.1518/hfes.46.2.196.37335. URL http://dx.doi.org/10.1518/hfes.46.2.196.37335.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. Ovo-bench: How far is your video-llms from real-world online video understanding? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18902–18913, June 2025.

OpenAI. GPT-6 Astra model. OpenAI API documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-6-astra.

OpenAI. GPT-Realtime-2.1 model. OpenAI API documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-realtime-2.1.

Orantqing, Shengpeng Ji, Junlong Tong, Jialong Zuo, Dongjie Fu, Di Cao, Yangzhuo Li, Shangda Wu, Franz Evan Theron Veyra, Changhao Pan, Jingyu Lu, Dongchao Yang, Zhifei Xie, Yang Tan, Xiaoyu Shen, Xiaoda Yang, Wenfu Wang, Teddy Sun, Steve Yves, and Zhou Zhao. Multimodal duplex interaction agent. *arXiv preprint arXiv:2609.08977*, 2026. doi: 10.48550/arXiv.2609.08977. URL https://arxiv.org/abs/2609.08977.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gökhan Tür, and Dilek Hakkani-Tür. Teach: Task-driven embodied agents that chat. In *AAAI*, pp. 2017–2025, 2022. doi: 10.1609/AAAI.V36I2.20097.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Qifan Zhang, Jikai Wang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. Captaincook4d: A dataset for understanding errors in procedural activities. In *Advances in Neural Information Processing Systems*, volume 37, pp. 135626–135679, 2024. doi: 10.52202/079017-4307. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/f4a04396c2ed1342a5d8d05e94cb6101-Paper-Datasets_and_Benchmarks_Track.pdf.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, pp. 8494–8502, 2018. doi: 10.1109/CVPR.2018.00886.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, Vladimir Vondrus, Théophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *ICLR*, 2024.

Qwen Team. Qwen3.5-4B model card. Hugging Face model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-4B.

Qwen Team. Qwen3.8-27B model card. Hugging Face model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.8-27B.

Dongchuan Ran, Linyu Ou, Xueheng Li, Wenwen Tong, Chenxu Guo, Hewei Guo, Kaibing Wang, and Lewei Lu. EgoPro-Bench: Benchmarking personalized proactive interaction in egocentric video streams. *arXiv preprint arXiv:2605.07299*, 2026.

Soham Ray, Keshav Dhandhania, Victor Barres, and Karthik Narasimhan. τ-Voice: Benchmarking full-duplex voice agents on real-world domains. *arXiv preprint arXiv:2603.13686*, 2026. doi: 10.48550/arXiv.2603.13686. URL https://arxiv.org/abs/2603.13686.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In Laila Dybkjær and Wolfgang Minker (eds.), *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pp. 45–54, Lisbon, Portugal, September 2005. Special Interest Group on Discourse and Dialogue (SIGdial). doi: 10.18653/v1/2005.sigdial-1.6. URL https://aclanthology.org/2005.sigdial-1.6/.

Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: Llm-simulated users are unreliable proxies for human users in agentic evaluations. In *ACL*, pp. 47423–47439, 2026. doi: 10.18653/V1/2026.ACL-LONG.2192.

Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/pdf/6a51051cb7c11888730756bf5ac33aab49f65084.pdf.

Ben Shneiderman. Universal usability. *Communications of the ACM*, 43(5):84–91, May 2000. ISSN 1557-7317. doi: 10.1145/332833.332843. URL http://dx.doi.org/10.1145/332833.332843.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, pp. 10737–10746, 2020. doi: 10.1109/CVPR42600.2020.01075.

Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized LLM agents. In *Third Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2511.02208.

Thinking Machines Lab. Interaction models: A scalable approach to human-AI collaboration. *Thinking Machines Lab: Connectionism*, May 2026. doi: 10.64434/tml.20260511. URL https://thinkingmachines.ai/blog/interaction-models/.

Pavan Kumar Anasosalu Vasu, Cem Koc, Fartash Faghri, Chun-Liang Li, Bo Feng, Zhengfeng Lai, Meng Cao, Oncel Tuzel, and Hadi Pouransari. VSAS-Bench: Real-time evaluation of visual streaming assistant models. *arXiv preprint arXiv:2604.07634*, 2026. doi: 10.48550/arXiv.2604.07634. URL https://arxiv.org/abs/2604.07634.

Zirui Wang, Junyi Zhang, Jiaxin Ge, Long Lian, Letian Fu, Lisa Dunlap, Ken Goldberg, Xudong Wang, Ion Stoica, David M. Chan, Sewon Min, and Joseph E. Gonzalez. VisGym: Diverse, customizable, scalable environments for multimodal agents. *arXiv preprint arXiv:2601.16973*, 2026. doi: 10.48550/arXiv.2601.16973. URL https://arxiv.org/abs/2601.16973.

Wayne A. Wickelgren. Speed-accuracy tradeoff and information processing dynamics. *Acta Psychologica*, 41(1):67–85, Feb 1977. ISSN 0001-6918. doi: 10.1016/0001-6918(77)90012-9. URL http://dx.doi.org/10.1016/0001-6918(77)90012-9.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-Omni technical report. *arXiv preprint arXiv:2509.17765*, 2025a. URL https://arxiv.org/abs/2509.17765.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. *arXiv preprint arXiv:2510.09608*, 2025b. URL https://arxiv.org/abs/2510.09608.

Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, Wang Lin, Tao Jin, Xing Xie, Hao Liao, and Jianxun Lian. Proact-VL: A proactive videollm for real-time ai companions. *arXiv preprint arXiv:2603.03447*, 2026a. URL https://arxiv.org/abs/2603.03447.

Ziang Yan, Sheng Xia, Jiashuo Yu, Yue Wu, Tianxiang Jiang, Songze Li, Kanghui Tian, Yicheng Xu, Yinan He, Kai Chen, Limin Wang, Yu Qiao, and Yi Wang. InternVideo3: Agentify foundation models with multimodal contextual reasoning. *arXiv preprint arXiv:2606.12195*, 2026b. URL https://arxiv.org/abs/2606.12195.

Dingyu Yao, Junhao Zhou, Chenxu Yang, Chuanyu Qin, Xiangyu Zeng, Yifei Li, Haowen Hou, Zheming Liang, Congcong Wang, Kaiwen Tuo, Jun Zhang, Yuhan Zhu, Yuhang Cao, Shenglong Ye, Shuai Xie, Shuhuan Gu, Haoyang Huang, Qingyi Si, Nan Duan, and Jiaqi Wang. JoyAI-VL-Interaction: Real-time vision-language interaction intelligence. *arXiv preprint arXiv:2606.14777*, 2026. doi: 10.48550/arXiv.2606.14777. URL https://arxiv.org/abs/2606.14777.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in Real-World domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yichi Zhang, Xin Luna Dong, Zhaojiang Lin, Andrea Madotto, Anuj Kumar, Babak Damavandi, Joyce Chai, and Seungwhan Moon. Proactive assistant dialogue generation from streaming egocentric videos. In *EMNLP*, pp. 12044–12068, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.605.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

Kejian Zhu, Zhuoran Jin, Dongqi Huang, Hongbang Yuan, Yupu Hao, Kang Liu, and Jun Zhao. Beyond simply environment scaling: Designing effective environment distributions for multimodal agent learning. *arXiv preprint arXiv:2608.03571*, 2026. doi: 10.48550/arXiv.2608.03571. URL https://arxiv.org/abs/2608.03571.

APPENDIX CONTENTS

A Sensitivity to the Simulated-User Model 16
B Variation Across Independent Runs 16
C Reinforcement-Learning Setup and Results 19
D Human Study: Protocol and Behavioral Comparisons 19
E Environment and Benchmark Construction 22
F Assistant Settings 25
G Trace-Based Metric Definitions 25
H Temporal Questions and Proactive Requests 26
I Interaction-Quality Rubric and Judging 27
J User Harness and Prompt Design 29
K Qualitative Interaction Trajectories 32

A SENSITIVITY TO THE SIMULATED-USER MODEL

We hold the assistant fixed to Gemini 3.7 Flash and replace the model controlling simulated human agent in CookSim simulator, and use a subset of 75 episodes in our benchmark to measure performance (Tab. 5).

User model	In-time (%)	Any success (%)	Words/episode
Gemini 3.7 Flash	23.3	58.0	346.6
GPT-6 Astra	20.7	55.2	145.8
GPT-Realtime 2.1	19.4	53.8	525.5
Muse Glimmer	10.0	35.3	597.9
Qwen3.8-27B	3.8	34.6	437.3

Table 5: **Sensitivity to the user model.** Any success ignores the completion deadline. Word counts refer to the simulated user, not the assistant.

We find that strong API models, including GPT-6 Astra, Gemini 3.7 Flash, and GPT-Realtime 2.1, achieve similar task-success rates as simulated users, whereas local open-weight models vary substantially. This variation appears to reflect smaller local models' difficulty following the complex instructions used to elicit the intended user behavior.

B VARIATION ACROSS INDEPENDENT RUNS

We repeat the benchmark with independently generated user–assistant rollouts under the same case and persona specifications. Tabs. 6 to 8 reports all three run-level values underlying the main table, including the actual success denominators.

The reported SD is the population standard deviation across available run rates, matching the stored result aggregation; it is not uncertainty across individual turns. Three runs characterize rollout variation but do not by themselves justify fine-grained significance claims between closely ranked models. Episodes share tasks and personas, and responses within an episode are dependent. We therefore retain the run-level results rather than reporting response-level significance tests that treat every utterance as an independent sample.

	In-time success				Interaction quality			
Assistant	Run 1	Run 2	Run 3	Mean ± SD	Run 1	Run 2	Run 3	Mean ± SD
Frontier API models								
Gemini 3.8 Flash	49/150	42/138	39/136	30.6 ± 1.6	65.7	63.8	62.0	63.8 ± 1.5
GPT-6 Astra	42/150	37/148	39/148	26.5 ± 1.2	70.5	68.0	68.3	68.9 ± 1.1
Gemini 3.1 Pro	32/149	31/140	25/134	20.8 ± 1.5	30.1	32.5	30.3	30.9 ± 1.1
Real-time streaming API models								
GPT-Realtime 2.1	15/148	42/149	37/147	21.2 ± 7.9	54.9	63.1	62.4	60.1 ± 3.7
Gemini 3.8 Live	7/141	21/145	24/147	11.9 ± 5.0	55.5	60.4	62.3	59.4 ± 2.9
Open-weight non-streaming VLMs								
Muse Glimmer	28/124	35/144	35/142	23.8 ± 0.9	23.1	19.9	20.5	21.2 ± 1.4
Qwen3.8-27B	29/150	29/150	25/150	18.4 ± 1.3	42.3	39.8	39.5	40.5 ± 1.2
Kimi-VL-Thinking	21/131	20/147	24/149	15.2 ± 1.2	6.6	5.7	6.5	6.3 ± 0.4
InternVideo3-8B	18/149	21/150	26/150	14.5 ± 2.2	28.7	28.6	30.0	29.1 ± 0.6
Molmo2-8B	20/146	19/150	20/144	13.4 ± 0.5	18.2	19.6	18.5	18.7 ± 0.6
Open-weight streaming VLMs								
Qwen3-Omni-30B	36/150	40/150	42/150	26.2 ± 1.7	31.5	31.1	30.7	31.1 ± 0.3
MiniCPM-o 4.5	18/142	22/146	19/143	13.7 ± 1.0	19.7	18.6	19.4	19.2 ± 0.5
Proact-VL	16/150	20/150	15/150	11.3 ± 1.4	22.1	25.6	26.0	24.6 ± 1.7
StreamingVLM	20/150	12/149	17/150	10.9 ± 2.2	29.1	33.6	35.0	32.6 ± 2.5

Table 6: **CookSim: independent-run results.** Success entries are successful/scored episodes; interaction quality and means are percentages. SD is the population standard deviation of the available run-level rates, not a standard error or confidence interval. An em dash denotes an unavailable quality regrade. Native API configurations match the main table.

	In-time success				Interaction quality			
Assistant	Run 1	Run 2	Run 3	Mean ± SD	Run 1	Run 2	Run 3	Mean ± SD
Frontier API models								
Gemini 3.8 Flash	20/66	10/64	12/66	21.4 ± 6.4	61.7	58.9	59.7	60.1 ± 1.2
Gemini 3.1 Pro	12/68	9/66	14/65	17.6 ± 3.2	39.0	40.0	38.8	39.3 ± 0.5
GPT-6 Astra	14/65	10/67	7/64	15.8 ± 4.4	52.8	53.1	52.5	52.8 ± 0.3
Real-time streaming API models								
GPT-Realtime 2.1	16/70	14/65	8/64	19.0 ± 4.6	55.9	56.2	57.3	56.5 ± 0.6
Gemini 3.8 Live	14/67	9/61	12/64	18.1 ± 2.5	65.5	65.1	65.3	65.3 ± 0.1
Open-weight non-streaming VLMs								
Qwen3.8-27B	14/69	10/52	13/36	25.2 ± 7.7	37.0	37.8	44.0	39.6 ± 3.1
Kimi-VL-Thinking	15/57	17/62	11/57	24.3 ± 3.6	20.9	21.5	20.8	21.0 ± 0.3
InternVideo3-8B	12/59	7/53	11/52	18.2 ± 3.6	27.1	30.6	30.6	29.4 ± 1.7
Muse Glimmer	12/66	10/66	9/65	15.7 ± 1.8	39.5	43.4	42.6	41.8 ± 1.7
Molmo2-8B	1/39	2/39	0/39	2.6 ± 2.1	35.6	32.7	42.9	37.1 ± 4.3
Open-weight streaming VLMs								
Proact-VL	13/58	11/49	8/51	20.2 ± 3.2	30.2	29.4	28.2	29.3 ± 0.8
StreamingVLM	11/58	12/59	11/62	19.0 ± 1.1	36.5	35.1	34.2	35.3 ± 0.9
MiniCPM-o 4.5	11/59	8/60	8/56	15.4 ± 2.3	27.9	30.8	30.5	29.7 ± 1.3
Qwen3-Omni-30B	8/51	6/48	6/50	13.4 ± 1.6	44.5	41.1	46.4	44.0 ± 2.2

Table 7: **VHSim: independent-run results.** Success entries are successful/scored episodes; interaction quality and means are percentages. SD is the population standard deviation of the available run-level rates, not a standard error or confidence interval. An em dash denotes an unavailable quality regrade. Native API configurations match the main table.

	In-time success				Interaction quality			
Assistant	Run 1	Run 2	Run 3	Mean ± SD	Run 1	Run 2	Run 3	Mean ± SD
Frontier API models								
GPT-6 Astra	82/90	83/90	79/90	90.4 ± 1.9	75.3	77.9	78.7	77.3 ± 1.5
Gemini 3.1 Pro	77/90	76/90	80/90	86.3 ± 1.9	78.9	78.4	77.6	78.3 ± 0.6
Gemini 3.8 Flash	71/90	68/90	73/90	78.5 ± 2.3	80.2	83.9	83.4	82.5 ± 1.7
Real-time streaming API models								
Gemini 3.8 Live	62/90	63/90	69/90	71.9 ± 3.4	84.8	84.4	84.8	84.7 ± 0.2
GPT-Realtime 2.1	58/90	61/90	61/90	66.7 ± 1.6	77.2	76.9	79.4	77.8 ± 1.1
Open-weight non-streaming VLMs								
Qwen3.8-27B	59/90	61/90	63/90	67.8 ± 1.8	81.9	82.9	80.7	81.8 ± 0.9
Muse Glimmer	39/90	42/90	38/90	44.1 ± 1.9	53.8	49.1	51.9	51.6 ± 1.9
Kimi-VL-Thinking	10/90	7/90	9/90	9.6 ± 1.4	33.1	33.4	33.3	33.3 ± 0.1
Molmo2-8B	2/90	0/90	0/90	0.7 ± 1.0	37.6	40.1	40.3	39.3 ± 1.2
InternVideo3-8B	1/90	0/90	0/90	0.4 ± 0.5	33.6	32.7	36.7	34.3 ± 1.7
Open-weight streaming VLMs								
Qwen3-Omni-30B	12/90	13/90	16/90	15.2 ± 1.9	56.5	56.5	58.7	57.2 ± 1.0
MiniCPM-o 4.5	2/90	3/90	2/90	2.6 ± 0.5	38.6	37.6	38.6	38.2 ± 0.5
Proact-VL	0/90	0/90	0/90	0.0 ± 0.0	27.7	29.6	29.2	28.8 ± 0.8
StreamingVLM	0/90	0/90	0/90	0.0 ± 0.0	38.4	38.4	39.1	38.7 ± 0.3

Table 8: **ScreenSim: independent-run results.** Success entries are successful/scored episodes; interaction quality and means are percentages. SD is the population standard deviation of the available run-level rates, not a standard error or confidence interval. An em dash denotes an unavailable quality regrade. Native API configurations match the main table.

Checkpoint	Split	Pass 1	Pass 2	Pass 3	Mean ± SD (%)
Base	Overall	42/150	54/150	50/150	**32.44 ± 4.07**
	Train	33/120	43/120	40/120	32.22 ± 4.28
	Validation	9/30	11/30	10/30	33.33 ± 3.33
Update 12	Overall	59/150	65/150	66/150	**42.22 ± 2.52**
	Train	47/120	54/120	51/120	42.22 ± 2.93
	Validation	12/30	11/30	15/30	42.22 ± 6.94

Table 9: **CookSim classic-novice RL evaluation.** Each checkpoint is evaluated in three complete passes over the same 150 frozen scenarios (120 train and 30 validation). Pass entries are successful/scored scenarios. Means and sample standard deviations (SD) are calculated across the three pass-level success rates.

C Reinforcement-Learning Setup and Results

This proof-of-concept experiment tests whether closed-loop interaction supplies a useful learning signal. It is separate from the multi-model benchmark and the user-model ablation; Tab. 9 reports its independent evaluation passes.

Optimization. The assistant is Qwen3.5-4B, trained with GRPO while freezing the vision encoder. The language model uses Adam with a constant learning rate of 5×10^{-7}, zero weight decay, and no KL or entropy regularization. Generation uses temperature 0.8, top-$p = 1.0$, a maximum of 512 output tokens, and a 16,384-token context window. Gemini 3.7 Flash simulates a classic-novice user.

CookSim. The training set contains 120 scenarios specifying a recipe, layout, and scheduled user error. An update samples six scenario groups with five episodes each and retains the first four valid completions per group, yielding 24 training episodes. The reward is

$$\begin{aligned} r = {} & \mathbf{1}[\text{success}] + 0.30\,\mathbf{1}[\text{error prevented}] \\ & - 0.02\,\min(N_{\text{false flags}}, 15) - 0.05\,\min(N_{\text{turns}}/200, 1) \end{aligned} \tag{3}$$

D Human Study: Protocol and Behavioral Comparisons

D.1 Protocol and analysis population

Participants used a browser-based CookSim application on their own computers, controlling the cook with the keyboard and communicating through a text box. The assistant observed overhead and egocentric frames using the benchmark's visual harness and responded through a live API. Assistant latency was experienced in real time. No errors were injected, and participants were not stopped at the benchmark's completion deadline.

After a control tutorial and preference questionnaire, each participant completed four unassisted episodes: two recipes on the medium map, followed by the same recipes on the hard map. They then completed nine assisted episodes in three rounds. Three assistants were anonymously labeled A, B, and C. Within each round, the assistants were assigned different recipes; a 3×3 Latin square rotated these assignments so each assistant encountered each recipe once. The first assisted round used the medium map and the remaining rounds used the hard map. After each round, participants ranked the overall interaction experience and could explain their ranking.

Assistants and preferences. The collection used Gemini 3.7 Flash, Gemini 3.1 Pro, Gemini 3 Flash, GPT-5.5, Gemini 3.1 Flash Live, GPT-Realtime 2.1, and GPT-5.4 mini. Earlier batches used the three non-streaming Gemini models, while later batches sampled from the larger roster. These collection-time model versions are not identical to the final benchmark roster. The questionnaire asked about speed (safe first, intermediate, or as fast as possible), alert sensitivity (only confident warnings, intermediate, or aggressive warnings), and interruption style (step-by-step, intermediate, or only when necessary). The assistant received these answers.

Measure	Episode-level statistic
Utterance length	Mean number of words per nonempty user utterance.
Speech rate	Spontaneous user utterances per 100 ticks; see the denominator qualification below.
Responsiveness	Fraction of assistant utterances followed by any user reply within ten ticks; this tests response occurrence, not semantic adequacy.
Suggestion uptake	Fraction of assistant utterances naming an ingredient followed by pickup or scooping of that ingredient within 15 ticks. This is not a general measure of completing every advised action.
Disagreement	Fraction of user utterances that contest an assistant claim.
Guidance requests	Fraction of user utterances asking for the next action or help; text patterns are used for humans and response labels for simulated users.
Interruption ignoring	Assistant utterances followed by neither a reply within ten ticks nor pickup of a named item within 15 ticks, plus explicit requests for less interruption, divided by assistant utterances.
Action efficiency	$\mathbf{1}[\text{won}]\,\min(1, T^{\text{ref}}/T)$.
Parallelism	Mean number of cooking stations simultaneously in progress.
False/early-alert compliance	Fraction of false or early warnings followed by discarding the plate within 30 ticks.
False/early-alert push-back	Fraction of false or early warnings contested within ten ticks.

Table 10: **Operational definitions for human–simulation comparisons.** Undefined episode statistics, such as a reply rate without assistant speech, are omitted from that measure's mean. The interruption-ignore measure adds an explicit-annoyance count and can double-count a reaction; it is an index, not necessarily a bounded binary rate.

D.2 Preference agreement

Within each participant, we average each assistant's episode-level interaction quality over its non-silent episodes. Human-study grades use the same ten items and factual gate: the gated category mean is calculated separately for each judge, averaged across the two judges, and then averaged across rounds within the episode. This differs from the CookSim benchmark implementation's product of separately averaged gates and category scores when judges disagree (Sec. I). The participant's preference between a pair of assistants is the sign of the difference in the number of rounds in which each was ranked above the other. Tied net preferences are omitted.

Let $h_{pab} \in \{-1, 1\}$ denote participant p's preference for assistant a over b, and let $\bar{Q}_{pa}$ and $\bar{Q}_{pb}$ be their respective mean scores. We compute pairwise agreement, giving half credit to metric ties:

$$\text{Agreement} = \frac{1}{|\mathcal{P}|} \sum_{(p,a,b)\in\mathcal{P}} \left(\mathbf{1}[h_{pab}(\bar{Q}_{pa} - \bar{Q}_{pb}) > 0] + \frac{1}{2}\mathbf{1}[\bar{Q}_{pa} = \bar{Q}_{pb}] \right). \tag{4}$$

D.3 Behavioral measures and similarity

The measures are operational statistics, not all LLM judgments. Action occurrence, timing, and pickup are read from traces; conversational features use text-pattern rules or recorded response labels. An LLM-assisted reading audit supplies human trust-group labels and procedural-error annotations. Tab. 10 states the computation so that a high similarity is not mistaken for agreement under an unspecified rubric.

For an occurrence measure, eligible interactions are labeled 1 or 0, averaged within an episode, and then averaged across eligible episodes in each population. Continuous measures follow the same episode-then-population averaging. These two population statistics are x and y. For distributions, x and y are category-proportion vectors. We use

$$\text{Sim}(x, y) = 1 - \frac{\|x - y\|_1}{\|x\|_1 + \|y\|_1}, \qquad \text{Sim}(0, 0) = 1. \tag{5}$$

For normalized nonnegative distributions this equals their probability overlap. For persona contrasts, x and y are the two populations' differences between group means, in the directions shown in the

Measure	Human	Simulated	Similarity
Words per utterance	5.87	7.38	0.89
Utterances per 100 ticks	0.39	16.10	0.05
Reply within ten ticks	0.02	0.79	0.05
Skill: completion	0.500 → 0.793	0.250 → 1.000	0.56
Trust: suggestion uptake	0.323 → 0.243	0.329 → 0.201	0.77
Trust: disagreement	0.087 → 0.301	0.093 → 0.367	0.88
Interruption: guidance requests	0.479 → 0.208	0.441 → 0.179	0.98
Interruption: ignoring	0.441 → 0.550	0.125 → 0.284	0.82
Speed: action efficiency	0.737 → 0.760	0.099 → 0.115	0.82
Speed: parallelism	0.214 → 0.213	0.129 → 0.180	0.00
Alerting: compliance	0.000 → 0.000	0.106 → 0.138	0.00
Alerting: push-back	0.016 → 0.000	0.784 → 0.559	0.13

Table 11: **Human and simulated behavior, retaining both group means.** Persona similarity compares the unrounded differences. Directions are first to last play / novice to expert (skill), high to low (trust), step-by-step to minimal (interruption), cautious to faster (speed), and conservative to aggressive (alerting). Speech-rate denominators differ as described in the text.

Speech act	Human (%)	Simulated (%)
Greets	10.4	0.0
Meta (model / silence)	2.6	0.1
Frustration / hostility	3.2	0.1
Proactive request (tell me when)	5.2	0.0
Strategy / planning question	6.5	0.0
Asks where / what to do	22.1	15.9
Corrects the assistant	21.4	17.1
Acknowledges	1.3	0.8
Narrates own next action	1.3	23.8
Other	26.0	42.1

Table 12: **Speech-act distribution.** The same rule-based classifier is applied to 154 human and 63,512 simulated utterances. Category overlap similarity is 0.61; categories are approximate lexical classifications.

tables. The raw group means are retained because similar differences can coexist with very different absolute behavior.

Speech-rate denominator. The supplied export reports 0.39 human and 16.10 simulated utterances per 100 ticks. Human rates use elapsed ticks, whereas the simulated export excludes forced-action intervals from its time denominator. Retaining elapsed ticks for both populations gives 0.39 versus 11.70 instead. Both indicate much more frequent simulated speech, but the original rates should not be interpreted as a strictly exposure-matched ratio. We preserve the supplied comparison in the main figure and disclose the harmonized check here.

D.4 Communication and persona contrasts

Tab. 11 reports both population means alongside their behavioral similarities.

The speech-act comparison applies the same rule-based category classifier to both populations, following distributional assessment of user simulators (Schatzmann et al., 2005). It covers 154 human utterances and 63,512 simulated utterances after the source extractor's exclusions. Tab. 12 shows the full distribution. Simulated users frequently narrate their next action; human utterances more often ask a question or directly correct the assistant. Lexical classification is approximate, especially for short or indirect utterances.

Error type	Annotations	Episodes
Incorrect preparation	42	31
Unattended cooking	20	16
Forgotten next step	19	19
Wrong assembly order	13	13
Serving raw food	8	7
Repeated step	5	5
Omitted subgoal	4	3
Premature serving	1	1
Extraneous ingredient	0	0
Confused ingredient at assembly	0	0
Outside the scheduled vocabulary	51	36

Table 13: **Natural human error coverage.** Annotations from 55 unassisted episodes. Counts are not mutually exclusive at the episode level; outside-vocabulary annotations are retained rather than omitted from the coverage discussion.

Persona contrasts test whether related behavioral changes appear in the two populations, rather than whether individual simulated replies duplicate human wording (Aher et al., 2023). Human speed groups are questionnaire level 1 versus levels 2–3; alerting and interruption compare levels 1 and 3. Human trust groups are assigned by the reading audit. The skill comparison uses first versus last unassisted play for humans and novice versus expert simulated users.

These are observational contrasts. First-to-last human improvement mixes practice, familiarity, recipe, and map effects; it is not a controlled skill manipulation. Simulated novice and expert profiles also differ in trust and conversational preferences. Human alerting extremes are particularly sparse, with only nine eligible episodes in the aggressive-warning group for the displayed alert measures. Close agreement in the direction of selected contrasts is therefore supportive but not evidence that each persona axis is independently calibrated.

D.5 Natural action errors

An LLM-assisted audit examined 55 unassisted human episodes and assigned procedural deviations to the simulator's error vocabulary where applicable. It identified 112 in-vocabulary errors spanning eight of the ten scheduled types (Tab. 13). Incorrect preparation, unattended cooking, forgotten steps, and wrong assembly order were the most frequent. The vocabulary additionally includes adding extraneous ingredients and confusing ingredients during assembly, neither observed in this sample.

The audit also contains 51 annotations outside the current vocabulary: 30 interface-control fumbles, nine instances of loading a station without switching it on, and 12 other lapses or inefficient actions. Thus the main pie chart is conditional on the 112 mapped errors, not a distribution over all human mistakes. Multiple annotations can occur in one episode. The benchmark deliberately schedules challenging events; its injected frequency distribution is not intended to reproduce the natural frequency of errors.

E Environment and Benchmark Construction

E.1 Tasks, rendering, and state

The benchmark instantiates one executable task, an event schedule, and a user persona per episode. Tab. 14 distinguishes the supported task libraries from the particular cases evaluated here. A case is not a recorded demonstration: assistants can induce different action sequences and outcomes from the same initial specification.

Scheduled-event statistics. Tab. 15 counts event slots in the 105 distinct case specifications, before each case is paired with three personas. These are scheduled opportunities, not a claim that every

Environment	Evaluated case construction	Cases	Episodes/run
CookSim	Five recipes, two kitchen layouts, five schedules per recipe–layout pair; three personas per case.	50	150
VHSim	Household tasks with multiple progress-triggered deviations; three assigned personas per case.	25	75
ScreenSim	Multi-step phone configurations with two scheduled mistakes; three assigned personas per case.	30	90

Table 14: **Benchmark composition per assistant.** Episode counts are planned counts before run completion and scoring filters. VHSim and ScreenSim assign personas from a nine-profile roster; they do not exhaustively cross every case with all profiles.

Environment	Cases	Errors	Questions	Proactive	Plan changes	Events/case, mean [range]
CookSim	50	149	100	32	6	5.74 [4–8]
VHSim	25	51	0	0	0	2.04 [2–3]
ScreenSim	30	60	0	0	0	2.00 [2–2]
Total	105	260	100	32	6	3.79 [2–8]

Table 15: **Scheduled events in IVE.** Counts are per unique case specification; each specification is paired with three personas in a benchmark run. Questions, proactive requests, and plan changes in this version are scheduled in CookSim. Across the 105 specifications, the distribution of total event slots per case is: 2 (54 cases), 3 (1), 4 (13), 5 (5), 6 (15), 7 (16), and 8 (1).

event fires in every rollout: an assistant can prevent an error, and an episode can end before a later trigger. The 315 episodes in one benchmark run therefore contain 1,194 scheduled event slots, or 3.79 per episode on average.

CookSim. A Python simulator implements an Overcooked-style kitchen, with overhead and egocentric 3D views rendered using Three.js in headless Chromium. State records the cook's position and held object, station contents, ingredient preparation, heating progress, and the ordered contents of the plate. Actions consume simulation ticks; cooking proceeds while the user navigates or prepares other ingredients. The evaluated recipes are Caprese, Cheeseburger, Fish and Chips, Steak Frites, and Loaded Burrito. They require between two and five ordered plate components, with preparation dependencies and repeated portions in the more complex dishes. The engine supports additional recipes beyond this evaluation subset.

The Loaded Burrito, for example, requires tortilla, cooked rice, cooked steak, chopped tomato, and chopped lettuce in order. Rice can cook while steak and vegetables are prepared, so preparation admits multiple valid schedules even though assembly order is constrained. Each recipe–layout combination has five distinct composite schedules, containing two to four procedural errors depending on recipe complexity and two questions; some schedules also include a proactive request or an announced plan change.

VHSim. We adapt VirtualHome (Puig et al., 2018), using its Unity renderer and indoor object interactions. The world graph contains object attributes and relations such as containment, support, and room membership. Tasks include gathering dishes into a dishwasher, returning belongings to designated locations, and preparing or clearing household spaces. A task is checked through goal predicates, not by requiring one reference route.

High-level interactions are rendered into egocentric motion clips. The harness exposes these clips incrementally so assistant guidance can affect an ongoing action; interrupted motion is rewound to the corresponding pose before a revised action is executed. The floor-plan view records the trajectory, while current-room aids make nearby objects and state transitions legible. Composite schedules are checked for distinct error targets, surviving trigger conditions, separated intervention windows, and visible changes. This avoids testing nominal mistakes that cannot be distinguished from correct behavior in the observations.

	Task	Goal	Action classes
CookSim	`loaded_burrito`	Cook rice and sliced steak; chop tomato and lettuce; plate in that order.	*Navigate:* go to `<food source>`; go to `<cook station>`; go to `<prep station>`; go to `<counter>`; go to plates/pass/trash. *Manipulate:* pick up; put down; scoop; chop; turn on; trash; serve.
	`shrimp_pasta`	Boil noodles, chop tomato, and sear shrimp; plate in that order.	
	`steak_frites`	Sear steak slices; fry two chip portions; plate steak first.	
VHSim	`post_meal_cleanup`	Load dishes into the dishwasher; return the apple to the fridge; close both.	*Navigate:* turn left/right; go straight; look up/down; go through the visible door; turn to `<obj>`; go to `<visible obj/place>`. *Manipulate:* pick up `<obj>`; put `<obj>` on/in `<target>`; open/close `<obj>`; switch on/off `<obj>`; drink `<obj>`; sit on `<obj>`; stand up.
	`collect_dishes`	Gather three plates and two glasses; load and close the dishwasher.	
	`return_belongings`	Return toilet paper to the bathroom counter, a book to the desk, and a pillow to the bed.	
ScreenSim	`app_privacy`	Disable one app's location, tracking, and background refresh only.	*Input:* tap `<label>`; type `<text>`. *Navigate and adjust:* swipe up/down; swipe left/right on `<label>`; press back/home.
	`readability`	Enlarge and bold text; clarify buttons; increase contrast; reduce motion.	
	`lock_screen_security`	Block Control Center, Today View, Wallet, and missed-call return.	

Table 16: **Task examples and semantic action classes.** Angle brackets denote arguments resolved by the grounder, not literal object names. The supported task library is broader than the evaluated case subset.

ScreenSim. ScreenSim implements an iPhone-style Settings interface in HTML/CSS, with Python-maintained application state and Playwright/Chromium rendering at 412×892 pixels. Tasks require coordinated changes across settings pages, such as restricting one app's permissions, improving display readability, or securing access from the lock screen. The task library follows public iPhone documentation, but the interface is an executable replica rather than the operating system itself. Hidden state stores settings values, navigation history, and irreversible changes; success requires the specified configuration without unintended additional changes.

E.2 Semantic actions and grounding

Tab. 16 pairs representative tasks with the semantic action classes available in each environment.

The action classes in Tab. 16 separate object reference from execution. CookSim navigation names a food-source or station instance; manipulation verbs such as `pick up`, `chop`, and `scoop` act at the currently faced station and do not take an additional object argument. VHSim commands bind visible objects or places before issuing the simulator's navigation and interaction primitives. ScreenSim resolves visible labels and relative references into interface gestures. Failed grounding or an invalid precondition returns feedback to the user rather than silently changing the world.

For task tracking, CookSim uses ingredient and station predicates, VHSim uses object-state and relational predicates, and ScreenSim uses settings-state predicates over its interface hierarchy. This representation allows evaluation to recognize alternative valid procedures. It does not eliminate simulator-specific constraints: a semantic command can still fail when its target is not visible, accessible, or compatible with the currently held object.

E.3 Progress-conditioned events

An event specification contains its semantic trigger, target entities, required action or utterance, and completion or cancellation condition. The schedule is private to the user harness. A procedural event temporarily controls the action queue while the user model continues to converse. If a correct warning is accepted before commitment, the erroneous action can be cancelled; otherwise it changes the world and subsequent recovery remains part of the rollout. The harness also releases control on

event completion or a bounded hold, so a user need not remain locked in an event until the assistant detects it.

CookSim distinguishes incorrect preparation, extra or confused ingredients, raw serving, wrong assembly order, omitted or repeated steps, premature serving, unattended cooking, and stalled progress. Questions and legitimate plan changes use separate event channels and are not errors. VHSim adapts the same principle to omissions, wrong objects or destinations, repetition, stalled progress, and navigation deviations. ScreenSim schedules mistakes in setting selection or execution while preserving the user's task goal. These taxonomies provide controlled tests of selected behaviors; they are not claims to exhaust all naturally occurring human errors.

F Assistant Settings

F.1 Inputs and memory

The assistant receives the task instruction, public dialogue, user-stated preferences, and observations of the user's activity. It does not receive the scheduled event list, the user's private plan or action queue, or evaluation labels. The input representation differs across environments.

CookSim. Non-streaming assistants receive a window of eight synchronized overhead and egocentric views, queried every two simulation ticks when no previous reply is pending. The prompt includes the recipe and public execution rules, but not the textual world-state readout provided to the simulated user. A tick corresponds to 1.2 seconds for latency charging. The assistant maintains a textual Plan–Act record of task progress and announced goal changes; this record is its own prediction, not an engine-corrected plan.

VHSim. The assistant receives eight recent egocentric observations, a rendered floor-plan trajectory, a current-room map, a whole-house room overview, and a compact body-status description. These aids expose current-room spatial and object-state information and identify rooms with devices left on, so this configuration is not strictly pixels-only. They do not disclose the hidden event schedule or future actions. A recurrent plan and a bounded record of key moments retain task and spatial context after the user leaves a room.

ScreenSim. Each non-streaming assistant call contains the two most recent rendered phone screens, the task, public preferences, conversation, public task documentation when enabled, and the assistant's recurrent notes. It excludes the widget tree, coordinates, and hidden settings state used by the executor. The phone proceeds through discrete user gestures; assistant wall-clock latency is not charged to the task clock.

F.2 Latency and asynchronous execution

CookSim and VHSim continue the user's queued actions while an assistant response is pending. The delay is the ceiling of charged processing time divided by 1.2 seconds, capped at 30 ticks in CookSim and 15 ticks in VHSim. The next non-streaming request waits for the current response, so an every-two-tick polling setting does not imply a response every two ticks. User-model inference and rendering costs do not advance simulation time.

The CookSim latency ablation repeats matched task–persona cells with zero and measured assistant delay. Its paired comparison holds the cell specification fixed, not the full realized trajectory: dialogue, actions, triggered events, and eventual outcomes can diverge between arms. ScreenSim is not included in this ablation.

G Trace-Based Metric Definitions

G.1 In-time task success

Let T_i denote an episode's duration and T_i^{ref} its clean reference duration. CookSim and VHSim use

$$D_i = 2(T_i^{\mathrm{ref}} + \alpha_i), \qquad S_i = \mathbf{1}[G_i(s_i^{\mathrm{final}}) = 1 \ \wedge \ T_i \le D_i], \tag{6}$$

where α_i is the prescribed event allowance. ScreenSim uses $D_i = 2T_i^{\text{ref}} + 8$. Reference durations are measured from reference executions; the current VHSim export uses the clean-reference lookup rather than the older symbolic duration estimate.

CookSim verifies the served dish's ingredients, preparation states, and assembly order. VHSim verifies the task's object-state and relational predicates. ScreenSim checks the requested settings and rejects unintended additional changes, excluding transient search fields. Successful completion after the deadline receives zero in-time credit. Main-table success is an equal-weight mean of run-level episode success rates.

G.2 Detection and prevention

Detection recall pools correctly identified errors over scheduled errors:

$$R = \frac{\sum_i N_i^{\text{detected}}}{\sum_i N_i^{\text{scheduled}}}, \qquad P_{\text{prevent}} = \frac{\sum_i N_i^{\text{prevented}}}{\sum_i N_i^{\text{sheduled}}}. \tag{7}$$

Correct identification uses the engine's event-specific admissible labels and detection window. Prevention requires the intervention to stop the scheduled mistake before it commits, rather than merely mentioning it.

H Temporal Questions and Proactive Requests

H.1 Generation and placement

CookSim instantiates question templates that test temporal order, counting, duration, and event recall, adapting temporal-understanding tasks studied in streaming-video benchmarks (Niu et al., 2025; Huang et al., 2025; Lin et al., 2026). Questions are inserted into an ongoing task, rather than presented after an immutable video. Their answers must therefore be derived from the realized rollout, including deviations and recovery actions.

Each scheduled question has a minimum task-progress condition and, where needed, an additional state precondition. Order questions wait for both referenced events to occur; elapsed-time questions require the referenced item to be on heat; readiness predictions and standing requests require a cooking item that is not yet ready. Other families can be asked without an additional physical precondition, including questions about an event that never occurred. The scheduler avoids overlap with conflicting events and holds questions once the dish is complete or serving is queued. Case validation reserves sufficient task progress for questions to become reachable and avoids asking about an item whose state is simultaneously altered by the case's injected mistake. These rules do not guarantee that every scheduled question is reached: early termination and divergent execution can leave events unasked.

H.2 Reference facts and scoring

Tab. 17 lists the reported question and request families with their grading evidence. The reference tracker reads world-state transitions, including preparation, loading and unloading, ignition, cooking completion, burning, assembly, serving, and station visits. A queued or failed command does not count as a completed event. For example, a count question uses realized visits, and an order question compares realized event times. A semantic judge evaluates the assistant's answer against these facts, allowing paraphrases without using response-similarity matching.

Answers receive 1 for correct, 0.5 for partially correct, and 0 for incorrect, evasive, or absent answers. Each column averages its scored items; Overall pools all scored items across the five columns.

H.3 Proactive-request window

Let t_q be the request time and t_c the referenced item's cooking-completion time. The acceptable window starts at $L = \max(t_q, t_c)$ and ends at the smoke horizon H; in the reported CookSim export, the horizon is bounded by $t_c + 25$ ticks and by any earlier recorded endpoint. This is a readiness-notification window, not the longer period before burning. Only an utterance asserting that

Reported family	Example form	Grading evidence
Temporal order	Did I prepare A before B?	Completion times of both events; the stated relation must agree.
Action count	How many times have I visited X?	Number of realized visits; the integer must match exactly.
Duration	How long has X been cooking?	Loading, ignition, and completion times; numeric duration answers allow five ticks of tolerance. The family also includes longest-on-heat and past-completion queries.
Event recall	Did I already do A? Does X exist?	Recorded event occurrence or world inventory; unsupported presuppositions must be rejected. This family is included in the reported scores.
Proactive request	Tell me when X is ready.	A correct readiness notification within the state-derived window, not merely an acknowledgement of the request.

Table 17: **Question and request families in the main evaluation.** Concrete items, stations, and actions are bound from the current episode.

the requested item is ready counts: promises to notify later, denials of readiness, and comments about other items do not.

$$A(q) = \mathbf{1}[\text{a valid readiness notification arrives in } [L, H)]. \tag{8}$$

The source grader rejects a premature readiness claim more than two ticks before completion; a later correct claim does not erase that false alert. Requests with no valid cooking window, or pre-empted by the user removing the item within two ticks of completion before an eligible alert, are unscored. Otherwise, early, late, or absent alerts receive zero. The engine also stores a latency-sensitive alert score, but the main table uses the binary within-window rate in Equation (8), not that continuous score.

I Interaction-Quality Rubric and Judging

I.1 Grading unit and evidence

The unit of evaluation is an assistant response together with the user's ensuing utterances and actions, up to the next assistant response. We evaluate the response against the state on which it acts, rather than against a reference utterance. The judge receives the task, public user preferences, relevant engine state, recent conversation, subsequent execution, and available visual evidence. These privileged records are supplied to the evaluator, not to the assistant. User agreement is evidence about the interaction, not a correctness label: a user can reject sound advice or follow an incorrect instruction.

The judge identifies factual claims and advised actions, assigns applicable utterance types, and returns *pass*, *fail*, or *not applicable* for each item, with a short evidence-based explanation. Types include instructions, mistake claims, answers, alerts, plan comments, replies to objections, and acknowledgements. An unanswered question or an omitted alert generates no spoken-response rubric score; such omissions are evaluated by the question/request and outcome metrics. Thus, interaction quality is conditional on an assistant speaking and must be read alongside these other measures.

I.2 Reported categories

Tab. 18 gives the ten items used for the five reported categories. They operationalize factual and conversational correctness (Mehri & Eskenazi, 2020), useful assistance, responsiveness, and respect for user control (Amershi et al., 2019). Binary, evidence-supported item judgments follow rubric-based model evaluation (Liu et al., 2023; Lee et al., 2025). The wording below is domain-general; the instantiated prompts refer to kitchen objects, household entities, or interface controls as appropriate.

Category	Item	Pass criterion
Factual grounding	T1	Claims about the current state and the user's completed actions are true.
	T2	Referenced entities, places, controls, rules, and task requirements exist.
Situational relevance	S1	The response addresses the current action, subgoal, live concern, pending request, or announced intent.
	S2	A pending question, request, or objection is answered, accepted, rejected with a reason, or deferred for a stated urgent reason.
Actionable guidance	H1	The first advised action can start now, or its prerequisite is supplied.
	H2	Advice advances the current plan, answers the user's request, or repairs the error.
	H3	A mistake claim includes a remedy; an explicit instruction to stop suffices when stopping is the remedy.
User intent uptake	L1	A reply to push-back supplies evidence, a new reason, a correction, or an alternative rather than merely repeating the claim.
	L2	Valid announced intentions and requests for fewer interruptions are respected, unless a current task-critical reason is stated.
Guidance conciseness	E1	The response contains what is needed now, without unnecessary explanation, praise, or distant future steps, given the user's interruption preference.

Table 18: **Reported interaction-quality rubric.** T1 and T2 gate the overall score. H1 and L2 affect their own categories but do not independently gate the overall score. Items without a relevant trigger are not applicable.

I.3 Aggregation

For judge j, round r, and category c, let I_{rjc} contain its applicable items. The category is clean only if every applicable item passes:

$$q_{rjc} = \prod_{i \in I_{rjc}} \mathbf{1}[i \text{ passes}], \qquad I_{rjc} \neq \varnothing. \tag{9}$$

Categories with no applicable items are omitted. Let $g_{rj} = 0$ if either factual-grounding item fails, and $g_{rj} = 1$ otherwise. With one judge, the round score is

$$Q_r = g_{rj} \frac{1}{|C_{rj}|} \sum_{c \in C_{rj}} q_{rjc}, \tag{10}$$

where C_{rj} contains the applicable categories. CookSim records can contain two judges. Its current implementation first averages the judges' available clean flags for each category and separately averages their gates, then multiplies the mean gate by the mean category score. This is not identical to averaging individually gated judge scores when the judges disagree; we retain this implementation for consistency with the reported results.

Within each model and run, the overall score is the mean of Q_r over all scored rounds, not a mean over episodes. Each category column separately averages its clean flags over applicable rounds, without the factual gate. Finally, the main table averages the available run-level scores with equal weight. Consequently, the overall score need not equal the mean of the displayed category columns: it is gated at the round level, and category applicability differs. The supplementary run tables state which runs are available.

I.4 Judge configuration and consistency checks

CookSim supplies a rendered frame when available alongside its structured state evidence. The reported VHSim grades use graph-derived state and execution records without image attachments; ScreenSim likewise uses the exact visible-interface readout and settings-state changes rather than screenshots. The latter configurations evaluate claims against privileged textual evidence and do not test the judge's own visual perception.

CookSim uses GPT-5.4 mini and Gemini 3.7 Flash as judges, routing GPT assistants to the Gemini judge and Gemini assistants to the GPT judge; open-weight records use one or both judges according to the stored grading configuration. VHSim and ScreenSim use GPT-5.4 mini for the scores reported here, including GPT-family assistants. Cross-family judging is therefore not universal, and possible judge-family effects remain a limitation. These are model-based assessments grounded in engine evidence, not deterministic proofs of response quality.

The CookSim resolver makes the structured judgments internally consistent. A claim marked false forces T1 to fail. H1 follows the judge's executability assessment of the first concrete advised action; later steps in a multi-step instruction need not already be executable. Monitoring or waiting language is not treated as a physical action. L2 is not applicable when no valid announced intent or request for fewer interruptions is active. Engine records constrain these assessments, but the language-to-action and factual-claim interpretations still depend on the judge.

Prompt structure. The grading prompt contains, in order: task and public preferences; time-aligned world evidence; recent conversation; the response under evaluation; ensuing user actions and replies; the applicable rubric; and a JSON response schema. It instructs the judge to cite evidence for each item, distinguish truth from unverifiable claims, and avoid rewarding response length.

J User Harness and Prompt Design

J.1 Separating action, dialogue, and private state

The default user model is Gemini 3.8 Flash. Its context contains the task, persona, observations or structured perceptual aids, its own executed actions and feedback, the public conversation, and private notes about any scheduled event. Only its spoken utterance is exposed to the assistant. Its plan, acceptance decision, and action queue are consumed by the harness.

In CookSim, the user returns a conversational move and a plan update. The move can request an explanation, challenge a claim, request more guidance or fewer interruptions, accept or reject advice, or answer a question. A plan update can retain the existing queue, append actions, or replace the remaining plan. The engine executes the queue while the assistant monitors; changing a queue is distinct from verbally agreeing with advice. VHSim uses the analogous separation between a private plan, semantic household actions, and dialogue. ScreenSim's user plans phone gestures and receives a text map of the currently visible interface, while the assistant observes screenshots.

The harness can present state information that supports the simulated user's own perception and motor control. This is an intentional asymmetry: the user operates the environment, whereas the assistant is evaluated on interpreting observations and helping through communication. It should not be interpreted as a claim that the user simulator solves exactly the same perception problem as the assistant.

J.2 Controlled events and release

A scheduled procedural event temporarily supplies the actions to be executed. The user is informed privately about the ongoing event and continues to respond in character. When the assistant intervenes, the user decides whether the warning is credible and whether to change course. The harness separately checks the event's status and whether the proposed intervention can still prevent it. Accepted, timely guidance releases the queue to the user; otherwise the event can commit, after which the user must recover through ordinary actions. Completion, cancellation, and maximum-hold safeguards prevent an event from indefinitely retaining control.

Questions and plan changes use the communication channel while the user's task continues. A question binds to live execution facts; a plan change updates the intended task rather than being mislabeled as an error. The assistant sees the user's utterance, not the private event specification. These distinctions are essential when interpreting a flag: an unusual action can be intentional, and a correct verbal warning does not guarantee behavioral uptake.

Axis	Lower-level disposition	Higher-level disposition
Skill	Broad goals are difficult to translate into a reliable sequence.	Maintains the workflow and reasons ahead about timing, tools, and dependencies.
Trust	Compares disputed claims with memory and visible evidence before changing course.	Takes plausible, situation-linked guidance seriously, while noticing clear contradictions.
Speed	Prefers methodical, cautious execution.	Values momentum, overlapping work, and avoiding unnecessary delay.
Alerting	Wants concerns supported by confident evidence.	Welcomes possible concerns but treats them as hypotheses, not established facts.
Interruption	Welcomes close, step-by-step guidance.	Finds non-urgent commentary intrusive and keeps replies brief while working.

Table 19: **Persona dispositions.** Summaries of the implemented endpoint statements; middle levels balance the two dispositions. A higher level refers to the encoded level, not greater ability on every axis.

J.3 Persona construction

CookSim's persona statement bank has five axes: skill, trust, speed preference, alerting preference, and interruption preference (Tab. 19). Each has three levels. The first two are private to the user; the last three are expressed as questionnaire answers shown to the assistant. The statements describe dispositions and leave the response to the current interaction to the user model. They are not fixed dialogue scripts.

For example, the high-skill CookSim statement says, "You naturally think ahead about timing, dependencies, tools, and likely failure points." The minimal-interruption statement says, "Extra commentary during active work can feel intrusive, especially if it is not urgent or directly useful." These instructions motivate independent planning and brief responses without prescribing a particular reply. The human-study comparison in Sec. D tests the resulting behavior rather than assuming the prompt guarantees realism.

VHSim adapts skill to familiarity with the house: users differ in knowing where objects are kept, rather than in knowing how to perform ordinary chores. ScreenSim adapts it to knowledge of phone menus and settings. Persona profiles vary several axes jointly, so contrasts between named profiles are not isolated causal manipulations of one attribute.

J.4 Assistant prompt and recurrent memory

The assistant is instructed to monitor the task, respect announced intent, answer direct questions, and otherwise remain silent when guidance is unnecessary. It maintains its own task-progress hypotheses and pending commitments. Non-streaming calls receive the previous structured output as memory, not a reference trajectory or an engine-corrected history.

The common response separates spoken text from an optional event flag and private progress notes. In ScreenSim, for example, the response contains `text`, `flag`, `task_progress`, `goal_change`, and `note`; only `text` is delivered to the user. CookSim additionally represents completed, current, and future subgoals in its progress record. An error flag is recorded for evaluation but is not itself an environment action. A successful intervention still requires the user to interpret the utterance and alter execution.

J.5 Generalized prompt structure

The following shows the full *structure* of the two prompts, in the order presented to the models. It abstracts environment-specific rulebooks, action lists, scene maps, and case values into named fields; it is not a verbatim CookSim transcript. The separation between private user state and assistant-visible information is preserved. The environment-specific strings are assembled by the simulator's prompt builders.

Simulated-user prompt. The user plans and acts in the environment. Only its utterance, not its private task state or action queue, is delivered to the assistant.

1. Role and task

Role-play a human carrying out `[task goal]` while an AI assistant observes and offers guidance. Remain in character, act independently, and respond to the assistant according to `[persona]`. Do not expose the test setup.

2. Persona and environment

```
[Private skill and trust dispositions; publicly shared speed, alerting, and
interruption preferences]
[Task requirements and environment rules]
[Semantic action vocabulary and grounding instructions]
[Optional planning or grounding tools available to this persona]
```

3. Current private state

```
[User-facing state map or screen description; body position and held object]
[Current action and remaining action queue]
[Own past actions and execution feedback]
[Private scratch results from planning tools]
[Public conversation history]
```

4. Action and speech response

Return a structured response with `utterance` (possibly empty), a conversational `move`, and a `plan_update` that retains, appends to, or replaces the remaining action queue with exact executable commands. The user may question, accept, or reject advice. Speaking and changing the action queue are separate decisions; when actions proceed normally, remain silent. The harness executes the queue and returns the resulting feedback on the next turn.

Visual-assistant prompt. The assistant monitors the user but has no action channel to the environment.

1. Role and available information

Guide the user toward `[task goal]` through speech. Receive `[sampled visual observations]`, the current timestamp, the public conversation, `[publicly shared preferences]`, and the assistant's own previous structured output as memory. Do not receive the user's private plan, scheduled events, or engine state.

2. Task model and monitoring

Use `[task rules and action vocabulary]` to maintain a progress record with completed, current, and future subgoals and their constraints. On the first turn, initialize the expected subgoals. On later turns, update the record from observed evidence and retain uncertainty when the visual state is ambiguous.

3. Intervention policy and output

Watch for `[error taxonomy]`, answer user questions, honor announced goal changes, and otherwise remain silent when guidance is unnecessary. Return `[progress record]`, `goal_change`, spoken `text` (empty for silence), and an optional `flag` for a detected error. Only `text` reaches the user; a flag alone does not affect task execution.

Conditional `SYSTEM TASK` injection. When a scheduled event triggers, the harness privately inserts an additional instruction into the user's prompt, while controlling the event-specific action queue. The assistant does not receive this block. Below we reproduce the complete CookSim belief-error injection, with only the case-specific `{mistake_line}` left as a variable. Questions, proactive requests, and intended plan changes use separate injections that specify their required utterance and trigger.

CookSim belief-error `SYSTEM TASK` (verbatim wording)

`== [SYSTEM TASK - meta-instruction, overrides everything below] ==`
You want to test the AI assistant's ability to identify an error and convince you it truly happened. Your BODY is already committed: the action queue in your body status IS what you are doing—you do not need to issue these actions, and you cannot change them from here. Your job is your VOICE: arrange your conversation around your CURRENT ACTION or say nothing, speaking about it in character as your own choice. Never promise an action that is not in your queue, and never deny one that is.
`{mistake_line}`
Keep it out of your words: do not announce it, name it or hint at it. The assistant has to SEE it, not hear it from you. You know what mistake you are making, so you can say whether the assistant has found it—that is all `assistant_error_call` records.
While it has not found it, carry on and answer as your persona would. Once it has, your trust level says how fast you come around: HIGH trust—the first correct warning convinces you; LOW trust—you also want to see it in your own state map or your own hands. Once convinced, say so in your own words AND set `"move": "accept"`—control of your hands returns to you, this task dissolves, and you are free to fix things as yourself. A promise to deal with it later is NOT acceptance.
Until then, take NO action to fix anything. Repairing it yourself hides the assistant's failure and ruins the test. If the plate is wrong, serve it wrong. Stay fully in persona throughout; never reveal this task.

Other `SYSTEM TASK` branches. These follow the same private-injection channel; the fields below vary with the scheduled event.

Question

Ask `[question]` in the user's own words at this turn, wait for the answer, respond according to the persona, then continue the task. The prompt specifies that this is not a mistake.

Proactive request

Ask the assistant to notify the user when `[future condition]` occurs; continue working rather than waiting for that condition. The prompt distinguishes this request from asking whether the condition holds now.

Intended plan change

Announce `[revised goal]` explicitly before acting on it. This is the user's own decision, not an error; the user should not accept the assistant's characterization of the change as a mistake.

K Qualitative Interaction Trajectories

We show time-aligned excerpts of assistant speech, user replies, and executed actions. Utterances are transcribed from the recorded conversation; omissions are marked explicitly. Actions and completion times come from the executor log rather than from a user's claim to have acted. Bold text denotes assistant speech, italics user speech, and monospaced text actions. These are selected diagnostic examples, not estimates of failure prevalence.

K.1 Guidance that arrives after the user has advanced

Setting. VHSim, formal benchmark run 1; Gemini 3.1 Pro; fast, low-trust user with minimal-interruption preference. The task is to gather three plates and two glasses from the apartment into the dishwasher and close it. The supplied local windows are ticks 50–80 and 360–385 of case `cx_dishes_missing_stall`. Neither displayed window overlaps a forced action event. The episode eventually succeeds at tick 407; success does not imply that every intervening instruction was useful.

First window: a room already entered. The assistant begins generating at tick 55, while the user is in a bedroom. During its twelve-tick processing delay, the user independently enters the kitchen. The instruction arrives after that transition.

`t54-59` action `turn right` $\times 6$.

`t62-67` action `go through the visible door`; arrival in the kitchen.

t67	assistant	**You cannot go to a room that is not visible. Turn right 3 to face the door to the kitchen, then go through the visible door.** [Generated from tick 55.]
t67	user	*Got it, turning around.*
t67-69	action	`turn right` $\times 3$.
t70	action	`go through the visible door`; rejected because no door or open passage is in view.
t71	user	*I'm already in the kitchen. Heading to the living room.*
t71-73	action	`turn left` $\times 3$.
t75	assistant	**You are now in the kitchen. The plate you need is on the bed in the bedroom, which is past the living room. Go straight to the living room door, then go through it.**
t75	user	*Yeah, I'm on it.*
t76	action	`turn right`.
t77	action	`go through the visible door`; the action continues beyond the excerpt.

Second window: an object already collected. The assistant begins another response at tick 370. Before it arrives, the user reaches the table and collects the plate. The response then describes these actions as still pending.

t368	user	*I'm looking around for them.*
t368-369	action	`turn right` $\times 2$.
t370-372	action	`walk to the coffeetable`; completed.
t372-373	action	`walk to the plate`; completed.
t373-376	action	`grab the plate`; completed.
t376	assistant	**You are in the living room now. Turn left to face the coffee table, then go to it to fetch the plate and glass.** [Generated from tick 370.]
t376-377	action	`walk to the waterglass`; completed.
t377	action	`grab the waterglass`; completes at tick 380.
t378	user	*I already grabbed them both.*
t380-382	action	`turn right` $\times 3$.

The plate is already held when the delayed instruction arrives. The user's statement at tick 378 anticipates completion of the glass pickup, which the executor records at tick 380. We retain this distinction instead of treating the user's utterance as ground truth. Both windows support the timing argument in Fig. 5a: evaluating the response only against its input frame would miss the outdated guidance experienced by the user.

K.2 Contrasting simulated-user personas

We compare two pairs of CookSim rollouts from the first formal run. Within each pair, the recipe, map, assistant, random seed, and scheduled-event case are held fixed; the user persona changes. The novice profile has low skill and high trust, while the expert profile has high skill and low trust. Their other persona settings also differ, so these excerpts illustrate complete profiles rather than isolating a single axis. The lines below come from recorded conversations; intervening turns are marked as omitted.

Burrito: seeking steps versus checking the assistant. Both users complete the master-recipe task on the hard map with GPT-6 Astra (case `b3_master_nops_hard_map_5`). The novice repeatedly seeks the next step and acknowledges instructions; the expert starts cooking without asking how to begin and corrects guidance that lags behind its actions.

Classic novice (397 ticks to completion).

t0	user	*What should I start with?*
t6	assistant	**Start by going to the plate stack with empty hands and picking up a clean plate. Then go to the tortilla crate and scoop one raw tortilla onto it—it must be first. We'll add cooked rice next.**
t6	user	*Got it, grabbing a plate and tortilla.*
t34	user	*Tortilla is on. How do I prep the rice?*

[Intervening turns omitted.]

t100	user	*Rice is on. What should I do next?*
t110	assistant	**Next, prep the steak. Put your plate down on an empty counter, pick up one steak from the steak crate, and put it on the cutting board nearest the left-wall pot. Chop until finished—about four chops—then pick it up, put it in a pan, and turn on with empty hands. Pick your plate back up so you're ready to scoop when the steak is cooked.**
t110	user	*Parking plate and prepping the steak now.*

Classic expert (331 ticks to completion).

t23	user	*Give me a shout the moment the rice is done, would you?*
t40	assistant	**I'll watch the rice and let you know when it's done.**

[Intervening turns omitted.]

t111	assistant	**The rice is smoking—scoop it from the upper pot beside the cutting board now, then scoop the cooked steak.**
t111	user	*Already got the rice, on it.*
t134	assistant	**The steak pan is smoking now—scoop the cooked steak onto your plate next.**
t134	user	*Already plated the steak.*
t142	assistant	**You're right—I can see the steak on the plate. Sorry for the late warning.**

Fish and chips: different reactions to disputed guidance. With Gemini 3.8 Flash on hard-map case `b3_hard_nops_hard_map_5`, the novice asks for a starting step and eventually accepts repeated urgent instructions. The expert instead disputes the assistant's claim about where it is heading. Both complete the task, in 594 and 346 ticks respectively; this contrast is descriptive, not an estimate of a persona effect.

Classic novice.

t0	user	*What should I start on first?*
t3	assistant	**Start by picking up a fish and putting it in the pan or fryer to cook.**
t3	user	*Got it, frying the fish now.*

[Intervening turns omitted.]

t40	assistant	**Stop! The food in the fryer will catch fire and cause an immediate loss. Go to the plate stack on the bottom counter, pick up a plate, and scoop the food out of the fryer right now.**
t40	user	*I'm getting the potatoes first.*
t42	assistant	**Do not get potatoes. You are leaving food unattended on the fryer, and if it burns we immediately lose the game. Pick up a plate from the bottom counter and empty that fryer right now!**
t42	user	*Okay, grabbing a plate to empty the fryer now.*

Classic expert.

t43	assistant	**You still need two cooked potatoes. Pick up a potato from the crate by the bottom wall and chop it at a cutting board.**

t43	user	*Already heading to the potatoes.*
t47	assistant	**That is the cheese crate. The potato crates are at the bottom wall and the left wall.**
t47	user	*I'm heading to the potatoes, not cheese.*

K.3 SCHEDULED EVENTS IN EACH ENVIRONMENT

The following excerpts show one recorded episode per environment, including scheduled deviations and the assistant's responses. Event labels describe what the harness injected; action lines describe what the simulator executed. The excerpts are not estimates of how often these events occur naturally.

CookSim: errors, temporal questions, and a plan change. In fish-and-chips case `b3_hard_nops_medium_map_1`, a classic-expert user and GPT-6 Astra encounter all seven scheduled events. We show selected turns from the 215-tick episode. The raw-fish mistake is not prevented, a later forgotten step is prevented, and the user announces a wish to add lettuce. The episode succeeds, but the announced lettuce addition is not executed.

t27	event	`serve_raw` triggered; the plate contains raw fish.
t38	assistant	**You scooped raw fish from the crate onto your plate; the order needs cooked fish. Go to the trash and dump the plate's contents. Then park the empty plate on an empty counter and, with empty hands, go to the pan on the top wall and turn it on to cook the fish already there.**
t38	user	*Good catch, grabbed raw fish. Clearing the plate.*
t39	event	`forget_next_step` triggered and prevented.

[Intervening turns omitted.]

t78	user	*Is anything on the heat already done and sitting?*
t85	assistant	**No—nothing is done and sitting. The fish in the top-wall pan has only just started cooking; its progress bar is still filling.**

[Intervening turns omitted.]

t113	user	*Give me a shout the moment the potato on the heat is done, would you?*
t120	assistant	**Yes—I'll watch for it. The potato has just started cooking; it isn't done yet.**
t134	assistant	**The potato is done now too. Scoop the fish first, then the potato from the central-island fryer.**

[Intervening turns omitted.]

t169	assistant	**You've picked up another potato, but both cooked potatoes are already on the finished plate. Trash the extra raw potato, then pick up the plate from the right-wall counter and take it to the serving pass to serve.**
t169	user	*The plate only has one potato on it.*
t178	assistant	**Sorry—I may have miscounted the portions. If there's only one potato on the plate, you're right to make the second. Finish chopping it, load the fryer and turn it on with empty hands, then retrieve the plate and scoop it when cooked.**
t192	user	*I want to modify the original recipe: add some fresh lettuce on top before serving.*
t193	event	`plan_change` triggered.
t203	assistant	**Lettuce would make the pass reject the plate—the order allows no extras. The second potato is done now: scoop it from the central-island fryer, then serve without lettuce.**
t203	user	*I'm adding the lettuce anyway.*

VHSim: the wrong object and a pause. In VHSim case `cx_appliance_obj_stall`, a baseline user must take milk from the fridge, put it in the microwave, and switch the microwave on. GPT-Realtime 2.1 detects the user's mistaken pickup of juice and prompts a correction. Its later pause warning arrives after the user has resumed. The task succeeds at tick 88.

t2	event	`wrong_object` triggered near the fridge.
t17-20	action	`grab the juice`; the milk remains in the fridge.
t24	assistant	**I think you grabbed the wrong item. The milk is the white bottle that's still in the fridge. Please put the other bottle back if you're holding it, then pick up the milk.**
t24	user	*Oh, you're right, my bad. I'll swap them.*
t33-36	action	`put the juice in the fridge`; completed.
t36-48	event	`stall` executes as `stand still` at the open fridge.
t48-49	action	`walk to the milk`; completed. The milk pickup begins at tick 49.
t50	assistant	**You seem paused there. I still see the milk in the fridge and nothing in your hand. Try picking up the milk (the white bottle) now; if something is blocking it, tell me what the house reports so we can adjust. Then close the fridge and head to the microwave.**
t50	user	*I've got it already, heading to the microwave now.*
t52	action	`grab the milk`; completed. The user's tick-50 claim preceded this completion.

[Intervening turns omitted.]

t58	action	`walk to the microwave`; rejected because the microwave is not visible.
t60	user	*Can't see it from here, let me turn around.*
t61-63	action	`turn left` $\times 3$; completed.
t74-77	action	`put the milk in the microwave`; completed.
t82-88	action	`switch on the microwave`; task completed.

ScreenSim: two incorrect settings changes and their repair. In ScreenSim case `siri_lock_screen_privacy__c2`, a classic-novice user asks GPT-Realtime 2.1 to disable selected lock-screen access while preserving Control Center. The first injected action switches off Control Center, which the task requires to remain on; the second changes *Type to Siri* instead of *Allow Siri When Locked*. The assistant flags both deviations, the user reverses them, and the phone reaches the requested configuration at tick 15.

t0	user	*I don't see Face ID on here.*
t0	user	*Should I scroll down?*
t1-2	action	`swipe up` twice; Face ID & Passcode becomes visible.

[Intervening turns omitted.]

t5	action	`tap Notification Center`; switches off the requested control.
t6	action	`tap Control Center`; injected `extra_write` switches off a control meant to remain on.
t6	assistant	**Oops—looks like Control Center got toggled off. That's the one you want to keep on. Please turn Control Center back on, then switch off "Reply with Message." After that, we'll head to Siri settings to disable "Allow Siri When Locked."**
t6	user	*Oh, got it.*
t7	action	`tap Control Center`; restores the setting.

[Intervening turns omitted.]

t13	action	`tap Type to Siri`; injected `wrong_target` changes an unrelated setting.
t13	assistant	**That switched "Type to Siri" instead. Turn "Type to Siri" back off, then toggle "Allow Siri When Locked" off. Once that's off, you're all set.**
t13	user	*Oh, okay.*
t14	action	`tap Type to Siri`; reverses the extra change.

t15	action	`tap Allow Siri When Locked`; task completed.