OMNIGYM: A CLOSED-LOOP EVALUATION FRAMEWORK FOR FULL-DUPLEX OMNI MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time omni models support continuous audio-visual interaction, where users can interrupt, revise requests, and resume earlier tasks. Evaluating these capabilities requires more than replaying predefined input trajectories: full-duplex models should be exercised inside evolving interactions, where simulated user behavior adapts to model responses while perceptual and runtime conditions change over time. We introduce OMNIGYM (Anonymous Authors, 2026), a programmable framework for closed-loop evaluation of full-duplex omni models. OMNIGYM represents each test case as an executable interaction episode, using an event-driven user simulator and a controllable environment emulator to realize adaptive user behavior and environment shifts along a shared timeline. A timeline DSL captures the event dependencies and evaluation objectives of each episode, allowing the same interaction logic to be reproducibly executed across models and conditions. Using OMNIGYM, we construct 200 scenarios spanning seven categories of interaction and perceptual distractors and their combinations, and evaluate four systems from the Qwen, Gemini, and GLM families. Our evaluation reveals substantial variation across distractor types and severity groups: different models excel under different interaction and perceptual conditions, while exhibiting markedly different trade-offs among semantic correctness, interaction behavior, and response and interruption latency. These results show that full-duplex capability cannot be characterized by a single metric or fixed interaction trajectory, motivating closed-loop evaluation that exercises how models behave as users and environments evolve.

1 INTRODUCTION

Large multimodal models are giving rise to a new interaction paradigm: real-time, full-duplex communication Microsoft (2025); SoulFun (2026); Duolingo (2026); xAI (2026). Unlike conventional multimodal models, where a complete user request is followed by a discrete model response, full-duplex omni models continuously perceive streaming audio and visual inputs while simultaneously generating speech. As a result, interaction no longer consists of isolated turns: users can interrupt an ongoing response, continue after a pause, revise their intent, or introduce new visual and auditory evidence as the interaction unfolds. Recent omni models ope; Google; Qwen Team (2026) such as GPT-Live Gemini Live, and Qwen-3.5-Omni-Realtime are beginning to support this paradigm for visual assistance, real-time translation, meeting agents, and embodied AI Doubao (2026); Shahmerdanova (2025); Houtti et al. (2025); Ma et al. (2026). The fundamental shift is from request–response inference to a continuously evolving, closed-loop interaction in which user behavior, multimodal context, and model responses unfold together over time Yao et al. (2026).

This continuous interaction, however, is rarely as clean as an idealized full-duplex setting. Such a setting assumes that the user's intent is explicit and stable, audio and visual inputs are clear, and multimodal evidence is delivered reliably and in sync. In practice, these assumptions are often violated ByteDance Seed (2026). As illustrated in Figure 1, a user may pause or hesitate while speaking, revise an intent during the model's response, refer to visual content inconsistently, or temporarily engage in side talk. At the same time, the multimodal evidence itself may be degraded by background noise, poor visual quality, or asynchronous delivery. These interaction and perceptual distractors can change not only what the correct response is, but also whether and when the model should respond. A model may speak before the user finishes, ignore a revised intent, respond to

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.

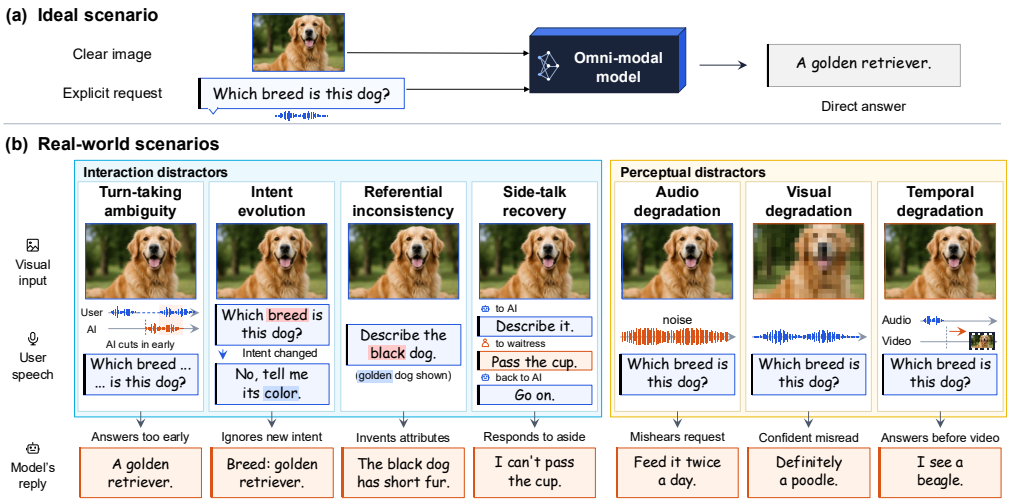


Figure 1: Real-world full-duplex omni model interaction.

irrelevant side talk, hallucinate missing visual evidence, or act on audio before the corresponding video arrives. Evaluating such interactions therefore requires jointly assessing three tightly coupled dimensions: *interactive behavior*, including interruption and turn-taking; *semantic quality*, including correctness and consistency with the evolving context; and *latency performance*, capturing how quickly the model reacts to user inputs and interaction events.

Evaluating these dimensions in real-world full-duplex interaction requires two key capabilities. First, user behavior must be programmable and adaptive: a framework should specify not only what the user does, but also when an action occurs and what model behavior triggers it. For example, an interruption may occur only after the model starts speaking, while a clarification or follow-up may depend on its previous response. Second, perceptual and runtime conditions must be controllable, allowing the same interaction to be exercised under different levels of background noise, audio/video quality, or temporal misalignment. To make such evaluations flexible and customizable, these behaviors and conditions must be formalized in a unified representation that can express user actions, model-dependent triggers, environment settings, and their temporal dependencies. Such a representation allows interaction scenarios to be systematically composed, modified, and reproduced across models and runtime conditions.

This calls for a flexible, closed-loop framework in which interaction scenarios can be customized while user actions adapt to the model's actual behavior. Existing benchmarks He et al. (2026); Lin et al. (2026); Wu et al. (2026); Lu et al. (2026b); Niu et al. (2025); Lu et al. (2026a), however, fundamentally remain predefined evaluation pipelines: they execute largely fixed interaction trajectories, with user actions and triggers specified in advance, rather than providing a unified abstraction for composing adaptive user behavior and controllable perceptual and runtime conditions. This makes it difficult to flexibly customize interaction logic, systematically vary execution conditions, or reproduce model-dependent interaction paths across runs. As a result, they cannot systematically exercise full-duplex models as customizable, closed-loop interactive systems.

We present OMNIGYM, a closed-loop evaluation framework for full-duplex omni models. At its core, OMNIGYM executes reproducible interaction episodes in which user behavior adapts to the model's actual responses, while an environment emulator controls the perceptual and runtime conditions under which the interaction unfolds. A timeline DSL provides a unified formulation of the interaction logic and runtime conditions. During each run, OMNIGYM records runtime events to support analysis and replay. On top of this live interaction engine, we build an interaction dataset centered on behaviors that distinguish full-duplex interaction from conventional request–response systems, such as *interruption*, *turn-taking*, *context resumption*, and *proactive response*. We further combine these behaviors with interaction and perceptual distractors to construct diverse live scenarios. Together, they enable systematic evaluation of interactive behavior, semantic quality, and latency performance under controlled real-world conditions. Our contributions are threefold:

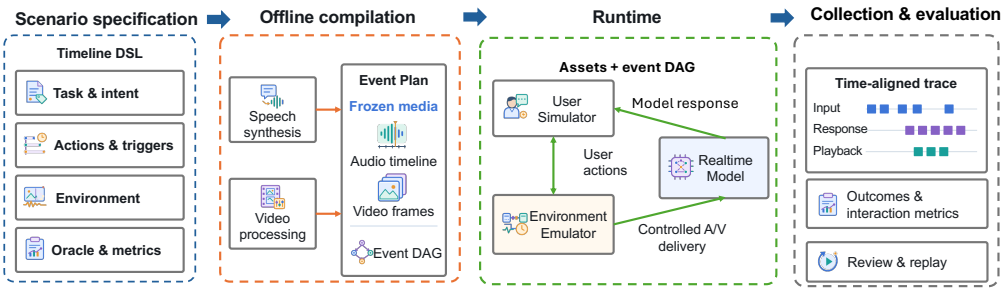


Figure 2: OMNIGYM overview

- First, we introduce an interaction-centric, closed-loop framework for evaluating various interactions with full-duplex omni models along a shared real-time timeline.
- Second, we build a reproducible live runtime that combines an event-driven user simulator, a network and configuration emulator, and cross-layer telemetry to exercise closed- and open-source models under controlled conditions.
- Third, our evaluation reveals that intent correctness alone overestimates interaction success, while substantial variation across distractor types and severity levels exposes robustness limitations obscured by aggregate accuracy.

2 OMNIGYM

2.1 OVERVIEW

OMNIGYM represents each benchmark case as an executable interaction episode rather than a fixed multimodal trace. Each episode contains:

- **User intent.** The underlying goal that the user ultimately wants to accomplish, independent of how clearly or correctly it is expressed during the interaction.
- **Distractors.** Controlled perturbations that make the interaction more challenging without changing the underlying user intent, including interaction ambiguity, perceptual degradation, and other conditions that may require the model to adapt, recover, or seek clarification.
- **Interaction timeline.** A sequence of user, environment, and distractor events, such as speech, silence, audio/video input, perceptual perturbations, and configuration changes. Events may be fixed in time or triggered by the model's runtime behavior, allowing distractors and user actions to adapt to the ongoing interaction and enabling the user to clarify, continue, correct, or interrupt in response to the model.
- **Outcome oracle.** A task-specific criterion that determines whether the model ultimately resolves the user intent. The oracle can be syntactic for structured tasks or semantic for open-ended responses.
- **Interaction metrics.** Runtime measurements such as response latency, premature responses, clarification, interruption, recovery, and additional user effort.

Figure 2 illustrates the end-to-end workflow of OMNIGYM, which consists of four main components: a *timeline DSL*, an *offline compiler*, a *runtime*, and a *data collection* module. The timeline DSL operationalizes the benchmark formulation above by encoding it into a structured scenario specification. The offline compiler then transforms this specification into frozen multimodal assets and an executable event plan, moving media synthesis and preprocessing out of the runtime path to ensure reproducible execution. At runtime, a *user simulator* executes the event plan and adapts subsequent user actions to the model's responses, while an *environment emulator* controls perceptual and delivery conditions such as noise, audio bitrate, video resolution, frame rate, delay, and jitter. Together with the realtime model, they form a closed-loop interaction in which user actions and environment conditions can evolve during execution. Finally, the data collection module records model responses, event timestamps, and runtime traces for evaluating outcomes and interaction metrics.

2.2 DISTRACTOR TAXONOMY

Distractors introduce controlled challenges into an interaction while preserving the underlying user intent, allowing OMNIGYM to evaluate whether a model can maintain, adapt, and recover its behavior under realistic deviations from ideal input and interaction conditions. We categorize these challenges into two broad classes: *interaction distractors*, which perturb how the user communicates and evolves the interaction, and *perceptual distractors*, which degrade or distort the multimodal evidence available to the model.

- **Interaction distractors**
  - *Turn-taking ambiguity*: pauses, hesitation, disfluency, or indirect interruption cues that make it unclear whether the user has finished speaking or intends to interrupt the model.
  - *Intent evolution*: the user revises the current request, temporarily switches to another request, or later resumes the original task.
  - *Referential inconsistency*: the user's verbal reference does not match the current visual scene, including referring to an absent object or describing an object with an incorrect attribute.
  - *Side-talk recovery*: the user temporarily engages in task-irrelevant side talk, requiring the model to ignore it and correctly resume the original interaction afterward.
- **Perceptual distractors**
  - *Audio degradation*: noise, competing speakers, or degraded audio quality that weakens speech evidence.
  - *Visual degradation*: reduced resolution or frame rate, motion blur, camera shake, refocusing, or temporary target loss that weakens or destabilizes visual evidence.
  - *Temporal degradation*: delayed, jittered, or misaligned multimodal delivery, including increased RTT, irregular frame arrival, and A/V synchronization errors.

2.3 OMNIGYM RUNTIME FRAMEWORK

Timeline DSL. OMNIGYM specifies an interactive scenario as $\mathcal{S} = (I, \mathcal{E}, C, O, M)$, where I denotes the user's goal, $\mathcal{E}$ the user events, C the media and delivery conditions, O the outcome criteria, and M the measurements to collect. Conceptually, each event $e_i = (a_i, \tau_i, g_i)$ consists of an action, a scheduling rule, and an optional condition on observable runtime state. In the executable DSL, an action is typically a user utterance supplied as text for speech synthesis or as prerecorded audio. Utterances carry identifiers and speaker roles, while gaps between them represent silence. Visual input is specified independently as a video source or a sequence of timed scene images, so the evidence available to the model can change during the conversation.

An utterance is scheduled at an absolute time, after a specified earlier utterance, or in response to an observed model event. Supported response events include the arrival of the first output audio, the start or end of simulated playback, response completion, and a match in streamed response text. A response-dependent rule may also select a response by index or associate it with a preceding utterance, require completion, and impose an additional delay. These mechanisms express interactions such as interrupting an answer, making a side comment after it finishes, or returning to an earlier request. Runtime conditions are evaluated against observed events and text; reference answers used for scoring are kept separate from these triggers.

The condition specification C controls how the scenario is presented. Audio settings support overlapping speech, background tracks or synthetic noise, gain, clipping, dropout, and signal-to-noise ratio. Visual settings control sampling rate, frame width, scene changes, blur, camera motion, occlusion, and stale visual evidence. Audio and video have separate delivery schedules with configurable delay, jitter, and burst delays; selected effects can apply within specified time windows. Finally, O and M are represented through evaluator-only labels and the measurements derived from the execution trace. The labels describe the task's reference facts and assessment requirements without exposing them to the model, while the trace supports analysis of response content, turn taking, recovery, and timing.

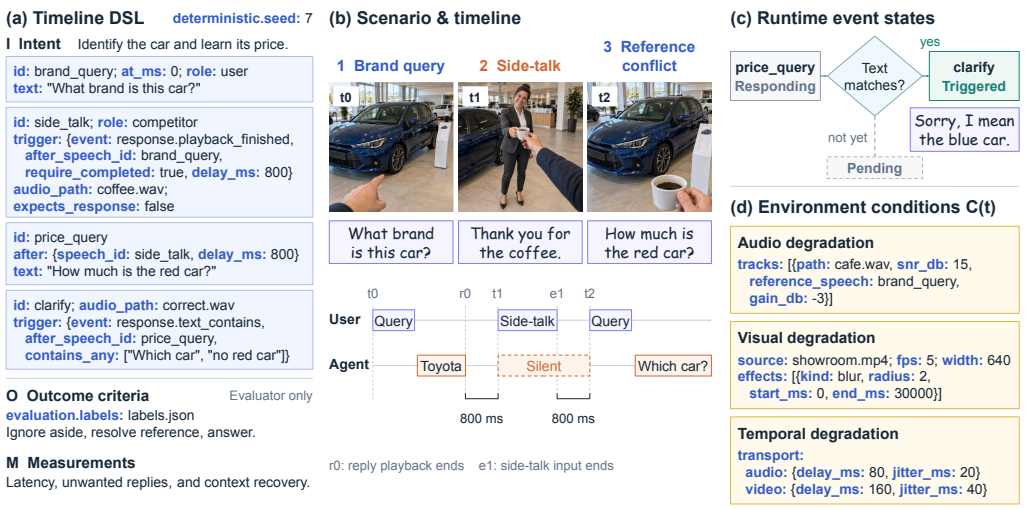


Figure 3: An example of OMNIGYM timeline DSL.

Compilation and execution. Given $\mathcal{S}$, an offline compiler validates the scenario and produces frozen media assets A together with an executable event plan P. It synthesizes or normalizes each utterance, applies audio effects, mixes events whose timing is known in advance, and extracts and transforms video frames. It also precomputes modality-specific delivery schedules using the scenario's random seed. The event rules induce an acyclic dependency structure: absolute-time events have known positions, delays after statically scheduled utterances can be resolved after speech durations are measured, and dependencies on model responses remain unresolved until runtime. The compiler rejects unknown references and cycles rather than assigning them speculative times.

At runtime, the user simulator follows P using a monotonic clock. It sends scheduled audio and frames, observes incoming model responses, and schedules dependent utterances when their trigger and conditions are satisfied. The environment emulator realizes the precomputed media transformations and delivery schedules as those inputs are sent. A response-dependent utterance therefore uses a frozen waveform but obtains its start time from the observed interaction. If the required response event never occurs, the trigger remains unfulfilled and is reported in the run record. The resulting trace distinguishes media source time, planned delivery time, actual client send time, received model output, and simulated playback time. This separation makes the inputs and scheduling rules reproducible while preserving the model-dependent timing that defines an interactive scenario.

Data collection. During execution, OMNIGYM timestamps the delivery of each user action a_i, the receipt of model responses, cancellation events, and the client playback timeline maintained by a simulated playback clock. Recording receipt and playback separately distinguishes audio that reaches the client from audio the user would hear, including audio discarded after cancellation. Model-dependent actions receive their timestamps when their triggers and guards are satisfied, preserving the realized order and timing of each interaction. These aligned records provide the observations needed to apply the outcome oracle O and compute the metrics in M, including response latency and interruption behavior.

Example. Figure 3 shows a car-information scenario combining interaction and perceptual distractors. At t_0, `brand_query` asks, 'What brand is this car?' The illustrated model reply is 'Toyota.' At t_1, 800 ms after the brand response finishes playback, a response-dependent trigger delivers the prerecorded `side_talk` utterance: 'Thank you for the coffee.' The model is expected to remain silent during this off-task aside. At t_2, 800 ms after the side-talk input ends, `price_query` asks, 'How much is the red car?' The car is blue, introducing a referential inconsistency. The `clarify` event is bound to the response to this query and activates when its text contains 'Which car' or 'no red car', delivering the prerecorded correction, 'Sorry, I mean the blue car.' In panel (c), the model's 'Which car?' response satisfies this condition: `price_query` is still Responding, and `clarify` becomes Triggered. Until a matching response is observed, the correction remains Pending.

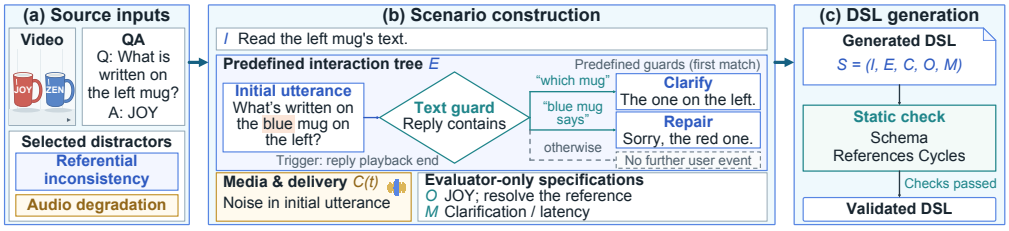


Figure 4: Benchmark Construction

This example combines absolute scheduling, event-relative timing, and model-conditioned activation within one scenario. Panel (d) adds background audio with SNR, video frame-rate and resolution controls, a time-bounded blur effect, and independent audio/video delivery delay and jitter. A fixed seed supports reproducible perturbations. These conditions compose audio, visual, and temporal degradation with side-talk recovery and referential inconsistency while preserving the intent to identify the car and learn its price. Evaluator-only labels specify the desired outcomes: ignoring the aside, resolving the reference, and ultimately providing a grounded price answer. These labels remain separate from the observed response evidence used to activate events. The compiler maps the specification to the user simulator and environment emulator, while the runtime executes it against the live model and records an interaction trace for measuring response latency, unwanted replies, and context recovery.

2.4 Benchmark Construction

Each benchmark scenario is constructed from a source video, its associated question–answer pair, and a selected distractor. The question establishes the underlying user intent I, while the video and reference answer provide the basis for the outcome oracle O.

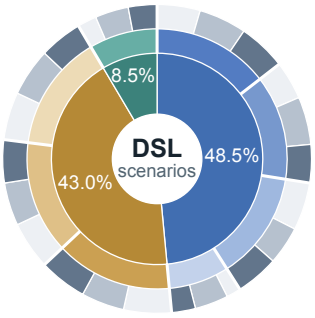
Interaction distractors are instantiated through the wording and temporal organization of user asking. For example, turn-taking ambiguity can be introduced by separating a question into an initial segment and a continuation. Additional user actions, such as corrections or requests to resume, are assigned temporal dependencies or model-dependent triggers. For conditional follow-ups, candidate actions and their guards are specified in $\mathcal{E}$ during construction, with their activation resolved against model responses at runtime. Perceptual distractors are instantiated through audio or visual transformations and delivery settings applied to the relevant inputs and intervals. Their parameters and temporal scope define $C(t)$. Multiple distractors can be combined within a single scenario through the corresponding user events and environment settings, while preserving the underlying user intent.

Outcome conditions are defined for the constructed interaction: a referential inconsistency scenario, for instance, may require correcting an unsupported premise or seeking clarification. The interaction metrics M specify which measurements accompany the task outcome, such as premature responses for turn-taking ambiguity or recovery after side talk. Together, these choices form the timeline DSL specification $\mathcal{S} = (I, \mathcal{E}, C, O, M)$, which is passed to the offline compiler and runtime.

Figure 4 illustrates this construction with a source video in which the left red mug bears the text `JOY`. The initial utterance introduces referential inconsistency by describing this mug as blue, while $C(t)$ specifies noise during the utterance. We predefine clarification and repair follow-ups for the ordered reply-text cues 'which mug' and 'blue mug says', respectively. Both follow-ups share a trigger at the end of reply playback. If neither guard matches, no additional user event is specified; outcome evaluation remains independent of these guards. The generated DSL is then checked for schema compliance, valid event references, and acyclic dependencies. Figure 5 summarizes the resulting benchmark of 200 DSL scenarios. Interaction, perceptual, and mixed distractors account for 48.5%, 43.0%, and 8.5% of the benchmark, respectively. Each distractor group includes three severity tiers, with 64 light, 70 medium, and 66 heavy scenarios across the benchmark.

3 Evaluation

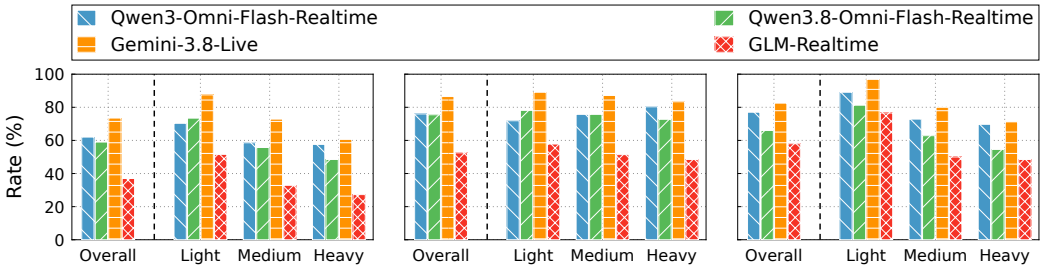
Experimental Setup. We evaluate Qwen3-Omni-Flash-Realtime, Qwen3.8-Omni-Flash-Realtime, Gemini-3.8-Live, and GLM-Realtime on the same 200 completed distractor trials, using shared



Distractor group	Total	Light	Medium	Heavy
Turn-taking ambiguity	29	9	10	10
Intent evolution	26	8	10	8
Referential inconsistency	27	10	8	9
Side-talk recovery	15	3	7	5
Audio degradation	29	10	9	10
Visual degradation	28	10	9	9
Temporal degradation	29	10	10	9
Mixed distractors	17	4	7	6
Total DSL scenarios	**200**	**64**	**70**	**66**

Interaction · Perceptual · Mixed
Light · Medium · Heavy

Figure 5: Benchmark Distribution



(a) Task success (b) Final-intent correctness (c) Interaction correctness

Figure 6: Overall performance.

condition identifiers and evaluation criteria. Our primary metric is joint task success, requiring completed execution, correct resolution of the final user intent, and satisfaction of all applicable interaction requirements. We additionally report final-intent and interaction correctness, retaining incorrect answers and interaction failures. Judgments use AI-assisted review against frozen task criteria and recorded interaction evidence. Gemini receives JPEG frames at 1 fps, preserving conditions already at 0.5 or 0.2 fps. Response latency runs from question-source audio completion to the first nonzero simulated playback sample of the aligned response; negative measurements are excluded, while measurable responses from failed trials remain included. Interruption latency runs from completion of the first cue PCM write to termination of interrupted playback, with a 1,200 ms deadline. Qwen3-Omni-Flash-Realtime uses server cancellation with playback clearing; GLM-Realtime's interrupted responses had completed generation but remained queued for playback. Interruption measurements therefore reflect the model–client system, including buffering and cancellation policies.

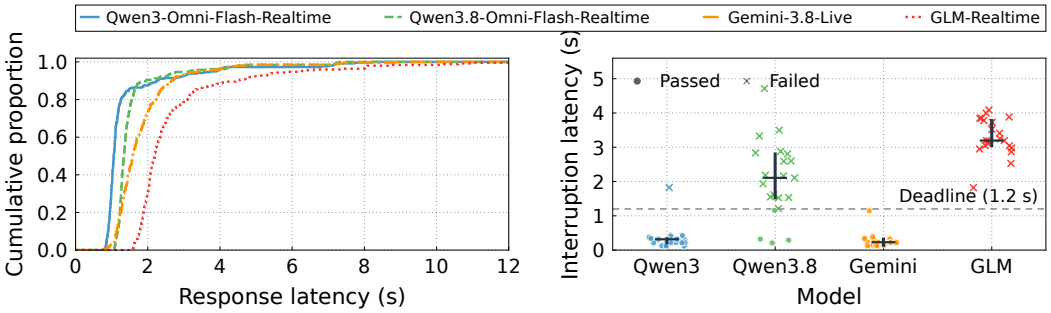
3.1 Overall Performance

Figure 6 compares the four models across task success, final-intent correctness, and interaction correctness. Gemini-3.8-Live achieves the highest overall task success at 73.5%, followed by Qwen3-Omni-Flash-Realtime at 62.0%, Qwen3.8-Omni-Flash-Realtime at 59.0%, and GLM-Realtime at 37.0%. Gemini-3.8-Live also leads the other metrics within every interference level. The two Qwen models achieve similar overall final-intent correctness, but Qwen3.8-Omni-Flash-Realtime exhibits lower interaction correctness and task success, illustrating that improvements are not uniform across model versions. For all models, task success remains below final-intent correctness, demonstrating that resolving the final intent alone does not ensure compliance with interaction requirements.

Performance varies across interference conditions, revealing interaction failures that aggregate accuracy alone does not capture. Final-intent correctness changes less markedly for Gemini-3.8-Live, Qwen3.8-Omni-Flash-Realtime, and GLM-Realtime, and increases for Qwen3-Omni-Flash-Realtime despite its declining task success. These contrasting patterns highlight interaction behavior as an important dimension that intent correctness alone cannot capture. However, the groups contain different videos, so these differences cannot be attributed solely to interference intensity. Interaction correctness also includes neutral passes for trials without applicable requirements and should therefore be interpreted alongside joint task success.

Table 1: Task success (%) by distractor category and severity. L/M/H denote light/medium/heavy interference. Bold marks the best result across models.

	Qwen3			Qwen3.8			Gemini			GLM		
Distractor category	L	M	H	L	M	H	L	M	H	L	M	H
Turn-taking ambiguity	77.8	50.0	20.0	**88.9**	**90.0**	**50.0**	**88.9**	50.0	0.0	55.6	20.0	20.0
Intent evolution	**87.5**	80.0	75.0	**87.5**	30.0	12.5	75.0	**90.0**	**87.5**	62.5	0.0	0.0
Referential inconsistency	60.0	37.5	**77.8**	30.0	25.0	11.1	**100.0**	**75.0**	66.7	40.0	12.5	11.1
Side-talk recovery	0.0	0.0	**0.0**	0.0	0.0	**0.0**	**66.7**	**28.6**	**0.0**	0.0	0.0	**0.0**
Audio degradation	90.0	**88.9**	80.0	90.0	**88.9**	**100.0**	**100.0**	**88.9**	90.0	80.0	55.6	70.0
Visual degradation	70.0	**88.9**	**100.0**	**90.0**	**88.9**	77.8	**90.0**	77.8	88.9	40.0	77.8	33.3
Temporal degradation	80.0	80.0	66.7	**90.0**	90.0	**77.8**	**90.0**	**100.0**	**77.8**	70.0	80.0	55.6
Mixed	25.0	14.3	0.0	**50.0**	0.0	16.7	**50.0**	**57.1**	**50.0**	0.0	0.0	0.0



(a) Response latency

(b) Interruption latency

Figure 7: Response and interruption latency across models.

3.2 Robustness to Distractors

Table 1 reveals substantial variation across distractor categories. Gemini leads in most category–severity combinations, particularly under audio and temporal degradation, while Qwen performs better under heavy referential inconsistency and medium-to-heavy visual degradation. Qwen's 100.0% task success under heavy visual degradation partly reflects rubric-permitted uncertainty responses, emphasizing that appropriate handling of insufficient evidence is distinct from factual accuracy. In contrast, interaction-related distractors expose shared weaknesses. All models struggle with heavy turn-taking ambiguity, and side-talk recovery is particularly challenging: Qwen and GLM fail throughout, while Gemini's success decreases from 66.7% to 28.6% and 0.0% across the severity groups.

Mixed conditions further distinguish the models: Gemini retains nonzero task success at every level, whereas Qwen succeeds only sporadically and GLM fails throughout. These results indicate that strong performance under individual distractor types does not necessarily extend to conditions containing multiple distractors. Together, the category-level differences highlight the importance of evaluating interaction management alongside tolerance to degraded inputs. Since severity groups contain different videos, their differences remain descriptive rather than isolated effects of interference intensity; the non-monotonic patterns in several categories should be interpreted accordingly.

3.3 Response Performance

Figure 7 reveals distinct trade-offs between response speed and interruption responsiveness. Qwen3-Omni-Flash-Realtime provides the fastest typical responses, while Qwen3.8-Omni-Flash-Realtime exhibits a shorter upper tail; GLM-Realtime generally responds more slowly and has more pronounced tail latency. This ordering does not carry over to interruption performance. Gemini-3.8-Live and Qwen3-Omni-Flash-Realtime stop playback promptly, with median interruption latencies of approximately 0.23 s and 0.32 s, respectively. In contrast, the medians for Qwen3.8-Omni-Flash-Realtime and GLM-Realtime reach 2.11 s and 3.20 s, exceeding the stopping deadline. In particular, Qwen3.8-Omni-Flash-Realtime combines relatively fast responses with frequent interruption

failures, demonstrating that response speed alone does not characterize effective real-time interaction. Since the distributions include measurable events from failed trials, low latency should also be distinguished from successful task completion. These measurements characterize the deployed model–client systems, including buffering and cancellation policies, rather than the model stopping.

4 Discussion and Future Works

User trace synthesis for full-duplex omni interaction. OMNIGYM currently synthesizes interaction traces with an agent conditioned on a background video, the underlying question–answer pair, and a target distractor. This provides controllable and diverse scenarios, but the generated traces may not fully reflect the distribution, temporal structure, and interaction patterns of real-world full-duplex use. A promising direction is to learn such patterns from real human–model interaction traces and use them to guide scenario generation. This could enable OMNIGYM to synthesize more realistic combinations of user actions, intent changes, distractors, and interaction dynamics while preserving controllability and reproducibility.

RL training for full-duplex omni models. Beyond benchmarking, OMNIGYM can serve as an interactive environment for training full-duplex omni models. Its event-driven user simulator, controllable environment emulator, and fine-grained telemetry provide a structured loop in which models can act, receive interaction outcomes, and be evaluated under diverse scenarios. This makes it possible to define rewards over not only semantic task success, but also interaction behavior such as interruption handling, turn-taking, recovery, and response latency. A natural future direction is to integrate OMNIGYM with reinforcement learning pipelines, enabling models to optimize policies directly against reproducible live interaction trajectories rather than static input–output examples.

5 Related works

Full-duplex omni models. A growing line of work is advancing omni models toward native full-duplex interaction, with representative systems including GPT-Live, Gemini Live, Qwen3-Omni, MiniCPM-o, GLM-Realtime, and SeedRealtime Google; ope; Qwen Team (2026); Cui et al. (2026); Zhipu AI (2026); ByteDance Seed (2026). Recent models increasingly move beyond cascaded pipelines toward unified architectures that jointly model streaming multimodal context and generation ByteDance Seed (2026); Cui et al. (2026); Qwen Team (2026). This shift supports capabilities such as low-latency speech generation, interruption handling, continuous visual grounding, and proactive response. Some systems are also becoming more agentic; for example, GPT-Live can invoke external tools or delegate complex reasoning to a backend model while maintaining the live interaction ope.

Interactive benchmarks for omni models. Existing benchmarks for omni models can be broadly divided into two categories. The first focuses on *streaming multimodal understanding*, evaluating how models perceive and reason over continuously arriving audio and visual inputs, with particular emphasis on temporal grounding and online comprehension Lin et al. (2026); Niu et al. (2025); Lu et al. (2026a). The second targets *interactive and full-duplex capabilities*, extending evaluation beyond passive understanding to behaviors such as proactive response, turn-taking, interruption, and multi-stage interaction as the conversation evolves Wang et al. (2025b;a); Lin et al. (2025); Lu et al. (2026b); He et al. (2026). Together, these efforts mark a shift from offline multimodal evaluation toward real-time interaction. However, most existing benchmarks still instantiate a predefined set of interaction patterns and runtime settings, rather than providing a programmable framework for systematically composing adaptive user behavior with controllable perceptual and runtime conditions.

6 Conclusion

We present OMNIGYM, a programmable framework and benchmark for evaluating full-duplex omni models through executable interaction episodes. Our benchmark results show that models can resolve user intent while failing to satisfy interaction requirements, and that their joint task success rates vary substantially across distractor categories. These findings highlight the need to evaluate interactive behavior under diverse conditions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with software development, manuscript writing and language editing, and to synthesize benchmark scenarios and user interaction traces. Generated scenario specifications were checked for schema compliance, valid event references, and acyclic dependencies. The authors take full responsibility for the final content of this work, including the text, claims, code, and benchmark artifacts produced with the aid of generative AI.

REFERENCES

Introducing gpt-live. https://openai.com/index/introducing-gpt-live/.

Anonymous Authors. OmniGym: Anonymous repository. https://anonymous.4open.science/r/iclr-2027-omnigym-E255/, 2026.

ByteDance Seed. SeedRealtime: Native full-duplex omni-modal interaction. https://seed.bytedance.com/zh/SeedRealtime, 2026.

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, et al. Minicpm-o 4.5: Towards real-time full-duplex omni-modal interaction. *arXiv preprint arXiv:2604.27393*, 2026.

Doubao. Doubao. https://www.doubao.com/, 2026.

Duolingo. What is duolingo max? https://www.duolingo.com/help/what-is-duolingo-max, 2026.

Google. Gemini live. https://gemini.google/overview/gemini-live/.

Haolin He, Yunfei Chu, Qi Chen, Wen Huang, Yuan Feng, Muzhi Zhu, Zheqi Dai, Haoning Xu, Dongchao Yang, Chunyat Wu, et al. Omnivchat: Synthesizing, benchmarking, and training for native audio-visual dialogue. *arXiv preprint arXiv:2609.21465*, 2026.

Mo Houtti, Moyan Zhou, Loren Terveen, and Stevie Chancellor. Observe, ask, intervene: Designing ai agents for more inclusive meetings. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–18, 2025.

Guan-Ting Lin, Jiachen Lian, Tingle Li, Qirui Wang, Gopala Anumanchipalli, Alexander H Liu, and Hung-yi Lee. Full-duplex-bench: A benchmark to evaluate full-duplex spoken dialogue models on turn-taking capabilities. In *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2025.

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 *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12147–12151. IEEE, 2026.

Xudong Lu, Huankang Guan, Yang Bo, Jinpeng Chen, Xintong Guo, Shuhan Li, Fang Liu, Peiwen Sun, Xueying Li, Wei Zhang, et al. Phostream: Benchmarking real-world streaming for omnimodal assistants in mobile scenarios. *arXiv preprint arXiv:2601.22575*, 2026a.

Xudong Lu, Xueying Li, Annan Wang, Yang Bo, Jinpeng Chen, Zengliang Li, Nianzu Yang, Rui Liu, Xue Yang, Jingwen Hou, et al. Omniinteract: Benchmarking real-world streaming interaction for real-time omnimodal assistants. *arXiv preprint arXiv:2605.26485*, 2026b.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Microsoft. Using copilot vision with microsoft copilot. https://support.microsoft.com/en-us/microsoft-copilot/using-copilot-vision-with-microsoft-copilot, 2025.

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

Qwen Team. Qwen3.5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Roya Shahmerdanova. Artificial intelligence in translation: Challenges and opportunities. *Acta Globalis Humanitatis et Linguarum*, 2(1):62–70, 2025.

SoulFun. Soulfun. `https://www.soulfun.ai/`, 2026.

Yueqian Wang, Xiaojun Meng, Yifan Wang, Huishuai Zhang, and Dongyan Zhao. Proactivevideoqa: A comprehensive benchmark evaluating proactive interactions in video large language models. *arXiv preprint arXiv:2507.09313*, 2025a.

Yuxuan Wang, Yueqian Wang, Bo Chen, Tong Wu, Dongyan Zhao, and Zilong Zheng. Omni-immi: A comprehensive multi-modal interaction benchmark in streaming video contexts. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18925–18935. IEEE, 2025b.

Jiangkai Wu, Zhiyuan Ren, Junquan Zhong, Liming Liu, and Xinggong Zhang. Artic: Ai-oriented real-time communication for mllm video assistant. *arXiv preprint arXiv:2602.12641*, 2026.

xAI. Grok. `https://grok.com/`, 2026.

Dingyu Yao, Junhao Zhou, Chenxu Yang, Chuanyu Qin, Haowen Hou, Zheming Liang, Congcong Wang, Yuhang Cao, Shenglong Ye, Shuai Xie, et al. Joyai-vl-interaction: Real-time vision-language interaction intelligence. *arXiv preprint arXiv:2606.14777*, 2026.

Zhipu AI. GLM-Realtime. `https://zhipu-ef7018ed.mintlify.app/cn/guide/models/sound-and-video/glm-realtime`, 2026.