STREAMMIND: PROACTIVE VIDEO AGENTS FOR EVOLVING USER INTENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models are shifting from offline passive understanding toward online proactive interaction. However, prevailing methods largely rely on explicit requests while miss the implicit user intent (*e.g.,*"where should I click next?" while a user navigate a website), meanwhile fail to update the state simultaneously as user requests might update continuously. These limitations make them less suited to intent-aware interaction in practical. To address these challenges, we introduce **StreamMind-Bench**, a new benchmark serving as testbed for user intent interaction, with 4.1K evaluation samples and 80K training samples spanning explicit, implicit, and dynamic intents. Building on this benchmark, we develop **StreamMind-VL**, a streaming proactive model that infers implicit user intents from continuous observations and dynamically tailors its interactions as these intents evolve. We further introduce **StreamMind-Harness**, which maintains user intents as persistent states with a fleixble lifecycles via a hierarchical memory to track the latest user intent and environmental state. Our experiments reveal three key findings: *(i)* StreamMind-VL preserves strong general video understanding while improving response timing and answer quality by 10.7% over other baselines; *(ii)* existing proactive models remain limited in understanding user intent and deciding when to respond, especially under implicit and dynamic intents; and *(iii)* StreamMind-Harness further improves response consistency and adaptation over sustained interaction, resulting in a 12.3% gain in proactive interaction.

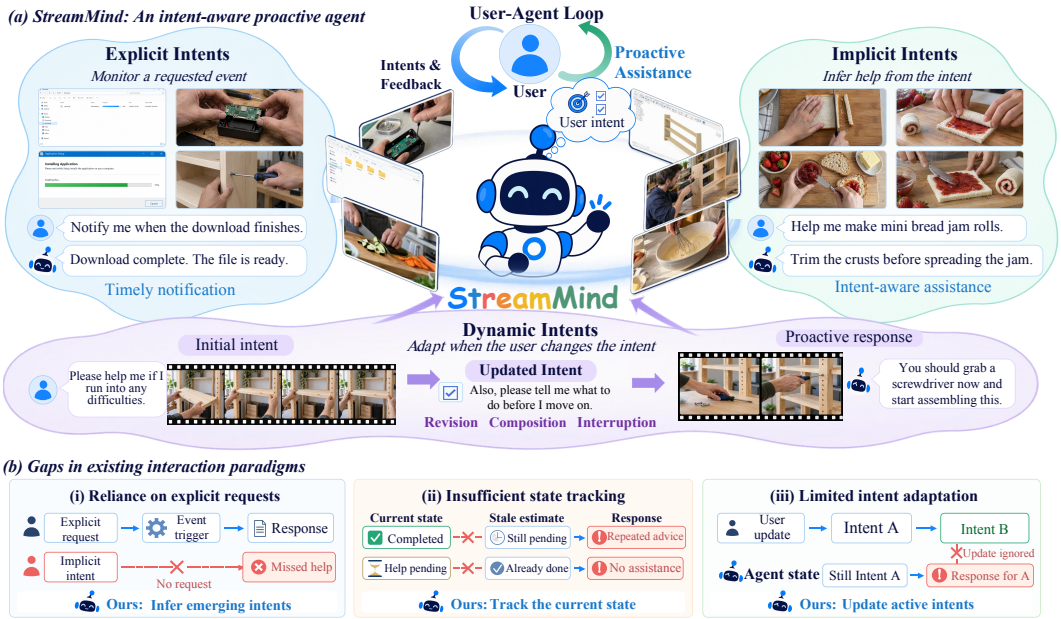


Figure 1: **Overview of StreamMind.** (a) StreamMind studies sustained proactive interaction under explicit, implicit, and dynamically evolving user intents, where the agent continuously observes the environment and decides when and how to assist. (b) Existing paradigms rely on explicit intents, with limited tracking of interaction state and evolving intents. StreamMind addresses these limitations through proactive streaming policy learning and persistent interaction state modeling that tracks task progress and updates active intents over time.

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.

1 INTRODUCTION

"Mind is activity, a range of dynamic processes of interaction between organism and world."

— Stanford Encyclopedia of Philosophy, John Dewey

Vision-language models (Lu et al., 2026; Wang et al., 2026) are advancing from offline content understanding (Wang et al., 2025c; Zhu et al., 2025) toward continuous proactive assistance in real-world streaming scenarios (Cui et al., 2026; Wang et al., 2025d; Orantqing et al., 2026). However, existing proactive interaction approaches are largely designed for explicit queries (e.g. "Notify me when a goal is scored."), and isolated, single-turn requests (Li et al., 2026a; Kundu et al., 2026), as illustrated in Figure 1(i).

However, these paradigm is less suited to real-world open-ended settings across three challenges, where ***(i)*** *implicit user intents* (e.g. "Help me finish this task whenever I get stuck.") must be inferred from ongoing observations and interactions. Beyond identifying user intent, effective assistance requires ***(ii)*** *task-relevant state tracking* As shown in Figure 1(ii), prior interactions can change the environment and task progress, making a response that was appropriate earlier unnecessary or incorrect later. Although existing streaming models can retain long interaction histories (Yao et al., 2026a; Wang et al., 2025a), they still struggle to maintain an up-to-date state, resulting in repeated, outdated, or missed responses. Lastly, as shown in Figure 1(iii), ***(iii)*** *user intents is evolving* as the interaction unfolds (Tack et al., 2026; Tao et al., 2026; Yang et al., 2026d). Later feedback may introduce new goals, revise existing constraints, or shift the current focus, while existing methods often fail to update which intents remain relevant and instead continue to rely on earlier requests. These challenges make proactive interaction more than a problem of real-time long-context modeling: it requires the agent to continuously infer and update user intent while tracking the evolving interaction state. This motivates three key research questions forward "**Intent-aware interaction**":

How can we construct a testbed for proactive interaction with implicit user intent?
How can such intent-aware interaction be internalized as a native model capability?
How to flexible manage context to track environmental and user intent changes?

In this work, we address these three questions through benchmarking, modeling, and agentic design. Firstly, we introduce StreamMind-Bench, a unified intent-aware benchmark for sustained proactive interaction with 4.1K evaluation samples and 85K training samples spanning explicit, implicit, and dynamic user intents. Building on this benchmark, we further develop StreamMind-VL, a streaming proactive model that infers implicit user intents and adapts its interactions as these intents evolve. Streaming supervised fine-tuning (SFT) jointly learns when and how to assist, while localized trajectory reinforcement learning (RL) focuses optimization on interaction-critical moments to improve response timing and quality. Lastly, we further develop StreamMind-Harness, which maintains user intents as persistent, updatable states with flexible lifecycles, while hierarchical memory retrieves relevant history to keep subsequent decisions aligned with the current interaction context and user intent. Our main contributions are as follows:

- To the best of our knowledge, we are the first work to study user intents in proactive streaming interaction. To this end, we construct StreamMind-Bench, a unified benchmark with 4.1K evaluation and 85K training samples spanning explicit, implicit, and dynamic intents.
- We introduce StreamMind-VL, a proactive streaming video model that jointly learns when and how to assist under implicit and evolving user intents.
- We develop the StreamMind-Harness, which maintains updatable intent states with flexible lifecycles and hierarchical memory for consistent behavior over sustained interaction.

2 RELATED WORK

Streaming Video Benchmark. Early video benchmarks focus on offline understanding (Fu et al., 2025a; Zhao et al., 2025), while online benchmarks evaluate sequential perception and response before video completion (Li et al., 2025b; Huang et al., 2025). Recent proactive benchmarks further require models to decide when to respond, often using annotated trigger points and extending to egocentric or task-guidance settings (Wang et al., 2025e; Kundu et al., 2026; Ran et al., 2026). However, most focus on a single intent type or environment and evaluate responses independently.

Table 1: **Comparison with other proactive interaction benchmarks.** StreamMind-Bench covers Explicit, Implicit, and Dynamic Intents across both physical and digital environments, and supports both Pointwise and Trajectory evaluation. ✓, ▲, and ✗denote full, partial, and no support, respectively. Dur. denotes the average sample duration in minutes; Evid. and Traj. denote the numbers of evidence points and evaluation trajectories.

Benchmark	User Intent Type			Dur. (min)	Evid.	Environment		Eval Setting		Traj.
	Explicit	Implicit	Dynamic			Physical	Digital	Pointwise	Trajectory	
ProactiveVideoQA (Wang et al., 2025e)	✓	✗	✗	2.1	3.5K	✓	✗	✗	✓	1.4K
EgoProactive (Kundu et al., 2026)	▲	✓	✗	3.5	5.4K	✓	✗	✓	✗	700
EgoPro-Bench (Ran et al., 2026)	✓	✓	✗	3.0	2.4K	✓	✗	✗	✓	2.4K
IPIBench (Li et al., 2026b)	✓	✗	✓	1.0	3.7K	✓	✗	✓	▲	1.8K
EgoServe (Sitong et al., 2026)	▲	✓	▲	6.2	3.4K	✓	✗	✗	✓	293
GuideMe (Liu et al., 2026)	✓	✓	✗	5.5	47.8K	✓	✗	✓	▲	2.5K
GUIDE (Yang et al., 2026c)	✓	✓	▲	33.8	0.7K	✗	✓	✓	✗	120
StreamMind-Bench (Ours)	✓	✓	✓	**8.3**	**20.5K**	✓	✓	✓	✓	**4.1K**

StreamMind-Bench instead unifies explicit, implicit, and dynamic intents across physical and digital tasks, with both pointwise and trajectory evaluation.

Proactive Interaction Models. Online video-language models support continuous perception and low-latency responses (Zeng et al., 2025; Cui et al., 2026), while real-time and task-guidance systems further learn when to speak or provide assistance during ongoing interaction (Sitong et al., 2026; Liu et al., 2026; Lu et al., 2026). However, most existing approaches center on explicitly specified requests or task objectives, with limited support for tracking interaction state and adapting to evolving user intent over time (Tack et al., 2026; Tao et al., 2026). StreamMind-VL targets open-ended proactive interaction, where intents must be inferred from user goals, environment changes, and task progress, and updated throughout the interaction.

Memory and Agent Harness. Agentic systems increasingly incorporate persistent memory (Xu et al., 2025), planning (Xiong et al., 2025), reflection (Yuan et al., 2025), and reusable skills (Zheng et al., 2025) to preserve and reuse historical context over long-horizon tasks. Recent memory systems further study how to retrieve, integrate, and abstract historical information into reusable experience for long-term interaction (Yu et al., 2026; Chhikara et al., 2025). However, these methods mainly organize and access past information without explicitly tracking evolving user intents and task progress, which can cause proactive responses to drift from the current interaction state. StreamMind-Harness addresses this gap by maintaining user intents and task progress as persistent, updatable interaction states and retrieving relevant history before each decision.

3 StreamMind-Bench: Intent-aware Proactive Benchmark

3.1 Task Formulation

We formulate proactive assistance as a causal streaming decision process. The agent observes a video stream $\mathcal{V} = \{v_t\}_{t=1}^{T}$ while the user may express one or more intents i_j over time, explicitly or implicitly, possibly before earlier intents are fulfilled. Each intent becomes available to the agent at time τ_j. Let $\mathcal{I}_{\leq t}$ denote all intents available by time t, and $\mathcal{A}_t \subseteq \mathcal{I}_{\leq t}$ the currently active ones. At each timestep, the agent accesses only $c_t = \langle \mathcal{V}_{\leq t}, \mathcal{I}_{\leq t}, \mathcal{H}_{<t} \rangle$ and predicts $a_t = \pi(c_t) \in \{\text{SILENCE}, \text{RESPONSE}\}$. If responding, it generates y_t for some $i_j \in \mathcal{A}_t$, without access to future video or user inputs.

Unlike retrospective VideoQA, relevant evidence may emerge only after an intent is expressed. Each intent i_j can correspond to one or more evidence points ξ_{jk}, each with a valid response window $[s_{jk}, e_{jk}]$ and reference response $\hat{y}_{jk}$. An evidence point is covered if the agent responds within its window with a semantically correct response to the corresponding intent. An intent remains active until all its evidence points are covered. The agent must therefore determine both when to respond and what to respond, while avoiding premature, missed, repeated, or irrelevant responses.

3.2 Hierarchical Difficulty

StreamMind-Bench characterizes proactive interaction along two dimensions: *intent explicitness* and *intent dynamics*. Existing benchmarks primarily focus on **(i) Level 1: Explicit Intents**, where the user specifies a condition, event, or state to monitor, and the agent responds when it occurs, e.g. Let me know when the water starts boiling. We further introduce **(ii) Level 2: Implicit Intents**, where the user provides only a high-level goal, requiring the agent to infer when and how to assist from ongoing observations and task progress, e.g. Help me keep an eye on this dish. Finally, **(iii) Level 3: Dynamic Intents** considers intents that evolve throughout the interaction. As the task

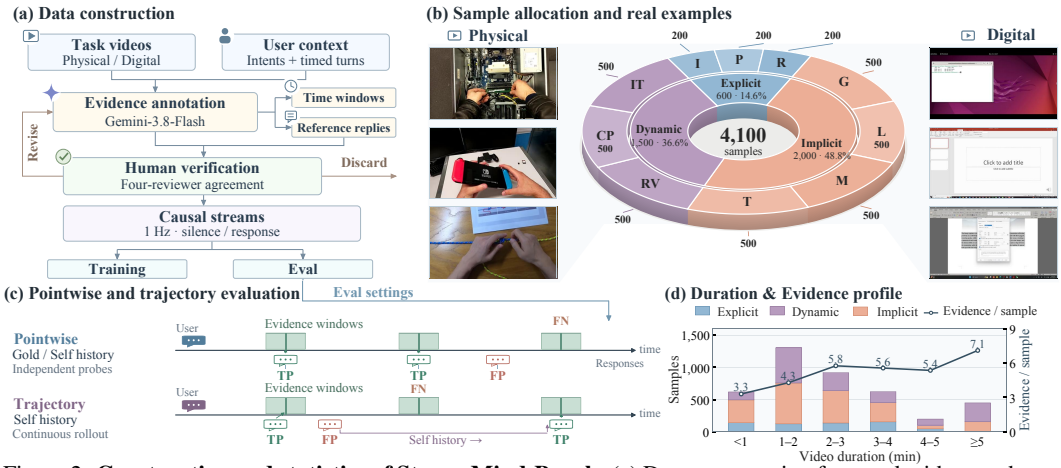


Figure 2: **Construction and statistics of StreamMind-Bench.** (a) Data construction from task videos and user context to evidence annotation, human verification, and causal stream generation. (b) Sample allocation across three requirement types, with examples from physical and digital environments. (c) Pointwise and trajectory evaluation settings for local response decisions and continuous interaction behavior. TP: true positives; FP: false positives; FN: false negatives. (d) Joint distribution of video duration, sample count, and average number of evidence points per sample.

unfolds, the user may provide multiple new inputs, requiring the agent to continually update the currently valid task state. We consider three common forms: *Revision (RV)*, which modifies an existing intent; *Composition (CP)*, which adds a new intent or constraint while preserving existing ones; and *Interruption (IT)*, which introduces an immediate request while the ongoing task remains active. These changes may occur repeatedly and in combination within a single interaction.

3.3 Benchmark Construction

Task Collection. We collect task videos from both physical and digital environments to cover two complementary forms of proactive interaction. In (i) *physical* environments, assistance mainly relies on understanding continuous visual processes, including user activities, task progress, and emerging intents. In contrast, user intent in (ii) *digital* environments is often implicit in interaction trajectories such as navigation, content editing, and tool use, requiring the agent to infer user goals, constraints, and task state from the current interface and interaction history. The two settings thus capture user intent through physical behavior and digital interaction, respectively, covering proactive assistance across different interaction environments. Specifically, physical tasks mainly include egocentric daily activities and task execution, while digital tasks focus on GUI and computer-use scenarios.

Annotation and Verification. For each video-request pair, we use Gemini-3.8-Flash (Google DeepMind, 2026) to clean the data and identify candidate help points with response windows, preferred times, and reference answers. For dynamic intents, we further construct timestamped multi-turn interactions in which intents may be revised, extended, or interrupted as the task unfolds. Four reviewers then independently verify the user requests and response timing, correcting or removing samples without sufficient evidence and retaining only those approved by all reviewers. The validated annotations are aligned to 1 FPS streams and converted into causal interaction sequences for training and evaluation. Details are provided in Appendix C.3.

3.4 Data Statistics

Categories and Scale. Building on the three proactive interaction settings in Section 3.2, we organize StreamMind-Bench into finer-grained task subtypes, as illustrated in Figure 2(b). The 600 *Explicit Intents* samples span Single Event (I), Transition (P), and Recurring Event (R). For *Implicit Intents*, 2,000 samples cover Guidance (G), Learning (L), Monitoring (M), and Tool (T). The remaining 1,500 *Dynamic Intents* samples consist of Revision (RV), Composition (CP), and Interruption (IT). Subtypes are evenly distributed within each category. In total, StreamMind-Bench contains 4,100 evaluation samples, 20,546 help points, and 183.4 hours of video. Additional statistics and detailed explanations are provided in the Appendix C.

Training Data Curations. We construct the StreamMind-85K training set, consisting of 70K samples for supervised fine-tuning and 15K tasks for reinforcement learning. During supervised

fine-tuning, we further incorporate 768K general streaming-video samples from sources including TimeChat-Online (Yao et al., 2025) and VideoChat-Flash (Li et al., 2025a) to improve general streaming-video modeling. Detailed components are provided in Appendix C.4.

3.5 Evaluation Protocol

Settings. As illustrated in Figure 2(c), two complementary evaluation settings are adopted to distinguish local proactive decision-making from sustained interaction behavior.

(i) Pointwise Evaluation samples positive and negative timestamps around annotated help points and independently predicts whether the model should respond. Both *Gold History* and *Self History* are considered, where the former isolates local decision ability and the latter measures the effect of model-generated history. Independent evaluation avoids error accumulation across time.

(ii) Trajectory Evaluation runs the model continuously over the full video stream, with each decision conditioned on prior observations and self-generated interaction history. This setting captures error propagation, requirement tracking, and repeated or missed responses. Together, the two settings evaluate local decisions under a given history and sustained interaction under self-generated history.

Metrics. We evaluate models along two dimensions: response timing and response quality. Timing is measured by Precision, Recall, F1, and False Positive Rate (FPR), capturing response accuracy, coverage, and false activations. Quality is evaluated using Answer Score for overall correctness and relevance, and Content F1 for key-content alignment with reference answers.

4 StreamMind: Grounding Evidence, Tracking Intent

Overview. As shown in Figure 3, the formulation in Section 3.1 highlights two key challenges in proactive interaction. **(i) Evidence Grounding.** The policy π must ground each active intent i_j in emerging visual evidence ξ_{jk} and determine when and how to respond. We address this with a dedicated training framework that directly improves the streaming video policy. **(ii) State Tracking.** As intents evolve, maintaining an up-to-date active set $\mathcal{A}_t$ and interaction history $\mathcal{H}_{<t}$ becomes challenging for the policy alone. We therefore introduce the StreamMind-Harness ϕ, which explicitly maintains these interaction states and provides π with continuously updated context, $\tilde{c}_t = \phi(c_t, \mathcal{A}_t)$. This separates video-grounded response decisions from persistent state tracking: the Harness tracks dynamic intents and task progress, while response decisions and generation remain with π. Details of the framework and workflow are provided in Appendix D.

4.1 StreamMind-Harness Framework

Interaction State. The Harness maintains an explicit interaction state $\mathcal{S}_t = \langle \mathcal{A}_t, \mathcal{P}_t, \mathcal{R}_t \rangle$, where $\mathcal{A}_t$ denotes the currently active intents, $\mathcal{P}_t$ captures task progress and environment state, and $\mathcal{R}_t$ records previously delivered responses. As new user inputs and observations arrive, the Harness incrementally updates $\mathcal{S}_{t-1}$ to $\mathcal{S}_t$ by adding, revising, or terminating intents and updating task progress. State changes are applied only when supported by sufficient interaction or perceptual evidence, avoiding premature updates from uncertain observations.

Intent Lifecycle. Each intent i_j is maintained as an independent stateful instance, allowing multiple intents to remain active and evolve concurrently. The Harness tracks its current status, relevant evidence, and completion state, and updates the lifecycle as the interaction unfolds. Persistent intents may remain active across multiple events, whereas one-time intents are closed once satisfied. Importantly, the lifecycle only determines which intents remain valid; when and how to respond are still decided by the agent policy π.

Hierarchical Memory. The Harness organizes past observations and interactions into hierarchical memory $\mathcal{M}_{<t}$, preserving relevant history without continuously expanding the agent context. Recent information is retained in fine-grained form, while older content is progressively compressed to preserve task-relevant entities, actions, constraints, and state changes. Given the active intents $\mathcal{A}_t$ and task state $\mathcal{P}_t$, the Harness retrieves relevant history as $m_t = \mathrm{Retrieve}(\mathcal{M}_{<t}, \mathcal{A}_t, \mathcal{P}_t)$, prioritizing recent and task-relevant information. The retrieved memory serves as additional evidence for the current decision without directly modifying the persistent interaction state.

State Integration. Before each decision, the Harness integrates the persistent interaction state and retrieved memory with the causal context to obtain $\tilde{c}_t = \phi(c_t, \mathcal{S}_t, m_t)$, which is then passed to the agent policy π for response decision and generation. After a response is delivered, the resulting interaction is written back to the state and memory for subsequent decisions.

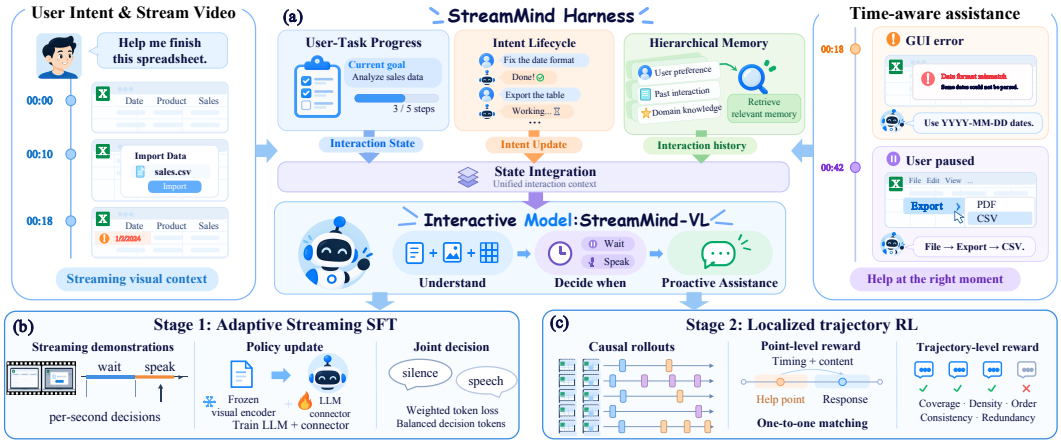


Figure 3: **Overview of StreamMind.** (a) During inference, StreamMind-Harness maintains persistent interaction state, tracks each intent through an independent lifecycle, and retrieves relevant history from hierarchical memory, providing StreamMind-VL with a unified context for response decisions. (b) Adaptive streaming SFT balances wait/speak decisions and jointly supervises timing and content. (c) Localized trajectory RL samples causal rollouts around help points, combining point-level timing and content rewards with trajectory-level rewards on coverage, density, order, consistency, and redundancy.

4.2 Evidence-Grounded Policy Learning for Streaming Interaction

***Stage 1*: Adaptive Streaming SFT.** We introduce adaptive streaming supervision, which adjusts decision-token weights to the current training distribution while separately optimizing response generation. This helps balance varying silence/response ratios across training sources and reduces bias toward excessive silence or over-response. Our SFT objective is

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{Y}|}\sum_{j\in\mathcal{Y}} \omega_j \log \pi_\theta\left(z_j \mid c_{t(j)}, z_{<j}\right), \qquad \omega_j = \begin{cases} \gamma_{B,c(j)}, & j \in \mathcal{A}, \\ \lambda_{\text{text}}, & j \in \mathcal{R}, \end{cases} \tag{1}$$

where $\mathcal{Y}$ denotes supervised assistant tokens, $\mathcal{A}$ and $\mathcal{R}$ denote decision and response tokens, and $c(j) \in \{\text{response}, \text{silence}\}$ is the decision class. The adaptive weight $\gamma_{c(j)}^{(t)}$ follows the current training distribution, while λ_{text} preserves response-generation learning.

***Stage 2*: Localized Trajectory RL.** Building on SFT, we introduce localized trajectory reinforcement learning. Full-trajectory optimization mixes correct and erroneous behaviors, leading to diffuse credit assignment and reward interference. We therefore construct short causal rollouts around annotated help points and optimize them with GRPO (Shao et al., 2024) using complementary point-level and trajectory-level rewards.

(i) Point-Level Reward. For each evidence point g and predicted response p, the temporal offset is defined as $\Delta_{g,p} = t_p - t_g$. The timing score $\tau(\Delta_{g,p}) \in [0,1]$ measures alignment within an asymmetric response window, while $\rho(g,p) \in [0,1]$ measures LLM-judged content quality. Their product $\tau(\Delta_{g,p})\rho(g,p)$ serves as the matching score, and one-to-one matching yields the set $\mathcal{M}$, where each evidence point and response can be matched at most once. The matched scores are then aggregated while accounting for missed evidence points and excessive responses.

$$R_{\text{align}} = \frac{(1+\alpha)\sum_{(g,p)\in\mathcal{M}} \tau(\Delta_{g,p})\rho(g,p)}{N_{\text{pred}} + \alpha N_{\text{gt}}} \tag{2}$$

Here, N_{pred} and N_{gt} denote the numbers of predicted responses and evidence points, respectively, while α controls the trade-off between missed evidence and over-response.

(ii) Trajectory-Level reward. We further introduce a trajectory-level reward R_{glob} to evaluate each rollout as a whole. Since point-level reword alone does not ensure coherent interaction, R_{glob} captures the overall consistency and naturalness of the response sequence, discouraging redundant or conflicting responses and favoring coherent streaming behavior. The final reward is defined as $R = \lambda R_{\text{align}} + (1-\lambda)R_{\text{glob}}$, with $\lambda = 0.7$ to place greater emphasis on response timing and content quality while preserving trajectory-level coherence. Since response formatting is already well learned during SFT, the RL stage optimizes only these two objectives without introducing an additional format reward. Further details on training are provided in Appendix A.1.

5 EXPERIMENTS

In this section, we conduct extensive experiments to answer the following research questions:

- ♠ **RQ1**: How top-performing baseline behave and can StreamMind-VL outperform them in intent-aware interaction setting?
- ♠ **RQ2**: What limitations do current proactive agents exhibit in sustained proactive interaction?
- ♠ **RQ3**: What gains does the StreamMind-Harness provide beyond the end-to-end policy?

5.1 SETUP

Benchmarks. We evaluate StreamMind across two settings: ***(i) Proactive Interaction*** on StreamMind-Bench, General Timing (Yao et al., 2026b), EgoProactive (Kundu et al., 2026), and IPIBench (Li et al., 2026b); and ***(ii) General Video Understanding*** on MVBench (Li et al., 2024), Video-MME (Fu et al., 2025a), OVO-Bench (Li et al., 2025b), and OVBench (Huang et al., 2025). External suites retain their official protocols. Further details are provided in Appendix A.

Baselines. We benchmark StreamMind against representative models from the three categories used in Table 2: *(i) Proprietary Models*, including Gemini-3.1-Pro and Gemini-3.8-Flash (Google DeepMind, 2026), and GPT-Realtime-2.1 (OpenAI, 2026); *(ii) Open-Source Offline Models*, including Qwen3-VL-8B (Bai et al., 2025), Qwen3.5-9B (Qwen Team, 2026), and JoyAI-VL-Interaction (Yao et al., 2026b); and *(iii) Open-Source Streaming Models*, including MMDuet2 (Wang et al., 2025d), Mage-VL (Yang et al., 2026d), LiveStar (Yang et al., 2025), ViSpeak (Fu et al., 2025b), and VideoLLM-Online (Chen et al., 2024).

Evaluation Settings. For StreamMind-Bench, we use the protocol and metrics defined in Section 3.5, with Claude Sonnet 4.6 (Anthropic, 2026) as a shared judge and response quality scored on a 0–4 scale. For external benchmarks, we strictly follow their official evaluation protocols and metrics. See Appendices C.5 and F.4 for evaluation details.

5.2 ANALYSIS

♠ **RQ1:** ***StreamMind-VL achieves stronger proactive interaction while preserving general video understanding.*** Table 2 shows that StreamMind-VL consistently outperforms open-source baselines on StreamMind-Bench. Under Pointwise Evaluation, it achieves the highest Content F1 while remaining competitive with proprietary models in response quality. RL further improves over SFT, with larger gains on dynamic intents. This trend persists under Trajectory Evaluation, where RL yields higher Content F1 and response quality, indicating stronger sustained interaction when decisions depend on prior history.

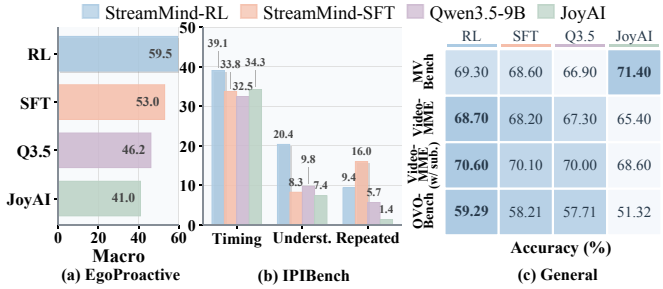


Figure 4: Generalization to external benchmarks. (a) Macro score on EgoProactive; (b) submetric scores on IPIBench; (c) accuracy on general video benchmarks, with darker cells indicating higher scores within each row.

The gains also transfer to external benchmarks. Figure 4(a-b) shows competitive performance on EgoProactive and consistent improvements over other models on IPIBench. Figure 4(c) further shows that our model outperforms its base model across all general video benchmarks, indicating that proactive policy learning preserves general video understanding. Detailed results and analysis are provided in Appendix B.

♠ **RQ2:** ***Current proactive agents struggle to sustain reliable behavior over continuous interaction.*** Figure 5 and Figure 6 reveal three main findings. *(i) Intent understanding remains difficult.* Implicit and Dynamic Intents are markedly harder than Explicit Intents, indicating that agents struggle to determine what help is needed when intents are unstated or keep changing. *(ii) Response*

Table 2: Main results on StreamMind-Bench (±2 s tolerance). *Pointwise evaluation* scores each decision given the gold interaction history, whereas *trajectory evaluation* runs the model over the full stream with its self-generated history. Streaming models do not support pointwise evaluation and are thus reported only under trajectory evaluation. **bold** and underline mark the best and second-best results within each block.

Model	Explicit						Implicit						Dynamic					
	P↑	R↑	F1↑	FPR↓	ANS↑	C-F1↑	P↑	R↑	F1↑	FPR↓	ANS↑	C-F1↑	P↑	R↑	F1↑	FPR↓	ANS↑	C-F1↑
Pointwise Evaluation *(gold interaction history)*																		
Proprietary Models																		
Gemini-3.1-Pro	**62.34**	68.52	65.28	**41.40**	45.22	29.52	70.92	85.58	77.56	43.57	67.46	52.32	62.87	84.59	72.13	58.81	51.15	36.89
GPT-Realtime-2.1	57.78	74.09	64.93	64.87	51.10	33.18	69.67	70.95	70.30	29.83	55.92	39.31	**62.99**	84.47	**72.17**	55.37	45.64	32.94
Gemini-3.8-Flash	60.34	77.40	**67.81**	50.69	51.37	**34.84**	**81.55**	80.45	**81.00**	**27.09**	82.63	**66.93**	62.96	84.52	72.16	60.25	**60.56**	**43.70**
Open-Source offline Models																		
Qwen3-VL-8B	51.25	82.28	63.16	89.05	40.91	25.84	56.38	57.85	57.11	83.71	80.51	45.98	54.37	87.56	67.08	93.88	45.01	30.19
Qwen3.5-9B	52.95	**91.36**	67.04	81.45	44.59	29.89	57.42	61.23	59.26	49.17	78.37	46.45	56.74	**90.93**	69.88	78.89	47.50	33.19
JoyAI-VL-Interaction	52.50	90.10	66.34	81.88	40.06	26.58	58.38	79.21	67.22	67.32	81.63	54.87	57.48	72.20	64.00	62.60	45.23	28.95
Ours																		
StreamMind-VL (SFT)	57.14	62.90	59.88	47.90	49.89	29.88	59.71	85.64	70.36	71.34	**85.92**	60.46	58.56	58.85	58.70	**49.13**	56.68	33.27
StreamMind-VL (RL)	57.33	64.89	60.88	46.19	**55.32**	33.68	61.59	**86.66**	72.01	67.91	85.11	61.28	58.94	75.86	66.34	62.93	59.25	39.31
Trajectory Evaluation *(self-generated interaction history)*																		
Open-Source offline Models																		
Qwen3-VL-8B	3.56	40.05	6.54	19.30	36.78	2.40	3.99	35.54	7.17	26.46	40.34	2.89	2.88	48.89	5.44	18.64	25.31	1.38
Qwen3.5-9B	9.37	47.62	15.66	19.30	16.63	2.60	7.94	52.48	13.79	14.56	44.98	6.20	4.87	51.65	8.90	20.95	27.67	2.46
JoyAI-VL-Interaction	4.67	**95.44**	8.90	30.87	20.21	1.80	4.39	37.96	7.87	18.87	46.83	3.69	3.15	**76.70**	6.05	64.52	21.64	1.31
Open-Source Streaming Models																		
MMDuet2	19.45	38.65	25.88	7.90	22.71	5.88	15.98	36.50	22.23	4.13	**74.26**	16.51	13.06	35.51	19.10	5.24	39.10	7.47
Mage-VL	20.70	50.25	29.32	6.11	10.23	3.00	8.56	51.94	14.70	5.50	8.19	1.20	11.97	52.88	19.52	6.20	6.10	1.19
LiveStar	14.96	19.95	17.10	4.81	3.60	0.62	6.84	24.95	10.74	6.27	25.43	2.73	7.29	12.92	9.32	2.86	8.05	0.75
ViSpeak	20.66	74.71	**32.37**	12.75	10.19	3.30	8.41	**74.03**	15.10	12.67	62.98	9.51	13.51	72.32	**22.77**	14.19	16.85	3.84
VideoLLM-Online	12.29	5.04	7.15	**1.44**	17.89	1.28	6.33	17.66	9.32	5.32	31.88	2.97	**17.86**	14.75	16.16	**1.28**	8.69	1.40
Ours																		
StreamMind-VL (SFT)	18.20	29.39	22.48	6.33	38.93	8.75	18.49	28.51	22.43	3.80	61.79	13.86	10.01	43.20	16.25	12.25	38.98	6.34
StreamMind-VL (RL)	**21.53**	34.48	26.51	2.02	**40.86**	**10.83**	**24.57**	28.20	**26.26**	**1.49**	72.83	**19.13**	14.80	30.65	19.96	5.68	**44.62**	**8.91**

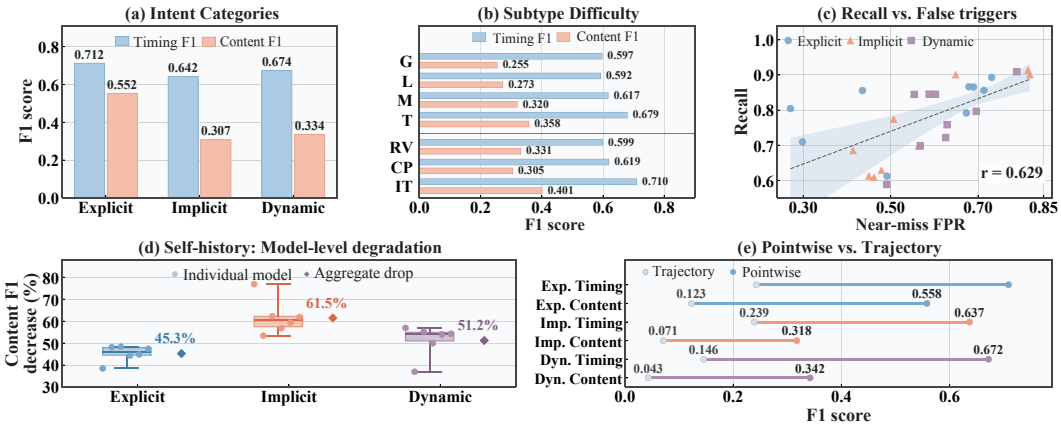


Figure 5: **Diagnostic analysis of proactive agents**. (a)-(b) Timing and Content F1 across intent categories and subtypes; (c) Recall versus near-miss FPR across models; (d) Content F1 degradation under accumulated self-generated history; (e) pointwise versus trajectory F1 per category and metric.

selectivity is insufficient. Higher Recall often comes with more false triggers, suggesting that current methods cannot yet balance coverage against unnecessary responses. *(iii) Sustained interaction amplifies these problems.* Timing F1 declines steadily with interaction length; Revision shows a sharper late-stage Recall drop, and Composition exhibits weaker content quality, indicating difficulty in continuously updating intents while maintaining multiple constraints. Correct judgments at individual moments are thus insufficient for stable proactive behavior throughout interaction.

♠ **RQ3:** ***Policy learning improves proactive decision-making, while state tracking enhances sustained interaction stability.*** As shown in Figure 6(2), SFT makes the model more sensitive to potential assistance opportunities, leading to higher Recall but also more unnecessary triggers. RL further improves decision selectivity, substantially reducing FPR while also improving answer quality. Figure 6(3) further shows that, with the policy fixed, adding the Harness continues to suppress unnecessary responses and improves content quality in most settings, with the clearest gains on Dynamic Intents. These results suggest that policy learning mainly improves local proactive decisions,

whereas state tracking helps preserve valid intents and task progress over long-horizon interactions, thereby reducing outdated, redundant, or state-inconsistent outputs.

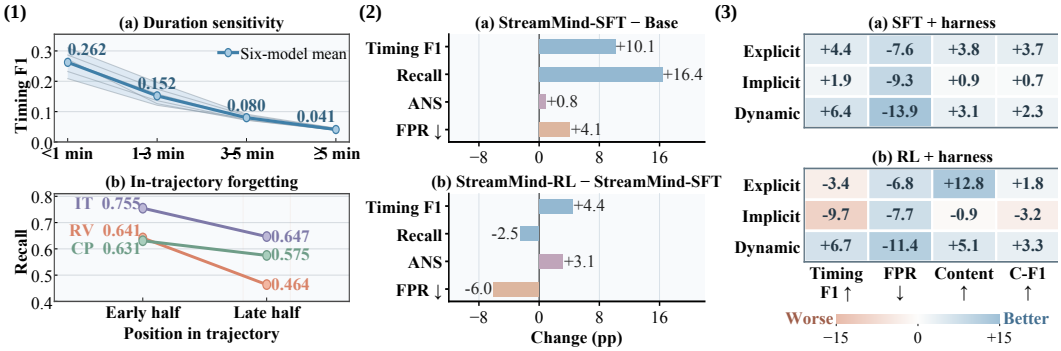


Figure 6: **Effects of interaction length, policy learning and the harness.** (1) Timing F1 versus video duration, and Recall in the early versus late half of the trajectory for each Dynamic Intent subtype. (2) Absolute metric changes of SFT over the base model and of RL over SFT. (3) Gains from adding the Harness to the SFT and RL policies across intent categories. All results use a $\tau = 2$ s tolerance.

Qualitative Analysis. Figure 7 shows representative physical and digital interactions of StreamMind-VL. Across multiple assistance points within the same task, it grounds responses in the current visual evidence and task state, providing assistance at appropriate moments. In contrast, streaming baselines often respond before sufficient evidence appears or fail to update later responses as the task evolves, yielding premature, inconsistent, or redundant assistance. These cases show that our agent keeps both response timing and content consistent throughout sustained interaction. Additional qualitative examples are provided in Appendix E.2.

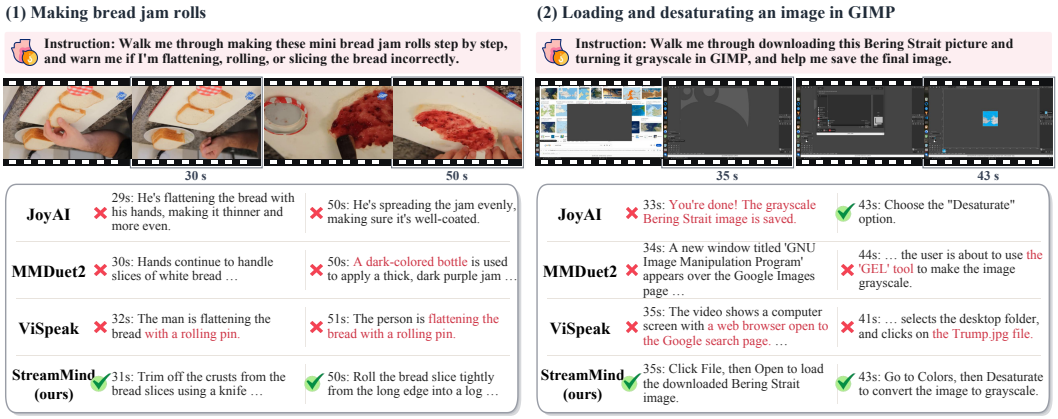


Figure 7: **Case studies of implicit proactive interaction.** Across physical and digital tasks, StreamMind-VL better aligns response timing and content with the evolving visual evidence and task state, while streaming baselines more often produce premature, missing, or inconsistent assistance.

6 Conclusion

We study proactive interaction over streaming video, where user intents may be implicit, evolve over time, and must be tracked alongside a changing environment. We introduce StreamMind-Bench, a unified benchmark spanning explicit, implicit, and dynamic intents with pointwise and trajectory evaluation; StreamMind-VL, a proactive streaming policy trained with adaptive streaming SFT and localized trajectory RL; and StreamMind-Harness, which maintains user intents and task progress as persistent interaction states. Experiments show that our model improves response timing and quality while preserving general video understanding, while explicit state tracking further reduces outdated and redundant responses. More broadly, real-world proactive interaction may involve richer intent evolution and tighter coupling between user needs and environmental changes than captured in our current setting. New observations, task outcomes, and prior assistance may jointly reshape both user needs and the environment, leading to new goals and constraints. Future work could further study such co-evolving interactions, particularly the complex feedback between intents and environmental states over longer horizons.

REFERENCES

Anthropic. Introducing claude sonnet 4.6. Anthropic model announcement, 2026. URL https://www.anthropic.com/news/claude-sonnet-4-6.

Shuai Bai et al. Qwen3-vl technical report. 2025. URL https://arxiv.org/abs/2511.21631.

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. 2022. URL https://arxiv.org/abs/2104.00650.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. URL https://openaccess.thecvf.com/content_cvpr_2015/html/Heilbron_ActivityNet_A_Large-Scale_2015_CVPR_paper.html.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding. 2025a. URL https://arxiv.org/abs/2406.10819.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. 2024. URL https://arxiv.org/abs/2406.11816.

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video llm with streaming speech transcription at scale. 2025b. URL https://arxiv.org/abs/2504.16030.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. 2025. URL https://arxiv.org/abs/2504.19413.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

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. 2026. URL https://arxiv.org/abs/2604.27393.

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971. doi: 10.1037/h0031619.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. 2025a. URL https://arxiv.org/abs/2405.21075.

Shenghao Fu, Qize Yang, Yuan-Ming Li, Yi-Xing Peng, Kun-Yu Lin, Xihan Wei, Jian-Fang Hu, Xiaohua Xie, and Wei-Shi Zheng. Vispeak: Visual instruction feedback in streaming videos. 2025b. URL https://arxiv.org/abs/2503.12769.

Google DeepMind. Gemini 3 model documentation. Google AI for Developers, 2026. URL https://ai.google.dev/gemini-api/docs/models.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, et al. Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives. 2024. URL https://arxiv.org/abs/2311.18259.

Mingfei Han, Linjie Yang, Xiaojun Chang, Lina Yao, and Heng Wang. Shot2story: A new benchmark for comprehensive understanding of multi-shot videos. 2025. URL https://arxiv.org/abs/2312.10300.

Kairui Hu, Penghao Wu, Fanyi Pu, Wang Xiao, Yuanhan Zhang, Xiang Yue, Bo Li, and Ziwei Liu. Video-mmmu: Evaluating knowledge acquisition from multi-discipline professional videos. 2025. URL https://arxiv.org/abs/2501.13826.

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. 2025. URL https://arxiv.org/abs/2501.00584.

Xiangru Jian, Shravan Nayak, Kevin Qinghong Lin, Aarash Feizi, Kaixin Li, Patrice Bechard, Spandana Gella, and Sai Rajeswar. Cua-suite: Massive human-annotated video demonstrations for computer-use agents. 2026. URL https://arxiv.org/abs/2603.24440.

Patrick Knab, Orgest Xhelili, Inis Buzi, Drago Andres Guggiana Nilo, Mohd Saquib Khan, Lorenz Kolb, Manuel Scherzer, Kerem Yildirir, Christian Bartelt, and Philipp Johannes Schubert. Barista: A multi-task egocentric benchmark for compositional visual understanding. 2026. URL https://arxiv.org/abs/2605.12074.

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. 2026. URL https://arxiv.org/abs/2606.04970.

J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977. doi: 10.2307/2529310.

Ao Li, Zihan Xiao, Zihao Yue, Boshen Xu, Linli Yao, Jiaze Li, Pei Fu, Jianzhong Ju, Jian Luan, and Qin Jin. Streampro: From reactive perception to proactive decision-making in streaming video. *arXiv preprint arXiv:2605.16381*, 2026a.

Jinzhao Li, Yinuo Chen, Wenxuan Song, Yijia Lei, Yichi Zhang, Honglei Yan, Panwang Pan, and Miao Liu. Ipibench: Evaluating interactive proactive intelligence of mllms under continuous streams. 2026b. URL https://arxiv.org/abs/2605.27074.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. Mvbench: A comprehensive multi-modal video understanding benchmark. 2024. URL https://arxiv.org/abs/2311.17005.

Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, Yu Qiao, Yali Wang, and Limin Wang. Videochat-flash: Hierarchical compression for long-context video modeling. 2025a. URL https://arxiv.org/abs/2501.00574.

Yifei Li, Junbo Niu, 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? 2025b. URL https://arxiv.org/abs/2501.05510.

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. 2026. URL https://arxiv.org/abs/2607.02991.

Xudong Lu, Yang Bo, Jinpeng Chen, Shuhan Li, Xintong Guo, Huankang Guan, Fang Liu, Dunyuan Xu, Peiwen Sun, Heyang Sun, Rui Liu, and Hongsheng Li. Aura: Always-on understanding and real-time assistance via video streams. 2026. URL https://arxiv.org/abs/2604.04184.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Videogpt+: Integrating image and video encoders for enhanced video understanding. 2024. URL https://arxiv.org/abs/2406.09418.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. 2023. URL https://arxiv.org/abs/2308.09126.

Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, Si Qin, Liqun Li, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Gui-360°: A comprehensive dataset and benchmark for computer-using agents. 2025. URL https://arxiv.org/abs/2511.04307.

OpenAI. Gpt-realtime-2.1 model documentation. OpenAI API documentation, 2026. 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. 2026. URL https://arxiv.org/abs/2609.08977.

Viorica Pătrăucean, Lucas Smaira, Ankush Gupta, Adrià Recasens Continente, Larisa Markeeva, Dylan Banarse, Skanda Koppula, Joseph Heyward, Mateusz Malinowski, Yi Yang, Carl Doersch, Tatiana Matejovicova, Yury Sulsky, Antoine Miech, Alex Frechette, Hanna Klimczak, Raphael Koster, Junlin Zhang, Stephanie Winkler, Yusuf Aytar, Simon Osindero, Dima Damen, Andrew Zisserman, and João Carreira. Perception test: A diagnostic benchmark for multimodal video models. 2023. URL https://arxiv.org/abs/2305.13786.

Qwen Team. Qwen3.5-9b. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B.

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. 2026. URL https://arxiv.org/abs/2605.07299.

Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. 2022. URL https://arxiv.org/abs/2203.14712.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. 2024. URL https://arxiv.org/abs/2402.03300.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. 2016. URL https://arxiv.org/abs/1604.01753.

Gunnar A. Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. 2018. URL https://arxiv.org/abs/1804.09626.

Gong Sitong, Tianyu Yan, Caixin Kang, Bo Zheng, Xiang Ruan, Huchuan Lu, Kaipeng Zhang, Yoichi Sato, and Yifei Huang. Vinci2: Providing proactive assistance in continuous egocentric videos. 2026. URL https://arxiv.org/abs/2607.11523.

Enxin Song, Wenhao Chai, Weili Xu, Jianwen Xie, Yuxuan Liu, and Gaoang Wang. Video-mmlu: A massive multi-discipline lecture understanding benchmark. 2025. URL https://arxiv.org/abs/2504.14693.

Jihoon Tack, Philippe Laban, and Jennifer Neville. Llms get lost in evolving user intent. 2026. URL https://arxiv.org/abs/2607.20734.

Meiling Tao, Yiling Tao, and Peng Wang. Intent-driven situation tracking for user-centric multi-turn agents. 2026. URL https://arxiv.org/abs/2608.15755.

Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. Streambridge: Turning your offline video large language model into a proactive streaming assistant. 2025a. URL https://arxiv.org/abs/2505.05467.

Pengyu Wang, Chenkun Tan, Shaojun Zhou, Qirui Zhou, Yanxin Chen, Xingyang He, Huazheng Zeng, Jijun Cheng, Chenghao Wang, Xiaomeng Qian, Pengfei Wang, Zhan Huang, Shanqing Gao, Wei Huang, Longjun Cao, Wu Ran, Jie Liu, Changtai Zhu, Hongkai Wang, Yixian Tian, Chenghao Liu, Zhen Ye, Xinghao Wang, Botian Jiang, Guoguo Feng, Zhaoye Fei, Ruixiao Li, Mingshu Chen, Yang Gao, Qinyuan Cheng, Shimin Li, and Xipeng Qiu. Moss-vl technical report, 2026. URL https://arxiv.org/abs/2608.15045.

Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. 2023. URL https://arxiv.org/abs/2309.17024.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Haotian Yao, Ziwei Chen, Qizheng Gu, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents. 2025b. URL https://arxiv.org/abs/2508.09123.

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, Conghui He, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. Internvid: A large-scale video-text dataset for multimodal understanding and generation. 2024.

URL https://arxiv.org/abs/2307.06942.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, Min Dou, Kai Chen, Wenhai Wang, Yu Qiao, Yali Wang, and Limin Wang. Internvideo2.5: Empowering video mllms with long and rich context modeling. 2025c. URL https://arxiv.org/abs/2501.12386.

Yueqian Wang, Songxiang Liu, Disong Wang, Nuo Xu, Guanglu Wan, Huishuai Zhang, and Dongyan Zhao. Mmduet2: Enhancing proactive interaction of video mllms with multi-turn reinforcement learning. 2025d. URL https://arxiv.org/abs/2512.06810.

Yueqian Wang, Xiaojun Meng, Yifan Wang, Huishuai Zhang, and Dongyan Zhao. Proactivevideoqa: A comprehensive benchmark evaluating proactive interactions in video large language models. 2025e. URL https://arxiv.org/abs/2507.09313.

Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning. 2026. URL https://arxiv.org/abs/2512.21334.

Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. Mpo: Boosting llm agents with meta plan optimization. 2025. URL https://arxiv.org/abs/2503.02682.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. 2025. URL https://arxiv.org/abs/2502.12110.

Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, Bei Ouyang, Zhengyu Lin, Marco Cominelli, Zhongang Cai, Yuanhan Zhang, Peiyuan Zhang, Fangzhou Hong, Joerg Widmer, Francesco Gringoli, Lei Yang, Bo Li, and Ziwei Liu. Egolife: Towards egocentric life assistant. 2026a. URL https://arxiv.org/abs/2503.03803.

Qinglong Yang, Haoming Li, Haotian Zhao, Xiaokai Yan, Jingtao Ding, Fengli Xu, and Yong Li. Fingertip 20k: A benchmark for proactive and personalized mobile llm agents. 2026b. URL https://arxiv.org/abs/2507.21071.

Saelyne Yang, Jaesang Yu, Yi-Hao Peng, Kevin Qinghong Lin, Jae Won Cho, Yale Song, and Juho Kim. Guide: A benchmark for understanding and assisting users in open-ended gui tasks. 2026c. URL https://arxiv.org/abs/2603.25864.

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, Xiang An, Yin Xie, Zhening Liu, Xun Guo, Jiahao Li, Shicheng Zheng, Jinglu Wang, Zongyu Guo, Wenxuan Xie, Zihan Zheng, Yuxuan Luo, Bin Li, and Yan Lu. Mage-vl: An efficient codec-native streaming multimodal foundation model. 2026d. URL https://arxiv.org/abs/2607.24904.

Zhenyu Yang, Kairui Zhang, Yuhang Hu, Bing Wang, Shengsheng Qian, Bin Wen, Fan Yang, Tingting Gao, Weiming Dong, and Changsheng Xu. Livestar: Live streaming assistant for real-world online video understanding. 2025. URL https://arxiv.org/abs/2511.05299.

Dingyu Yao, Shuhuan Gu, Qingyi Si, Junhao Zhou, Chenxu Yang, Chuanyu Qin, Naibin Gu, Zheng Lin, Weiping Wang, Nan Duan, and Jiaqi Wang. Harnessing streaming video in the wild, 2026a. URL https://arxiv.org/abs/2606.08615.

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. 2026b. URL https://arxiv.org/abs/2606.14777.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, Lingpeng Kong, Qi Liu, Yuanxing Zhang, and Xu Sun. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. 2025. URL https://arxiv.org/abs/2504.17343.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. 2026. URL https://arxiv.org/abs/2601.01885.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. 2025. URL https://arxiv.org/abs/2501.11425.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. Streamforest: Efficient online video un-

derstanding with persistent event memory. 2025. URL https://arxiv.org/abs/2509.24871.

Jun Zhang, Teng Wang, Yuying Ge, Yixiao Ge, Xinhao Li, Ying Shan, and Limin Wang. Timelens: Rethinking video temporal grounding with multimodal llms. 2026. URL https://arxiv.org/abs/2512.14698.

Yilun Zhao, Lujing Xie, Haowei Zhang, Guo Gan, Yitao Long, Zhiyuan Hu, Tongyan Hu, Weiyuan Chen, Chuhan Li, Junyang Song, Zhijian Xu, Chengye Wang, Weifeng Pan, Ziyao Shangguan, Xiangru Tang, Zhenwen Liang, Yixin Liu, Chen Zhao, and Arman Cohan. Mmvu: Measuring expert-level multi-discipline video understanding. 2025. URL https://arxiv.org/abs/2501.12380.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. 2025. URL https://arxiv.org/abs/2504.07079.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: Benchmarking multi-task long video understanding. 2025. URL https://arxiv.org/abs/2406.04264.

Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards automatic learning of procedures from web instructional videos. 2017. URL https://arxiv.org/abs/1703.09788.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. 2025. URL https://arxiv.org/abs/2504.10479.

A Appendix

A.1 Training Details

This section provides additional details for the adaptive streaming supervision and localized reinforcement learning described in Section 4.2.

A.1.1 Adaptive Streaming Supervision

Streaming supervision. Each training trajectory interleaves visual observations, user interactions, and assistant outputs. Supervision is applied only to assistant tokens, which are divided into decision tokens $\mathcal{A}$ and response tokens $\mathcal{R}$. Decision tokens indicate whether the model should remain silent or respond, while response tokens supervise the generated assistance content.

Adaptive decision weighting. The $\mathrm{silence}/\mathrm{response}$ ratio varies across training sources and training steps. Rather than using fixed class weights, we adapt the decision supervision to the current training distribution. Let $n_c^{(t)}$ denote the number of decision tokens of class $c \in \{\mathrm{silence}, \mathrm{response}\}$ at training step t. We first compute the effective count

$$E_c^{(t)} = \frac{1 - \beta^{\max(n_c^{(t)}, 1)}}{1 - \beta}. \tag{3}$$

The adaptive class weight is then defined as

$$\gamma_c^{(t)} = \frac{2\left(E_c^{(t)}\right)^{-1}}{\left(E_{\mathrm{silence}}^{(t)}\right)^{-1} + \left(E_{\mathrm{response}}^{(t)}\right)^{-1}}. \tag{4}$$

This increases the contribution of the less frequent decision class while keeping the overall scale of decision supervision stable. We use $\beta = 0.9996$.

For each supervised token j, the weight is

$$\omega_j = \begin{cases} \gamma^{(t)}_{c(j)}, & j \in \mathcal{A}, \\ \lambda_{\text{text}}, & j \in \mathcal{R}, \end{cases} \tag{5}$$

where $c(j) \in \{\text{silence}, \text{response}\}$ denotes the decision class of token j, and λ_{text} controls response-generation supervision. We use $\lambda_{\text{text}} = 1.5$.

The resulting SFT objective is

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{Y}|} \sum_{j \in \mathcal{Y}} \omega_j \log \pi_\theta \left(z_j \mid c_{t(j)}, z_{<j} \right), \tag{6}$$

where $\mathcal{Y}$ denotes the supervised assistant tokens, z_j is the target token, and $c_{t(j)}$ is the causal context available at its decision time. The adaptive weighting follows the current training distribution rather than enforcing a fixed global response ratio.

A.1.2 Localized Reinforcement Learning

Short-trajectory sampling. Full long-horizon rollouts contain many routine silence decisions and may mix correct interventions, missed evidence points, and false responses. This can cause reward cancellation and diffuse credit assignment. We therefore construct short local trajectories around annotated evidence points and sample multiple causal rollouts from the same context. In our setting, each local trajectory is approximately 10–15 seconds and contains one scored evidence point, with $K = 8$ rollouts sampled for each prompt.

Temporal quality. For an evidence point g at time t_g and a predicted response p at time t_p, we define their temporal offset as

$$\Delta_{g,p} = t_p - t_g. \tag{7}$$

We use an asymmetric temporal function that allows greater tolerance for anticipatory responses than delayed ones:

$$\tau(\Delta) = \begin{cases} 1 - \dfrac{|\Delta|}{8}, & -4 \le \Delta < 0, \\ 1 - \dfrac{\Delta}{4}, & 0 \le \Delta \le 2, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The score is maximal at the annotated time and decreases more slowly for early than for delayed responses.

Content quality. For each temporally eligible pair (g, p), the Claude Sonnet 4.6 (Anthropic, 2026) judge assigns a content quality score

$$\rho(g, p) \in [0, 1]. \tag{9}$$

The judge considers the user task, available evidence, reference assistance, and candidate response, and evaluates whether the response is grounded, correct, and useful.

Point-level alignment. Timing and content are combined multiplicatively:

$$q(g, p) = \tau(\Delta_{g,p}) \rho(g, p). \tag{10}$$

Evidence points are processed chronologically, and each evidence point is matched to the highest-scoring unused eligible response. This produces a one-to-one matching set $\mathcal{M}$ and prevents multiple responses from receiving credit for the same evidence point.

The alignment reward is

$$R_{\text{align}} = \frac{(1 + \alpha) \sum_{(g,p) \in \mathcal{M}} \tau(\Delta_{g,p}) \rho(g, p)}{N_{\text{pred}} + \alpha N_{\text{gt}}}. \tag{11}$$

Here, N_{pred} and N_{gt} denote the numbers of predicted responses and annotated evidence points, respectively. The coefficient α balances missed and excessive interventions. Larger α emphasizes coverage, whereas smaller α places relatively greater pressure on suppressing unnecessary responses. We use $\alpha = 2$.

Trajectory-level alignment. Point-level alignment can score individual interventions accurately without ensuring that the resulting sequence forms a coherent interaction. Across a long-running interaction, locally appropriate responses may repeat prior guidance, occur too densely, appear out of order, or conflict with earlier responses and the evolving context. We therefore introduce a trajectory-level reward R_{glob} based on a sample-specific rubric that jointly evaluates coverage, response density, ordering, cross-response consistency, and redundancy over the complete rollout. This reward encourages a coherent, non-repetitive sequence of interventions rather than a collection of independently plausible responses.

The final reward is

$$R = \lambda R_{\text{align}} + (1 - \lambda) R_{\text{glob}}, \tag{12}$$

where we use $\lambda = 0.7$. This places greater emphasis on local intervention timing and response quality while retaining trajectory-level constraints. We do not include an additional formatting reward because output formatting is already reliably learned during SFT.

GRPO optimization. For each local context, GRPO samples K candidate trajectories and compares their rewards within the group. Let R_i denote the reward of rollout i. The normalized advantage is

$$A_i = \frac{R_i - \mu_R}{\max(\sigma_R, \epsilon)}, \tag{13}$$

where μ_R and σ_R denote the mean and standard deviation of rewards within the rollout group, and ϵ prevents unstable scaling when the group variance is small. The resulting relative advantages are used for clipped policy optimization with KL regularization toward the frozen SFT policy.

B Additional Evaluation Results

We report Pointwise evaluation with gold or model-generated interaction history and Trajectory evaluation with model-generated history. The matching tolerances are ± 2, ± 1, and ± 0 s; tighter tolerances rescore the same predictions. The following model-comparison tables report StreamMind-VL without the Harness, followed by separate Harness comparisons. **Bold** and underline denote the best and second-best eligible results in each column; FPR is lower-is-better and all other metrics are higher-is-better.

B.1 Pointwise Evaluation with Gold Interaction History

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	70.92	85.58	77.56	43.57	67.46	52.32	**62.34**	68.52	65.28	41.40	45.22	29.52	62.87	84.59	72.13	58.81	51.15	36.89
GPT-Realtime-2.1	69.67	70.95	70.30	29.83	55.92	39.31	57.78	90.09	**70.41**	64.87	51.10	**35.98**	**62.99**	84.47	**72.17**	55.37	45.64	32.94
Gemini-3.8-Flash	**81.55**	80.45	**81.00**	27.09	82.63	**66.93**	60.34	77.40	67.81	50.69	51.37	34.83	62.96	84.52	72.16	60.25	**60.56**	**43.70**
Open-Source Offline Models																		
Qwen3.5-9B	57.42	61.23	59.26	49.17	78.37	46.44	52.95	91.36	67.04	81.45	44.59	29.89	56.74	90.93	69.88	78.89	47.50	33.19
Qwen3-VL-8B	56.38	**90.87**	69.59	83.71	80.51	56.03	51.25	**95.28**	66.65	91.05	47.91	31.93	54.37	**97.56**	69.83	93.88	50.91	35.55
JoyAI-VL-Interaction	58.38	79.21	67.22	67.32	81.63	54.87	52.50	90.10	66.34	81.88	40.06	26.58	57.48	72.20	64.00	62.60	45.23	28.95
Open-Source Streaming Models																		
MMDuet2	15.98	36.50	22.23	**4.13**	74.26	16.51	19.45	38.65	25.88	7.90	22.71	5.88	13.06	35.51	19.10	5.24	39.10	7.47
Mage-VL	8.56	51.94	14.70	5.50	8.19	1.20	20.70	50.25	29.32	6.11	10.23	3.00	11.97	52.88	19.52	6.20	6.10	1.19
LiveStar	6.84	24.95	10.74	6.27	25.43	2.73	14.96	19.95	17.10	4.81	3.60	0.62	7.29	12.92	9.32	2.86	8.05	0.75
ViSpeak	8.41	74.03	15.10	12.67	62.98	9.51	20.66	74.71	32.37	12.75	10.19	3.30	13.51	72.32	22.77	14.19	16.85	3.84
VideoLLM-Online	6.33	17.66	9.32	5.32	31.88	2.97	12.29	5.04	7.15	**1.44**	17.89	1.28	17.86	14.75	16.16	**1.28**	8.69	1.40
Ours																		
StreamMind-VL (SFT)	60.19	89.31	71.91	73.11	85.02	61.14	55.74	69.79	61.98	56.35	53.65	33.25	57.56	79.68	66.84	69.58	57.67	38.55
StreamMind-VL (RL)	61.59	86.66	72.01	67.91	**85.11**	61.29	57.33	60.89	59.06	46.19	**55.32**	32.67	58.94	75.86	66.34	62.93	59.25	39.31

Table 3: ***Pointwise evaluation with gold interaction history*** on StreamMind-Bench without the Harness (± 2 s tolerance).

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	67.47	81.43	73.80	51.13	68.92	50.86	53.46	58.76	55.98	52.96	46.73	26.16	58.03	78.07	66.57	68.27	52.94	35.24
GPT-Realtime-2.1	63.10	64.25	63.67	42.00	57.79	36.79	50.76	79.14	61.85	77.83	51.78	32.03	55.68	74.67	63.79	69.60	46.60	29.73
Gemini-3.8-Flash	**79.04**	77.97	**78.50**	31.60	83.41	**65.48**	53.48	68.61	60.11	61.10	52.45	31.53	**59.01**	79.21	67.63	67.94	**61.25**	**41.42**
Open-Source Offline Models																		
Qwen3.5-9B	54.78	58.42	56.54	54.27	79.23	44.80	51.40	88.69	65.08	84.61	44.91	29.23	55.39	88.77	68.22	82.03	47.61	32.48
Qwen3-VL-8B	55.38	**89.25**	68.35	86.65	81.16	55.47	50.52	**93.93**	**65.70**	92.66	47.82	31.42	54.18	**97.22**	**69.58**	94.37	50.97	35.46
JoyAI-VL-Interaction	57.78	78.40	66.53	68.79	81.96	54.53	51.82	88.94	65.49	83.25	40.27	26.37	56.31	70.74	62.71	64.73	45.40	28.47
Open-Source Streaming Models																		
MMDuet2	4.77	10.91	6.64	7.13	75.87	5.04	5.49	10.90	7.30	11.10	24.07	1.76	7.12	19.35	10.41	6.95	38.66	4.02
Mage-VL	1.97	11.93	3.38	10.19	9.84	0.33	7.58	18.40	10.74	9.78	9.65	1.04	3.05	13.48	4.97	10.38	7.07	0.35
LiveStar	2.95	10.75	4.63	7.93	27.01	1.25	7.31	9.75	8.36	5.98	3.85	0.32	2.61	4.63	3.34	3.73	10.36	0.35
ViSpeak	2.16	19.01	3.88	19.12	64.28	2.49	5.52	19.96	8.65	19.06	10.99	0.95	4.86	26.02	8.19	19.10	16.76	1.37
VideoLLM-Online	2.53	7.07	3.73	**6.56**	29.77	1.11	6.66	2.73	3.87	**1.71**	19.82	0.77	15.24	12.59	13.79	**1.51**	5.52	0.76
Ours																		
StreamMind-VL (SFT)	59.83	88.77	71.48	74.09	85.14	60.86	53.46	66.95	59.45	59.72	54.21	**32.23**	56.36	78.01	65.44	72.00	58.14	38.05
StreamMind-VL (RL)	61.01	85.85	71.33	69.38	**85.24**	60.80	**54.45**	57.83	56.09	49.81	**55.90**	31.35	57.45	73.94	64.66	65.71	59.70	38.60

Table 4: ***Pointwise evaluation with gold interaction history*** on StreamMind-Bench without the Harness (± 0 s tolerance).

B.2 Pointwise evaluation with self-generated interaction history

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	66.28	**82.45**	73.49	33.17	40.95	30.09	68.23	70.21	69.21	29.31	29.20	20.21	73.03	80.05	76.38	28.13	36.70	28.03
GPT-Realtime-2.1	65.42	72.84	68.93	34.54	54.10	37.29	61.47	85.03	71.36	50.95	**48.94**	34.92	66.86	79.83	72.77	42.50	41.99	30.56
Gemini-3.8-Flash	**89.93**	75.22	**81.92**	5.79	71.88	**58.88**	79.24	79.49	79.36	13.77	46.90	**37.22**	**85.98**	**82.75**	**84.33**	7.27	**56.27**	**47.45**
Open-Source Offline Models																		
Qwen3.5-9B	66.55	29.86	41.22	8.24	68.04	28.05	56.04	89.04	68.79	67.93	22.90	15.75	62.64	80.52	70.46	51.20	32.39	22.82
JoyAI-VL-Interaction	64.78	67.33	66.03	29.64	68.08	44.95	78.54	**91.23**	**84.41**	14.37	21.65	18.27	83.76	59.29	69.43	4.54	33.53	23.28
Open-Source Streaming Models																		
MMDuet2	15.98	36.50	22.23	**4.13**	**74.26**	16.51	19.45	38.65	25.88	7.90	22.71	5.88	13.06	35.51	19.10	5.24	39.10	7.47
Mage-VL	8.56	51.94	14.70	5.50	8.19	1.20	20.70	50.25	29.32	6.11	10.23	3.00	11.97	52.88	19.52	6.20	6.10	1.19
LiveStar	6.84	24.95	10.74	6.27	25.43	2.73	14.96	19.95	17.10	4.81	3.60	0.62	7.29	12.92	9.32	2.86	8.05	0.75
ViSpeak	8.41	74.03	15.10	12.67	62.98	9.51	20.66	74.71	32.37	12.75	10.19	3.30	13.51	72.32	22.77	14.19	16.85	3.84
VideoLLM-Online	6.33	17.66	9.32	5.32	31.88	2.97	12.29	5.04	7.15	**1.44**	17.89	1.28	17.86	14.75	16.16	**1.28**	8.69	1.40
Ours																		
StreamMind-VL (SFT)	69.05	76.03	72.37	27.09	70.42	50.96	77.54	62.79	69.39	12.51	42.46	29.46	76.98	69.37	72.98	16.38	45.94	33.53
StreamMind-VL (RL)	73.58	76.67	75.09	17.66	72.43	54.39	**84.28**	65.75	73.87	5.00	45.71	33.77	82.70	68.45	74.90	8.05	48.53	36.35

Table 5: ***Pointwise evaluation with self-generated interaction history*** on StreamMind-Bench without the Harness (± 2 s tolerance).

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	54.69	**68.03**	60.63	59.37	41.25	25.01	45.91	47.24	46.57	56.49	27.20	12.67	50.21	55.03	52.51	64.44	33.65	17.67
GPT-Realtime-2.1	57.03	63.50	60.09	51.52	55.53	33.37	**49.81**	68.90	57.82	70.04	49.80	**28.79**	**54.90**	65.55	**59.75**	63.23	43.11	**25.76**
Gemini-3.8-Flash	**72.18**	60.37	**65.75**	32.78	73.90	**48.59**	35.61	35.72	35.66	65.59	49.80	17.76	43.63	42.00	42.80	66.43	**58.46**	25.02
Open-Source Offline Models																		
Qwen3.5-9B	53.43	23.97	33.09	18.94	69.09	22.86	48.11	**76.44**	**59.05**	82.85	23.04	13.61	52.34	**67.28**	58.88	70.42	31.84	18.75
JoyAI-VL-Interaction	47.90	49.78	48.82	61.53	68.28	33.33	27.04	31.40	29.06	85.19	21.03	6.11	37.12	26.27	30.77	52.47	34.70	10.68
Open-Source Streaming Models																		
MMDuet2	4.77	10.91	6.64	7.13	**75.87**	5.04	5.49	10.90	7.30	11.10	24.07	1.76	7.12	19.35	10.41	6.95	38.66	4.02
Mage-VL	1.97	11.93	3.38	10.19	9.84	0.33	7.58	18.40	10.74	9.78	9.65	1.04	3.05	13.48	4.97	10.38	7.07	0.35
LiveStar	2.95	10.75	4.63	7.93	27.01	1.25	7.31	9.75	8.36	5.98	3.85	0.32	2.61	4.63	3.34	3.73	10.36	0.35
ViSpeak	2.16	19.01	3.88	19.12	64.28	2.49	5.52	19.96	8.65	19.06	10.99	0.95	4.86	26.02	8.19	19.10	16.76	1.37
VideoLLM-Online	2.53	7.07	3.73	**6.56**	29.77	1.11	6.66	2.73	3.87	**1.71**	19.82	0.77	15.24	12.59	13.79	**1.51**	5.52	0.76
Ours																		
StreamMind-VL (SFT)	47.87	52.70	50.17	69.48	67.37	33.80	33.89	27.44	30.33	54.36	46.02	13.96	40.67	36.65	38.56	63.87	45.61	17.59
StreamMind-VL (RL)	47.36	49.35	48.33	67.32	69.37	33.53	31.78	28.02	29.78	46.10	**50.35**	14.99	39.12	37.49	38.29	57.34	49.03	18.77

Table 6: ***Pointwise evaluation with self-generated interaction history*** on StreamMind-Bench without the Harness (± 0 s tolerance).

B.3 Trajectory evaluation with self-generated interaction history

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	32.74	**79.21**	46.33	4.60	85.60	39.66	17.73	58.78	27.24	12.80	52.12	14.20	14.23	57.82	22.84	8.92	64.95	14.83
GPT-Realtime-2.1	9.70	55.56	16.52	11.63	66.84	11.04	14.91	56.26	23.57	15.71	48.21	11.36	9.69	52.67	16.37	14.49	46.86	7.67
Gemini-3.8-Flash	**47.69**	79.10	**59.50**	**2.37**	**87.49**	**52.06**	**21.03**	71.82	**32.53**	12.94	**59.96**	**19.50**	16.00	66.17	**25.77**	8.42	**72.48**	**18.68**
Open-Source Offline Models																		
Qwen3.5-9B	7.94	52.48	13.79	14.56	44.98	6.20	9.37	47.62	15.66	19.30	16.63	2.60	4.87	51.65	8.90	20.95	27.67	2.46
JoyAI-VL-Interaction	4.39	37.96	7.87	18.87	46.83	3.69	4.67	**95.44**	8.90	90.87	10.21	0.91	3.15	**76.70**	6.05	64.52	21.64	1.31
Open-Source Streaming Models																		
MMDuet2	15.98	36.50	22.23	4.13	74.26	16.51	19.45	38.65	25.88	7.90	22.71	5.88	13.06	35.51	19.10	5.24	39.10	7.47
Mage-VL	8.56	51.94	14.70	5.50	8.19	1.20	20.70	50.25	29.32	6.11	10.23	3.00	11.97	52.88	19.52	6.20	6.10	1.19
LiveStar	6.84	24.95	10.74	6.27	25.43	2.73	14.96	19.95	17.10	4.81	3.60	0.62	7.29	12.92	9.32	2.86	8.05	0.75
ViSpeak	8.41	74.03	15.10	12.67	62.98	9.51	20.66	74.71	32.37	12.75	10.19	3.30	13.51	72.32	22.77	14.19	16.85	3.84
VideoLLM-Online	6.33	17.66	9.32	5.32	31.88	2.97	12.29	5.04	7.15	**1.44**	17.89	1.28	**17.86**	14.75	16.16	**1.28**	8.69	1.40
Ours																		
StreamMind-VL (SFT)	11.55	40.66	17.99	11.40	57.97	10.43	13.13	47.91	20.61	15.59	38.06	7.84	5.38	56.76	9.83	26.13	35.88	3.53
StreamMind-VL (RL)	15.80	41.68	22.91	8.27	60.07	13.76	18.02	43.64	25.51	9.68	41.74	10.65	7.60	52.58	13.28	17.11	39.55	5.25

Table 7: ***Trajectory evaluation with self-generated interaction history*** on StreamMind-Bench without the Harness (± 2 s tolerance).

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models																		
Gemini-3.1-Pro	19.21	**46.49**	27.19	8.43	85.54	23.26	9.01	29.88	13.85	16.13	55.51	7.69	7.74	31.46	12.42	11.71	69.94	8.69
GPT-Realtime-2.1	4.84	27.70	8.24	14.90	71.20	5.87	6.62	24.96	10.46	19.32	52.70	5.51	4.34	23.61	7.33	17.57	51.95	3.81
Gemini-3.8-Flash	**27.51**	45.63	**34.33**	**6.29**	**88.34**	**30.33**	**10.60**	36.20	**16.40**	17.04	**64.64**	**10.60**	8.52	35.23	13.72	11.70	**76.23**	**10.46**
Open-Source Offline Models																		
Qwen3.5-9B	3.69	24.41	6.41	17.85	44.26	2.84	4.75	24.15	7.94	22.00	17.36	1.38	2.34	24.81	4.28	23.79	29.24	1.25
JoyAI-VL-Interaction	3.23	27.92	5.79	20.05	41.05	2.38	4.53	**92.53**	8.64	91.20	10.04	0.87	2.92	**70.92**	5.61	65.11	21.13	1.19
Open-Source Streaming Models																		
MMDuet2	4.77	10.91	6.64	7.13	75.87	5.04	5.49	10.90	7.30	11.10	24.07	1.76	7.12	19.35	10.41	6.95	38.66	4.02
Mage-VL	1.97	11.93	3.38	10.19	9.84	0.33	7.58	18.40	10.74	9.78	9.65	1.04	3.05	13.48	4.97	10.38	7.07	0.35
LiveStar	2.95	10.75	4.63	7.93	27.01	1.25	7.31	9.75	8.36	5.98	3.85	0.32	2.61	4.63	3.34	3.73	10.36	0.35
ViSpeak	2.16	19.01	3.88	19.12	64.28	2.49	5.52	19.96	8.65	19.06	10.99	0.95	4.86	26.02	8.19	19.10	16.76	1.37
VideoLLM-Online	2.53	7.07	3.73	6.56	29.77	1.11	6.66	2.73	3.87	**1.71**	19.82	0.77	**15.24**	12.59	**13.79**	**1.51**	5.52	0.76
Ours																		
StreamMind-VL (SFT)	6.47	22.79	10.08	13.49	52.78	5.32	7.62	27.78	11.96	17.91	38.43	4.60	3.92	41.33	7.16	27.76	35.18	2.52
StreamMind-VL (RL)	7.90	20.84	11.46	10.71	56.67	6.49	9.01	21.81	12.75	12.20	42.85	5.46	4.98	34.48	8.70	19.03	41.00	3.57

Table 8: ***Trajectory evaluation with model-generated interaction history*** on StreamMind-Bench without the Harness (± 0 s tolerance).

B.4 General Timing and External Benchmarks

The general-timing results use Pointwise or Trajectory evaluation as indicated. EgoProactive, IP-IBench, and OVO-Bench retain their benchmark-specific evaluation protocols.

Model	$\tau = 2$ s						$\tau = 0$ s					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models												
Gemini-3.1-Pro	73.78	78.71	76.17	34.17	47.57	36.23	65.06	69.40	67.16	47.97	49.82	33.46
GPT-Realtime-2.1	65.29	82.16	72.76	48.30	40.64	29.57	56.15	70.66	62.57	65.34	41.09	25.71
Gemini-3.8-Flash	**75.14**	79.58	**77.30**	35.71	47.59	36.79	**68.79**	72.85	70.76	45.70	49.06	34.71
Open-Source Offline Models												
Qwen3.5-9B	52.64	83.96	64.71	88.56	61.52	39.81	51.17	81.61	62.90	92.05	61.44	38.65
JoyAI-VL-Interaction	69.34	86.15	76.84	53.41	67.11	51.57	67.44	83.80	**74.74**	56.90	67.67	50.58
Open-Source Streaming Models												
MMDuet2	37.06	45.76	40.95	9.17	49.16	20.13	14.98	18.50	16.55	14.40	45.18	7.48
Mage-VL	35.96	37.99	36.95	4.84	22.73	8.40	9.64	10.18	9.90	10.18	22.58	2.24
LiveStar	23.57	40.34	29.75	12.18	28.80	8.57	12.79	21.89	16.15	15.72	34.38	5.55
ViSpeak	28.21	84.84	42.34	22.66	33.06	14.00	12.19	36.67	18.30	31.91	32.39	5.93
VideoLLM-Online	45.06	17.46	25.17	**3.40**	12.54	3.16	30.65	11.88	17.12	**4.47**	14.40	2.47
Ours												
StreamMind-VL (SFT)	59.39	**96.93**	73.65	94.32	**71.32**	52.53	59.22	96.66	73.44	94.72	**71.35**	52.40
StreamMind-VL (RL)	60.37	96.60	74.30	85.47	71.28	**52.96**	59.90	**97.91**	74.33	86.53	71.30	**53.00**

Table 9: ***Pointwise evaluation with gold interaction history*** on general timing without the Harness.

Model	$\tau = 2$ s						$\tau = 0$ s					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Proprietary Models												
Gemini-3.1-Pro	29.75	91.35	44.88	26.46	68.26	30.64	20.62	63.33	31.11	31.85	68.00	21.15
GPT-Realtime-2.1	30.17	68.86	41.96	17.84	60.90	25.55	17.84	40.72	24.81	23.25	61.91	15.36
Gemini-3.8-Flash	**48.33**	85.50	**61.75**	12.53	**72.78**	**44.94**	30.38	53.75	**38.82**	18.63	**74.77**	**29.03**
Open-Source Offline Models												
Qwen3.5-9B	24.21	31.53	27.39	5.36	34.18	9.36	9.79	12.75	11.08	8.99	32.45	3.60
JoyAI-VL-Interaction	30.00	70.61	42.11	26.05	54.85	23.10	20.74	48.82	29.11	30.27	51.49	14.99
Open-Source Streaming Models												
MMDuet2	37.06	45.76	40.95	9.17	49.16	20.13	14.98	18.50	16.55	14.40	45.18	7.48
Mage-VL	35.96	37.99	36.95	4.84	22.73	8.40	9.64	10.18	9.90	10.18	22.58	2.24
LiveStar	23.57	40.34	29.75	12.18	28.80	8.57	12.79	21.89	16.15	15.72	34.38	5.55
ViSpeak	28.21	84.84	42.34	22.66	33.06	14.00	12.19	36.67	18.30	31.91	32.39	5.93
VideoLLM-Online	45.06	17.46	25.17	**3.40**	12.54	3.16	**30.65**	11.88	17.12	**4.47**	14.40	2.47
Ours												
StreamMind-VL (SFT)	22.95	86.92	36.31	49.13	47.87	17.38	19.51	73.89	30.87	51.65	43.09	13.30
StreamMind-VL (RL)	25.44	**92.28**	39.88	44.94	53.80	21.46	20.42	**74.06**	32.01	48.47	48.06	15.38

Table 10: ***Trajectory evaluation with self-generated interaction history*** on general timing without the Harness.

Model	EgoProactive				IPIBench			
	Macro	G-mean	Interrupt	Silent	Timing	Underst.	Repeated	Macro
Proprietary Models								
Gemini-3.1-Pro	58.75	58.75	59.42	58.09	53.40	33.12	7.55	31.36
GPT-Realtime-2.1	45.28	45.28	67.09	23.46	41.52	18.25	8.02	22.60
Gemini-3.8-Flash	**64.96**	**64.96**	66.19	63.72	**58.77**	**37.82**	12.74	**36.44**
Open-Source Offline Models								
Qwen3.5-9B	46.20	43.20	62.60	29.80	32.50	9.80	5.70	16.00
JoyAI-VL-Interaction	41.00	28.60	**70.40**	11.60	34.30	7.40	1.40	14.40
Open-Source Streaming Models								
MMDuet2	38.21	30.96	15.82	60.61	30.88	6.49	5.66	14.34
Mage-VL	40.52	35.48	60.10	20.95	26.65	11.47	5.19	14.44
LiveStar	53.62	53.16	46.58	60.67	14.65	3.40	0.47	6.17
ViSpeak	45.21	43.84	34.17	56.26	19.89	0.43	0.00	6.77
VideoLLM-Online	44.71	39.14	23.10	**66.32**	7.67	0.38	2.36	3.47
Ours								
StreamMind-VL (SFT)	53.04	52.20	62.45	43.63	33.80	8.30	**16.00**	19.40
StreamMind-VL (RL)	59.50	59.50	58.90	60.00	39.10	20.40	9.40	23.00

Table 11: Results on EgoProactive and IPIBench without the Harness.

Model	OVO-Bench			
	BT	RT	FAR	Overall
Open-Source Offline Models				
Qwen3.5-9B	53.94	63.94	55.24	57.71
JoyAI-VL-Interaction	47.09	67.99	38.87	51.32
Open-Source Streaming Models				
MMDuet2	47.02	61.59	37.91	48.84
Mage-VL	46.76	**81.12**	43.50	**63.94**
LiveStar	42.25	55.12	4.29	33.88
ViSpeak	**57.00**	65.80	53.61	60.55
VideoLLM-Online	31.99	29.58	5.69	22.42
Ours				
StreamMind-VL (SFT)	54.12	64.12	60.10	59.44
StreamMind-VL (RL)	53.42	64.55	**62.00**	59.29

Table 12: Results on OVO-Bench without the Harness.

B.5 StreamMind-Harness

We compare StreamMind-VL with and without the Harness under Trajectory evaluation at ±2 s, using the corresponding base model as a reference. The external-benchmark comparison also reports the matched SFT pair evaluated with and without the Harness.

Model	Explicit						Implicit						Dynamic					
	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1	P	R	F1	FPR	ANS	C-F1
Ours + Harness																		
StreamMind-VL (SFT) + Harness	18.49	28.51	22.43	3.80	61.79	13.86	18.20	29.39	22.48	6.33	38.93	8.75	10.01	43.20	16.25	12.25	38.98	6.33
StreamMind-VL (RL) + Harness	**24.57**	16.20	19.53	**1.49**	**72.83**	**14.22**	**21.53**	29.48	24.89	**2.02**	40.86	10.17	**14.80**	30.65	**19.96**	**5.68**	**44.62**	**8.91**
Ours																		
StreamMind-VL (SFT)	11.55	40.66	17.99	11.40	57.97	10.43	13.13	**47.91**	20.61	15.59	38.06	7.84	5.38	**56.76**	9.83	26.13	35.88	3.53
StreamMind-VL (RL)	15.80	41.68	**22.91**	8.27	60.07	13.76	18.02	43.64	**25.51**	9.68	**41.74**	**10.65**	7.60	52.58	13.28	17.11	39.55	5.25
Open-Source Offline Models																		
Qwen3.5-9B	7.94	**52.48**	13.79	14.56	44.98	6.20	9.37	47.62	15.66	19.30	16.63	2.60	4.87	51.65	8.90	20.95	27.67	2.46

Table 13: ***Trajectory evaluation*** of StreamMind-VL with and without the Harness on StreamMind-Bench (±2 s tolerance).

Model	General timing						IPIBench				EgoProactive			
	P	R	F1	FPR	ANS	C-F1	Timing	Underst.	Repeated	Macro	Macro	G-mean	Interrupt	Silent
Ours + Harness														
StreamMind-VL (SFT) + Harness	48.39	68.25	56.63	12.57	61.44	34.79	44.13	15.67	38.68	32.83	46.71	43.48	**63.75**	29.66
StreamMind-VL (RL) + Harness	**53.23**	67.22	**59.41**	6.84	**62.88**	**37.36**	**45.34**	16.88	**40.05**	**35.67**	48.79	44.0	3.87	29.80
Ours														
StreamMind-VL (SFT)	22.95	86.92	36.31	49.13	47.87	17.38	34.85	13.40	28.77	25.67	35.74	23.33	62.82	8.66
StreamMind-VL (RL)	25.44	**92.28**	39.88	44.94	53.80	21.46	39.10	**20.40**	29.40	23.00	**53.04**	**52.20**	62.45	**43.63**
Open-Source Offline Models														
Qwen3.5-9B	24.21	31.53	27.39	**5.36**	34.18	9.36	32.50	9.80	5.70	16.00	46.20	43.20	62.60	29.80

Table 14: StreamMind-Harness comparison on general timing (Trajectory, ±2 s), IPIBench, and EgoProactive.

B.6 Category Difficulty Across Intent Types

Building on the overall experimental results above, we further analyze performance across different subtypes to obtain a more fine-grained understanding of proactive interaction capabilities. Specifically, we examine *(i) the relative difficulty of different subtypes for existing systems*, *(ii) how model performance changes across evaluation settings*, and *(iii) how RL shifts response behavior and performance trade-offs across subtypes*. All analyses use a matching tolerance of ±2 s. Figures 8–10 report subtype-level results under the Pointwise and Trajectory evaluation settings used in Tables 3 and 7. Figure 8 summarizes the average performance of all available models across subtypes under each evaluation protocol to characterize the overall difficulty of different task types. Pointwise includes seven models, while Trajectory additionally includes five native streaming baselines.

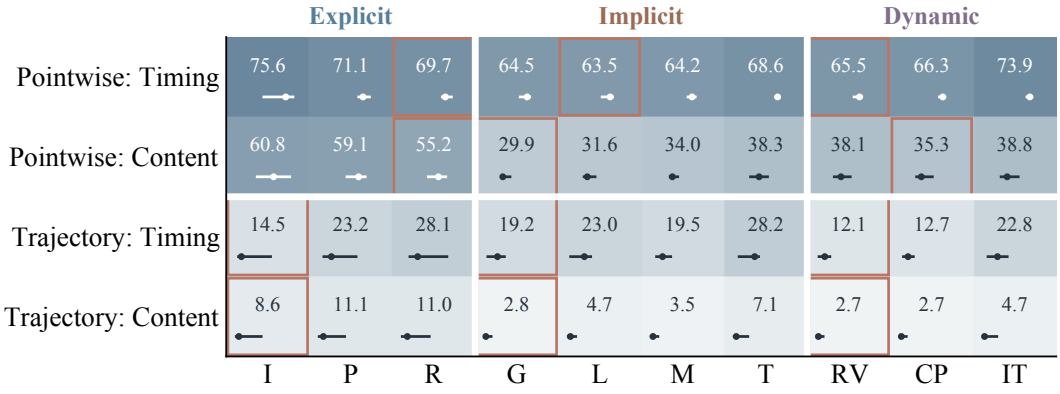


Figure 8: **Category difficulty across StreamMind-Bench.** Each cell gives the mean score across all models available in that setting; the line spans the per-model range and the dot marks the median. Outlines identify the lowest mean within each intent type and metric. Pointwise uses gold interaction history (n=7 models), whereas Trajectory uses model-generated interaction history and also includes native streaming baselines (n=12 models). All results use the ±2 s tolerance.

Trajectory evaluation reveals substantially greater difficulty under sustained interaction. As shown in Figure 8, performance drops markedly from Pointwise to Trajectory evaluation across all

subtypes. Under Pointwise evaluation, Timing F1 remains relatively high at 63.5–75.6, indicating that models can often recognize locally appropriate response moments when evaluated at isolated timestamps. In contrast, Trajectory Timing F1 falls to 12.1–28.2 and Content-F1 to 2.7–11.1, revealing a substantial gap between local perception and sustained interaction. This suggests that maintaining reliable assistance over a full trajectory requires more than recognizing individual events: models must continuously update active intents, track task state, and regulate responses based on their evolving interaction history.

Figure 8 further shows that subtype difficulty changes under sustained interaction. For Explicit intents, I achieves the highest Pointwise Timing F1 at 75.6 but the lowest Trajectory Timing F1 at 14.5, whereas R reaches 28.1. Among Implicit intents, Tool/GUI remains relatively strong, while Guidance becomes more challenging. For Dynamic intents, Revision and Composition underperform Interruption, highlighting the difficulty of continuously revising and composing active requirements. Overall, Pointwise evaluation primarily reflects local recognition and timing, whereas Trajectory evaluation exposes the additional challenges of maintaining coherent and reliable behavior throughout continuous, evolving interaction.

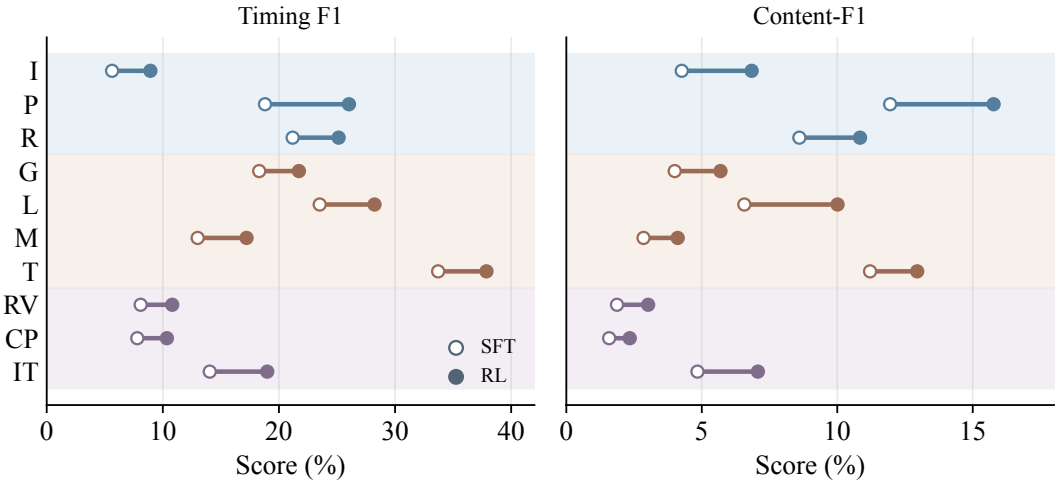


Figure 9: **Absolute subtype profiles of StreamMind-VL (SFT) and StreamMind-VL (RL) under Trajectory evaluation.** Hollow and filled endpoints show SFT and RL, respectively; the connecting segment exposes both direction and magnitude. Background bands encode the three intent types. Scores are percentage points at ± 2 s.

Different subtypes exhibit distinct performance patterns. As shown in Figure 9, different subtypes exhibit clear differences in both response timing and content quality. StreamMind-VL performs best on Tool/GUI, achieving a Timing F1 of 37.9 and a Content-F1 of 13.0, while Monitoring reaches only 17.2 and 4.1, respectively, suggesting that tasks with explicit operational goals are easier to handle reliably than continuous monitoring. For Explicit intents, P and R achieve Timing F1 scores of 26.0 and 25.1, substantially higher than I at 8.9, indicating that single-event cases are more prone to missed or mistimed responses during sustained interaction. For Dynamic intents, Revision and Composition both achieve Timing F1 scores around 10, whereas Interruption reaches 19.0 and also attains higher Content-F1, suggesting that continuously revising or composing existing requirements places greater demands on maintaining the currently valid intent and task state.

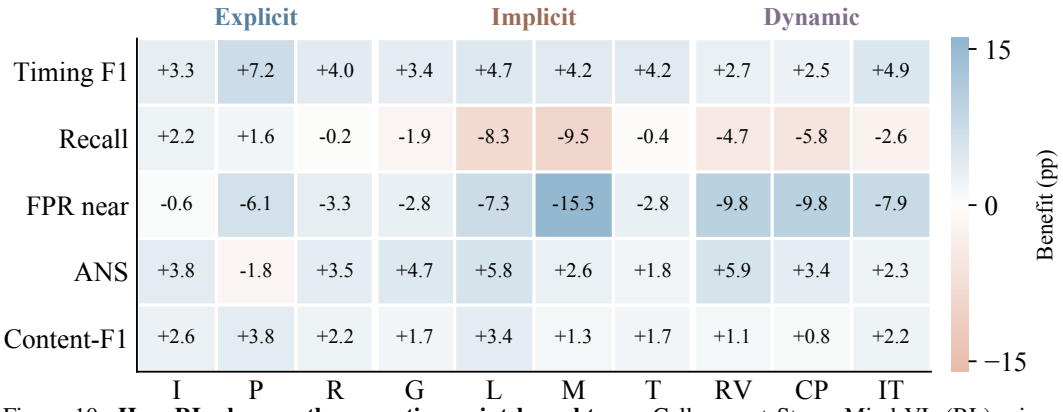


Figure 10: **How RL changes the operating point by subtype.** Cells report StreamMind-VL (RL) minus StreamMind-VL (SFT) in percentage points under Trajectory evaluation. Numbers retain the metric's natural sign; color encodes benefit, so a negative $\mathrm{FPR}_{\mathrm{near}}$ change is blue. The visualization jointly exposes timing, coverage, false interventions, answer quality, and Content-F1.

RL shifts the policy toward more selective intervention. As shown in Figure 10, RL consistently improves Timing F1 and Content-F1 while reducing $\mathrm{FPR}_{\mathrm{near}}$ across all subtypes, indicating that its primary effect is to suppress unnecessary triggers and concentrate responses around more appropriate moments. The gains vary across subtypes: P shows the largest Timing F1 improvement of $+7.2$, while Monitoring exhibits the largest FPR reduction of -15.3, suggesting that RL is particularly effective in settings prone to over-triggering. RL makes proactive behavior more selective and better aligned with appropriate intervention opportunities across subtypes.

C Dataset Construction and Evaluation Details

This section consolidates the taxonomy, source provenance, training mixture, evaluation cohorts, and metrics summarized in the main paper.

C.1 Overview of Evaluation Datasets

Our evaluation covers proactive interaction and general video understanding. The former tests when to intervene and whether the response supports the ongoing task; the latter checks whether proactive policy learning preserves video perception and temporal reasoning. We describe the datasets below; protocols and metrics are detailed in Appendix C.5.

StreamMind-Bench. Our primary benchmark evaluates sustained proactive interaction under Explicit, Implicit, and Dynamic Intents. Its 4,100 evaluation samples cover ten subtypes, ranging from responding to specified events to inferring useful assistance and tracking revised or interleaved requests. Pointwise evaluation isolates response decisions at individual timestamps, while Trajectory evaluation tests continuous interaction with model-generated history. The following subsections describe its taxonomy, sources, and construction.

General-Timing. We construct General-Timing as a separate evaluation subset drawn from the publicly released training data of JoyAI-VL-Interaction (Yao et al., 2026b). The selected samples are held out from our training: they are used for neither supervised fine-tuning nor reinforcement learning, and are reserved exclusively for evaluation. This subset tests response timing and content quality on additional streaming-video interactions beyond StreamMind-Bench. We apply our Pointwise and Trajectory protocols and report results in Tables 9 and 10.

EgoProactive. EgoProactive (Kundu et al., 2026) evaluates proactive procedural assistance from wearable, egocentric videos. Its annotations include out-of-plan deviations and recovery guidance, requiring an assistant to recognize task progress, decide when an interruption is useful, and provide appropriate corrective assistance. We use its official evaluation protocol to assess proactive assistance in physical task execution.

IPIBench. IPIBench (Li et al., 2026b) evaluates interactive proactive intelligence under continuous video streams. It covers proactive monitoring, proactive task management, and interleaved reactive–proactive requests, including multi-turn interactions in which users add, modify, or cancel requests. It complements StreamMind-Bench by testing the coordination of response timing and evolving interaction requirements under an external protocol. We follow its official evaluation setup; results for IPIBench and EgoProactive are reported in Table 11.

MVBench. MVBench (Li et al., 2024) evaluates video understanding across diverse temporal tasks that require reasoning beyond a single frame. We use it to assess whether proactive policy learning preserves general temporal perception and reasoning, following its official evaluation protocol.

Video-MME. Video-MME (Fu et al., 2025a) assesses video understanding across varied content domains and video durations. It complements MVBench by testing broader video comprehension over different temporal scales. We follow its official evaluation protocol and report the corresponding benchmark score.

OVO-Bench. OVO-Bench (Li et al., 2025b) evaluates timestamp-aware online video understanding through backward tracing, real-time perception, and forward active responding. These settings test whether a model can recall relevant past events, interpret the current observation, and respond as a stream unfolds. We retain its official task definitions and evaluation metrics.

OVBench. OVBench (Huang et al., 2025) evaluates online video understanding as visual observations arrive over time. We use it to assess comprehension of ongoing video streams alongside the proactive interaction benchmarks, following its official evaluation protocol and reported metrics.

C.2 Task Taxonomy and Evaluation Sources

Each example contains a video stream, an initial user instruction, optional timestamped user turns, and one or more verified evidence points. An evidence point stores an acceptable intervention window, a preferred time, and a gold response. The ten subtypes in Table 15 describe what must be tracked rather than merely the visual domain. Selected data-construction prompts are provided in Appendix F.1.

Source coverage. Physical-task videos include BARISTA (Knab et al., 2026), Assembly101 (Sener et al., 2022), HoloAssist (Wang et al., 2023), YouCook2 (Zhou et al., 2017), and Ego-Exo4D (Grauman et al., 2024). Learning sources additionally include ActivityNet (Caba Heilbron et al., 2015), Video-MMLU (Song et al., 2025), and Video-MMMU (Hu et al., 2025); monitoring sources include EgoSchema (Mangalam et al., 2023), Perception Test (Pătrăucean et al., 2023), Charades (Sigurdsson et al., 2016), and MLVU (Zhou et al., 2025). GUI sources include AgentNet (Wang et al., 2025b), VideoCUA (Jian et al., 2026), FingerTip (Yang et al., 2026b), GUI-World (Chen et al., 2025a), and GUI-360 (Mu et al., 2025). Source videos are reused with our intent-specific requests and evidence-point annotations; their original benchmark scores are not used as StreamMind-Bench labels.

Annotation and provenance. Candidate evidence points are produced from the source video and task record, then checked by four reviewers for request fidelity, visible evidence, timing, and assistance content. Accepted videos are serialized at 1 FPS; at time t, the record exposes only frames and user turns available by t. The normalized metadata retains the source collection and video identifier, duration, subtype, initial and updated requests, evidence-point windows, gold responses, review record, and negative-probe type. Dynamic examples additionally retain the active intent transition, making superseded and newly valid constraints distinguishable.

Category	Subtype	Operational definition	Samples	Evidence	Hours
Explicit	I	One requested event or answer; verified Explicit Intents pool	200	200	7.5
	P	State transition or irreversible point; verified Explicit Intents pool	200	723	8.2
	R	Repeated occurrences of a requested event; verified Explicit Intents pool	200	794	6.4
Implicit	G	Guidance for the next task action	500	3,702	26.7
	L	Learning and explanation	500	3,287	18.4
	M	State monitoring for task progress and changes	500	2,701	18.9
	T	Tool use in software interfaces	500	2,861	15.3
Dynamic	RV	Revision replaces an active intent; Dynamic Intents construction pool	500	1,885	25.2
	CP	Composition adds a intent while preserving prior ones; Dynamic Intents construction pool	500	2,340	31.2
	IT	Interruption inserts an immediate request while the proactive task remains active; Dynamic Intents construction pool	500	2,053	25.6
		Target balanced evaluation allocation	**4,100**	**20,546**	**183.4**

Table 15: **StreamMind-Bench target allocation and source coverage.** "Evidence" counts proactive gold evidence points; reactive questions are excluded. Implicit and Dynamic statistics are measured on their balanced manifests. Explicit evidence-point and duration statistics are proportional estimates from the qualified I/P/R pool at 200 samples per subtype.

C.3 Human Review Criteria and Inter-Annotator Agreement

Review unit and acceptance rule. The review unit is a complete candidate sample: the source video, initial instruction, timestamped user turns, evidence windows, preferred response times, and reference answers. Four human reviewers independently assess the same sample against Table 16. A reviewer approves a sample only when all applicable criteria are satisfied; a required correction counts as non-approval of the current version. A sample is retained only if all four reviewers approve the same final version. Thus, three approvals are insufficient. Samples with any objection must be corrected and reviewed again by all four reviewers, or discarded; changing an annotation invalidates approval of its earlier version. Unresolvable ambiguity or insufficient visual evidence requires exclusion rather than a guessed reference answer.

Independent agreement audit. Unanimous acceptance is a quality-control rule, not itself evidence of high inter-annotator agreement. The agreement audit must use the original, independent judgments on a fixed set of candidate samples *before* discussion, correction, or acceptance filtering, including candidates that are not retained. Each reviewer supplies a binary decision, approve or require revision/reject, under the same rubric. Repeated versions of a candidate must not be counted as independent samples. The audit should record the candidate count and decision distribution for each intent type.

Human–human agreement. Fleiss' κ (Fleiss, 1971) measures chance-corrected agreement among the four reviewers. For N candidates, let n_{ic} be the number of reviewers assigning candidate i to decision $c \in \{0, 1\}$, with $\sum_c n_{ic} = 4$. Then

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_c n_{ic}(n_{ic} - 1)}{4 \cdot 3}, \qquad p_c = \frac{\sum_i n_{ic}}{4N}, \qquad \kappa_{\mathrm{F}} = \frac{\bar{P} - \sum_c p_c^2}{1 - \sum_c p_c^2}.$$

The audit should additionally report the raw pairwise agreement $\bar{P}$ and the fraction of candidates with four identical decisions. Computing κ only on retained, unanimously approved samples is invalid as evidence of general reviewer reliability: filtering enforces agreement, and a single decision class makes the chance-correction denominator zero.

Table 16: **Human review rubric.** Each reviewer checks every applicable criterion. Acceptance requires four independent approvals of the same sample version, with no unresolved issue.

Criterion	Standard for approval
Intent fidelity	The instruction is coherent and compatible with the task. The assigned Explicit, Implicit, or Dynamic category matches the interaction; inferred intents are supported by the user's context, not invented.
Evidence and causality	Every annotated intervention has identifiable visual or interaction evidence available by its response time. Future frames and later user turns are not needed to justify the response.
Coverage and non-redundancy	Distinct, necessary intervention opportunities are covered, without duplicate evidence points or repeated assistance after the relevant intent has already been satisfied.
Temporal validity	Each response window and preferred timestamp correspond to a moment when the evidence is sufficient and assistance is still useful; the preferred time lies within the window and neither anticipates evidence nor misses the actionable moment.
Reference-answer quality	The reference is correct, relevant, and useful for the active intent. It avoids unsupported details, contradictory advice, and unnecessary instructions, while remaining consistent with the visible state.
Dynamic-intent consistency	For Dynamic samples, timestamped revisions, compositions, and interruptions are coherent. References follow the latest valid intent and preserve unaffected constraints without acting on superseded ones; this criterion is not applicable to other categories.
Record integrity	The video, source identifier, user-turn timestamps, evidence windows, and reference answers are complete and aligned. Each decision refers to an identifiable sample version.

Gemini–human agreement. For this comparison, Gemini-3.8-Flash must independently assign the same binary review decision to the same frozen candidates, using the rubric without access to human judgments or post-review corrections. Its original candidate annotations alone are not binary audit decisions. Compute Cohen's κ (Cohen, 1960) separately between Gemini and each reviewer, $\kappa_{\mathrm{C}}^{(r)} = (p_o^{(r)} - p_e^{(r)})/(1 - p_e^{(r)})$, where $p_o^{(r)}$ is observed agreement and $p_e^{(r)}$ is agreement expected from their respective decision marginals. Report the mean of the four pairwise coefficients and their range, rather than treating the four human judgments as one independent rater.

Table 17: **Inter-annotator agreement.** Human–human agreement is measured by Fleiss' κ; Gemini–human agreement is the mean of four pairwise Cohen's κ coefficients, with ranges in parentheses.

Audit subset	Candidates	Human–Human Fleiss' κ ↑	Gemini-3.8-Flash–Human Cohen's κ ↑ (mean of four)
Explicit	872	0.87	0.79 (0.76–0.82)
Implicit	2,946	0.84	0.75 (0.72–0.78)
Dynamic	2,187	0.82	0.73 (0.70–0.77)
All candidates	6,005	0.84	0.76 (0.70–0.82)

Analysis. As shown in Table 17, human–human Fleiss' κ ranges from 0.82 to 0.87 across all categories, falling within the "almost perfect agreement" band (>0.81) (Landis & Koch, 1977), indicating that the four reviewers maintained highly consistent judgments under the shared rubric. Explicit achieves the highest agreement (0.87), while Dynamic is slightly lower (0.82), consistent with the inherently greater complexity of annotating evolving intents. Gemini–human Cohen's κ ranges from 0.73 to 0.79, within the "substantial agreement" band (0.61–0.80), with narrow per-pair ranges, indicating that Gemini's review decisions are well aligned with human judgments. Together, these results confirm the reliability and quality of the StreamMind-Bench annotation pipeline.

C.4 Training Mixture and Evaluation Cohorts

Table 18 reports the SFT data manifest containing 70K task-aligned examples and 768K general streaming examples. Task-aligned Explicit and Implicit examples preserve their original task structure, while the general streaming mixture broadens coverage across visual domains, narration styles,

temporal scales, and GUI scenarios. Localized RL starts from a separate balanced 15K seed pool, consisting of 5K tasks each from Implicit Intents, Explicit Intents, and general streaming data.

Training component	Recorded source families	Examples	Use
Task-aligned Explicit	Explicit Intents event families	35,000	Streaming SFT
Task-aligned Implicit	Guidance, learning, monitoring, and tool/GUI	35,000	Streaming SFT
General streaming core	JoyAI mixture: Live-WhisperX, Shot2Story, TimeLens, GUI-World, VideoGPT+, Charades-Ego, InternVid, ActivityNet, WebVid, and smaller sources	763,607	Streaming SFT
General streaming auxiliary	TimeChat-Online, VideoChat-Flash, TimeLens, EgoIT, and StreamO-Instruct	4,393	Streaming SFT
	Full SFT manifest	**838,000**	**1 FPS streams**
Localized RL seed	Implicit / Explicit / general streaming, 5K each	15,000	Local rollout windows

Table 18: **Training-data provenance.** The SFT mixture contains 70K task-aligned examples and 768K general streaming examples. Localized RL uses a separate 15K seed pool, yielding 85K task-oriented training examples in total when combined with task-aligned SFT. Dataset names indicate source families and do not imply a shared annotation protocol.

General-streaming sources. The inherited mixture includes Live-WhisperX (Chen et al., 2025b), Shot2Story (Han et al., 2025), TimeLens (Zhang et al., 2026), VideoGPT+ (Maaz et al., 2024), Charades-Ego (Sigurdsson et al., 2018), InternVid (Wang et al., 2024), WebVid (Bain et al., 2022), and the ActivityNet and GUI-World sources cited above. Auxiliary data comes from TimeChat-Online (Yao et al., 2025), VideoChat-Flash (Li et al., 2025a), TimeLens, EgoIT-99K (released with EgoLife) (Yang et al., 2026a), and Streamo-Instruct (Xia et al., 2026).

C.5 Evaluation Protocols and Metrics

Evaluation	What is scored	Reported measures	Judge / scorer
Pointwise	Independent probes; supplied history	Precision, recall, F1, $\mathrm{FPR}_{\mathrm{near}}$, ANS, Content-F1	Temporal matcher; Claude Sonnet 4.6 (Anthropic, 2026)
Trajectory	Complete 1 FPS rollout; generated history	Same six scores at ± 2, ± 1, ± 0 s	Same matcher and judge
General Timing	Streaming transfer cohort	Same timing and content scores	Same protocol
EgoProactive / IPIBench	Session-level interaction	Macro, G-mean, Interrupt, Silent; Timing, Understanding, Repeated	Official protocols
Video suites	MVBench, Video-MME, OVO-Bench, OVBench	Accuracy or official aggregate	Official scorers

Table 19: **Metrics and judges used in the paper.** External benchmarks retain their official evaluation protocols and are not rescored by the StreamMind-Bench content judge.

Temporal and content scores. At tolerance τ, predictions and evidence points within $\pm\tau$ are matched chronologically and one-to-one. Thus $P = \mathrm{TP}/N_{\mathrm{pred}}$, $R = \mathrm{TP}/N_{\mathrm{gold}}$, and F1 is their harmonic mean; $\mathrm{FPR}_{\mathrm{near}}$ measures responses on designated nearby negative probes. For each temporally matched response, Claude Sonnet 4.6 (Anthropic, 2026) assigns a 0–4 content score r that is normalized to $s_{g,p} = r/4 \in [0, 1]$. ANS is its mean over matched responses. Content-F1 is $2\sum_{(g,p)\in\mathcal{M}} s_{g,p}/(N_{\mathrm{pred}} + N_{\mathrm{gold}})$, so missed intents and extra interventions remain penalized instead of disappearing from a conditional answer average. For Dynamic Intents, ANS and Content-F1 score proactive responses only; reactive answers remain in the interaction history.

D StreamMind-Harness Implementation Details

This section provides implementation details for the four functional components introduced in Section 4.1. StreamMind-Harness maintains persistent interaction state and organizes causal evidence for the interaction policy. These components jointly support a single interaction process and should not be interpreted as four independent models. We use Qwen3-VL-4B-Instruct (Bai et al., 2025) as an auxiliary visual-memory model for memory summarization and consolidation with greedy decoding, while the StreamMind-VL checkpoint under evaluation serves as the interaction policy. The auxiliary model is used off-the-shelf and receives no additional training.

Table 20: **Harness configuration.** Temporal parameters correspond to the chunked-streaming setting; benchmark-specific native adapters are described separately below. Retrieval limits denote maximum capacities rather than fixed numbers of items injected into the policy context.

Component	Setting
Memory model	Qwen3-VL-4B-Instruct; separate inference process
Memory input	At most 64 frames per call; 262,144 pixels per frame
Chunk and submission schedule	$T_s = 32$ steps; submission lookahead $= 8$ steps
Consolidation and retention	$M = 5$ mid-term summaries; $L = 8$ long-term blocks
Evidence selection	Up to 3 facts and 3 ordered transitions
Compact state evidence	First rendered evidence line, capped at 180 characters
Skill recall	10 fixed Skills; at most 1 selected per context construction
Procedural memory	Up to 8 completed steps and 12 temporal anchors

D.1 Interaction State

At time t, the interaction state is represented as $\mathcal{S}_t = \langle \mathcal{A}_t, \mathcal{P}_t, \mathcal{R}_t \rangle$, where $\mathcal{A}_t$ stores active intents, $\mathcal{P}_t$ records observed task progress, and $\mathcal{R}_t$ tracks responses that have actually been delivered. State changes are represented as typed events with an identifier, timestamp, source, and payload. A deterministic update function applies valid events to the corresponding state fields and ignores duplicate updates. User instructions may introduce, revise, or retire requests; visual observations may establish task-relevant events; and observed progress or explicit user confirmation may update procedural state. A delivery record is created only after a response has been emitted to the user.

We enforce source-specific update constraints so that information retains its semantic role across the interaction. In particular, a request specifies what should be monitored rather than what has already occurred; a reference plan describes possible future steps rather than completed actions; and an assistant suggestion is not treated as evidence of user progress. Fields without sufficient support remain unresolved. Historical evidence may be exposed to the policy as context, but cannot by itself create, revise, or retire an intent.

Procedural memory retains at most eight completed steps and twelve evidence intervals that serve as temporal retrieval anchors. Progress is updated only when supported by observation or explicit confirmation. For event-monitoring requests, the auxiliary model additionally extracts and normalizes evidence from causal frames within a two-second window. This event parser updates evidence and event state only; it does not generate user-facing responses and therefore does not constitute a second action policy.

D.2 Intent Lifecycle

Each active intent is assigned a stable identifier and maintains an independent record of its criterion, associated event, evidence reference, and delivery status. A newly registered criterion begins in an evidence-waiting state. Once sufficient evidence supports a match, the criterion is linked to a concrete event instance. If the policy subsequently responds, the resulting delivery record identifies both the addressed event and the associated intents. This design allows multiple intents to remain active concurrently without treating a response to one intent as completion of the others.

The lifecycle explicitly separates *request validity* from *event coverage*. Answering a one-time question closes that pending question. By contrast, reporting an occurrence under a standing request marks only that event instance as addressed while keeping the request active for future evidence.

Closing an event returns the corresponding instance to an evidence-waiting state, whereas explicit user cancellation retires the request itself. These transitions update interaction bookkeeping rather than policy behavior: neither an active intent nor a matched event deterministically triggers a response.

Skill library and selection. The Harness maintains ten fixed, read-only Skill records covering visual question answering, multiple-choice answering, event detection, explanation, hazard monitoring, milestone reporting, mistake monitoring, narration, next-step guidance, and stage summarization. Each Skill specifies request cues, evidence to inspect, response and waiting conditions, interaction cadence, and answer form. Skills are reusable behavioral specifications rather than neural models. Each active criterion maintains its own execution instance, including phase, evidence reference, event association, and delivery status.

Skill selection first restricts candidates to those compatible with the current interaction type and requires non-empty observed memory, procedural context, or event context. A pending question prioritizes the question-answering Skill, followed by direct matches to active criterion types. If neither applies, lexical overlap between the request and observed context is used as a fallback, with at least two shared terms required. At most one Skill is selected for each context construction, and selecting no Skill is a valid outcome. For example, next-step guidance is conditioned on observed task progress, whereas event detection tracks a requested transition and whether the corresponding occurrence has already been reported. Importantly, selecting a Skill does not imply that its response condition has been satisfied.

D.3 Hierarchical Memory

Storage and consolidation. The memory system contains three levels: recent raw visual context, timestamped mid-term summaries, and older consolidated text blocks. Under the chunked-streaming configuration, we use $T_s = 32$ steps per chunk, consolidate every $M = 5$ mid-term summaries, and retain up to $L = 8$ long-term blocks. At one step per second, each chunk corresponds to 32 seconds of video. The auxiliary model processes frames from the short-term buffer in chronological order, uniformly subsampling when the 64-frame input limit is exceeded, and conditions each summary on the current request. Consolidation of older summaries is performed using text-only input.

Mid-term summaries preserve task-relevant object states, quantities, on-screen text, and temporal transitions. Intervals without meaningful change are collapsed, unsupported attributes remain unknown, and the most recently observed state is retained when visually established. Each ordered transition records its temporal interval, actor, action, object, prior and resulting states, temporal relation, and confidence. The mid-term summarization prompt is capped at 220 words, while procedural fields distinguish observed current and completed steps from request-supported references to possible next steps.

Every M consecutive mid-term summaries are consolidated into one long-term block using a 300-word prompt budget. Consolidation removes redundant static descriptions while preserving timestamps, distinct actions, temporal order, and supported procedural information. Each mid-term summary belongs to exactly one non-overlapping consolidation group. Once the number of long-term blocks reaches the capacity L, older blocks are evicted in first-in, first-out order.

To preserve causality under asynchronous execution, summarization is submitted before the chunk boundary using only frames already observed at submission time. Frames arriving afterward remain in the next short-term buffer, preventing gaps in temporal coverage. Mid-term inputs remain available while long-term consolidation is pending. If consolidation fails or produces an empty output, their concatenation is retained instead, preserving the available evidence without assuming that generated summaries are lossless.

Indexing and retrieval. Summary facts are indexed using normalized text together with stable identifiers, source intervals, and lexical terms. Repeated observations of the same fact update its observation interval rather than creating duplicate entries. Ordered transitions are indexed separately to preserve event sequence. Request descriptions and reference-plan fields are excluded from the factual index, and response records remain separate from visual evidence.

At retrieval time, the Harness constructs a focus string from the pending question, current request, and relevant procedural state. Candidate entries are ranked primarily by lexical overlap with this focus and secondarily by recency, with additional relevance assigned to entries overlapping task-relevant temporal anchors. Facts and ordered transitions are selected independently, with up to three candidates from each category and at most six entries before context projection. If no candidate matches the focus, the most recent entries are retained as a fallback. Selected transitions are re-ordered chronologically before projection. For native event-monitoring adapters, target retrieval is further restricted to at most three matching evidence lines, excluding negative or uncertain statements. No embedding model or learned retriever is used.

The retrieval capacity is intentionally decoupled from the amount of evidence exposed to the policy. In the compact lifecycle projection, only the first rendered evidence line is retained and truncated to 180 characters; this line may contain a joined excerpt of multiple selected facts. For a pending question, the bounded fact and transition blocks may instead be exposed directly. Retrieved items are always treated as timestamped historical context, and current observations take precedence whenever they supersede an earlier state.

D.4 State Integration

Before each policy call, the Harness compiles the relevant state fields and retrieved evidence into a compact textual context and combines it with the causal video input,

$$\tilde{c}_t = \phi(c_t, \mathcal{S}_t, m_t),$$

where c_t denotes the causal interaction context, $\mathcal{S}_t$ the persistent interaction state, and m_t the retrieved memory. The projected context explicitly distinguishes pending questions, observed progress, event status, delivery records, uncertainty, and historical evidence. Inactive or empty fields are omitted, and the full memory store is not appended by default. Within a context epoch, the header remains fixed while subsequent observations and actions are appended, allowing the policy to reuse the shared prompt prefix.

Skill Projection for Policy Inference. Although each Skill stores a complete behavioral specification, the policy receives only the subset required for the current interaction. For single-criterion guidance, monitoring, and explanation, the projected context includes the selected Skill's execution phase, evidence reference, response basis, and a compact cadence description. Question-answering and multi-criterion contexts expose only the answer-form field, while single-criterion narration retains a compact symbolic representation. The complete Skill specification is therefore not copied into the prompt, and Skill metadata does not replace policy generation. Request changes update the interaction type causally; after an intervening question is answered, the context of the standing request is restored.

The interaction policy π remains the sole component that decides whether to remain silent or respond and, when responding, generates the user-facing text. Once a non-empty response is delivered, its timestamp and intent/event associations are committed to $\mathcal{R}_t$, while its exact text is preserved in the interaction history. Subsequent context projections may retain the most recent response verbatim and compress older responses into action markers. Assistant outputs are excluded from visual summarization. Response verification, cadence gating, and post-generation rewriting are disabled. Consequently, this feedback path records delivered assistance for future state construction without introducing new visual evidence or modifying the policy output.

E Qualitative Examples and Case Studies

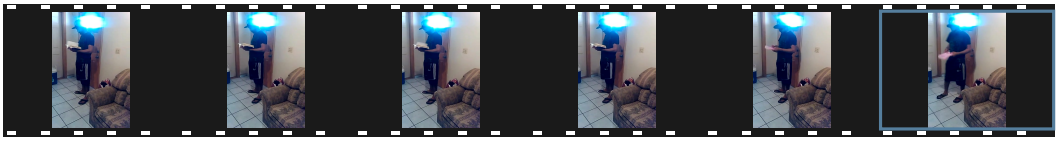
This section presents annotated examples from all ten task subtypes, followed by additional comparisons of model responses during sustained interaction.

E.1 Representative Samples from Every Subtype

Figures 11–17 show two verified samples per subtype. Each full-width sample contains the original instruction, a chronological six-frame filmstrip, and timestamped reference assistance. Three annotated interventions are shown per sample, except for single-event I, whose definition permits only one gold evidence point. Each pair of adjacent frames shows the second before an intervention and its annotated time; gaps between pairs are indicated by the timestamps. The outlined frame marks the intervention. Dynamic examples include the user updates applicable to the displayed timeline.

I | One requested event — Example A

User instruction: Notify me as soon as the man throws the book onto the armchair.



Evidence 1 (6s): The man is throwing the book onto the armchair.

I | One requested event — Example B

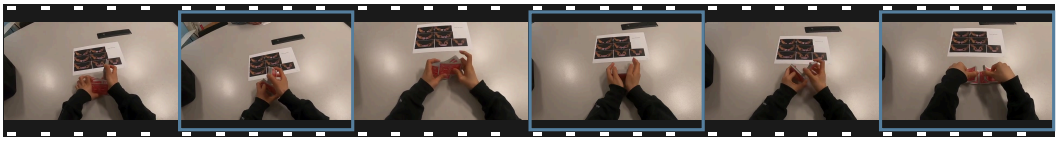
User instruction: Notify immediately when a woman holding a beverage cup appears in the office.



Evidence 1 (24s): A woman holding a beverage cup has entered the office area.

P | State transition — Example A

User instruction: Monitor the state of the playing card deck and report whenever it transitions between being a single stack and being split into two stacks.



Evidence 1 (4s): The deck has transitioned from a single stack to two stacks.

Evidence 2 (10s): The deck has transitioned from two stacks to a single stack.

Evidence 3 (14s): The deck has transitioned from a single stack to two stacks.

Figure 11: **Annotated samples (1/7).** Explicit Intents: single-event requests (I, two samples) and state transitions (P, first sample). I has one annotated evidence point per sample by definition.

P | State transition — Example B

User instruction: Monitor the raccoon and identify transitions in its eating state (eating vs not eating).



13s 14s 37s 38s 45s 46s

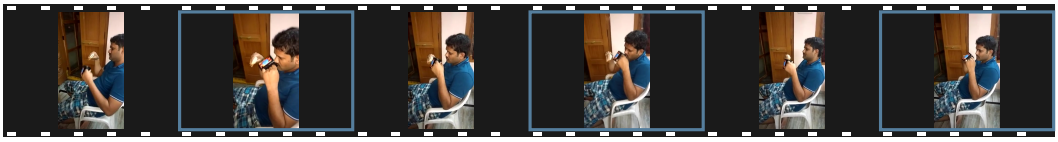
Evidence 1 (14s): The raccoon has finished the kibble and stopped eating.

Evidence 2 (38s): The raccoon has reached the bowl on the counter and started eating.

Evidence 3 (46s): The raccoon was pulled away from the food bowl and stopped eating.

R | Repeated occurrences — Example A

User instruction: Notify me every time the man takes a sip from his mug.



7s 8s 16s 17s 27s 28s

Evidence 1 (8s): The man has started taking his first sip from the mug.

Evidence 2 (17s): The man has started taking a second sip from the mug.

Evidence 3 (28s): The man has started taking a third sip from the mug.

R | Repeated occurrences — Example B

User instruction: Notify me each time the mechanic spins the bicycle's front wheel.



43s 44s 58s 59s 87s 88s

Evidence 1 (44s): The mechanic spins the front wheel for the first test.

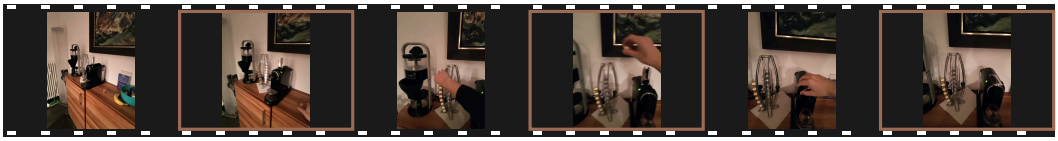
Evidence 2 (59s): The mechanic spins the front wheel for a second inspection.

Evidence 3 (88s): The mechanic spins the front wheel for a third check after tool adjustment.

Figure 12: **Annotated samples (2/7).** Explicit Intents: state transitions (P, second sample) and repeated events (R, two samples), with three reference interventions per sample.

G | Hands-on guidance — Example A

User instruction: Walk me through using this capsule espresso machine step by step as I prepare my drink, and let me know if I do something incorrectly.



Evidence 1 (3s): Now go ahead and pick a coffee capsule from the holder, then place it into the top slot of the machine.

Evidence 2 (8s): Now push the top lever all the way down to lock the capsule firmly into place.

Evidence 3 (13s): Now press one of the top brew buttons to start extracting your espresso shot.

G | Hands-on guidance — Example B

User instruction: Guide me step by step through assembling this metal plant stand tray table, and stop me if I make any assembly errors.



Evidence 1 (120s): Cross the two leg frames into an X shape and line up the center screw holes where they intersect.

Evidence 2 (136s): Cross the two frame pieces at the central notch to form the X-base, and make sure the middle screw holes line up.

Evidence 3 (165s): Before inserting the central screw, double-check that the holes on both interlocked frame legs line up straight.

L | Learning — Example A

User instruction: Walk me through this hedge trimming tutorial, explaining the core trimming techniques, and leaf inspection methods.



Evidence 1 (3s): Before turning on the trimmer, taking stock of your garden layout helps you plan your cutting path and ensure your extension cord stays safely out of the way.

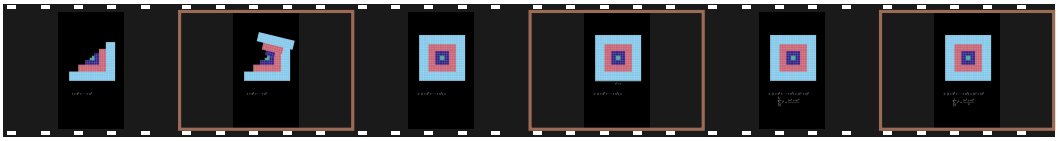
Evidence 2 (59s): Hold the trimmer firmly with both hands for full control, keeping the blade horizontal and parallel to the surface so you can guide it in smooth, sweeping motions.

Evidence 3 (120s): Checking the freshly trimmed surface by hand lets you feel for hidden uneven spots and ensure the blades made clean cuts rather than tearing the leaves.

Figure 13: **Annotated samples (3/7).** Implicit Intents: Guidance (G, two samples) and Learning (L, first sample), each with three evidence points.

L | Learning — Example B

User instruction: Walk me through this visual sum of cubes proof, and explain how the tile geometry translates into the algebraic summation formula.



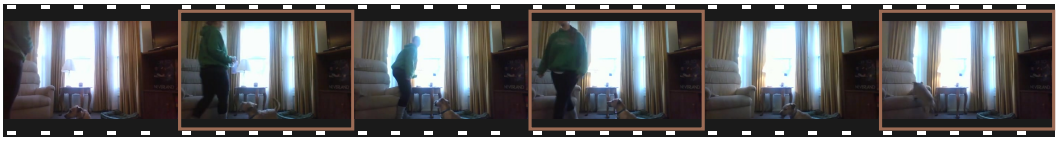
Evidence 1 (18s): Combining four identical tile sets fills in the open gaps to create a complete, symmetrical square, making it much easier to express the total area algebraically as four times the sum of ...

Evidence 2 (24s): That $n^2 + n$ side length comes from combining the dimensions of all n tile layers across the interlocked square.

Evidence 3 (39s): Since four identical tile stacks were combined to form that large square, dividing its total area by four isolates the formula for a single sum of cubes.

M | Monitoring — Example A

User instruction: Keep an eye on this living room camera feed, noting human movements coming through and any pets getting onto furniture or causing potential household disruptions.



Evidence 1 (2s): A person holding a spray bottle just entered the living room.

Evidence 2 (10s): Watch out — wet cleaning spray was left on the window glass, and the dog is now unattended in the room.

Evidence 3 (30s): The dog has climbed up onto the recliner armchair.

M | Monitoring — Example B

User instruction: Monitor this doorway safety test feed for items encroaching on the egress pathway or hazards blocking the partition door from closing smoothly.



Evidence 1 (4s): A purple ball was rolled into the doorway, directly obstructing the partition door path.

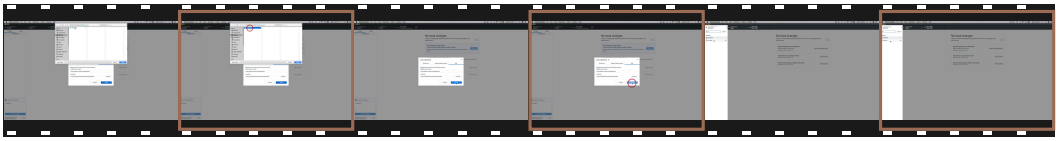
Evidence 2 (18s): A blue neck pillow was thrown into the doorway near the box, adding to pathway hazards.

Evidence 3 (25s): The green tinsel on the floor is getting caught directly beneath the closing partition door.

Figure 14: **Annotated samples (4/7).** Implicit Intents: Learning (L, second sample) and Monitoring (M, two samples), each with three evidence points.

T | Tool / GUI operation — Example A

User instruction: Walk me through cloning a repository in GitHub Desktop and managing my local folders, and steer me through switching between active projects.



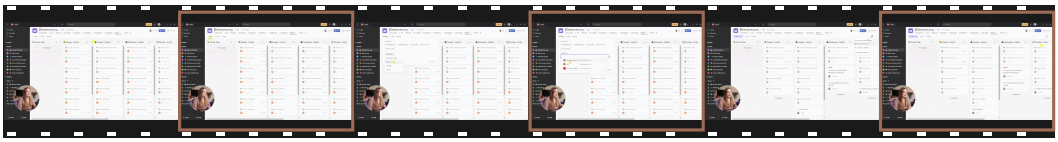
Evidence 1 (9s): Double-check that 'TestProject' is the intended target folder before confirming your selection in the file dialog.

Evidence 2 (19s): The cloning process is complete, and 'githubTest' has been automatically set as your active repository in the top bar.

Evidence 3 (39s): You've switched back to LiveALife as your active repository, so your original project workspace is restored and ready.

T | Tool / GUI operation — Example B

User instruction: Walk me through creating a personalized workload view in Asana, and guide me if I misstep setting up board filters or tab options.



Evidence 1 (21s): Select 'Just my tasks' from the quick filter menu to show only items assigned to you.

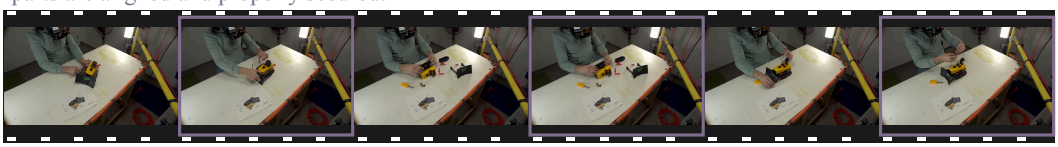
Evidence 2 (24s): Just double-check that 'Just my tasks' is selected for Natalia Kalinska under Assigned to before moving on.

Evidence 3 (30s): Your board view is now filtered to show just your assigned tasks, and the layout has been saved.

RV | Revision — Example A

User instruction: Please provide progress updates as I disassemble and rebuild the toy bulldozer.

User update (182s): Stop giving progress updates. Instead, check the quality of my assembly to ensure parts are aligned and properly secured.



Evidence 1 (36s): The front shovel blade has been completely detached from the bulldozer chassis.

Evidence 2 (259s): Before driving the screws in, double-check that the side track is sitting completely flush against the chassis frame.

Evidence 3 (329s): Looks like everything is back together. Give it a quick push across the desk to make sure the tracks roll smoothly and nothing is catching.

Figure 15: **Annotated samples (5/7).** Tool (T, two samples) and revision of Dynamic Intents (RV, first sample). Timestamped updates replace the earlier intent in RV.

RV | Revision — Example B

User instruction: Please provide next-step guidance as I make this lemon garlic aioli.

User update (79s): Stop with the next steps. Now, verify the quality of my aioli mixture.



Evidence 1 (6s): Now that you have the mayo and lemon juice, grab your garlic and a small mixing bowl to start building the base.

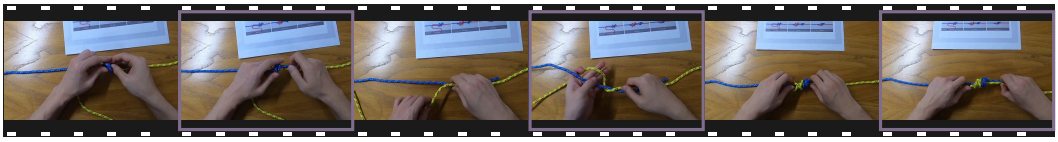
Evidence 2 (78s): Next, measure out about a teaspoon of the lemon juice and pour it into the mayonnaise.

Evidence 3 (152s): That looks perfectly smooth and thoroughly combined.

CP | Composition — Example A

User instruction: Let me know if I make a mistake while tying this knot.

User update (10s): Also, notify me whenever I complete a key step or finish the knot.



Evidence 1 (24s): You have completed the first knot.

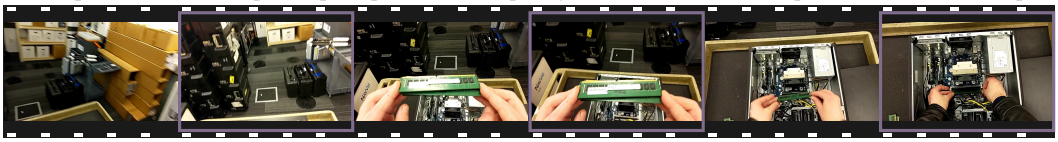
Evidence 2 (36s): [...] wrap the yellow cord twice around both strands, moving back towards the blue knot, before threading the end through.

Evidence 3 (51s): You've finished tying the double fisherman's bend.

CP | Composition — Example B

User instruction: Could you guide me on the next steps while I replace the RAM in my desktop computer tower?

User update (99s): Also, please prompt me to do a quality check whenever I complete an installation step.



Evidence 1 (76s): You've finished placing the desktop tower flat. Next, remove the side panel.

Evidence 2 (136s): You've finished inspecting the RAM stick. Next, align its notch with the ridge inside the motherboard slot.

Evidence 3 (160s): You've finished placing the RAM stick into the slot channels. Next, press straight down firmly on both ends.

Figure 16: **Annotated samples (6/7).** Dynamic Intents: revision (RV, second sample) and composition (CP, two samples). Updates replace or extend the active intent, respectively.

IT | Interruption — Example A

User instruction: I'm assembling this Nintendo Switch setup for the first time. Let me know if I attach a controller wrong or insert something incorrectly.

User update (31s): Are the Joy-Cons attached to the correct sides of the console?

User update (76s): Did I put the Joy-Cons on the correct sides of the grip?



Immediate answer 1 (31s): Yes, the blue Joy-Con is on the left and the red one is on the right.

Evidence 2 (52s): You attached the wrist strap upside down; align the plus and minus signs properly.

Evidence 3 (96s): You're inserting the microSD card into the game card slot; put it in the slot under the kickstand instead.

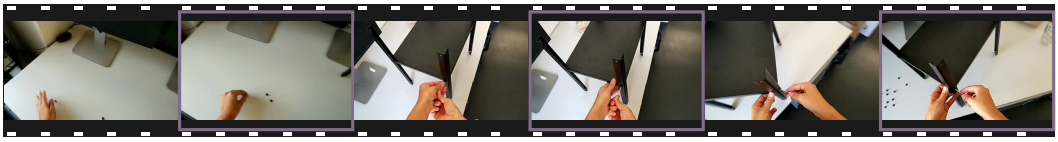
IT | Interruption — Example B

User instruction: I'm putting together this desk riser stand. Let me know if I attach any frame pieces upside down or miss a screw.

User update (48s): How many screws are left on the table?

User update (112s): What tool am I using to tighten the screws?

User update (174s): How many legs have I attached so far?



Immediate answer 1 (48s): There are 3 screws left on the table.

Evidence 2 (134s): You're attaching that leg bracket backwards. Turn it around so the lip faces inward.

Evidence 3 (188s): You're attaching that leg piece upside down. Flip it so the mounting holes align correctly.

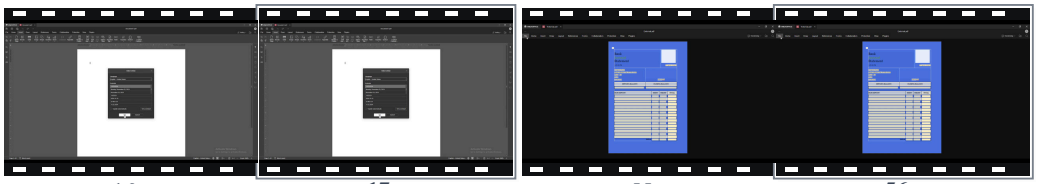
Figure 17: **Annotated samples (7/7).** Dynamic Intents: interruption (IT, two samples). An immediate answer is followed by two proactive evidence points, preserving the ongoing task.

E.2 Additional Qualitative Case Studies

As shown in Figures 18–32, these examples further illustrate the qualitative differences among methods in sustained proactive interaction. Compared with the baselines, StreamMind more accurately determines when to provide assistance by jointly considering task progress, interface state, and user goals, and generates concrete guidance that is consistent with the current interaction context. In contrast, other methods are more prone to responding too early or too late, missing critical evidence points, or merely describing the current observation without providing useful next-step guidance. Some methods also generate responses that are inconsistent with the actual interface state, rely on outdated information, or repeat previously provided content.

Case 01. Completing a document workflow

Instruction: Walk me through this ONLYOFFICE document to insert the current date, caption the chart, and turn on the autorecover setting.

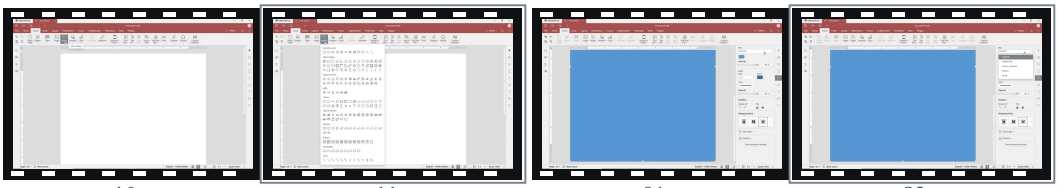


16 s | **17 s** | 55 s | **56 s**

JoyAI	✗ 17s: The chart is inserted into the document, and the user begins editing it.	✗ 56s: The user clicks the Autorecover button to open the settings dialog.
MMDuet2	✗ 15s: The cursor clicks the 'OK' button in the 'Date & Time' window, which closes, leaving the document with the inserted date visible.	✗ No response within ±2 s
ViSpeak	✗ 17s: The video shows a screenshot of a document in the ONLYOFFICE document editor. The cursor clicks on the "Insert" option in the toolbar, …	✗ 55s: … Go to the "File" tab, click on "Options," select "Save," and check the "Auto-recover" option.
StreamMind (ours)	✓ 18s: The date is inserted. Next, select the chart and add a caption.	✓ 58s: In the Advanced settings panel, check the box for 'Autorecover' to enable the feature.

Case 02. Drawing a rectangle and applying a gradient

Instruction: Walk me through adding a rectangle shape to this document, setting it behind the text, and applying a gradient fill.



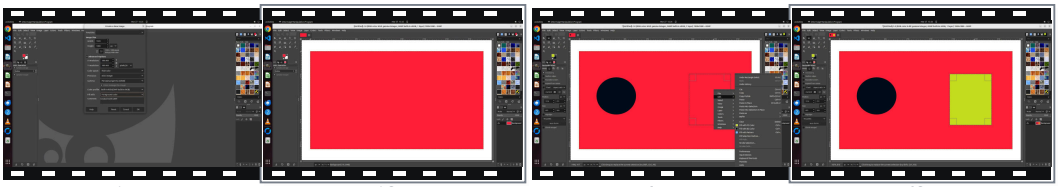
10 s | **11 s** | 81 s | **82 s**

JoyAI	✗ 11s: They select the Rectangle shape from the gallery, and a rectangle appears on the page.	✗ 82s: The gradient settings menu opens, and the user is fine-tuning the gradient.
MMDuet2	✓ 12s: … with the user hovering over the 'Rectangles' section.	✓ 84s: … the user is about to click on the 'Gradient fill' option within the 'Fill' dropdown menu.
ViSpeak	✓ 13s: … The person then selects a rectangle shape and clicks on the "Draw" button. The person then drags the rectangle to the center of the …	✓ 83s: The person clicks on the "Fill" option in the toolbar, selects "Gradient," and then selects the "Linear" option.
StreamMind (ours)	✓ 11s: Click the rectangle shape in the Shapes menu and draw it on the page.	✓ 82s: Select a gradient fill option from the dropdown menu in the Format Shape pane.

Figure 18: **Additional case studies: 01 and 02.** Completing a document workflow (top) and drawing a rectangle and applying a gradient (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 0 . Drawing and saving shapes in GIMP

Instruction: Walk me through drawing and coloring some basic shapes in GIMP, and make sure I save the project correctly when I'm done.

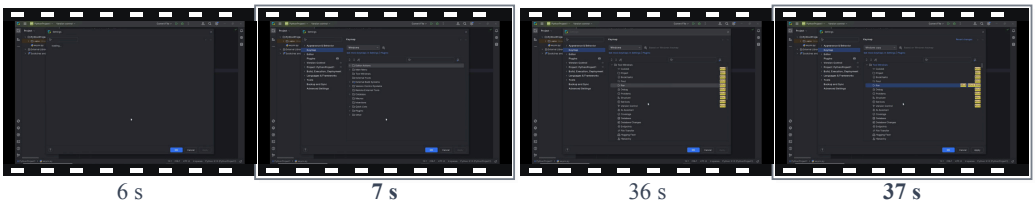


17 s | **18 s** | 67 s | **68 s**

JoyAI	✗ 20s: Great job! Now, let's add some color. Click the Paintbrush tool again and choose a different color from the color picker.	✗ 68s: Now, click the Paintbrush tool again to draw your shape. You can adjust the brush size in the tool options if needed.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 19s: The video shows a computer screen with a red background. The video is shot from a first-person perspective, with the cursor …	✗ 68s: The video shows a person drawing and coloring basic shapes in GIMP. The person first draws a circle and fills it with black, then draws a …
StreamMind (ours)	✓ 18s: Your canvas is set up. Now you can select the shape tool from the left toolbar to start drawing.	✓ 69s: Both shapes are drawn and colored. Now you can save the project by going to File and selecting Save.

Case 0 . Configuring shortcuts and text wrapping in PyCharm

Instruction: Walk me through setting up this IDE, specifically adding Enter as a second stroke for the Run shortcut and enabling wrap on typing.



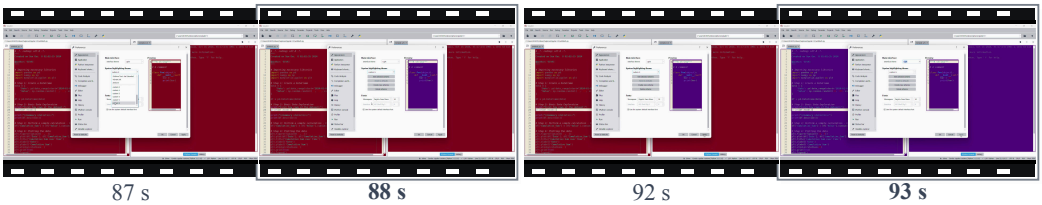
6 s | **7 s** | 36 s | **37 s**

JoyAI	✗ 6s: The user opens the settings menu, and the interface switches to a settings page with various options.	✗ 37s: The user clicks the save button again to confirm the settings.
MMDuet2	✓ 7s: … prepares to search for the 'Run' option to set it as the second stroke for Enter.	✗ 37s: The 'Keyboard Shortcut' window shows the first stroke set to 'Alt+4' and the second stroke field with 'Enter' entered, and the user is about to …
ViSpeak	✗ 5s: The video shows a computer screen with a text editor open. The text editor is being used to write code.	✗ 37s: … The cursor clicks on the "Tools" option in the menu bar, then clicks on "Add Shortcut" in the drop-down menu. Then the cursor …
StreamMind (ours)	✓ 6s: Open the Keymap settings and search for 'Run' to locate the shortcut.	✓ 36s: The Run shortcut is updated. Next, open the Editor settings to enable wrap on typing.

Figure 19: **Additional case studies: 03 and 04.** Drawing and saving shapes in GIMP (top) and configuring shortcuts and text wrapping in PyCharm (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 0 . Applying a custom color scheme in Spyder

Instruction: Walk me through setting up a custom purple color scheme in Spyder, and guide me through the preferences menu to apply it.

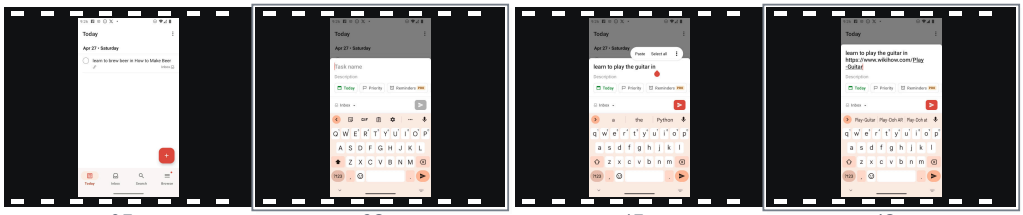


87 s | **88 s** | 92 s | **93 s**

JoyAI	88s: The user selects a custom purple theme from the list, and the preview updates to show the new color scheme.	93s: The user scrolls through the theme list, examining the different color options available.
MMDuet2	No response within ±2 s	No response within ±2 s
ViSpeak	89s: The video shows a computer screen with a purple background and a white text box in the center. The text box contains the words "Hello …	93s: The video shows a computer screen with a purple background and a white text box in the center. The text box contains the words "Hello …
StreamMind (ours)	89s: Click the 'Apply' button at the bottom of the Preferences window to save your new purple color scheme.	94s: Your custom purple color scheme is now applied across the editor and console. You can close the Preferences window to return to your workspace.

Case 0 . Saving a browser link as a to-do task

Instruction: Walk me through copying this guitar guide link from my browser and saving it as a new task in my to-do app while switching screens.



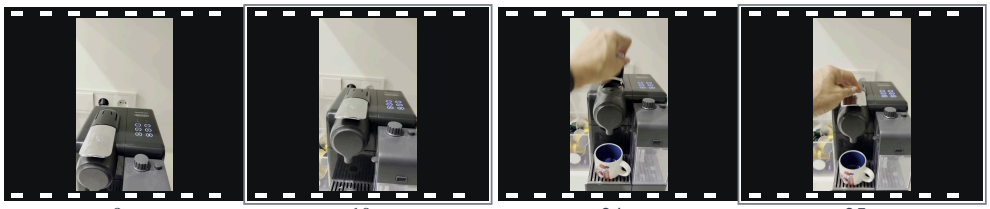
37 s | **38 s** | 47 s | **48 s**

JoyAI	38s: The user taps the back arrow again to return to the to-do app.	48s: The user taps the back arrow again to return to the to-do app.
MMDuet2	39s: … there is a red plus button at the bottom right for adding a new task.	48s: The 'Task name' field now contains the text 'learn to play the guitar in https://www.wikihow.com/Play - …
ViSpeak	38s: Copy the link of the guitar guide, paste it in the task name, and save it as a new task in the to-do app.	48s: The video shows a person typing "https://www.wikihow.com/Play-Guitar" into the search bar of a browser and then clicking on the "Add" …
StreamMind (ours)	38s: Tap the red plus button at the bottom right to create a new task, then paste the copied link.	48s: Tap the send button to save the task with the pasted link.

Figure 20: **Additional case studies: 05 and 06.** Applying a custom color scheme in Spyder (top) and saving a browser link as a to-do task (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 0 . Making capsule coffee

Instruction: Walk me step by step through making coffee with this capsule espresso machine, and jump in if I make any mistakes.

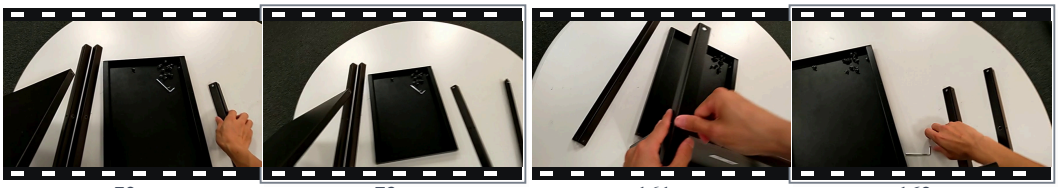


9 s | **10 s** | 24 s | **25 s**

JoyAI	✗ 10s: Now pull out the capsule holder, and you can see the capsule slot inside.	✗ 25s: You can see the capsule is being pushed into the machine.
MMDuet2	✗ No response within ±2 s	✓ 23s: Press the button on the machine, which dispenses coffee into the mug, and after dispensing, lift the coffee capsule out of the port.
ViSpeak	✗ 8s: The first step is to press the button on the machine.	✗ 26s: The person puts a cup under the coffee machine, presses the button, and then puts a coffee capsule into the machine.
StreamMind (ours)	✓ 9s: The machine is now powered on. Next, place your cup under the spout and insert a coffee capsule.	✓ 25s: The capsule is locked in. Press the coffee button on the control panel to start brewing.

Case 0 . Assembling a metal shelf

Instruction: Walk me through assembling this flat-pack metal shelf unit, and check that I am aligning the frame legs and securing every screw joint properly.



72 s | **73 s** | 161 s | **162 s**

JoyAI	✓ 73s: Great, now attach the top to the frame. Make sure it's aligned properly.	✗ 162s: Great, now attach the first shelf to the frame. Make sure it's aligned properly.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✓ 73s: The person is assembling a flat-pack metal shelf unit by placing the frame legs on the shelf and securing them with screws.	✗ No response within ±2 s
StreamMind (ours)	✓ 73s: Align the side leg with the mounting holes on the base frame and insert the screws to secure it.	✓ 162s: Tighten the screw on the side leg using the Allen key to secure the joint.

Figure 21: **Additional case studies: 07 and 08.** Making capsule coffee (top) and assembling a metal shelf (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 0 . Assembling a stool

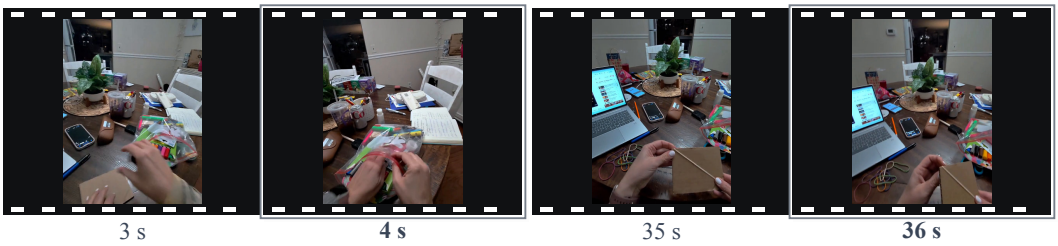
Instruction: Walk me through assembling this metal and plastic stool, making sure I align the frame and seat correctly and set the hardware properly.



JoyAI	✗ 7s: Okay, the stool is already assembled and placed on the desk. Now let's get started on the disassembly.	✗ 26s: Now, unscrew the screws on the right side of the seat.
MMDuet2	✓ 10s: Hands hold white metal legs and attach them to the white plastic seat with holes, using a tool to secure the connection by pushing it into place …	✓ 26s: Hands use a tool to tighten a screw on the white plastic seat with holes, then pick up a white plastic piece and position it over the seat, …
ViSpeak	✓ 6s: … The person first places the seat on the frame, then screws the screws on the seat.	✓ 27s: The person is assembling a metal and plastic stool by attaching the legs to the seat and tightening the screws.
StreamMind (ours)	✓ 8s: Align the leg frame with the seat base and insert the screws into the mounting holes.	✓ 26s: Tighten the screw securely with the Allen key, then move to the next mounting hole.

Case 1 . Making a diagonal rubber-band pattern

Instruction: Walk me through making a symmetric rubber band pattern on this cardboard square, and help me place the bands in the right sequence.



JoyAI	✗ No response within ±2 s	✓ 36s: Good job! Now, take another rubber band and place it on the cardboard. Try to keep it symmetric with the first two.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ No response within ±2 s	✗ 38s: The person is making a symmetric rubber band pattern on a cardboard square.
StreamMind (ours)	✓ 3s: Start by stretching a rubber band across the cardboard square to form your first diagonal line.	✓ 34s: Now stretch the second rubber band across the cardboard to form the second diagonal line.

Figure 22: **Additional case studies: 09 and 10.** Assembling a stool (top) and making a diagonal rubber-band pattern (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 11. Browning butter without scorching

Instruction: Walk me through making brown butter in this skillet, and warn me if the heat is wrong or if it starts to burn.

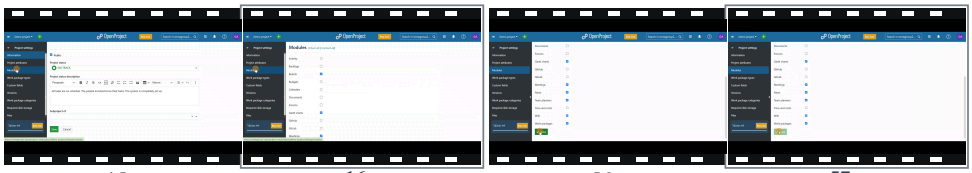


39 s **40 s** 79 s **80 s**

JoyAI	✗ 40s: The butter is turning a deeper golden color, and the aroma is getting stronger.	✗ 80s: The warning text remains the same, with a red overlay.
MMDuet2	✗ 39s: Chunks of butter are melting in a stainless-steel bowl on a stovetop burner, and a person in a dark chef's coat stands nearby observing the …	✗ 79s: Chunks of butter are melting in a stainless-steel bowl on a stovetop burner, and the melted butter has a frothy surface with numerous small …
ViSpeak	✓ 39s: … The person is adding butter to the pan and stirring it with a spatula.	✓ 81s: The heat is too high, and the butter is burning.
StreamMind (ours)	✓ 38s: Keep the heat at medium and stir the butter continuously with a spatula to prevent the milk solids from scorching.	✓ 78s: Immediately remove the pan from the heat to stop cooking and prevent the butter from burning.

Case 12. Enabling and saving OpenProject modules

Instruction: Walk me through the settings in this OpenProject workspace to add the calendars and activity modules to my project sidebar.



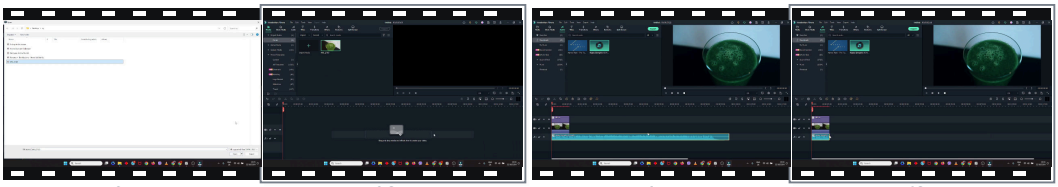
15 s **16 s** 56 s **57 s**

JoyAI	✗ 16s: The settings page refreshes, showing the updated modules list with "Calendars" and "Activity" now enabled.	✗ 57s: The user clicks the "Back" button to return to the previous settings page.
MMDuet2	✓ 17s: … the user checks the boxes next to 'Calendars' and 'Activity' under the 'Check all' option, and the 'Boards' module is already checked; the …	✓ 58s: A green 'Successful update.' message appears, indicating that the modules have been successfully added, and the cursor hovers over …
ViSpeak	✗ 15s: The video shows a person clicking on the settings button on the left side of the screen, then clicking on the "Add" button on the right side …	✗ 58s: The video shows a person clicking on the settings button on the left side of the screen, then clicking on the "Project" option, and finally …
StreamMind (ours)	✓ 16s: Select the checkboxes for 'Activity' and 'Calendars' to add them to your project sidebar.	✓ 58s: The 'Successful update' banner confirms the modules are saved. You can now close the settings menu to see the updated sidebar.

Figure 23: **Additional case studies: 11 and 12.** Browning butter without scorching (top) and enabling and saving OpenProject modules (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ± 2 s.

Case 1 . Editing and exporting a video in Filmora

Instruction: Walk me through editing this green smoothie video in Filmora, exporting it, and setting up the sharing permissions in Google Drive.

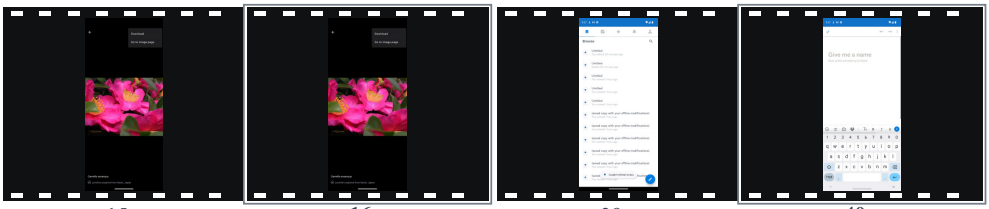


27 s | **28 s** | 67 s | **68 s**

JoyAI	✗ 28s: The user returns to the Filmora editing interface, where the video is still in the timeline.	✗ 68s: The user continues editing the video, adjusting the timeline and previewing the footage.
MMDuet2	✓ 29s: … text instructing to drag and drop media and effects to create a video.	✓ 66s: … the user is about to click the export button to save the project.
ViSpeak	✗ 27s: The video shows a computer screen with a Google Drive folder open. The user clicks on the "Share" button and selects "Invite People" …	✓ 66s: The video shows a person editing a green smoothie video in Filmora. The person then exports the video and sets up the sharing permissions in …
StreamMind (ours)	✓ 30s: Drag the green smoothie clip from the media library onto the timeline to begin editing.	✓ 70s: Click the 'Export' button in the export settings window to save your video.

Case 14. Downloading and inserting a flower photograph

Instruction: Walk me through downloading a flower image from Wikipedia and inserting it into my note document while navigating between apps.



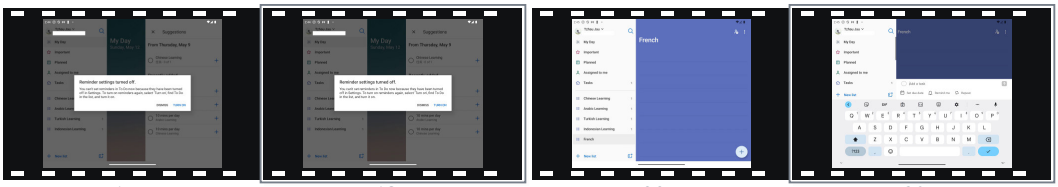
15 s | **16 s** | 39 s | **40 s**

JoyAI	✗ 15s: The user taps the download button, and the download progress bar appears.	✗ 40s: The user taps the new note button, and the note editing page opens with the keyboard.
MMDuet2	✓ 16s: Tap the three-dot menu icon on the Camellia sasanqua image, select 'Download', and prepare to save the image to the device's storage.	✗ No response within ±2 s
ViSpeak	✗ 17s: The video shows a person using a mobile phone to search for a flower image on Wikipedia and then inserting it into a note document.	✗ 39s: The video shows a person using a mobile phone to search for a flower image on Wikipedia, then downloading the image and …
StreamMind (ours)	✓ 15s: Tap 'Download' to save the flower image to your device.	✓ 40s: Tap the image icon in the toolbar to insert the downloaded flower image into the document.

Figure 24: **Additional case studies: 13 and 14.** Editing and exporting a video in Filmora (top) and downloading and inserting a flower photograph (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ± 2 s.

Case 1 . Setting a French-learning reminder

Instruction: Guide me through switching to the French course in Duolingo and creating a daily study task in Microsoft To Do on this tablet.

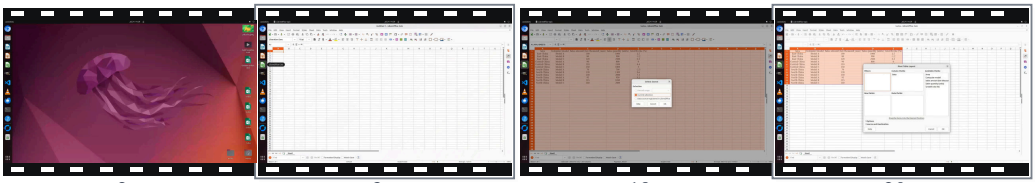


17 s | **18 s** | 29 s | **30 s**

JoyAI	✗ 17s: Now switch to the Microsoft To Do app. Open it from the home screen.	✗ 30s: You're back in Duolingo. Complete the lesson by selecting the correct image for "le chat."
MMDuet2	✗ No response within ±2 s	✗ 29s: After typing 'French' in the 'Enter list title' field, the 'CREATE LIST' button is clicked, resulting in a new 'French' list appearing on the right …
ViSpeak	✗ 19s: To switch to the French course in Duolingo, tap the French flag on the screen. Then tap the "Add" button on the screen. Then tap the "Add task" …	✗ 29s: The video shows a person using a tablet to switch to the French course in Duolingo and create a daily study task in Microsoft To Do on the tablet.
StreamMind (ours)	✓ 18s: Tap 'TURN ON' to enable reminders for your daily study task.	✓ 30s: Type '10 min per day' into the task field.

Case 1 . Loading sales data and configuring a pivot table

Instruction: Walk me through summarizing this sales data by computer model in a LibreOffice Calc pivot table, and help me generate a smooth line chart.



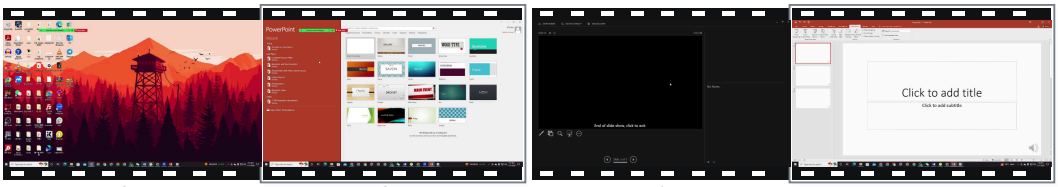
2 s | **3 s** | 19 s | **20 s**

JoyAI	✗ 3s: The user opens LibreOffice Calc, a spreadsheet program. The interface shows a blank sheet with grid lines.	✗ 20s: The user clicks "OK" to create the pivot table, and a blank pivot table appears on the new sheet.
MMDuet2	✗ No response within ±2 s	✗ 19s: A 'Select Source' window pops up asking for a named range or data source registration, with options for 'Named range' and 'Current …
ViSpeak	✗ 5s: The video shows a computer screen with a spreadsheet open. The spreadsheet contains data on sales of different computer models. The …	✗ 20s: The video shows a computer screen with a spreadsheet open. The spreadsheet contains data on sales, including the number of units sold, …
StreamMind (ours)	✓ 5s: Select the sales data file from the file browser and click Open to load it into the spreadsheet.	✓ 20s: Drag the 'Computer Model' field into the Row Fields box and the 'Sales' field into the Data Fields box.

Figure 25: **Additional case studies: 15 and 16.** Setting a French-learning reminder (top) and loading sales data and configuring a pivot table (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 1 . Recording and checking a PowerPoint voiceover

Instruction: Walk me through recording audio narration for this PowerPoint presentation, and help me verify that my voiceover actually saves to the slides.



2 s | **3 s** | 54 s | **55 s**

JoyAI	✗ 3s: Now the view switches to the PowerPoint interface, showing the slide thumbnails on the left and the main slide on the right. The user is …	✗ 55s: The view switches back to the PowerPoint interface, showing the second slide with a blank title slide. The user is back in PowerPoint.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 2s: The video shows a computer desktop with a forest background.	✓ 57s: The voiceover is saved to the slides.
StreamMind (ours)	✓ 4s: Open your presentation file from the Recent list to start setting up the audio recording.	✓ 56s: The recording is saved. Play the presentation to verify that your voiceover is attached to the slides.

Case 8. Running hostname and process-list commands

Instruction: Walk me through these basic Ubuntu terminal commands to check my network, find the hostname, and search for active processes.



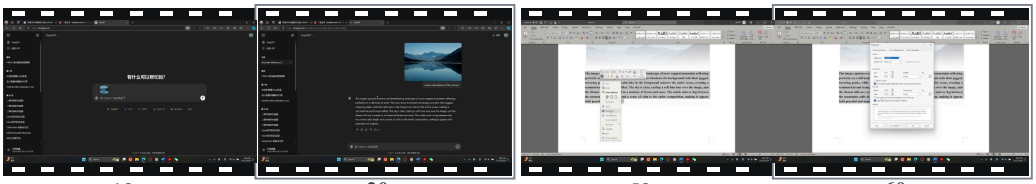
23 s | **24 s** | 47 s | **48 s**

JoyAI	✗ 25s: The user switches to the 'top' command, which provides a real-time view of system resource usage.	✗ 48s: The user types 'cd', which changes the current directory to the user's home directory.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 25s: The video shows a computer screen with a terminal window open. The terminal window displays the command "ping -c 4 google.com". …	✗ 48s: … The user then types in the command "ps" to find the hostname.
StreamMind (ours)	✓ 25s: Type `hostname` and press Enter to check your system's name.	✓ 50s: Type `ps aux` and press Enter to list all active processes.

Figure 26: **Additional case studies: 17 and 18.** Recording and checking a PowerPoint voiceover (top) and running hostname and process-list commands (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 1 . Copying a description and formatting Word text

Instruction: Walk me through getting an image description from ChatGPT, pasting it into my Word document, and finding the right font and indentation settings.

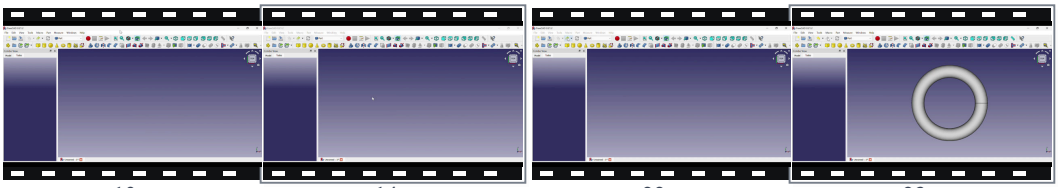


19 s | **20 s** | 59 s | **60 s**

JoyAI	✗ 20s: The user copies the selected text, and the screen switches to a Word document, where the text is pasted.	✗ 60s: The user adjusts the paragraph settings, and the interface displays the paragraph settings dialog box.
MMDuet2	✗ 21s: ChatGPT has generated a description of the image, which includes details about snow-capped mountains reflecting in a calm …	✗ 60s: … a Paragraph dialog box pops up with settings such as Alignment set to Justified, Line and Page Breaks, and Asian Typography tabs visible, …
ViSpeak	✗ 18s: The video shows a computer screen with a chat window open. The chat window has a text box and a button to upload an image. The …	✗ 61s: The video shows a computer screen with a document open in Microsoft Word. The document contains text and an image. The …
StreamMind (ours)	✓ 20s: Copy the generated description text from ChatGPT and paste it into your Word document.	✓ 60s: In the Paragraph dialog box, set the Indentation to 0.5 inches and click OK.

Case 2 . Restoring and sketching a shape in FreeCAD

Instruction: Walk me through this FreeCAD interface to practice using the undo and redo controls, and drawing 2D shapes on the canvas and in the Sketcher.



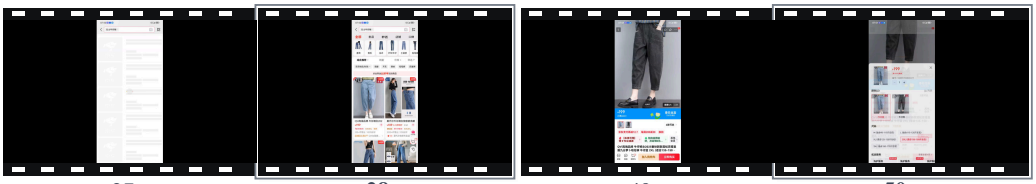
13 s | **14 s** | 22 s | **23 s**

JoyAI	✗ No response within ±2 s	✗ 23s: The user clicks the redo button, and the ring reappears, demonstrating the redo function.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 16s: The video shows a computer screen with a 3D modeling software interface. The interface has a toolbar at the top with various tools and …	✓ 25s: … The cursor clicks on the "Rectangle" button on the toolbar, and a rectangle appears on the screen. The cursor clicks on the "Rectangle" …
StreamMind (ours)	✓ 14s: Click the Redo button on the toolbar to restore the object you just deleted.	✓ 23s: The ring is back on the canvas. Next, you can draw a new 2D shape on the canvas or open the Sketcher to create a profile.

Figure 27: **Additional case studies: 19 and 20.** Copying a description and formatting Word text (top) and restoring and sketching a shape in FreeCAD (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 2 . Finding and selecting jeans in a shopping app

Instruction: Walk me through finding a pair of jeans on this shopping app, picking my size and color, and adding them to my cart.

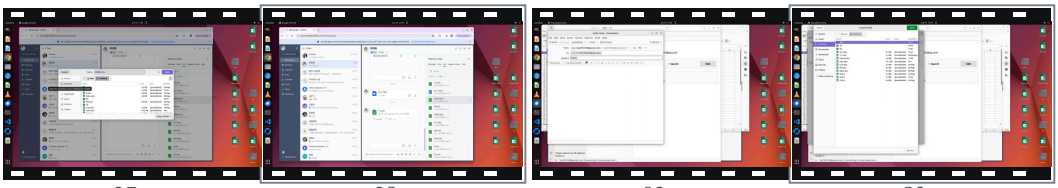


27 s **28 s** 49 s **50 s**

JoyAI	✗ 28s: The user taps the "Add to Cart" button, and the interface switches to the shopping cart page.	✗ 50s: The user taps the "Place Order" button, and the order confirmation page loads.
MMDuet2	✗ 29s: The search results page displays various styles of women's jeans including straight-leg, loose, and knitted, with options for sorting by …	✓ 51s: … options to select colors (including blue and black) and sizes (M, L, XL, 2XL) with weight ranges for each size; there is also a section …
ViSpeak	✗ 28s: The video shows a person using a shopping app to search for jeans. The person types in the search bar and selects the type of jeans they want …	✗ 49s: The person is browsing the shopping app and selecting a pair of jeans.
StreamMind (ours)	✓ 28s: Now that the search results are loaded, scroll through the list to find a pair of jeans you like.	✓ 50s: Select your preferred size and color from the options, then tap the red 'Add to Cart' button at the bottom.

Case 22. Opening and attaching a spreadsheet

Instruction: Walk me through downloading this dataset, building a pivot table in LibreOffice, and make sure I save and attach the updated file in Thunderbird.



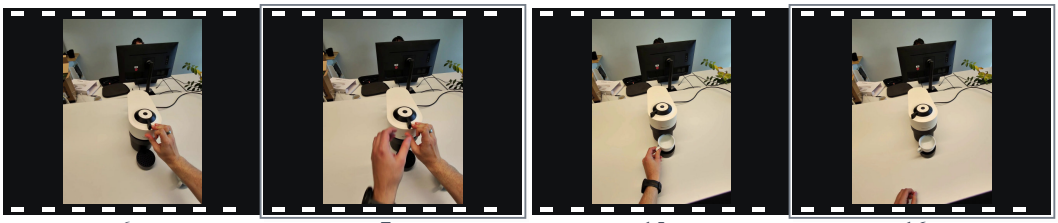
27 s **28 s** 82 s **83 s**

JoyAI	✗ 28s: Now, drag the 'Category' field to the 'Rows' area and the 'Sales' field to the 'Values' area.	✗ 83s: Right-click on the empty space and select 'New Document'.
MMDuet2	✗ No response within ±2 s	✓ 84s: … the user is about to click the 'Open' button to select a file to attach to the email.
ViSpeak	✓ 29s: … The file is then opened in LibreOffice. The user clicks on the "Insert" tab and selects "Pivot Table" to create a pivot table. The user …	✗ 83s: The video shows a person downloading a file from the internet, opening it in LibreOffice, and saving it to Thunderbird.
StreamMind (ours)	✓ 30s: Now that the dataset is downloaded, open LibreOffice Calc to start building your pivot table.	✓ 83s: Select the saved spreadsheet file from the file picker to attach it to your email draft.

Figure 28: **Additional case studies: 21 and 22.** Finding and selecting jeans in a shopping app (top) and opening and attaching a spreadsheet (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ± 2 s.

Case 2 . Preparing and brewing capsule coffee

Instruction: Walk me through making coffee with this Nespresso machine step by step, and warn me if I set up anything incorrectly.

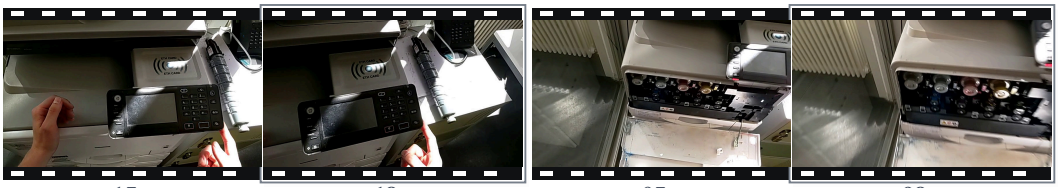


6 s | **7 s** | 15 s | **16 s**

JoyAI	✗ 7s: Now insert the capsule into the holder, making sure it's seated properly.	✗ 15s: The machine is brewing now. You can watch the coffee flow into the cup.
MMDuet2	✗ 6s: A hand presses down on the top part of the machine, which has a white and black color scheme, and the machine appears to be in the …	✗ 15s: A hand wearing a black wristband holds a white cup and positions it under the spout of the machine, which is actively brewing coffee with its …
ViSpeak	✓ 6s: … The person is putting the cup under the coffee machine.	✗ 15s: The person is holding a cup and placing it on the Nespresso machine.
StreamMind (ours)	✓ 7s: Place your cup under the spout and press the brew button to start.	✓ 16s: Press the brew button to start dispensing coffee.

Case 2 . Opening a printer and releasing a cartridge

Instruction: Walk me through replacing this office printer toner cartridge step by step, and warn me if I'm handling anything incorrectly or missing a release lock.



17 s | **18 s** | 97 s | **98 s**

JoyAI	✗ 19s: You're pulling out the old toner cartridge. Make sure to hold it by the edges and avoid touching the chip area.	✗ 98s: The paper tray is closed. You're all set.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ No response within ±2 s	✗ 98s: The person is handling the printer correctly.
StreamMind (ours)	✓ 18s: Press the button on the control panel to open the front access door.	✓ 98s: The front door is open. Now, locate and pull the release lever to remove the old toner cartridge.

Figure 29: **Additional case studies: 23 and 24.** Preparing and brewing capsule coffee (top) and opening a printer and releasing a cartridge (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 2 . Positioning arms and sustaining mannequin compressions

Instruction: Guide me through this adult CPR practice on the mannequin, keeping me on track with each step and correcting my posture and form.



25 s | **26 s** | 32 s | **33 s**

JoyAI	✓ 25s: Good, keep your hands in position and continue compressions.	✗ 33s: Keep going, maintain the head tilt and chin lift, and continue rescue breathing.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 25s: The man is demonstrating the correct way to perform CPR on a mannequin. He is kneeling on the floor and placing his hands on the …	✗ 33s: The man is performing CPR on a dummy. He is kneeling on the floor and pressing down on the dummy's chest with both hands.
StreamMind (ours)	✓ 26s: Keep your elbows locked and shoulders directly over your hands, then begin compressions at a steady pace of 100 to 120 beats per minute.	✓ 33s: Keep your elbows locked and shoulders directly over your hands, then begin compressions at a steady pace of 100 to 120 beats per minute.

Case 2 . Making successive creases for a paper butterfly

Instruction: Walk me through folding this paper butterfly step by step, and tell me exactly how to align and crease the paper for each fold.



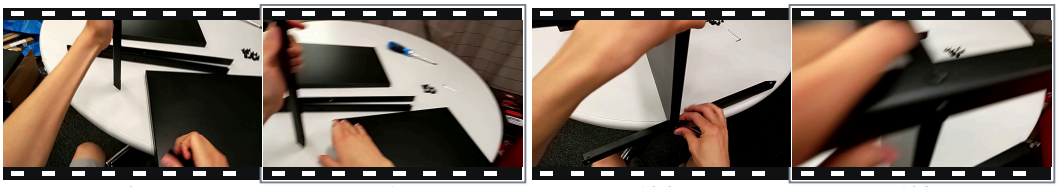
19 s | **20 s** | 23 s | **24 s**

JoyAI	✗ 20s: The hand folds the left corner down to the center crease, creating a new triangular shape.	✗ 24s: Then the bottom corner is folded up to meet the top, making the paper narrower.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ No response within ±2 s	✗ 26s: Fold the paper in half lengthwise, then unfold it. Fold the top corners of the paper to the center crease, then unfold it. Fold the bottom corners …
StreamMind (ours)	✓ 19s: Now that you've opened the paper back up, fold the top edge down to meet the bottom edge to create a horizontal crease.	✓ 23s: Now that you've opened the paper back up, fold the top edge down to meet the bottom edge to create a horizontal crease.

Figure 30: **Additional case studies: 25 and 26.** Positioning arms and sustaining mannequin compressions (top) and making successive creases for a paper butterfly (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 2 . Aligning and fastening a shelf frame

Instruction: Guide me step by step through assembling this black metal shelf unit and catch any errors I make along the way.

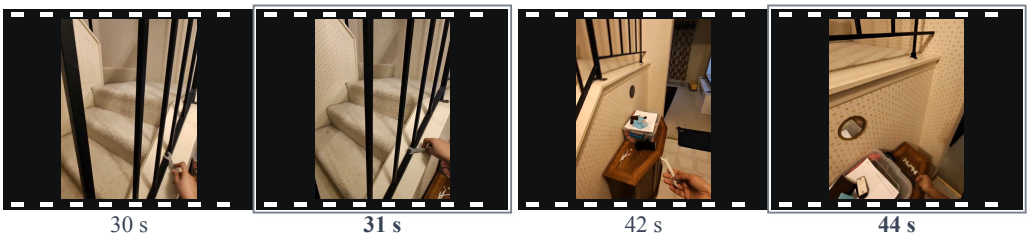


50 s | **51 s** | 196 s | **198 s**

JoyAI	51s: Now, take a look at the second shelf again. You should see it's already attached to the frame. Next, we'll add the third shelf.	198s: Good, now move on to the next screw. Repeat the same steps—tighten it until it's snug.
MMDuet2	No response within ±2 s	No response within ±2 s
ViSpeak	No response within ±2 s	199s: The person is assembling a black metal shelf unit.
StreamMind (ours)	51s: Align the side frame bar with the pre-drilled holes on the shelf panel and insert the screws.	198s: Align the side frame bar with the pre-drilled holes on the shelf panel and insert the screws.

Case 2 . Tightening baluster base bolts

Instruction: Walk me through tightening the base bolts on these staircase balusters, and let me know how to check that the handrail is completely secure.



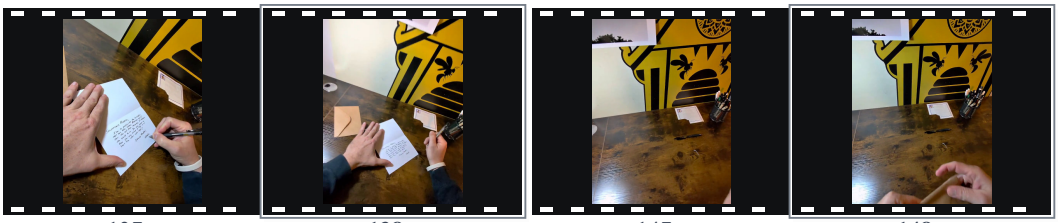
30 s | **31 s** | 42 s | **44 s**

JoyAI	31s: The hand is still gripping the base of the handrail, and you can see the bolts are being tightened further. The handrail is becoming more secure.	44s: The camera angle shifts again, and you can see the person's hand working on the bolts. The handrail is now much more stable.
MMDuet2	31s: A hand uses a wrench to tighten the base bolts on the balusters, ensuring they are snug but not overly tight, and then checks the handrail …	No response within ±2 s
ViSpeak	No response within ±2 s	No response within ±2 s
StreamMind (ours)	31s: Tighten the base bolt on this baluster with your wrench, then move to the next one.	43s: Tighten the base bolt on this baluster with your wrench, then move to the next one.

Figure 31: **Additional case studies: 27 and 28.** Aligning and fastening a shelf frame (top) and tightening baluster base bolts (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

Case 2 . Folding and addressing a thank-you card

Instruction: Walk me through writing this formal thank you note, and guide me on structuring the message and addressing the envelope.



JoyAI	✗ 138s: Make sure to write clearly and legibly.	✗ 148s: Now, write the date in the top right corner again.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✗ 136s: The envelope is addressed to the recipient's name and address.	✗ 148s: The envelope is addressed to the recipient's name and address, and the note is placed inside the envelope.
StreamMind (ours)	✓ 139s: Now that the message is complete, fold the card and place it inside the envelope.	✓ 150s: Write the recipient's name and address on the front of the envelope.

Case 3 . Folding a paper airplane along its center crease

Instruction: Walk me through folding this paper airplane step by step, and let me know if my creases and alignments look right as I go.



JoyAI	✗ No response within ±2 s	✓ 15s: … Next, fold the top two corners down to meet that center line.
MMDuet2	✗ No response within ±2 s	✗ No response within ±2 s
ViSpeak	✓ 4s: The person is folding the paper in half lengthwise.	✗ 15s: The person is folding the paper in half and then unfolding it.
StreamMind (ours)	✓ 4s: Fold the paper in half lengthwise, aligning the top and bottom edges, then press down firmly to create a sharp center crease.	✓ 14s: Now that the center crease is set, fold the top corners down so their edges meet right along that center line.

Figure 32: **Additional case studies: 29 and 30.** Folding and addressing a thank-you card (top) and folding a paper airplane along its center crease (bottom). Each case compares model responses at two annotated evidence points, with responses matched within ±2 s.

F Data Construction, Evaluation, and Judge Prompts

We report selected data-construction instructions and the prompt templates used by the interaction policy, Harness memory, and content judge. The Harness passes causally available state to the policy before generation; it does not select or rewrite the policy's response. Braced fields below are runtime variables: `t0` and `t1` delimit the observed interval, `focus` is the active request and its reference cues, `summaries` contains preceding memory blocks, and `previous` is the preceding normalized state. Images are supplied separately in chronological order. Prompt wording and output schemas follow the implementation. Rules, response types, and output fields are separated visually for readability.

F.1 Data-Construction Prompts

The following selected system instructions are used by the multi-stage video-annotation implementation for scenario-based proactive assistance. The stages describe the video, construct a scenario, identify evidence points, generate reference responses, and check annotation quality. Sample-specific inputs and output schemas are supplied by the corresponding stage-specific user messages; the boxes reproduce the system instructions with the original wording and clearer paragraph and list structure. Internal fields such as `demand_type` and `gt_only` retain their code names. The source pipeline supports a broader task-family catalogue, including long-form companionship; the final benchmark taxonomy remains that of Table 15.

Timestamped video frames and optional source facts are used to construct the intermediate visual description.

Data construction: video evidence description

You are a precise video annotator. You watch frames of a person completing a task and write a dense, timestamped caption of what actually happens: per segment, what the person is doing, how it is going (smooth / hesitating / struggling / just finished a step), and which objects/tools are visible.

Describe only what is visible; do not invent. This caption is an intermediate artifact that later steps rely on, so be concrete and grounded.

The caption and target source task family condition scenario construction. The instruction separates model-visible context from annotation-only fields.

Data construction: scenario construction

You design ONE scenario for a proactive-assistant benchmark, conditioned on a TARGET TASK FAMILY (the user's relationship to the activity: G execution / L learning / T tool-operation / M monitoring / C long-form companion). A scenario is the context that decides whether help is needed at a given moment: whether-to-help = f(visible frame, scenario). You are given a text caption of a task video (NOT the raw frames).

PHASE ONE: build ONLY the HIGH-NEED role — the person who, in THIS task, would want the broadest useful help (this is the upper bound other roles will later subtract from). Keep the visible role PLAIN and NATURAL: one line saying who the person is and one line saying they want to do the overall task well.

Do NOT state when help is wanted, what should be explained, what should be skipped, or which events deserve an alert. Positive and negative answer filters are both benchmark-label leakage. Do not invent complex personas, deadlines, or stacked constraints.

CRITICAL leak firewall: you output TWO groups of fields.

Data construction: scenario construction (continued)

(A) SUBJECT-VISIBLE (shown to the model under test): role_context (identity + help_preference), a GENERAL goal, ambient setting. These must NOT name any specific defect/mistake/need that should be discovered from the video.

(B) gt_only (NEVER shown to the subject): family_detail, success_criteria, blind_spots, constraints — the specific substance that later seeds and aligns the help points.

Write every natural-language value in English. If this video does not genuinely fit the target family, return {"supported": false}.

The scenario, caption, frame indices, and sampled frames condition candidate evidence points. The source code calls these help points; each must be traceable to the user request.

Data construction: evidence-point annotation

You annotate the gold moments where a proactive assistant should speak, for ONE scenario. You work in a SKELETON-DRIVEN way so that help points BLANKET the whole task, not just a few highlights.

LATENT INTENT (read it, it is not spelled out): the user's query states only a GENERAL task and merely *implies* what they care about — it does NOT tell you what help to give.

It is YOUR job to infer, from the query's latent need PLUS what actually happens in the video, the moments where a genuinely helpful proactive assistant would speak up and what kind of help it is. Do NOT assume any pre-declared intent (no 'they asked to be reminded/summarised'); derive each point's demand_type from the visible evidence and the inferred need.

STEP 1 — SKELETON: first read the video as a sequence of task units. Depending on the task family the unit is: an operation STEP (execution / tool use), a key EVENT (learning), a monitored TARGET / to-check item (monitoring), or a content STAGE (long companion). List these units in time order across the ENTIRE video, early to late.

STEP 2 — SKELETON POINTS (the task backbone, gated by the TRACEABILITY TEST): walk every skeleton unit, and at each one ask what the user SAID that makes this moment worth speaking at. Quote that span from the DEFAULT instruction into 'serves_request'. Cannot quote one? The point does not exist — drop it.

E.g. execution: next_step_guidance where the user asked to be walked through, progress_summary where they asked to know they got it right. A step's mere existence in the video justifies nothing. Walking the whole timeline is about not MISSING a moment the request reaches, never about filling slots.

STEP 3 — OVERLAY POINTS (only when a visible cue warrants it): on top of the skeleton, add the overlay demand points (given per family: error_correction, quality_verification, risk_alert, hidden_info_reveal, cross_reference, ...) ONLY where the frame actually shows the triggering cue (a real mistake, an inspectable result, a hazard, missed on-screen info, a link to an earlier moment). Overlay points must be grounded — do NOT invent or pad them.

BALANCE: the user instruction is the arbiter for BOTH kinds — every point carries a quote in 'serves_request' or it is dropped. Missing a moment the request reaches is the failure; so is a point no quote supports.

Do NOT tune toward a point count: the right number is whatever the request genuinely reaches, many for a broad request, few for a narrow one over the same video. Judge yourself as a human helper watching alongside this user would be judged.

Data construction: evidence-point annotation (continued)

TYPE FIDELITY: do not default every useful moment to next_step_guidance. When a grounded moment is more specifically a recap, correction, verification, risk, hidden-information, memory, knowledge, or event-notification need, use that exact allowed demand_type. Never invent a point or relabel it merely to balance counts; visible evidence and the user's request still decide.

The scenario, candidate point, requested assistance type, and local caption context condition the reference response. This step generates annotation content, rather than scoring a model output.

Data construction: reference-response generation

You write the CONCRETE help a good assistant naturally says at one specific moment. The gold 'reference' is what a model's response is scored against, so it MUST be answerable from ONLY what is knowable at this moment: the user's role, the goal, and the video UP TO this frame.

NEVER state a specific fact that can only be known by watching the FUTURE of the video (a value the user hasn't chosen yet, a result not yet on screen). If the correct help here is a normative action ('open the calendar and pick your departure date'), say that; do NOT invent the specific value ('October 2nd') unless the goal/role/current screen already fixes it.

TONE: sound like a real assistant helping in the moment, phrased for THIS task's situation (hands-on task -> second-person guidance 'you can now...'; a lesson -> explanatory narration; game watch -> commentary; monitoring -> a heads-up). Keep ONE consistent voice within the line.

PROACTIVE, NOT A REPLY (critical): the user did NOT ask a question at this moment. This is YOU, the assistant, deciding on your own to speak up right now because this moment warrants help. The reference is your unprompted, self-initiated remark — NOT the answer to any question.

The user's opening request (query) is only BACKGROUND telling you their general aim; do NOT phrase the reference as a response to it, do NOT echo or restate it, and never write as if replying ('to answer your question…', 'yes,…', 'what you're seeing is…'). Just say the helpful thing out of your own initiative.

HELP FORM IS FIXED BY THIS POINT'S demand_type (critical): the KIND of proactive help you give is determined by the point's demand_type (given below) — write EXACTLY that form of help:

- **next_step_guidance** -> proactively tell them the correct next action;
- **progress_summary** -> proactively give a short 'here's where things stand' recap;
- **knowledge_gap** -> proactively explain the key event / principle they need (the insight, not screen-reading);
- **goal_event_notify** -> proactively flag that the watched-for target/event just appeared;
- **error_correction** -> proactively point out the mistake AND give the fix;
- **risk_alert** -> proactively warn about the imminent hazard;
- **hidden_info_reveal** -> proactively surface the on-screen thing they haven't noticed;
- **quality_verification** -> proactively nudge them to check the result now inspectable;
- **callback_reuse / unfinished_recall / consistency_check / cross_reference** -> proactively bring up the relevant earlier moment (reuse it / remind them to finish it / flag the mismatch / link it).

Data construction: reference-response generation (continued)

ANTI-NARRATION (hard): except for progress_summary, the main content must be the demanded assistance, not a description of the visible action or state. For progress_summary, compress multiple completed steps or locate the user relative to the goal; merely restating one just-seen action/state is NOT a summary.

In particular, 'you have just/now done X' or 'X is ready' is invalid unless the same utterance adds the demand_type's genuinely useful next action, check, warning, correction, or goal-level context. Match whether the user is DOING vs WATCHING (a reviewer of footage is not the person on screen — serve their goal, don't tell them to perform on-screen actions).

NO ANSWER-KEY / NO AI VOICE (critical): you are a companion glancing over and casually saying something helpful in the moment — NOT answering an exam question and NOT lecturing. Do NOT write textbook 'X happens because Y' answer-key sentences, do NOT sound formal or explanatory.

GOOD: 'Heads up — that herd's really struggling on the loose sand, might be worth noting as a rough patch.'

BAD: 'They are migrating at night because it is much cooler, to avoid the extreme desert heat.'

SERVE THE USER'S NEED + BE WORTH SAYING (critical): the reference must advance the user's underlying aim (help them get it right / not miss the thing they care about / understand the key point / keep useful track) in the demand_type's form. Only say something a helpful assistant would actually bother to say.

Do NOT narrate trivial, low-value moments (e.g. 'logged that you're checking your phone', 'noted you looked around') — if the moment carries no real value for the user's need, the help should not exist. The remark should give the user something USEFUL: a next step, a catch, a meaningful milestone, an important detail they'd want — not a play-by-play of every movement.

NO STOCK OPENERS / NO TEMPLATE (critical): do NOT start with canned assistant phrases and do NOT reuse the same opener across moments.

BANNED openers/fillers: 'Here's a major turning point', 'Here's the next step', 'Heads up —' used every time, 'for your notes', 'for your analysis', 'worth noting', 'it's worth noting', 'Talk about a…', 'Make sure to note'. Never tell the user to 'note'/'record'/'log' something — that is THEIR job and naming it is the tell of an AI.

Just say the helpful thing itself, the way a friend watching alongside would blurt it out. Vary sentence shape point to point; two gold lines in the same sample must not share an opening formula.

Also do NOT merely restate what is on screen ('The male officer reveals he is an ally') — add the insight the user actually cares about (why it matters, what shifts), in one natural spoken breath.

The reviewer checks candidate annotations against the stated goal, role, and available evidence, and returns keep, fix-goal, or drop decisions.

Data construction: future-information leakage review

You audit each gold help point for CAUSAL VALIDITY in a proactive-video benchmark. The assistant may only use info knowable at the trigger moment: the user's role, the goal, and the video UP TO that frame.

Data construction: future-information leakage review (continued)

A point is a LEAK if its answer depends on information that only appears in the FUTURE of the video (a value the user has not chosen yet, a result not yet on screen) AND that info is not already fixed by the goal/role.

Use the demand_type as a prior, but judge each point by the causal test:

- next_step_guidance / error_correction / cross_reference: usually OK (they point at the current or past state). A next_step that names a specific unchooseable future value (e.g. 'pick Oct 2nd') is a LEAK unless the goal fixes that value.
- progress_summary / knowledge_gap / quality_verification / risk_alert: OK if grounded in what is already visible/known; LEAK if it states a not-yet-happened outcome.
- goal_event_notify: OK if the event has ALREADY appeared on screen by this frame.
- hidden_info_reveal: OK if the revealed thing already has a visible trace/cue by this frame (even if the user overlooked it — that is its value); LEAK only if there is NO trace yet and it is pure future knowledge.

For each point return a verdict: keep | fix_goal | drop.

- keep: causally valid as-is.
- fix_goal: the needed info is a legitimate TASK PARAMETER (a chosen date, target, constraint) that SHOULD have been stated in the goal; give 'goal_addition' = the short phrase to add to the goal so the answer becomes derivable.
- drop: the answer needs an arbitrary future value that does not belong in the goal; cannot be saved.

A separate annotation check evaluates the generated reference for grounding, usefulness, and natural proactive phrasing. This is distinct from the evaluation-time response judge below.

Data construction: reference-response quality check

You audit the GOLD assistant utterance of a proactive-video benchmark point. The utterance must be PROACTIVE — the assistant speaking up on its own initiative — NOT a reply to any user question.

Score it 0-4 on:

(a) correct & grounded in the stated evidence;

(b) DELIVERS THE demand_type's FORM OF PROACTIVE HELP and is WORTH SAYING — it must match the demand_type given (e.g. next_step_guidance -> gives the next action; knowledge_gap -> explains the key point; progress_summary -> recaps where things stand; goal_event_notify -> flags the event; error_correction -> names the mistake + fix; memory types -> bring up the relevant earlier moment) — and a helpful assistant would actually bother to say it;

(c) phrased as NATURAL, SELF-INITIATED in-the-moment help — NOT an exam-style answer-key sentence, NOT a reply to a question ('to answer that…', 'yes,…', 'what you're seeing is…'), NOT formal lecturing, NOT robotic.

4 = correct, delivers the demand_type's help, and naturally self-initiated.

<=2 if it narrates a TRIVIAL / low-value moment (e.g. 'logged that you're checking your phone', 'noted you looked around'), does NOT match the demand_type's help form, or is phrased as a reply to a query.

Data construction: reference-response quality check (continued)

0 = wrong/ungrounded, an answer-key / AI-sounding line, or an explicit answer to a user question.

Penalise 'X because Y' textbook phrasing, reply framing, formal tone, and play-by-play narration.

F.2 Online Interaction Policy

The SFT and RL checkpoints use the following system prompt in evaluation, including the Harness-enabled runs. The user input contains the active request and causally available frames. When the Harness is enabled, its selected state projection is prepended to the current user message as additional text, as described in Appendix D.4.

Online interaction: system prompt

You are a proactive real-time video assistant observing a continuous stream.

At every timestamp choose exactly one action using only frames visible up to that timestamp.

Output </silence> when evidence is irrelevant or insufficient.

Output </response> followed by a concise, grounded answer only when the evidence is sufficient.

Never use future information or make unsupported guesses.

F.3 HIERARCHICAL MEMORY

The Harness uses the ordered-event mid-term template to summarize observed chunks, followed by the long-term template to consolidate consecutive summaries. Both prompts preserve the distinction between observed evidence and request-derived reference conditions.

Mid-term memory: ordered evidence

You are the visual state tracker of a streaming video assistant. You see one chronological chunk covering {t0:.0f}s to {t1:.0f}s.

ACTIVE REQUEST AND REFERENCE CUES:

{focus}

Record compact, timestamped visual evidence for the next decision epoch. Track objects, settings, counts, locations and state CHANGES relevant to the request. Never infer intent, success, errors or completion unless the images establish them. Do not quote an assistant response and do not choose silence/response. Use at most 220 words in this exact shape:

VISUAL MEMORY FOCUS: <what visual evidence is being tracked; this is a request-derived focus, not an observation>

OBJECT STATES: <object -> state [time]; ... or none>

COUNTS/QUANTITIES: <thing: value [time]; ... or none>

ON-SCREEN TEXT: <verbatim text [time]; ... or none>

EVENTS:

EVENT [start-end]: actor=<visible actor or unknown>; action=<observable change>; object=<object or none>; from=<prior visible state or unknown>; to=<new visible state>; relation=<before/after relation or none>; confidence=<high|medium>. Write events in strict temporal order. If two actions overlap, keep separate lines and state the relation; never reverse cause and effect. Write '- none' only when there is no event.

CHUNK END [{t1:.0f}s]: <last visibly established state>

If the request is procedural, explanatory, or monitoring, append every line below; write 'unknown' when unsupported:

PLAN CURRENT: <visible current stage/action>

PLAN COMPLETED: <visibly completed stage(s), separated by | >

PLAN NEXT: <only if explicitly supported by the request/reference plan>

CUE CONTINUE SEEN: <evidence that an existing action merely continued>

CUE RESPONSE SEEN: <new completion, requested event, explainable transition, or error>

CUE DEVIATION SEEN: <visible mismatch/omission/unsafe state, or none>

RECOVERY: <reference correction only when a deviation is visible, or unknown>

Consistency rule: every newly visible PLAN COMPLETED item must also be represented in CUE RESPONSE SEEN, and every visible DEVIATION in CUE DEVIATION SEEN. Use CUE CONTINUE SEEN only when no new completion, requested transition, or deviation was established.

Long-term memory: evidence consolidation

You are the long-term memory of a streaming video system. Below are {n} consecutive mid-term summaries of one continuous video, covering {t0:.0f}s to {t1:.0f}s.

{summaries}

The downstream agent is still following this standing request:

<active_request>{focus}</active_request>

Consolidate the summaries into ONE evidence block.

Keep request-relevant state changes, exact counts, on-screen text, names, errors, completions, and criterion changes with absolute timestamps.

Merge repeated static states and delete narration that does not affect a future decision.

Do not turn a model reply, request condition, or inference into a visual fact.

Preserve strict event order plus actor/action/object/from/to/relation fields; do not merge two distinct actions into one or reverse a before/after relationship.

End with the latest visibly established state.

Use at most 12 bullets and 300 words.

Output only the consolidated evidence block.

Preserve any PLAN CURRENT, PLAN COMPLETED, PLAN NEXT, WATCH, DEVIATION, and RECOVERY fields verbatim; do not merge them into narrative prose.

F.4 Response-Quality Judge

Claude Sonnet 4.6 (Anthropic, 2026) applies the following reference-based rubric to temporally matched responses. The judge receives the reference and candidate reply as text, without video frames. The raw integer score $r \in \{0, 1, 2, 3, 4\}$ is normalized as $s_{g,p} = r/4$ for the content metrics in Appendix C.5. Temporal matching, missed responses, and extra interventions are handled by the evaluation protocol rather than by this content-only rubric. External benchmarks retain their own official scoring protocols.

Content judge: system instruction

You are grading a proactive video assistant's reply at one moment in a live stream. You are given the assistant's reply and a gold reference reply written for that same moment. The reference is a canonical example, NOT a literal checklist: a paraphrase that conveys the same substance is fully correct. You cannot see the video, so judge only against the reference.

Score an integer overall from 0 to 4 for how useful the reply is at this moment:

4 = correct, specific, actionable, and well fitted to the situation;

2 = partially correct or useful but vague;

0 = incorrect, off-topic, or non-actionable.

Do not penalize concise paraphrases.

Return ONLY {"overall": <0-4>}.