Plan, Watch, Recover: A Benchmark and Architectures for Proactive Procedural Assistance

Anonymous authors
Paper under double-blind review

Abstract

Proactive multimodal assistants must decide *when* guidance is needed and *what* to say, while tracking progress through extended procedures and helping users recover when they depart from the expected plan. Progress is limited by benchmarks that score isolated clips, cover narrow domains, or omit realistic deviations. We introduce **EgoProactive**, a smart-glasses dataset of procedures across four domains in which users commit step-level deviations and recover from them, paired with reference guidance. We combine it with five re-annotated egocentric and instructional datasets to form **Pro^{2}Bench**, a cross-domain benchmark that scores intervention timing, procedure completion, and response quality. Zero-shot, no frontier or open-weight model completes more than 26% of a procedure's interventions before its first error. We propose **PWR** (Plan, Watch, Recover), which keeps an explicit plan state in the model's context and revises it after every intervention, including recovery steps after a deviation. Across three open-weight backbones, training on Pro^{2}Bench without a plan barely changes performance, while adding the dynamic plan raises average procedure completion from 20% to 27% and improves turn-level decisions and response quality for every backbone. The gains hold on held-out datasets and on deviations never seen in training.

1 Introduction

Procedural tasks such as servicing a bike or cooking a soufflé are ordered sequences of steps, and errors in ordering, technique, or timing have real consequences. A proactive assistant on smart glasses must (i) track procedural state from streaming egocentric video, (ii) decide *when* to intervene, and (iii) decide *what* to say, including how to recover when the user goes off plan. It must do this without the user repeatedly asking *"What is the next step?"* These capabilities are coupled, since a wrong estimate of state produces mistimed interventions.

Recent work addresses these pieces separately, including online state and mistake detection (Seminara et al., 2024; Flaborea et al., 2024), cognitively grounded timing (Li et al., 2025a; Arakawa et al., 2024), and streaming video LLMs at interactive latency (Qian et al., 2025; Wang et al., 2025). ProAssist (Zhang et al., 2025) is the first end-to-end system for proactive assistance from streaming egocentric video. Its central finding is that dialogue quality is limited by procedural reasoning rather than perception. We ask how procedural reasoning should be evaluated and what a model needs in order to do it.

Evaluation lacks deviations and procedure-level scoring. Real users skip, reorder, and substitute steps, which we call out-of-plan (OOP) deviations. Existing corpora either assume execution without deviations or provide recovery guidance only for cooking and for errors within a single step (Bhattacharyya et al., 2026). Existing protocols also score each decision in isolation, which cannot tell whether a model carries a user through a procedure.

Knowing the procedure is not the same as knowing where the user is. A model given the task or even the full list of steps must still infer from video which step is in progress and whether the user has left the expected path. Neither prompting with a plan nor training

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

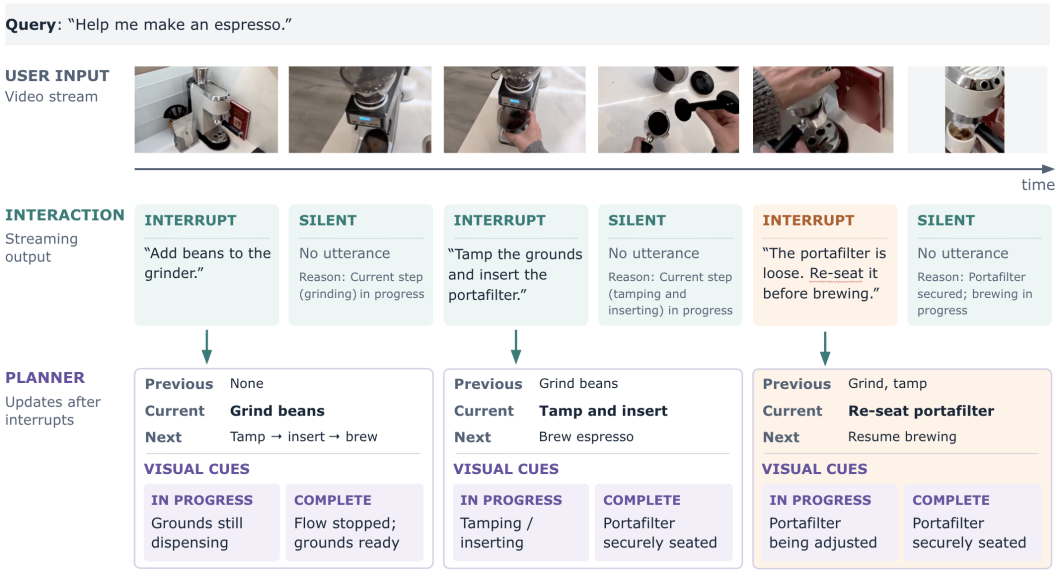


Figure 1: **Plan, Watch, Recover (PWR) on an out-of-plan deviation.** Given a request such as *"Help me make an espresso,"* (a) the **planner** writes the steps and the visual cues that mark each step's completion. (b) The **interaction model** watches the stream against the current step and emits SILENT or INTERRUPT. (c) When the user deviates, the planner replaces the remaining steps with **recovery steps**. More examples are in App. J.2.

on in-domain data closes this gap alone. The model needs an explicit account of its position that stays current as the user acts and deviates.

Our contributions.

1. **EgoProactive, a multi-domain dataset of step-level deviations and their recoveries.** We record 700 sessions (31.4 h) on Ray-Ban Meta glasses from about 30 participants at home, across cooking, crafts, DIY, and tutorials. In 247 sessions the participant omits, reorders, or substitutes a step and then recovers within one continuous take, so one video grounds both detection and recovery. To our knowledge, it is the only public dataset that pairs step-level deviations with recovery guidance across multiple domains (§2.2).[1]
2. **Pro^{2}Bench, a unified proactive procedural benchmark.** We re-annotate Ego4D, Ego-Exo4D, EPIC-KITCHENS, HoloAssist, and HowTo100M with plan states, intervention decisions, and reference guidance under one schema, and combine them with EgoProactive. Pro^{2}Bench provides 26,874 evaluation and 249,584 training decision points across 14 domains and 3 viewpoints. It scores systems per procedure and per turn (§2).
3. **PWR, dynamic procedural context for proactive assistance.** PWR places a plan state in the context of the interaction model and revises it after every intervention, so the model checks each observation against the current step and its visual cues. A planner writes the revisions, including recovery steps after a deviation, and runs asynchronously so the interaction model keeps deciding in real time (§3).
4. **Findings.** Zero-shot, no model completes more than 26% of a procedure's interventions before its first error. Across three open-weight backbones, training on Pro^{2}Bench without a plan barely changes performance, while adding the plan improves all nine backbone and metric comparisons, by +.070 Procedure Macro, +.068 Turn Macro-F1, and +.176 PQS on average. The gains extend to held-out sources and on OOP deviations never seen in training.

[1] `https://huggingface.co/datasets/facebook/wearable-ai`

2 Pro^{2}Bench: Proactive Procedural Benchmark

A proactive assistant must decide when to intervene and what to say throughout a multistep procedure. When users omit, reorder, or substitute steps (Peddi et al., 2024; Sener et al., 2022), it must also detect the deviation and guide them to a correct end state. Evaluating an assistant for deployment therefore requires intervention decisions with reference guidance, and deviation detection with recovery guidance. No existing dataset provides both across diverse domains and capture settings. Large egocentric and instructional corpora such as Ego4D (Grauman et al., 2022), EPIC-KITCHENS (Damen et al., 2022), Ego-Exo4D (Grauman et al., 2024), and HowTo100M (Miech et al., 2019) contain neither. ProAssist (Zhang et al., 2025) covers only procedures executed without deviations, CaptainCook4D (Peddi et al., 2024) lacks guidance, and HoloAssist (Wang et al., 2023) does not mark boundaries between steps.
QICD (Bhattacharyya et al., 2025) and Ego-MC-Bench (Bhattacharyya et al., 2026) provide recovery guidance, but only for cooking and for deviations within a single step, such as a wrong quantity. We close this gap with two contributions: we re-annotate five public corpora with intervention decisions and reference guidance under one schema, and we collect EgoProactive, a new smart-glasses dataset with step-level deviations and their recoveries. Together they form Pro^{2}Bench, with 26,874 evaluation and 249,584 training decision points from 29,367 videos across 14 domains and three viewpoints (Table 1).

2.1 Task and Annotation Schema

Task. A procedure is one user carrying out one task, beginning with a query q in which the user states the task. At time t the assistant receives the observation $o_t = (v_{1:t}, \mathcal{H}_{<t}, q)$, where $v_{1:t}$ is the video stream sampled at 2 fps and $\mathcal{H}_{<t}$ is the dialogue history. It emits an intervention decision $d_t \in \{\textsc{Silent}, \textsc{Interrupt}\}$ and, when $d_t = \textsc{Interrupt}$, an utterance u_t. An interrupt is warranted when the user completes a step, or departs from the procedure at the level of steps, which we call an out-of-plan (OOP) deviation.

Decision points. Each procedure is annotated as a time-ordered sequence of decision points. A decision point is a time t with a reference decision $d_t^\star$, the 8 s clip ending at t, and the reference dialogue history $\mathcal{H}_{<t}^\star$. Interrupt points lie at step boundaries and OOP onsets and carry a reference utterance $u_t^\star$. Silent points are placed between them. A silent point has no interrupt point within its clip or the following $\tau = 2$ s, which excludes frames just before a boundary where the correct decision is ambiguous (§2.3).

Plan states. Each decision point also carries a reference plan state $P_t^\star = \big((s_1, \sigma_1), \ldots, (s_N, \sigma_N); \mathbf{c}\big)$. Here s_i describes the i-th of N steps, $\sigma_i \in \{\texttt{completed}, \texttt{current}, \texttt{next}\}$ is its status at t, and the cue set $\mathbf{c}$ describes visual evidence that the `current` step is in progress or has completed. After an OOP deviation the step list includes the annotated recovery steps. Reference plan states supervise training (§3.3) and are never given to evaluated systems.

2.2 EgoProactive

EgoProactive records step-level deviations and their recoveries in continuous egocentric video from consumer smart glasses. It contains 700 sessions totaling 31.4 hours from about 30 participants, who wore Ray-Ban Meta glasses in their own homes for 2 to 5 minutes per session. Each session follows a script with a goal and 5 to 14 ordered steps (9.3 on average), drawn from 700 distinct scripts in cooking, crafts, DIY, and tutorials. In the 247 OOP sessions, the participant also commits a scripted deviation and fixes it within one continuous take that ends in the correct state, so one video grounds both detection and recovery. The deviations consist of omissions, reorderings, and substitutions. The remaining 453 sessions follow the script without deviation and test step-completion guidance on a device and in settings absent from training. Against Ego-MC-Bench (40 sessions, 7 participants), EgoProactive has about 17 times as many sessions and targets deviations across steps rather than within one, so the two are complementary.
EgoProactive carries the OOP evaluation, hence annotators reviewed every video in full: corrected steps and boundaries, marked each deviation onset by hand, linked to corresponding recovery steps, wrote the reference utterances for OOP decision points,

Source	View	Source Annot.	Doms	Vids	Train DPs	Eval DPs	OOP DPs
Ego4D[‡]	Ego (head-cam)	Narrations	14	24,474	214,511	–	0
EgoExo4D (EE4D)	Ego + Exo	Keysteps	8	462	13,726	2,734	0
EPIC-KITCHENS (EK)	Ego (head-cam)	Actions	1	468	8,954	6,399	0
HowTo100M (HowTo)	3rd-person	Transcripts	12	3,026	12,393	4,654	0
HoloAssist (HA)[†]	Ego (AR)	Dialogue	4	237	–	3,188	0
EgoProactive (ours)[†]	Ego (glasses)	Scripts + OOP	4	700	–	9,899	3,810
Pro^{2}Bench (ours)	All 3 views	Unified schema	**14**	**29,367**	**249,584**	**26,874**	**3,810**

Table 1: **Pro^{2}Bench composition. View** is the viewpoint of the procedure. **Source Annot.** is the annotation each source provided before conversion to the Pro^{2}Bench schema. **DPs** are decision points, counted after splitting, filtering, deduplication, and leakage checks. **OOP DPs** come from the 247 EgoProactive sessions with a deviation and its recovery. The last row counts unique domains and videos. [†]Evaluation only. [‡]Training only.

and edited the pipeline's candidates (§2.3) for step completions (protocol, demographics, and ethics in App. J).

2.3 Unified Annotation

Ego4D (Grauman et al., 2022), EPIC-KITCHENS (Damen et al., 2022), Ego-Exo4D (Grauman et al., 2024), HoloAssist (Wang et al., 2023), and HowTo100M (Miech et al., 2019) supply viewpoint, device, and domain diversity (Table 1). Their annotations, from action segments to transcripts, describe what the user does, not when to intervene or what to say. One pipeline maps these five corpora and EgoProactive onto the schema of §2.1. Source-specific converters recover the goal and steps where available, and a VLM proposes the remaining fields as candidates. This is followed by an iterative refinement stage, where a VLM judge checks each candidate's steps, boundaries, decisions, and guidance against the frames. For each failure, the judge returns localized frame evidence and identifies the affected fields. A separate correction pass receives the original candidate, the field-level critique, and the same frames; it revises only the flagged step text, temporal boundary, decision label, or guidance utterance while preserving accepted fields. The revised candidate is judged again, and any item that remains inconsistent is quarantined rather than released.

2.4 Benchmark Metrics

Procedure-level metrics. A system runs along all decision points of a procedure in time order until its first failure, a decision point where the predicted decision $\hat{d}_t$ differs from $d_t^\star$. Let $\mathcal{P}$ be the set of evaluated procedures. For procedure i, let L_i be the number of reference interrupt points and k_i the number of those interrupts correctly reached before the first failure. Correct silent decisions extend the valid sequence but do not increment k_i. We report Procedure Macro $= 1/|\mathcal{P}| \sum_{i \in \mathcal{P}} k_i/L_i$. Each decision point uses the reference dialogue history $\mathcal{H}^\star_{<t}$. Errors compound because evaluation stops at the first incorrect decision: no later interrupt contributes to k_i, even if it is predicted correctly.

Turn-level metrics. Turn-level metrics use all 26,874 frozen decision points, without class rebalancing or metric-specific subsampling (App. C.5). We report interrupt F1 (IF1), silent F1 (SF1), and Turn Macro-F1 $= (\text{IF1} + \text{SF1})/2$. An invalid or unparseable output is scored as an incorrect decision.

Proactive Quality Score (PQS). For each correct interrupt, GPT-5.6 scores the utterance $\hat{u}_t$ against $u_t^\star$ from 1 to 5 on relevance, specificity, actionability, and conciseness. With $\bar{r}_t$ their mean, the normalized quality is $g_t = (\bar{r}_t - 1)/4 \in [0, 1]$, and exact matches receive $g_t = 1$. Let $\mathcal{T}$ be the set of evaluation decision points and s_t equal 1 for a correct silence, g_t for a correct interrupt, and 0 otherwise. Then PQS $= \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} s_t$, which rewards speaking at the right time and saying something useful. Utterance quality alone is reported in App. B.6.

3 Method

A proactive assistant must know where the user is in the procedure. Knowing the steps is not enough. The assistant must also infer from video which step is in progress, whether it has just finished, and whether the user has left the expected path. We make this state explicit and keep it current. We introduce **PWR** (Plan, Watch, Recover), which places a plan state in the context of the interaction model and revises it as the procedure unfolds. A planner writes the initial plan and revises it after each intervention (*Plan*). The interaction model checks each observation against the current step and decides whether to speak (*Watch*). After an OOP deviation, the planner replaces the remaining steps with recovery steps (*Recover*). Both models are post-trained on plan-conditioned trajectories from Pro^{2}Bench (§3.3). We adopt the task of §2.1. In practice o_t contains the 8 s clip ending at t rather than the full stream $v_{1:t}$. Let $\mathcal{V}$ denote model vocabulary, so $u_t \in \mathcal{V}^*$ when $d_t = \textsc{Interrupt}$ and $u_t = \emptyset$ otherwise. We write P_t for a predicted plan state with the structure of §2.1, serialized as a structured text block in the input of the interaction model (Figure 1).

3.1 From Implicit to Dynamic Procedural Context

We consider three ways of giving procedural knowledge to an interaction model p_θ, differing only in input context. The backbone, video clips (App. A.1), and training data are held fixed.

Implicit and Static context. The two baselines condition on

$$p_\theta\big(d_t,\, u_t \mid o_t\big) \quad (1) \qquad\qquad p_\theta\big(d_t,\, u_t \mid o_t,\, P_0\big) \quad (2)$$

Implicit context (Eq. 1) is the formulation of Zhang et al. (2025). The procedure exists only in q and $\mathcal{H}_{<t}$, and one model must recover the steps, track the current one, and decide whether to speak. Static context (Eq. 2) adds a plan P_0 generated once from q. It states what the task involves. The model must still infer its position from the video and from steps it already confirmed in $\mathcal{H}_{<t}$. P_0 cannot change when the user departs from it.

Dynamic context (ours). PWR conditions on the most recent plan state (Eq. 3a) and revises it as the procedure unfolds (Eq. 3b),

$$p_\theta\big(d_t,\, u_t \mid o_t,\, P_{t-1}\big) \quad (3a) \qquad\qquad P_t = \mathcal{U}\big(P_{t-1},\, d_t,\, u_t,\, o_t\big) \quad (3b)$$

where $\mathcal{U}$ is an update rule realized by a learned planner (§3.2). Static context is the special case $\mathcal{U}(P_{t-1}, \cdot) = P_{t-1}$, so dynamic context subsumes it. With a current P_{t-1} the interaction model no longer has to localize the user in the task. It checks the observation against the `current` step and its cues. A step completion is a match with the completion cues, and an OOP deviation is a mismatch with the current step.

Static and dynamic contexts differ most when the plan itself must change. On procedures that follow the script, the dialogue history already records which steps the assistant has confirmed, and static and dynamic context perform similarly (App. C). After an OOP deviation, only a dynamic plan can replace the remaining steps with recovery steps. With reference plans, which remove planner errors from the comparison, dynamic context improves OOP procedure completion over static context for every backbone (§4). After an OOP deviation, only a dynamic plan can replace the remaining steps with recovery steps. Dynamic context thus handles step completion and recovery with one mechanism.

3.2 Updating the Plan

The plan changes only when a step completes or the user deviates. These are the moments at which the interaction model interrupts, so we revise the plan only at interrupts. We model $\mathcal{U}$ with a planner p_ϕ. Letting p denote the joint distribution over the outputs of the system at time t,

$$\boxed{p\big(d_t, u_t, P_t \mid o_t, P_{t-1}\big) = \underbrace{p_\theta\big(d_t, u_t \mid o_t, P_{t-1}\big)}_{\text{Interaction model}} \cdot \underbrace{p_\phi\big(P_t \mid d_t, u_t, o_t, P_{t-1}\big)}_{\text{Planner}}} \quad (4)$$

where the first factor is Eq. 3a and the second is a learned form of $\mathcal{U}$ with

$$p_\phi\big(P_t \mid d_t, u_t, o_t, P_{t-1}\big) = \begin{cases} \delta\big(P_t - P_{t-1}\big) & d_t = \textsc{Silent} \\ \text{learned (§3.3)} & d_t = \textsc{Interrupt} \end{cases} \quad (5)$$

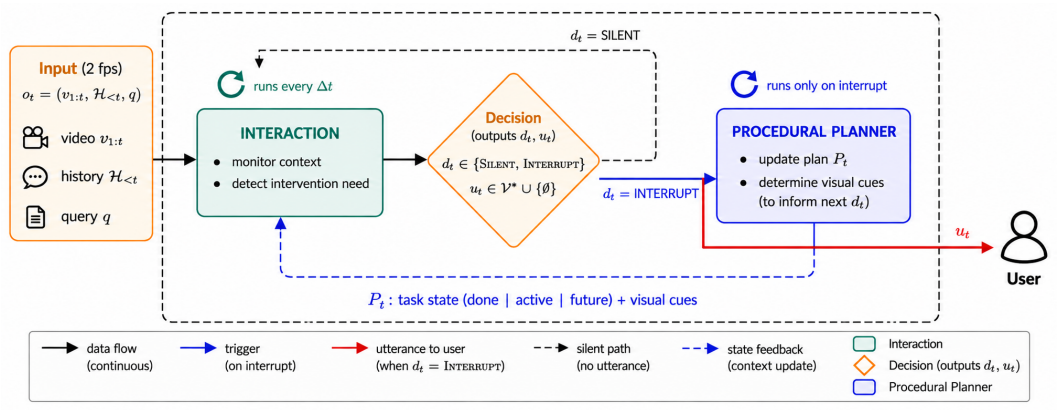


Figure 2: **PWR.** At observation o_t, the interaction model conditions on the latest plan state P_{t-1} to emit decision d_t and, when interrupting, utterance u_t. An interrupt triggers an asynchronous planner call. The revised state P_t replaces P_{t-1} for later decisions once it passes schema validation.

and δ the Dirac delta. Each procedure opens with the response of the assistant to q, treated as an interrupt at $t = 0$ with $P_{-1} = \emptyset$. On a step completion, the planner advances the `current` status and writes new cues. On an OOP deviation it writes a recovery plan. Conditioning on u_t keeps the revision consistent with what the user was just told. Because P_{t-1} already encodes the procedural history, the planner sees only the most recent clip.

We run the planner asynchronously so that the interaction model continues processing observations while emitting a `silent` token or a short instruction every 500 ms. A plan averages 114.9 tokens against 12.3 for a short instruction. On our hardware the plan output takes 0.9 to 1.6 s to decode, or two to three observation intervals. A single model that also revised its plan would stop observing for this time after every interrupt, precisely while the user acts on the guidance it just gave. Faster hardware, serving optimizations, and shorter plans reduce this cost, but they reduce it equally for PWR and are orthogonal to the design. Whatever a revision costs, a single model pays it inside the decision loop, and PWR pays it outside. The interaction model never waits on the planner, and the expected cost per timestep is $C_\theta + \Pr(\text{INTERRUPT})\, C_\phi$, where C_θ and C_ϕ are the per-call costs of the interaction model and the planner.

3.3 Training and Inference

A plan in context is necessary but not sufficient. Zero-shot models given a plan gain little (App. C), so we post-train. The planner and the interaction model are trained with the same supervised recipe on the same Pro^{2}Bench decision points. Each decision point pairs the reference decision $d_t^\star$ and utterance $u_t^\star$ with the latest reference plan state $P_{t-1}^\star$. The interaction model learns $p_\theta(d_t^\star, u_t^\star \mid o_t, P_{t-1}^\star)$ at every decision point. The planner learns $p_\phi(P_t^\star \mid d_t^\star, u_t^\star, o_t, P_{t-1}^\star)$ at interrupt points. A silent point teaches the interaction model to stay silent given a specific current step and its cues. An interrupt point teaches it to speak because the observation matched or broke that expectation. At inference, each interrupt triggers an asynchronous planner call, and user prompts and OOP recovery need no dedicated machinery. Objectives, sampling, and inference details are in App. A.

4 Experiments and Results

Setup. We evaluate on the Pro^{2}Bench evaluation split (Table 1) with the metrics of §2.4. Each decision point contains up to 16 chronological frames from an 8 s window sampled at 2 fps and the dialogue history. PWR carries its own predicted plan state forward and calls the planner only after a predicted INTERRUPT, as in Eq. 4. We report Turn Macro-F1, Procedure Macro derived from the same turn predictions, and PQS for the joint quality of when to speak and what to say (§2.4). We compare GPT-5.6, Gemini 3.6 Flash, and Claude Opus 5 with the open-weight Kimi K3, Inkling, Gemma 4 31B,

Muse Glimmer 30B, and Qwen3.8-VL-27B. Annotation tags, targets, and future mistake or recovery information are removed from every model input. Training inputs, controls and prompts are in App. C.

Training Recipe. For Gemma, Muse, and Qwen we train the planner and the interaction model with LoRA on EgoExo4D, EPIC-KITCHENS, HowTo100M, and Ego4D, using the recipe of §3.3. HoloAssist and EgoProactive are held out, so no OOP session is seen in training. We sample sources in proportion to the square root of their size and select checkpoints on a validation split whose procedures do not overlap training. The *FT, no plan* control trains the interaction model on the same data with the plan removed from its context, so it differs from PWR only in the input context. Details are in App. H.

4.1 Main Findings

Pro^{2}Bench is far from solved, and procedure-level scoring shows why. Zero-shot, no model completes more than 26% of reference interventions before its first error, and GPT-5.6 and Gemini complete under 7% (Table 2). Turn Macro-F1 ranges from .461 to .567. Turn level and procedure level metrics also rank the models differently. Qwen has the highest zero-shot Procedure Macro (.262) but amongst the lowest Turn Macro-F1 (.463), while Claude Opus 5 leads on turns (.567) and trails on procedures (.186). Deciding well on isolated turns does not mean a model can carry a user through a procedure.

The gains come from the plan, not from the training data. PWR improves over its zero-shot backbone on all three metrics for Gemma, Muse, and Qwen (Table 2). Training the same interaction model on the same data without a plan recovers little of this gain. Averaged over the three backbones, it changes Procedure Macro by +.017, Turn Macro-F1 by −.001, and PQS by +.023, and it lowers at least one metric for every backbone. Adding the plan then improves all nine comparisons, by +.070 Procedure Macro, +.068 Turn Macro-F1, and +.176 PQS on average. Gemma is the one backbone where data alone accounts for most of the procedure-level gain (.216 to .280 of .325). Because zero-shot models given a plan also gain little (App. C), neither the plan nor the training data suffices alone. The plan helps once the model is trained to read it. Figure 3 shows the corresponding model-wise profiles, including subjective response quality. The repeated gains across distinct backbones support the joint benefit of learned procedural state and an interaction model trained to consume it.

The gains hold on held-out sources and on out-of-plan deviations. On HoloAssist and EgoProactive, neither seen in training, PWR improves Turn Macro-F1 and PQS for all three backbones (Tables 15 and 16). The OOP sessions of EgoProactive are the hardest test, since every deviation is scripted but none appears in training (Table 3). PWR raises OOP Procedure Macro for every backbone, most for Qwen (.038 to .291), and raises OOP PQS by .094 to .250. This is also where revising the plan matters. Replacing the dynamic plan with a static one lowers OOP Procedure Macro for every backbone, by 2 to 8 points relatively.

5 Qualitative Results

Figure 4 shows two distinct failure modes: baselines can speak during an ongoing step or remain silent when guidance is warranted. In the tool-mistake case, Claude also interrupts but moves on to the handle without naming the needed tool change. PWR gives the specific correction. This exposes a guidance failure that an interrupt-only judgment would miss.

6 Related Work

Egocentric procedural understanding. Ego4D (Grauman et al., 2022), EPIC-KITCHENS (Damen et al., 2022), Ego-Exo4D (Grauman et al., 2024), and HowTo100M (Miech et al., 2019) provide scale and domain coverage, but target recognition rather than assistance. Assembly101 (Sener et al., 2022), CaptainCook4D (Peddi et al., 2024), and IndustReal (Schoonbeek et al., 2024) annotate mistakes without paired recovery guidance, while HoloAssist (Wang et al., 2023) provides 166 hours of expert-novice HoloLens 2 interaction with real-time guidance. Related methods include Seminara et al. (2024) on task graphs, Shen and Elhamifar (2024) on progress-aware streaming

		Zero-shot only					*Trained*		
Metric	**Config**	**GPT**	**Gem.**	**Opus**	**Kimi**	**Ink.**	**Gemma**	**Muse**	**Qwen**
Procedure Macro	Zero-shot	.068	.058	.186	.096	.249	.216	.078	.262
	FT, no plan	–	–	–	–	–	.280	.080	.248
	PWR	–	–	–	–	–	**.325**	**.213**	**.279**
	Δ_{plan}						↑.045	↑.133	↑.031
Turn Macro-F1	Zero-shot	.490	.461	.567	.465	.532	.539	.432	.463
	FT, no plan	–	–	–	–	–	.555	.425	.452
	PWR	–	–	–	–	–	**.592**	**.501**	**.542**
	Δ_{plan}						↑.037	↑.076	↑.090
PQS	Zero-shot	.416	.398	.461	.408	.412	.430	.280	.392
	FT, no plan	–	–	–	–	–	.400	.358	.412
	PWR	–	–	–	–	–	**.625**	**.468**	**.605**
	Δ_{plan}						↑.225	↑.110	↑.193

Table 2: **Results on Pro^{2}Bench.** Each metric is the equal-weight mean over the five evaluation datasets, with EgoProactive OOP and non-OOP sessions pooled as one source. *FT, no plan* trains the interaction model on the same data as PWR without a plan in context. PWR adds the plan and the learned planner. Δ_{plan} is PWR minus *FT, no plan*, the gain from the plan alone. **Bold** is the best configuration for each trained backbone. Ours.

		Trained		
Metric	**Config**	**Gemma**	**Muse**	**Qwen**
Procedure Macro	Zero-shot	.167	.024	.038
	FT, no plan	.185	.038	.194
	PWR	**.245**	**.134**	**.291**
	Δ_{plan}	↑.060	↑.096	↑.097
Turn Macro-F1	Zero-shot	.579	.451	.434
	FT, no plan	.524	.406	.498
	PWR	**.582**	**.484**	**.553**
	Δ_{plan}	↑.058	↑.078	↑.055
PQS	Zero-shot	.457	.254	.338
	FT, no plan	.433	.433	.251
	PWR	**.551**	**.504**	**.566**
	Δ_{plan}	↑.118	↑.071	↑.315

Table 3: **Out-of-plan recovery on EgoProactive.** Each metric is computed on the EgoProactive-OOP procedures. *FT, no plan* trains the interaction model on the same data as PWR without a plan in context. PWR adds the plan and learned planner. Δ_{plan} is PWR minus *FT, no plan*. **Bold**: best config for each trained model. Ours.

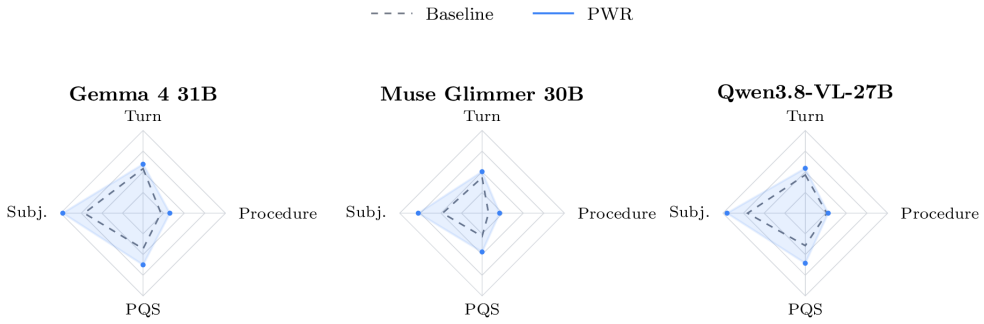


Figure 3: **Pooled performance profiles.** Zero-Shot and PWR are compared on Turn Macro-F1, Procedure Macro, PQS, and subjective response quality (with scores rescaled from 1–5 to 0–1).

segmentation, and Li et al. (2025b) on the broader procedural-understanding space. None combines wearable capture, per-decision-point intervention labels, paired deviation and recovery, and a shared multi-domain proactive-guidance schema.

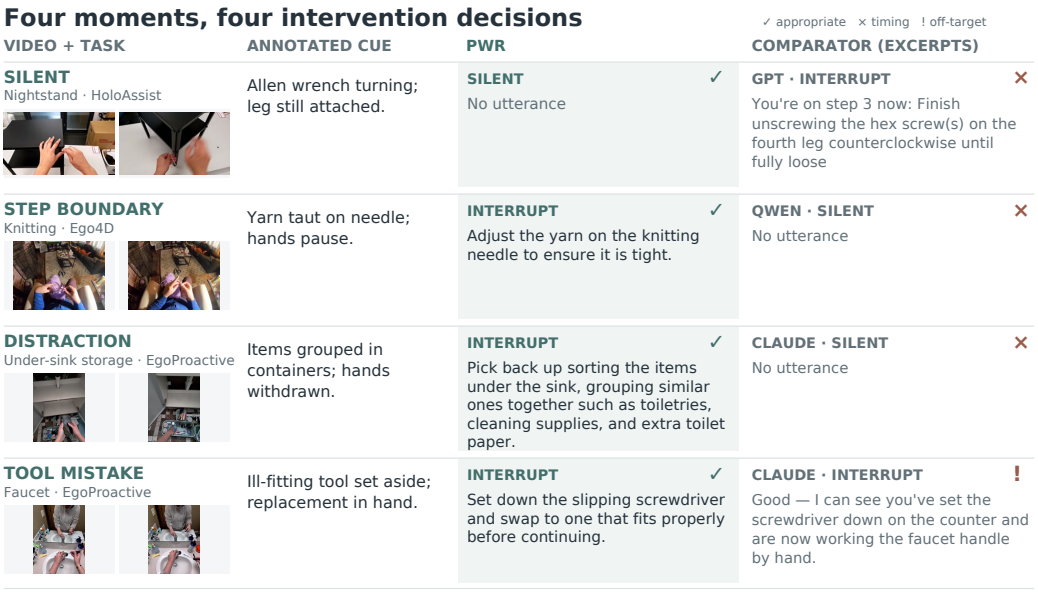


Figure 4: Qualitative analysis of four decision regimes. Frame pairs are sampled from each source clip. Cue descriptions are derived from the source annotations; PWR utterances are complete and longer comparator utterances are excerpted for display. More examples in App.G.

From detection to recovery. Anticipation methods use LLMs as mental simulators (Zhao et al., 2024) or translate actions into language for long-term prediction (Kim et al., 2024). Complementary work detects online egocentric mistakes (Flaborea et al., 2024), unifies procedural and execution-state detection (Patsch et al., 2025), or reasons about object-state effects (Guo et al., 2026). These methods identify that execution has diverged, but do not jointly evaluate when to intervene and how to guide recovery across multi-domain wearable procedures. EgoProactive makes this transition an explicit target.

Streaming perception and proactive intervention. A proactive assistant must process streaming video and intervene without disrupting the user. Dispider (Qian et al., 2025) separates perception, decision, and reaction, while StreamBridge (Wang et al., 2025) adapts offline video language models to streaming inference. For timing, Li et al. (2025a) trigger guidance from belief and task-graph divergence, Arakawa et al. (2024) use smartwatch sensing, and Chen et al. (2025) separate when and what to communicate; DARPA PTG (DARPA, 2022) field-tested augmented-reality guidance. These systems advance streaming perception and timing, but generally assume canonical execution, a single domain, or omit joint evaluation of timing and free-form recovery.

Proactive procedural assistance. Most closely related, Zhang et al. (2025) map streaming egocentric video to proactive dialogue using 30K synthetic dialogues and identify procedural reasoning as the main bottleneck. We evaluate that reasoning on real step-level deviations across domains and at the procedure level, and we address it with an explicit plan state that stays current as the user acts.

7 Conclusion

We introduced EgoProactive, a multi-domain smart-glasses dataset of step-level deviations and their recoveries, and Pro^{2}Bench, a benchmark that scores proactive assistants per turn and per procedure. Current models complete at most 26% of a procedure's interventions before their first error. PWR keeps an explicit plan state in context and revises it after every intervention. Across three backbones the gains come from the plan rather than the training data, and they hold on held-out sources and on deviations never seen in training. Revising the plan matters most where the plan must change, which makes planning under deviation the natural next target.

AI Use Statement

We used vision-language models (VLMs) and Large Language Models (LLMs) to align, check, and revise candidate procedural annotations K, and to evaluate response quality, as described in 2.4 and Appendices C, J, and K. The authors reviewed the AI-assisted material and take responsibility for the final data, analyses, figures, and claims.

References

Riku Arakawa, Hiromu Yakura, and Mayank Goel. PrISM-Observer: Intervention agent to help users perform everyday procedures sensed using a smartwatch. In *UIST*, 2024. doi: 10.1145/3654777.3676350.

Apratim Bhattacharyya, Bicheng Xu, Sanjay Haresh, Reza Pourreza, Litian Liu, Sunny Panchal, Pulkit Madan, Leonid Sigal, and Roland Memisevic. Can multi-modal LLMs provide live step-by-step task guidance? In *NeurIPS*, 2025.

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

Delong Chen, Tejaswi Kasarla, Yejin Bang, Mustafa Shukor, Willy Chung, Jade Yu, Allen Bolourchi, Theo Moutakanni, and Pascale Fung. Action100M: A large-scale video action dataset. In *CVPR Workshops*, pages 8832–8842, 2026.

Tuochao Chen, Nicholas Scott Batchelder, Alisa Liu, Noah A. Smith, and Shyamnath Gollakota. LlamaPIE: Proactive in-ear conversation assistants. In *Findings of ACL*, pages 13801–13824, 2025.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for EPIC-KITCHENS-100. *IJCV*, 130(1):33–55, 2022.

DARPA. Perceptually-enabled task guidance (PTG). Research program, 2022. URL `https://www.darpa.mil/research/programs/perceptually-enabled-task-guidance`.

Alessandro Flaborea, Guido Maria D'Amely di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella, and Fabio Galasso. PREGO: Online mistake detection in PRocedural EGOcentric videos. In *CVPR*, pages 18483–18492, 2024.

Kristen Grauman, Andrew Westbury, Eugene Byrne, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *CVPR*, pages 18995–19012, 2022.

Kristen Grauman et al. Ego-Exo4D: Understanding skilled human activity from first- and third-person perspectives. In *CVPR*, 2024.

Wenliang Guo, Yujiang Pu, and Yu Kong. Procedural mistake detection via action effect modeling. In *ICLR*, 2026.

Sanghwan Kim et al. PALM: Predicting actions through language models. In *ECCV*, 2024.

Chenyi Li, Guande Wu, Gromit Yeuk-Yin Chan, Dishita G. Turakhia, Sonia Castelo Quispe, Dong Li, Leslie Welch, Claudio Silva, and Jing Qian. Satori: Towards proactive AR assistant with belief-desire-intention user modeling. In *CHI*, 2025a. doi: 10.1145/3706598.3714188.

Junlong Li, Huaiyuan Xu, Sijie Cheng, Kejun Wu, Kim-Hui Yap, Lap-Pui Chau, and Yi Wang. Building egocentric procedural AI assistant: Methods, benchmarks, and challenges. *arXiv preprint arXiv:2511.13261*, 2025b.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019.

Constantin Patsch, Yuankai Wu, Marsil Zakour, Driton Salihu, and Eckehard Steinbach. MistSense: Versatile online detection of procedural and execution mistakes. In *ICCV*, 2025.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Qifan Zhang, Jikai Wang, Vasundhara Komaragiri, Eric Ragan, et al. Captain-Cook4D: A dataset for understanding errors in procedural activities. In *NeurIPS Datasets and Benchmarks Track*, 2024.

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video LLMs with active real-time interaction via disentangled perception, decision, and reaction. In *CVPR*, 2025.

Tim J. Schoonbeek, Tim Houben, Hans Onvlee, Peter H. N. de With, and Fons van der Sommen. IndustReal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting. In *WACV*, pages 4365–4374, 2024.

Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos. In *NeurIPS*, 2024.

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. In *CVPR*, pages 21064–21074, 2022.

Yuhan Shen and Ehsan Elhamifar. ProTAS: Progress-aware online action segmentation for egocentric procedural task videos. In *CVPR*, 2024.

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. In *NeurIPS*, 2025.

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. In *ICCV*, pages 20270–20281, 2023.

Yichi Zhang, Xin Luna Dong, Zhaojiang Lin, Andrea Madotto, Anuj Kumar, Babak Damavandi, Joyce Chai, and Seungwhan Moon. Proactive assistant dialogue generation from streaming egocentric videos. In *EMNLP*, pages 12044–12068, 2025.

Qi Zhao, Shijie Wang, Ce Zhang, Changcheng Fu, Minh Quan Do, Nakul Agarwal, Kwonjoon Lee, and Chen Sun. AntGPT: Can large language models help long-term action anticipation from videos? In *ICLR*, 2024.

A Method Details

This appendix gives the full details summarized in §3.3.

A.1 Plan-Anchored Clip Selection

The interaction model cannot consume $v_{1:t}$ in full at every timestep. Rather than a recency-only window, which discards long-range context in sessions spanning minutes to hours, we let the plan decide which past video to keep. At each timestep the interaction model receives up to $K = 15$ non-contiguous 8 s clips from $v_{1:t}$. One clip covers the most recent 8 s ending at t. The remaining clips are anchored at prior plan-update timestamps, keeping the $K - 1$ most recent when more updates have occurred. Since the session opens with a plan update at $t = 0$, at least one anchored clip is always available. The dynamic context therefore shapes both *what* the model looks for, through $\mathbf{c}$, and *where* in the timeline it looks.

Application to baselines. All implicit and static context baselines, including our reimplementation of Zhang et al. (2025), receive the same clip selection. For the implicit and static baselines, anchors are the timestamps of their own past interrupts, which is the same rule as for dynamic context, since plan updates occur exactly at interrupts. Differences between the three regimes therefore reflect the procedural context and not the visual input.

A.2 Post-Training on Plan-Conditioned Trajectories

This section expands the post-training summary in §3.3. We find that the gains of dynamic context are largest after post-training on data that pairs each decision with the plan state that should inform it (§4).

Plan-state densification. Pro^{2}Bench annotates interrupts sparsely. For a session with annotated interrupt timesteps $\tau_0 = 0 < \tau_1 < \cdots < \tau_M$, each τ_m carries a ground-truth utterance $u^*_{\tau_m}$ and a ground-truth plan state $P^*_{\tau_m}$. We convert these into dense plan-conditioned supervision. For every timestep t, the context plan is $P^*_{t-1} = P^*_{\tau_m}$ with τ_m the latest interrupt strictly before t. Every silent frame thus becomes an example of *staying silent given a specific current step and its cues*, and every interrupt becomes an example of *speaking because the observation matched or broke that expectation.* This turns a sparse event annotation into a trajectory over plan states, which is the form of data the dynamic context formulation requires.

Objectives. Both models are trained with supervised fine-tuning. The combined objective is $\mathcal{L} = \mathcal{L}_\theta + \mathcal{L}_\phi$ with

$$\mathcal{L}_\theta = -\mathbb{E}_t\big[\log p_\theta(y_t \mid o_t, P^*_{t-1})\big], \qquad \mathcal{L}_\phi = -\mathbb{E}_m\big[\log p_\phi(P^*_{\tau_m} \mid u^*_{\tau_m}, o_{\tau_m}, P^*_{\tau_{m-1}})\big],$$

where $y_t = (d_t, u_t)$ is the target sequence, equal to $(\textsc{Silent}, \emptyset)$ at silent timesteps and $(\textsc{Interrupt}, u^*_t)$ at interrupt timesteps, and $P^*_{\tau_{-1}} = \emptyset$ at the session-opening interrupt, where $o_{\tau_0} = (q, \emptyset)$. Silent timesteps contribute a single-token target and interrupt timesteps the full utterance, both under the same autoregressive cross-entropy. The parameters θ and ϕ appear in disjoint terms, so the two models can be trained independently.

Sampling. Silent frames dominate procedural video, which creates a severe class imbalance (Zhang et al., 2025). We sample silent and interrupt timesteps for $\mathcal{L}_\theta$ at roughly 1:1. Clip anchors come from ground-truth plan-update timestamps, and the same clip selection is used at training and inference. The planner is trained only on annotated interrupts, with the most recent clip as its visual input.

Teacher forcing on the plan. Training conditions on ground-truth plan states P^*, while inference conditions on predicted states $\hat{P}$. The interaction model never sees an imperfect plan during training. We quantify this gap by reporting both oracle-plan and predicted-plan variants (§4). Closing it, for example by training on the planner's own rollouts, is left to future work.

A.3 Inference

At each timestep t the interaction model receives (o_t, P_{t-1}) and produces (d_t, u_t). If $d_t = \textsc{Silent}$, then $P_t = P_{t-1}$ by Eq. 5 and the planner is not called. If $d_t = \textsc{Interrupt}$, the planner is called on (d_t, u_t, o_t, P_{t-1}) to produce P_t. Two further properties follow from keeping the procedure in context.

Reactive turns for free. The interaction model runs at every timestep regardless of who is speaking. A user question mid-task (e.g., *"How long should I tamp?"*) enters $\mathcal{H}_{<t}$, and the next forward pass can answer it with the current plan state in view.

Recovery without OOP-specific machinery. Step completions and OOP deviations trigger the same update. Only the planner's revision differs. No deviation-specific module or loss is used. Recovery follows from comparing the observation against an explicit plan state and rewriting that state when they disagree. We observe sensible recovery plans on held-out EgoProactive deviations with no training on that source (§4).

B Results: Detailed Analysis

This section expands the main findings in §4.1. It summarizes the matched evaluation conditions, reports cross-model and held-out-source trends, and provides additional analysis of the plan-conditioned controls. Complete per-dataset Turn and subjective results are reported in Tables 12–16.

B.1 Setup and Comparisons

Models. We evaluate three proprietary models, GPT-5.6, Gemini 3.6 Flash, and Claude Opus 5, and five open-weight models, Kimi K3, Inkling, Gemma 4 31B, Muse Glimmer 30B, and Qwen3.8-VL-27B. The three trained systems use Gemma, Muse, and Qwen model families. Each pairs a learned planner with a corresponding LoRA interaction model trained on EgoExo4D, EPIC-KITCHENS, HowTo100M, and Ego4D. HoloAssist and EgoProactive are excluded from training and therefore measure transfer to held-out sources.

Conditions. The main paper compares **Baseline**, **Fine-tuned, no plan**, and **PWR**. The fine-tuned no-plan row uses the same LoRA interaction checkpoint as PWR but omits planner state, separating training from planner conditioning. PWR pairs the learned planner with that interaction model. Appendix tables additionally report a **Plan** control. This is the stronger matched result from either a fixed reference plan or a row-local dynamic plan generated by the same model. The best-of selection favors the control; the same selected condition is then used for that model and dataset across Turn, Procedure, and subjective summaries. Full input and prompting definitions are provided in Appendix C.2.

Metrics. Turn Macro-F1 is the arithmetic mean of interrupt F1 and silent F1. Procedure Macro measures the normalized correct prefix before the first incorrect decision, using the same Turn predictions under reference dialogue history. PQS assigns full credit to correct silences, quality-weighted credit to correct interruptions, and zero credit to incorrect decisions. Subjective response quality is the mean of relevance, specificity, actionability, and conciseness on a 1–5 scale. Metric definitions and the frozen judging rubric are given in Appendix C.6 and Appendix C.7.

B.2 Aggregate Results

Training and planning contributions are separated. PWR improves over the unmodified Baseline in all nine aggregate comparisons across Turn Macro-F1, Procedure Macro, and PQS (Table 2). The fine-tuned no-plan control separates training from planner conditioning. PWR leads seven of eight completed comparisons with this control; Qwen Procedure Macro is .298 without a plan and .279 with PWR. These controls show that the aggregate gains combine training and plan conditioning.

The gains replicate across model families. Gemma improves from .539 to .592 in Turn Macro-F1, .216 to .325 in Procedure Macro, and .430 to .625 in PQS. Muse improves from .432 to .501, .078 to .213, and .280 to .468, respectively. Qwen improves from .463 to .542, .262 to .279, and .392 to .605. Gemma provides the strongest aggregate PWR result on all three reported metrics. Qwen provides the largest PWR gain over its unmodified baseline in Turn and PQS, while Muse provides the largest Procedure gain.

B.3 Generalization to Held-Out Sources

HoloAssist. PWR improves Turn Macro-F1 and PQS for all three trained model families on HoloAssist (Table 15). Gemma improves from .571/.451 to .600/.630, Muse from .464/.366 to .581/.542, and Qwen from .452/.372 to .504/.574, where each pair reports Turn Macro-F1/PQS.

EgoProactive. The same pattern holds on EgoProactive (Table 16). Gemma improves from .584/.478 to .598/.582, Muse from .449/.290 to .509/.533, and Qwen from .450/.354 to .563/.594. These gains are obtained without using either held-out source for interaction-model training.

B.4 Out-of-Plan Recovery

The canonical EgoProactive-OOP subset tests whether the system can recognize and recover from deviations rather than only follow a nominal procedure. PWR improves all nine comparisons with the unmodified no-plan baselines in Table 3. For Gemma, Turn Macro-F1, Procedure Macro, and PQS change from .579/.167/.457 to .582/.245/.551. Muse improves from .451/.024/.254 to .484/.134/.504. Qwen improves from .434/.038/.338 to .553/.291/.566. The largest Procedure improvement is obtained by Qwen, and the

largest PQS improvement is obtained by Muse. This result is especially important because EgoProactive and its OOP deviations are excluded from the training mixture. Table **??** separately isolates the effect of updating the plan within these OOP procedures.

B.5 Plan-Conditioned Controls

The Plan controls measure the effect of supplying procedural state to the interaction model. At the aggregate level, Gemma Plan obtains .560 Turn Macro-F1, .361 Procedure Macro, and .581 PQS; Muse Plan obtains .391, .030, and .320; and Qwen Plan obtains .484, .191, and .495. PWR exceeds these controls in eight of the nine matched aggregate comparisons. The only exception is Gemma Procedure Macro, where Plan reaches .361 and PWR reaches .325. PWR nevertheless provides Gemma's strongest Turn Macro-F1 and PQS, and it improves all three metrics over Plan for both Muse and Qwen. The control is therefore not uniformly weaker: procedural context can already be competitive, and the benefit of the learned planner depends on the model and metric. The per-dataset controls are intentionally retained in the appendix rather than the headline table. They show where a reference plan or same-model dynamic plan is already sufficient and where the matched PWR pairing adds a further gain. Tables 12–16 report the selected Plan control alongside Baseline and PWR under identical frozen examples and denominators.

B.6 Subjective Response Quality

PWR improves pooled subjective response quality over the unmodified baseline for every trained model family. On the original 1–5 scale, the pooled four-dimension mean increases from 3.58 to 4.87 for Gemma, 2.40 to 3.87 for Muse, and 3.50 to 4.75 for Qwen. Gemma PWR remains between 4.79 and 4.98 across the five per-dataset subjective means; Qwen remains between 4.56 and 4.90. Muse improves on every source and reaches 4.00 on the held-out EgoProactive set. The accompanying PQS gains show that these improvements are not obtained by evaluating response fluency in isolation: a response receives quality credit only after a correct decision to speak.

B.7 Evaluation Scope

All comparisons in this section use the same frozen decision points, frames, preceding canonical dialogue, and denominators. Each decision point is scored independently. Procedure results are deterministic aggregations of those Turn predictions and do not invoke a second model evaluation. PWR therefore tests the planner-to-interaction sequence defined in Appendix C.2. The benchmark evaluates annotated decision points and correct-prefix consistency; continuous-stream serving measurements are outside its metric definition.

C Evaluation Protocol

This section specifies the models, frozen inputs, planning conditions, prompts, and metrics used in the reported experiments.

C.1 Models and Decoding

We evaluate eight VLMs. All reported interaction predictions use deterministic decoding (temperature 0) and must emit either `$silent$` or `$interrupt$` followed by one concise instruction. Each decision point uses the same frozen frames, dialogue history, and decoding contract across matched conditions; the reference response is retained only for scoring. The three trained model families use a learned planner and a matched LoRA interaction model. Table 4 summarizes their evaluation roles.

Input sanitization. Model inputs contain only information available through the current decision point. Before prompt construction, we remove annotation-only tags such as `(mistake)`, `(interruption)`, and `(fix)`, together with event categories, target decisions, and reference responses. Plan context is sanitized in the same pass: it contains no future deviation, scripted mistake, recovery action, or later observation. These fields remain only in the scoring records and are never exposed to either the planner or interaction model. GPT-5.6 is used as a text-only judge under a frozen rubric. To avoid self-evaluation, we do not report PQS or response-quality scores for the GPT-

Table 4: Model configurations used in the reported evaluation.

Model	Evaluation role
GPT-5.6	Proprietary baseline; subjective judge, not self-scored
Gemini 3.6 Flash	Proprietary baseline
Claude Opus 5	Proprietary baseline
Kimi K3	Open-weight baseline
Inkling	Open-weight baseline
Gemma 4 31B	Baseline + LoRA interaction model + learned planner
Muse Glimmer 30B	Baseline + LoRA interaction model + learned planner
Qwen3.8-VL-27B	Baseline + LoRA interaction model + learned planner

5.6 candidate. Exact response matches receive deterministic full credit; all remaining true-positive interrupts from other models are judged with the same prompt and schema.

C.2 Frozen Evaluation Conditions

The main paper reports the baseline, a matched fine-tuned no-plan control, and PWR. Additional plan controls are retained in the appendix:

Baseline. No planner state is supplied to the unmodified interaction model.

Fine-tuned, no plan. No planner state is supplied to the LoRA interaction model used by PWR. This isolates interaction-model training from planner conditioning.

Reference plan. The interaction model receives a fixed reference plan for the current procedure.

Dynamic plan. The same model updates the reference-seeded plan from the current observation and preceding canonical history before producing the interaction response. Appendix tables label the stronger result from the reference-plan and dynamic-plan controls as **Plan**. This best-of control is deliberately favorable to plan conditioning; one selection is then shared across its Turn, Procedure, and subjective summaries.

PWR. The planner first produces P_0 after the user's procedural request. At decision point t, the LoRA interaction model receives the current frames o_t, preceding dialogue $\mathcal{H}_{<t}$, and latest plan P_{t-1}, then predicts $(\hat{d}_t, \hat{u}_t)$. A predicted interrupt invokes the planner, which advances the state to P_t for subsequent decision points; otherwise $P_t = P_{t-1}$.

The matched conditions use the same decision points, frames, history, reference response, and decoding configuration, so their denominators match exactly. PWR is evaluated separately using a dedicated learned planner paired with the LoRA interaction model.

Reported PWR information flow. The evaluation follows the same decision-then-update order as the architecture:

$$P_{t-1} \to (o_t, \mathcal{H}_{<t}, P_{t-1}) \to (\hat{d}_t, \hat{u}_t) \to \begin{cases} P_t = \mathcal{U}(o_t, \mathcal{H}_{<t}, P_{t-1}), & \hat{d}_t = \texttt{interrupt}, \\ P_t = P_{t-1}, & \hat{d}_t = \texttt{silent}. \end{cases}$$

The existing future-step context in P_{t-1} lets the interaction model announce the next action when it detects completion; that interrupt then advances the planner state used by later decision points. Visual observations and decision point identities remain frozen across matched conditions.

C.3 Frozen Interaction Prompt

The interaction model uses this system prompt. In no-plan mode, the planner-state line is omitted; plan-conditioned evaluations insert the appropriate reference or learned JSON plan.

```
You are a helpful, proactive multimodal assistant guiding a user through a
step-by-step procedural task from egocentric video.
Track the current plan step using the supplied planner state and prior
conversation.  Interrupt when the visible current step has completed and
provide only the next grounded action.  Interrupt when a visible mistake,
skipped step, wrong order, substitution, or out-of-procedure deviation
requires recovery.  If the step is underway, has not started, or completion
is uncertain, remain silent.
Output exactly one of:
- $silent$
- $interrupt$ followed immediately by one concise actionable instruction
Use only the progressive conversation, supplied plan state when present,
and chronological frames from the current authoritative interval.  Never
expose reasoning, narrate the completed action, repeat an already issued
instruction, or use future information.
```

C.4 Learned Planner Prompt and State

For PWR, the planner is called once after the initial user request to create P_0, then after each predicted interrupt to advance the plan. Update calls receive the current 16-frame window, the latest generated plan, and the preceding dialogue. Its frozen system prompt is:

```
You are a helpful planning agent.  You will be provided with a conversation
between a user and an assistant, the current visual context, and your
previous plan.
Identify whether the user is beginning or continuing a procedural task.
Produce the procedural plan that should guide the assistant's next decision.
Preserve the task sequence supported by the supplied context.  Include
current, next_few, last_few, eta, visual_cues, and every dataset-specific
field required by the plan.
Respond with only one valid JSON object.  Do not add an intent token, control
prefix, prose, markdown, or a code fence.
```

The canonical plan object contains `vertical`, `task`, `eta`, `visual_cues.{interrupt,silent}`, `last_few`, `current`, and `next_few`. Step identifiers must remain unique and ordered, and `eta` is constrained to $[0, 8]$. The planner output and interaction-model input must correspond to the same procedure and decision point. Any malformed JSON, duplicate identity, frame mismatch, cross-procedure state, future history, or incomplete denominator fails closed.

C.5 Evaluation Population and Metric Denominators

The evaluation population is the frozen manifest obtained after source-specific validity checks, deduplication, and train–evaluation leakage checks. We retain every eligible decision point. We do not balance the class labels, equalize source sizes, or draw separate samples for different metrics. Every completed model condition therefore predicts the same 26,874 decision points. A canonical procedure unit is identified by its source procedure, task, and reference step sequence; a source video may therefore contribute more than one procedure unit. Turn Macro-F1 computes interrupt and silent F1 over all decision points in one source and averages the two class scores. Procedure Macro reuses those predictions, orders them within each procedure unit, and averages the normalized correct-prefix score across the procedure units in that source. PQS again uses every decision point in the source: correct silences receive 1, incorrect decisions receive 0, exact true-positive responses receive deterministic full credit, and the remaining true-positive responses receive their normalized GPT-5.6 quality score. The headline value for each metric is the equal-weight mean of its five source-level values, not a micro-average over the 26,874 points. The EgoProactive-OOP analysis is a deterministic slice of the same frozen population. It contains 3,810 decision points grouped into 247 procedures. These points remain part of the pooled EgoProactive result and are not sampled again for the OOP table.

Table 5: Frozen evaluation denominators. Interrupt and silent counts are the decision-point denominators for Turn Macro-F1 and PQS. Procedure units are the denominators for Procedure Macro.

Source	Points	Interrupt	Silent	Procedures
EgoExo4D	2,734	1,551	1,183	286
EPIC-KITCHENS	6,399	4,172	2,227	667
HowTo100M	4,654	4,168	486	819
HoloAssist	3,188	1,695	1,493	237
EgoProactive	9,899	5,338	4,561	700
Total	**26,874**	**16,924**	**9,950**	**2,709**

Source	Always interrupt	Always silent
EgoExo4D	.362	.302
EPIC-KITCHENS	.395	.258
HowTo100M	.472	.095
HoloAssist	.347	.319
EgoProactive	.350	.315
Five-source mean	**.385**	**.258**

Table 6: Turn Macro-F1 for constant-decision baselines on the frozen evaluation population.

Trivial decision baselines. Table 6 reports the Turn Macro-F1 obtained by predicting one class at every decision point. These baselines make the effect of source imbalance explicit and remain far below the learned systems.

C.6 Turn and Procedure Metrics

Turn Macro-F1 is the arithmetic mean of the interrupt-class and silent-class F1 scores. Invalid outputs count as errors. G-mean recall is retained as a secondary diagnostic but is not the aggregate timing metric in Table 2. Procedure Macro measures sustained consistency across the ordered decision points of a procedure. Correct silent turns extend the valid prefix and correct interrupt turns increment the credited-interrupt count. The first wrong decision or invalid output ends that prefix, and later predictions receive no credit. For procedure p, let c_p be the number of gold interrupt points reached within the correct prefix and L_p the total number of gold interrupt points. Its correct-prefix score is c_p/L_p, and Procedure Macro is $|\mathcal{P}|^{-1} \sum_{p \in \mathcal{P}} c_p/L_p$. It measures correct-prefix consistency across a procedure under reference history. EgoProactive-OOP is obtained by a deterministic procedure-level metadata split and uses the same scoring rule.

Output validity. Invalid interaction outputs are counted as incorrect decisions. In the final PWR artifacts, all three model families produce a parseable decision for all 26,874 points, and all planner records pass the required JSON schema after validation.

C.7 Subjective Quality and PQS

Only true-positive interrupts are scored for response quality. GPT-5.6 assigns integer scores from 1 to 5 for relevance, specificity, actionability, and conciseness under this frozen rubric. Exact matches to the reference response receive deterministic full credit.

```
You are a STRICT evaluator of procedural AI assistant responses.  Your job is
to find flaws, not praise.
The assistant guides a user through step-by-step tasks via egocentric video.
When a step is complete, it should proactively interrupt with the next step.
Golden response:  {golden}
Model response:  {model}
Task context:  {task_context}
Rate relevance, specificity, actionability, and conciseness from 1-5.  A 5
means a near-perfect match to the golden response, 3 means acceptable but
flawed, and 1 means wrong or harmful.  Return only a JSON object containing
the four integer scores and one-sentence reasoning.
```

For a judged true positive i, let $q_i = (\bar{s}_i - 1)/4$, where $\bar{s}_i$ is the mean of its four 1–5 scores. Over all N decision points,

$$\mathrm{PQS} = \frac{\mathrm{TN} + \sum_{i \in \mathrm{TP}} q_i}{N}.$$

Thus correct silences receive full credit, true-positive interruptions receive quality-weighted credit, and false positives, false negatives, and invalid outputs receive zero credit.

Judge scope. PQS is a deterministic, reference-aligned comparison under one fixed rubric. We report its four response dimensions separately so that timing and language quality can be inspected independently. GPT-5.6 is excluded as a PQS candidate because it is the judge.

Statistical scope. All inference uses deterministic decoding and checkpoints selected before the frozen evaluation. Table 5 reports the exact procedure and decision-point denominators, preserving the procedure-level clustering used by Procedure Macro.

Reproducibility note. All matched conditions use the same frozen evaluation examples, visual inputs, dialogue histories, prompts, model versions, and deterministic decoding. Frames are sampled chronologically at 2 fps from each 8 s interval, yielding 16 frames when the complete interval is available. Procedure Macro is derived from Turn predictions with identical denominators and reference histories.

D Per-Dataset Statistics

This section reports detailed statistics for the five evaluation datasets and the six-source benchmark collection. Ego4D contributes training data but is not part of the frozen evaluation split.

Provenance. All values are computed deterministically from the released evaluation manifest and its canonical semantic-unit identifiers. The same 26,874 decision points are used for every matched model condition. The released statistics file and the accompanying processing scripts reproduce Tables 7–11.

D.1 Evaluation Set Distributions

The split contains 16,924 interrupt and 9,950 silent decision points. EgoProactive contains 3,810 rows from 247 out-of-plan procedures. Its canonical pooled score includes both out-of-plan and non-out-of-plan procedures.

D.2 Procedure Length Distributions

Table 8 summarizes the number of reference-plan steps per canonical evaluation procedure.

D.3 Response Length Distributions

Table 9 reports reference-response lengths for interrupt decision points. EgoProactive has the longest reference responses because deviation and recovery guidance often identifies both the error and the corrective action. HoloAssist has the shortest responses, reflecting concise assembly instructions.

Table 7: Detailed statistics for the frozen evaluation split. Int. and Sil. denote the percentages of interrupt and silent decision points. Response length is measured on reference responses for interrupt points only. OOP denotes all rows belonging to canonical EgoProactive out-of-plan procedures.

Dataset	Eval n	Int.%	Sil.%	Avg. resp. words	Procedures	OOP%
EgoExo4D	2,734	56.7	43.3	9.5	286	0.0
EPIC-KITCHENS	6,399	65.2	34.8	10.9	667	0.0
HowTo100M	4,654	89.6	10.4	11.0	819	0.0
HoloAssist	3,188	53.2	46.8	6.9	237	0.0
EgoProactive	9,899	53.9	46.1	16.4	700	38.5
Total	**26,874**	**63.0**	**37.0**	**12.1**	**2,709**	**14.2**

Table 8: Reference-plan step counts for evaluation procedures. Range gives the minimum and maximum number of steps in a canonical semantic unit.

Dataset	Range	Avg. steps	Procedures
EgoExo4D	2–15	5.4	286
EPIC-KITCHENS	2–15	6.3	667
HowTo100M	1–13	5.1	819
HoloAssist	2–20	7.2	237
EgoProactive	3–15	7.6	700
All	1–20	6.2	2,709

D.4 Domain and Viewpoint Coverage

Table 10 summarizes the activity and viewpoint coverage of each benchmark source.

D.5 Training and Evaluation Coverage

Table 11 identifies the sources used for training and evaluation in the reported experiments. Training and evaluation are separated at the canonical semantic-unit level, with dataset-specific video and identity leakage checks applied before release. All reported model conditions use the same frozen evaluation identifiers and denominators.

E Per-Dataset Score Details

Tables 12–16 report the complete per-dataset Turn and subjective breakdown. Turn columns report interrupt F1 (IF1), silent F1 (SF1), and their macro-average. Subjective columns report relevance (R), specificity (S), actionability (A), conciseness (C), their mean, and PQS. **No plan** omits planner state. **Plan** is the stronger of two matched controls: a fixed reference plan or a row-local dynamic plan produced by the same model. The selection is made independently for each model and dataset and is shared across its Turn, Session, and subjective summaries. **PWR** pairs the learned planner with the corresponding fine-tuned interaction model. Table labels abbreviate Gemini 3.6 Flash as Gemini-3.6, Claude Opus 5 as Opus-5, Gemma 4 31B as Gemma-31B, Muse Glimmer 30B as Muse-30B, and Qwen3.8-VL-27B as Qwen-27B. Fine-tuned configurations (Ours) . **Bold** is best and underline is second within a column.

F Per-Dataset Score Details

G More qualitative results

Figures 5 to 8 show single-decision examples for each behavior regime, and Figure 9 traces full multi-turn sessions against frontier baselines.

Table 9: Reference response lengths for interrupt decision points.

Dataset	Range (words)	Avg. words
EgoExo4D	3–36	9.5
EPIC-KITCHENS	2–38	10.9
HowTo100M	2–46	11.0
HoloAssist	2–17	6.9
EgoProactive	1–62	16.4
All	1–62	12.1

Table 10: Activity and viewpoint coverage across benchmark sources. Ego4D is used for training only; the remaining five sources form the frozen evaluation split.

Dataset	Viewpoint	Primary coverage
Ego4D	Egocentric	Broad everyday activities
EgoExo4D	Ego + exo	Skilled procedural activities
EPIC-KITCHENS	Egocentric	Cooking
HowTo100M	Third person	Broad instructional activities
HoloAssist	Egocentric AR	Assembly and repair
EgoProactive	Smart glasses	Cooking, tutorials, repair, arts and crafts

Per-turn behavior. On *silent* timesteps, where the user is mid-step and no intervention is warranted, PWR withholds while baselines frequently interrupt (Figure 5). At genuine *step boundaries* it interrupts with the correct next instruction rather than mistiming or mis-specifying the cue (Figure 6). The out-of-procedure cases are the most discriminative: under an external distraction PWR recognizes the deviation and re-engages with the task (Figure 7), and when the user makes a procedural mistake it both detects the error and names the specific corrective action, where baselines either miss it or offer generic advice (Figure 8).

Table 11: Source participation in the reported experiments. HoloAssist and EgoProactive are held out from planner and interaction-model training.

Source	Interaction train	Planner train	Evaluation
Ego4D	✓	✓	–
EgoExo4D	✓	✓	✓
EPIC-KITCHENS	✓	✓	✓
HowTo100M	✓	✓	✓
HoloAssist	–	–	✓
EgoProactive	–	–	✓

		Turn			*Subjective*					
Model	**Config**	**IF1**	**SF1**	**Macro-F1**	**R**	**S**	**A**	**C**	**Avg**	**PQS**
GPT-5.6	No plan	.443	.546	.494	2.78	2.85	4.19	4.86	3.67	0.434
GPT-5.6	Plan	.482	.558	.520	3.23	3.43	4.43	4.87	3.99	0.467
Gemini-3.6	No plan	.389	.532	.461	2.64	2.64	4.09	4.91	3.57	0.410
Gemini-3.6	Plan	.494	.529	.511	3.50	3.68	4.47	4.95	4.15	0.461
Opus-5	No plan	.604	.525	.564	3.11	3.18	4.41	4.47	3.79	0.468
Opus-5	Plan	.571	.529	.550	4.34	4.37	4.79	4.73	4.56	0.518
Kimi-K3	No plan	.436	.567	.501	2.92	3.03	4.33	4.72	3.75	0.451
Kimi-K3	Plan	.418	.561	.490	3.61	3.80	4.55	4.87	4.21	0.458
Inkling	No plan	.603	.494	.548	3.01	3.05	4.30	4.43	3.70	0.445
Inkling	Plan	.627	.498	.563	4.29	4.34	4.78	4.78	4.55	0.531
Gemma-31B	No plan	.661	.439	.550	2.94	2.80	4.15	4.98	3.72	0.445
Gemma-31B	Plan	.712	.334	.523	4.02	4.13	4.70	4.99	4.46	0.531
Gemma-31B	PWR	**.740**	.490	**.615**	**4.90**	**4.93**	**4.97**	**5.00**	**4.95**	**0.649**
Muse-30B	No plan	.438	.443	.441	2.48	2.68	3.20	1.65	2.50	0.304
Muse-30B	Plan	.342	.503	.423	2.86	3.11	3.25	1.65	2.72	0.350
Muse-30B	PWR	.467	**.646**	.557	4.84	4.87	4.11	1.85	3.92	0.524
Qwen-27B	No plan	.674	.250	.462	2.81	2.77	4.12	4.93	3.66	0.388
Qwen-27B	Plan	.582	.431	.506	4.43	4.41	4.76	4.97	4.64	0.488
Qwen-27B	PWR	.720	.358	.539	4.60	4.63	4.84	4.99	4.76	0.580

Table 12: Per-dataset Turn and subjective results on EgoExo4D.

		Turn			*Subjective*					
Model	**Config**	**IF1**	**SF1**	**Macro-F1**	**R**	**S**	**A**	**C**	**Avg**	**PQS**
GPT-5.6	No plan	.420	.498	.459	2.11	2.13	4.03	4.88	3.29	0.378
GPT-5.6	Plan	.524	.513	.519	2.71	2.81	4.19	4.75	3.62	0.427
Gemini-3.6	No plan	.324	.486	.405	2.15	2.12	3.95	4.93	3.29	0.356
Gemini-3.6	Plan	.413	.454	.434	3.21	3.25	4.32	4.88	3.92	0.377
Opus-5	No plan	.624	.457	.540	2.39	2.38	4.21	4.54	3.38	0.406
Opus-5	Plan	.624	.453	.538	3.78	3.75	4.57	4.64	4.18	0.474
Kimi-K3	No plan	.393	.507	.450	2.18	2.20	4.12	4.71	3.30	0.381
Kimi-K3	Plan	.394	.472	.433	3.13	3.24	4.40	4.72	3.87	0.379
Inkling	No plan	.607	.451	.529	2.17	2.11	4.05	4.35	3.17	0.380
Inkling	Plan	.643	.500	.571	3.95	3.94	4.65	4.77	4.33	0.519
Gemma-31B	No plan	.655	.448	.552	2.37	2.28	4.02	4.95	3.40	0.415
Gemma-31B	Plan	**.769**	.344	.556	3.82	3.91	4.63	4.96	4.33	0.563
Gemma-31B	PWR	.757	.538	**.648**	**4.65**	**4.65**	**4.88**	**4.97**	**4.79**	**0.655**
Muse-30B	No plan	.495	.403	.449	2.06	2.11	2.82	1.46	2.11	0.259
Muse-30B	Plan	.363	.446	.405	2.76	2.90	3.12	1.55	2.58	0.302
Muse-30B	PWR	.490	**.576**	.533	4.49	4.53	4.05	2.05	3.78	0.469
Qwen-27B	No plan	.738	.178	.458	2.14	2.12	4.01	4.95	3.30	0.366
Qwen-27B	Plan	.649	.283	.466	3.67	3.65	4.52	4.96	4.20	0.442
Qwen-27B	PWR	.748	.333	.540	4.43	4.42	4.79	4.96	4.65	0.587

Table 13: Per-dataset Turn and subjective results on EPIC-KITCHENS.

		Turn			*Subjective*					
Model	**Config**	**IF1**	**SF1**	**Macro-F1**	**R**	**S**	**A**	**C**	**Avg**	**PQS**
GPT-5.6	No plan	.544	.211	.377	2.66	2.71	4.16	4.85	3.59	0.301
GPT-5.6	Plan	.573	.216	.395	3.67	3.77	4.54	4.85	4.21	0.374
Gemini-3.6	No plan	.510	.222	.366	2.69	2.66	3.96	4.81	3.53	0.284
Gemini-3.6	Plan	.587	.226	.407	4.13	4.20	4.64	4.93	4.48	0.411
Opus-5	No plan	.738	.257	.498	2.80	2.85	4.26	4.43	3.59	0.420
Opus-5	Plan	.701	.229	.465	4.30	4.27	4.76	4.64	4.49	0.501
Kimi-K3	No plan	.420	.195	.308	2.79	2.88	4.28	4.62	3.64	0.243
Kimi-K3	Plan	.487	.204	.345	4.14	4.24	4.72	4.86	4.49	0.337
Inkling	No plan	.743	.187	.465	2.50	2.56	4.15	4.17	3.35	0.376
Inkling	Plan	.745	.198	.471	4.20	4.24	4.72	4.71	4.47	0.536
Gemma-31B	No plan	.663	.211	.437	2.73	2.63	4.03	4.96	3.59	0.363
Gemma-31B	Plan	.875	**.272**	**.573**	4.38	4.47	4.81	**4.99**	4.66	**0.723**
Gemma-31B	PWR	.753	.243	.498	**4.76**	**4.80**	**4.90**	4.98	**4.86**	0.608
Muse-30B	No plan	.555	.156	.355	2.39	2.54	3.10	1.68	2.43	0.183
Muse-30B	Plan	.419	.153	.286	2.53	2.80	3.03	1.65	2.50	0.147
Muse-30B	PWR	.434	.219	.327	4.64	4.64	4.19	2.08	3.89	0.274
Qwen-27B	No plan	**.877**	.105	.491	2.40	2.48	4.00	4.84	3.43	0.481
Qwen-27B	Plan	.800	.176	.488	4.09	4.13	4.64	4.93	4.45	0.590
Qwen-27B	PWR	.864	.268	.566	4.27	4.31	4.70	4.96	4.56	0.690

Table 14: Per-dataset Turn and subjective results on HowTo100M.

		Turn			*Subjective*					
Model	**Config**	**IF1**	**SF1**	**Macro-F1**	**R**	**S**	**A**	**C**	**Avg**	**PQS**
GPT-5.6	No plan	.417	.532	.474	2.41	2.64	4.16	4.85	3.52	0.412
GPT-5.6	Plan	.450	.540	.495	3.09	3.34	4.36	4.79	3.90	0.442
Gemini-3.6	No plan	.352	.523	.437	2.37	2.47	4.02	4.94	3.45	0.393
Gemini-3.6	Plan	.427	.493	.460	3.52	3.66	4.46	4.95	4.15	0.419
Opus-5	No plan	.600	.495	.547	2.90	3.08	4.37	4.30	3.66	0.441
Opus-5	Plan	.582	.429	.506	4.21	4.26	4.76	4.48	4.43	0.460
Kimi-K3	No plan	.440	.582	.511	2.57	2.77	4.25	4.78	3.59	0.455
Kimi-K3	Plan	.386	.524	.455	3.46	3.70	4.52	4.76	4.11	0.425
Inkling	No plan	.604	.495	.550	2.51	2.68	4.14	4.55	3.47	0.427
Inkling	Plan	.615	.534	.575	3.99	4.07	4.72	4.75	4.38	0.525
Gemma-31B	No plan	.636	.505	.571	2.67	2.67	4.03	4.98	3.59	0.451
Gemma-31B	Plan	.671	.394	.533	3.66	3.82	4.56	4.99	4.26	0.493
Gemma-31B	PWR	**.713**	.487	**.600**	**4.97**	**4.97**	**4.99**	**5.00**	**4.98**	**0.630**
Muse-30B	No plan	.386	.542	.464	2.19	2.46	3.08	1.63	2.34	0.366
Muse-30B	Plan	.195	.596	.396	2.55	2.91	3.07	1.67	2.55	0.422
Muse-30B	PWR	.490	**.671**	.581	4.67	4.66	4.00	1.80	3.78	0.542
Qwen-27B	No plan	.650	.255	.452	2.73	2.77	4.06	4.96	3.63	0.372
Qwen-27B	Plan	.643	.296	.469	4.32	4.35	4.69	4.95	4.58	0.481
Qwen-27B	PWR	.707	.300	.504	4.82	4.85	4.93	**5.00**	4.90	0.574

Table 15: Per-dataset Turn and subjective results on HoloAssist.

		Turn			*Subjective*					
Model	**Config**	**IF1**	**SF1**	**Macro-F1**	**R**	**S**	**A**	**C**	**Avg**	**PQS**
GPT-5.6	No plan	.602	**.686**	.644	2.72	2.70	4.15	4.89	3.61	0.557
GPT-5.6	Plan	.586	.665	.626	3.73	3.67	4.52	4.88	4.20	0.577
Gemini-3.6	No plan	.590	.685	.638	2.78	2.62	3.92	4.72	3.51	0.548
Gemini-3.6	Plan	.640	.685	.662	4.08	4.03	4.52	4.78	4.35	**0.615**
Opus-5	No plan	.695	.680	**.688**	2.91	2.93	4.29	4.46	3.65	0.568
Opus-5	Plan	.659	.621	.640	4.13	4.07	4.69	4.71	4.40	0.566
Kimi-K3	No plan	.462	.648	.555	2.83	2.87	4.30	4.39	3.60	0.510
Kimi-K3	Plan	.529	.662	.596	4.13	4.14	4.70	4.72	4.42	0.569
Inkling	No plan	.646	.493	.569	2.59	2.69	4.19	4.21	3.42	0.433
Inkling	Plan	.643	.522	.583	3.97	3.97	4.66	4.71	4.33	0.520
Gemma-31B	No plan	.592	.577	.584	2.83	2.65	4.01	4.83	3.58	0.478
Gemma-31B	Plan	.699	.534	.617	4.48	4.43	4.76	4.93	4.65	0.597
Gemma-31B	PWR	.602	.594	.598	4.68	4.68	4.84	**4.95**	4.79	0.582
Muse-30B	No plan	.502	.396	.449	2.57	2.71	3.32	1.84	2.61	0.290
Muse-30B	Plan	.352	.539	.445	2.79	3.07	3.40	1.90	2.79	0.380
Muse-30B	PWR	.347	.670	.509	4.68	4.75	4.28	2.29	4.00	0.533
Qwen-27B	No plan	.486	.413	.450	2.52	2.57	4.06	4.82	3.49	0.354
Qwen-27B	Plan	.577	.404	.491	4.40	4.36	4.74	4.91	4.60	0.472
Qwen-27B	PWR	**.706**	.420	.563	**4.81**	**4.80**	**4.90**	4.94	**4.87**	0.594

Table 16: Per-dataset Turn and subjective results on EgoProactive.

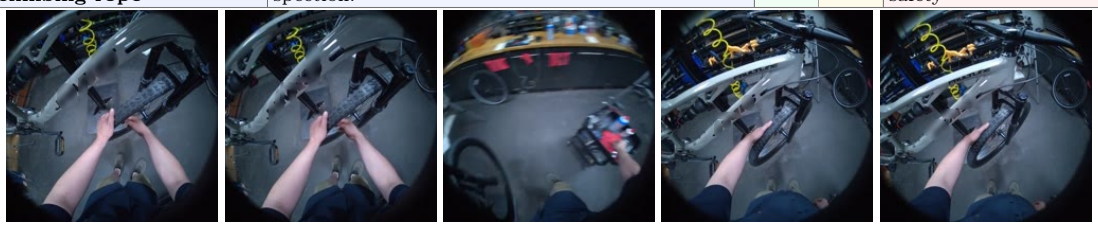
(A) **Silent Monitoring**: Step in Progress



Task: Ensure climbers are properly equipped and secured, and then ascend a rock face using proper climbing techniques and safety measures. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Inspect and put on climbing shoes

Plan (prev → curr)	Cue ($ silent$)	Golde	Ours	gemini ✗
Adjust and secure climbing harnesses and equipment → **Tie a secure knot on the climbing rope**	Hands threading/looping rope, tails loose and moving, knot open or asymmetrical, fingers adjusting segments, attention on manipulation rather than inspection.	[silent	[silent	Step 2 is complete. Now, Ascend the rock face while ensuring safety



Task: Remove the front wheel from the bicycle for potential repair or replacement by deflating the tire, removing the through axle, and pulling the wheel off the fork. *[Source: **EgoExo4D**]*

Dialog history:

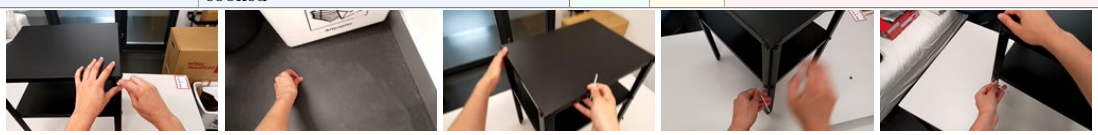
Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
→ **Unscrew the Presta valve to release air from the tire**	Hands rotating lock nut or depressing valve core, air visibly escaping, tire still firm, valve cap absent or partially removed.	[silent]	[silent]	You're skipping ahead—First, fully deflate the tire: remove the valve cap, unscrew the Presta valve tip a few turns, then press it in to release air until the tire is completely soft. Let me know when done and we'll move to the through axle.



Task: Prepare a meal by cracking eggs, mixing them with seasonings, cooking in a pan with butter, and cleaning up the workspace. *[Source: **EgoExo4D**]*

Dialog history: **1.** Sure! Let's get started. First, Open the kitchen cabinet and retrieve a spice or seasoning container. **2.** Light the gas stove and heat a pan. **3.** Crack eggs into a bowl and add seasonings. **4.** Add butter to the heated pan and stir it.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	claude ✗
Add butter to the heated pan and stir it. → **Pour the egg mixture into the pan and cook.**	Active stirring or manipulation of the eggs egg mixture is still being poured spatula is actively engaged with the eggs eggs are still visibly runny or not fully cooked	[silent]	[silent]	Good, it looks like you're pouring the egg mixture into the pan. Cook the eggs until they are fully set and no longer runny, stirring as needed. Once done, the next step will be to clean up the workspace.



Task: Disassemble a nightstand. *[Source: **HoloAssist**]*

Dialog history: **1.** Sure! Let's get started. First, Unscrew the hex socket heads of the first leg. **2.** Unscrew the hex socket heads of the second leg.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
Unscrew the hex socket heads of the third leg. → **Unscrew the hex socket heads of the fourth leg.**	Allen wrench actively turning a partially or fully seated screw, hand applying torque, leg still rigidly attached to frame.	[silent]	[silent]	You're on step 3 now: Finish unscrewing the hex screw(s) on the fourth leg counterclockwise until fully loose, then confirm the leg wiggles free. Next we'll remove the top shelf screws.

Figure 5. Qualitative examples: silent monitoring.



Task: Making Toast with Almond Butter and Honey. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Open the bread bag and take out two slices of bread. **2.** Place the two bread slices in the toaster oven, set the timer, and toast until lightly browned.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
Open the toaster oven, remove the toasted bread slices, and set them aside. → **Place the toasted slices on a cutting board.**	Hands actively placing toast slices, still in motion, not yet settled on the board.	[silent]	[silent]	Nice—both toasted slices are now flat on the cutting board. Next: open the almond butter jar, set the lid aside, and use a knife to scoop a small amount of almond butter.



Task: Navigate the car from the starting point on the highway to a gas station. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Drive forward on a two-lane highway with both hands on the steering wheel. **2.** Turn the steering wheel to the left to navigate a curve. **3.** Struggle with the steering wheel, making adjustments to its position. **4.** Drive on the highway, changing gears and steering smoothly.

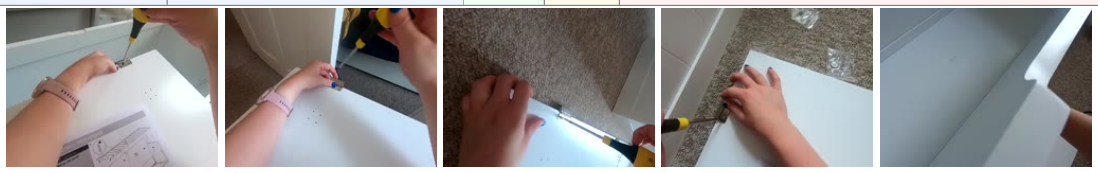
Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	qwen3 235B
Slow down and make adjustments to the steering wheel. → **Drive into a parking lot and turn the steering wheel clockwise.**	Hands gripping and rotating steering wheel clockwise, arms tense and in motion, motion blur or transitional scenery visible through windshield.	[silent]	[silent]	You have successfully navigated to the gas station. Park the car and turn off the engine.



Task: Drive a UTV through snowy roads, pull a blue sled through a snowy field, slide down a snowy hill on the sled, load the sled onto the UTV, and transport it to the final destination. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Drive the UTV through snowy residential areas and stop at a specific location. **2.** Exit the UTV and pull the blue sled through a snowy field.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
Slide down a snowy hill on the blue sled. → **Load the blue sled onto the UTV.**	Hands gripping or lifting sled edges, person bent at waist, sled partially tilted but still in contact with ground, vehicle stationary.	[silent]	[silent]	You're at the UTV with the blue sled visible on your right—before sliding, you need to pull the sled out into the snowy field and head toward the hill. Grab the sled's rope/handle and start walking it across the snow toward the slope.

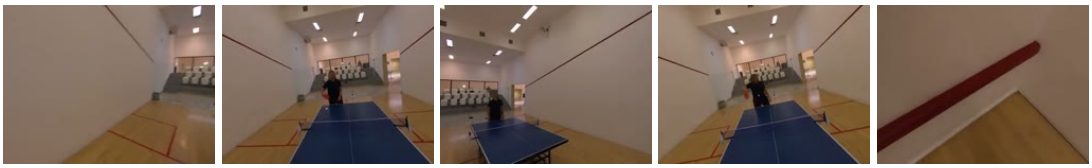


Task: Assemble a white piece of furniture by attaching hinges, metal brackets, and screws, and organize instructions and components to ensure all parts are properly secured and functional. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Screw a hinge onto a white piece of furniture using a screwdriver. **2.** Place a metal clip on the cabinet and tighten the screws. **3.** Attach metal attachments to the drawer. **4.** Assemble the sides of the white piece of furniture together. **5.** Screw screws into the corner of a white cabinet door. **6.** Attach a metal bracket to a white wardrobe door. **7.** Fix a door's hinge. **8.** Screw black brackets into the corners of a white object.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
Remove screws from a white surface. → **Attach metal brackets to the underside of a white piece of wood.**	Hand gripping bracket or screwdriver, bracket angled with gap from wood, screws partially inserted, bracket not yet aligned with mounting points.	[silent]	[silent]	It looks like you've removed the screws and set the panel down. Next, do Step 9: flip the white board so the underside is facing up, align the metal brackets with the pre-drilled holes, and screw them in flush (no gaps, screw heads level). If you don't see pilot holes, stop and check the instruction sheet to confirm you're on the correct side before drilling or forcing screws.

Figure 5. Silent monitoring (continued).



Task: Engage in a game of table tennis and then transition to a different activity by concluding the game and exploring the surroundings. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Serve the ball by bouncing it on the table before hitting it to the opponent's side.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	claude ✗
Hit the ball back and forth across the table to engage in a rally. → **Continue playing with consistent rallies.**	Paddle mid-swing in hitting motion, ball in flight between players, both in ready stance with eyes tracking the ball.	[silent]	[silent]	It looks like you've stopped playing and moved away from the table (no ball/paddle action visible). Go ahead and conclude the game (say "good game," stop the rally, and set the paddle/ball down), then transition to exploring by walking out of the court area and looking around the nearby indoor spaces (hallway/lounge/adjacent rooms).



Task: Clean the car's exterior, windows, and mirrors thoroughly, and maintain the surrounding area's cleanliness. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Clean the car's hood and headlights using a cloth held in one hand, moving it across the surface to remove dirt and grime. **2.** Clean the car's sides and windows using a larger towel, scrubbing from left to right. **3.** Dry the car using a green cloth, starting from the rear left quarter window, moving to the tail light, and then the rear windshield. **4.** Rinse a towel using a power washer, holding the green towel in one hand and the blue power washer in the other.

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	gpt ✗
Clean the ground using the power washer and towel. → **Clean the car's windows and mirrors using a towel.**	Hands wipe towel across glass, continuous towel contact, motion blur from rubbing or circular strokes, water droplets or soap film still present, person facing glass with focused gaze	[silent]	[silent]	You're holding the towel but it's not on the glass yet. Next, wipe the windows and mirrors: start at the driver-side window, then rear side window, then rear window, and finish with both side mirrors. Use firm overlapping strokes (top-to-bottom, then a quick left-to-right pass) until there are no droplets or streaks.



Task: Safely navigate through various traffic and weather conditions on the highway while making progress on the journey. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Accelerate and change lanes to overtake slow-moving vehicles on the highway **2.** Navigate through traffic by changing lanes and adjusting speed to maintain safe distance

Plan (prev → curr)	Cue ($ silent$)	Golden	Ours ✓	claude ✗
Drive through construction areas and cityscapes while adjusting speed and lane position → **Adjust speed and maintain safe distance while driving under snowy conditions**	Rapid closure on lead vehicle, or lead vehicle braking while ego vehicle not decelerating, or urgent steering corrections, or significant speed differential from surrounding traffic.	[silent]	[silent]	Good progress. You're currently driving under snowy/wet conditions approaching the city skyline. Maintain a safe following distance from the vehicle ahead and reduce speed as needed for the road conditions. Next step: Navigate through traffic under bridges and overpasses while maintaining safe distance.

(B) **Step Completion**: Proactive Interrupt



Task: Achieve the desired hair color and style for the clients by applying dye and product. *[Source: **Ego4D**]*

Dialog history: —

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gpt ✗
Using the blue brush, dip into the dye container and work the dye through the woman's hair section by section, using the free hand to pull and adjust for even coverage. → **Keep working dye through the woman's hair using the blue brush, re-dipping into the container as needed and using the free hand to section the hair for full coverage.**	Brush set aside and not in dye container, hands withdrawn, hair fully and evenly saturated, stylist posture shifting to next task.	Keep working dye through the woman's hair using the blue brush, re-dipping into the container as needed and using the free hand to section the hair for full coverage.	Continue applying dye to the woman's hair with the blue brush, ensuring thorough coverage by dipping the brush into the dye container multiple times and sectioning the hair with the other hand.	[silent]

Figure 6. Qualitative examples: step-completion interrupts.



Task: Knit a purple item using two knitting needles and purple yarn while sitting comfortably in a chair, taking breaks as necessary, and completing the item to the desired specifications. *[Source: **Ego4D**]*

Dialog history: —

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	qwen ✗
Settle into the white chair with legs crossed, pick up the knitting needles in both hands, and get ready to knit the purple cloth. → **Tighten the yarn on the knitting needle until it sits snug.**	Yarn taut and neatly wrapped on needle, hands stationary or withdrawing, fingers relaxed with no active tension adjustment.	Tighten the yarn on the knitting needle until it sits snug.	Adjust the yarn on the knitting needle to ensure it is tight.	[silent



Task: Achieve a collaborative and immersive discussion among group members using laptops and virtual reality headsets, and successfully conclude the discussion. *[Source: **Ego4D**]*

Dialog history: —

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gpt ✓
Jump into the collaborative discussion, working with laptops and VR headsets. → **Keep the collaborative project discussion going, working across laptops and VR headsets.**	Headsets coming off, laptops closing, participants leaning back and disengaging from devices, attention shifting away from the shared workspace.	Keep the collaborative project discussion going, working across laptops and VR headsets.	Continue the collaborative project discussion, utilizing laptops and virtual reality headsets.	Next step: assign roles and confirm everyone's setup. 1) Pick a facilitator and a note-taker/timekeeper. 2) Have each person open the shared doc/agenda on their laptop and confirm Wi-Fi/audio works. 3) If you're using VR, distribute headsets now and do a quick fit + pairing check (one person at a time).



Task: Prepare dough for baking by transforming it into uniform, shaped pieces, and positioning them for further processing or baking. *[Source: **Ego4D**]*

Dialog history: **1.** Sure! Let's get started. First, Retrieve a large wooden rolling pin from a bread baking rack and position it on a metal table. **2.** Move a large bread board from the rack to the metal table and rotate it to a suitable position. **3.** Cut a large lump of dough into smaller pieces using a bench scraper. **4.** Knead and shape the dough pieces into cylindrical forms and place them on a long wooden paddle.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	qwen ✗
Transfer the shaped dough pieces from the wooden paddle to a rack for further preparation or proofing. → **Mix additional dough with flour and water in a mixer.**	Dough uniformly combined in bowl, mixer stationary and off, hands withdrawn, ingredient containers empty or set aside.	Mix additional dough with flour and water in a mixer.	Add more dough to the flour mixture in the mixer.	[silent



Task: Making pan-seared halloumi with honey drizzle. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Prepare the halloumi: remove it from its packaging and slice the block into 1/2-inch thick pieces. **2.** Set up the cooking equipment: place a non-stick skillet on the stove, turn the burner to medium heat, and let the pan warm up. **3.** Eat a small snack (crackers) while the pan heats. (interruption, Body & Health) **4.** Add butter to the hot skillet, then place the halloumi slices in the pan without overlapping.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	qwen3 235B
Cook the halloumi: let it sizzle, flip the slices with a spatula, and continue cooking each side for 2-3 minutes until golden brown. → **Adjust the stove knob and pan position as needed to finish cooking the halloumi. (fix)**	Pan resting motionless on burner with knob off and utensil set aside, hands withdrawn as food browns evenly	Adjust the stove knob and pan position as needed to finish cooking the halloumi.	Adjust the stove knob and pan position as needed to finish cooking the halloumi.	The halloumi is now golden brown and fully cooked. Turn off the stove, then transfer the slices to a serving plate and arrange them neatly. Next, drizzle honey generously over the warm halloumi.

Figure 6. Step-completion interrupts (continued).

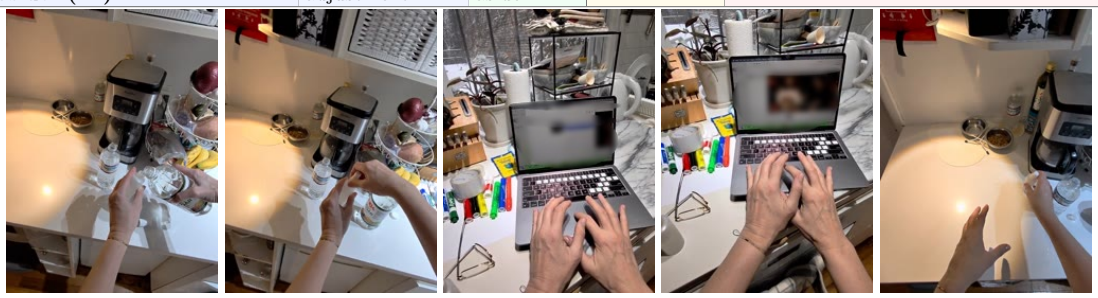
(C) **OOP Distractions**: Detection and Re-engagement



Task: How to pack a carry-on bag efficiently. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Lay out and organize clothing items on the bed, including shirts, pants, socks, and undergarments. **2.** Roll each clothing item tightly (shirts, pants, socks) to save space.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gpt ✓
(interruption, Object & Task-Management Interruptions) Pause to wipe the floor after noticing a spill, using a wipe from the package. → **Resume rolling the remaining clothing items and place them aside. (fix)**	All garments rolled and piled away, hands withdrawn, workspace cleared, rolls stable without adjustment.	**(fix)** Resume rolling the remaining clothing items and place them aside.	Pick back up rolling the remaining clothing items and set them aside.	You've paused rolling—resume by rolling the remaining flat garments (the patterned/striped shirt) into tight cylinders and move each finished roll into a separate 'done' pile off to one side. Once everything is rolled, tell me and we'll move to packing toiletries in a ziplock.



Task: How to remove wrinkles with a spray bottle. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Fill a clean spray bottle with distilled water and add a few drops of white vinegar. **2.** Shake the spray bottle to mix the solution.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	qwen ✗
(interruption, Media & Entertainment Interruptions) Switch from music to a podcast (or vice versa) while shaking the bottle. → **Resume shaking the spray bottle to ensure the solution is well mixed. (fix)**	Spray bottle still, upright and stable, grip relaxed, liquid settled, attention shifting to next action.	**(fix)** Pick back up shaking the spray bottle to make sure the solution is well mixed.	Resume shaking the spray bottle to ensure the solution is well mixed.	[silent]



Task: Making herbed cream cheese stuffed celery sticks. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Wash the celery stalks under running water and pat them dry with a paper towel. **2.** Trim the ends of the celery and cut the stalks into 4-inch sticks, then arrange the sticks on a plate.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	qwen3 235B
Switch the music or podcast playback on the smartphone. (interruption, Media & Entertainment Interruptions) → **Keep arranging the celery sticks on the plate until neatly set. (fix)**	All pieces arranged on serving plate, evenly spaced and aligned, hands withdrawn, layout settled and stable.	Keep arranging the celery sticks on the plate until neatly set.	Continue arranging the celery sticks neatly on the plate.	Open the cream cheese package, scoop the softened cream cheese into a bowl, chop fresh herbs, and add the herbs plus a pinch of salt to the bowl.

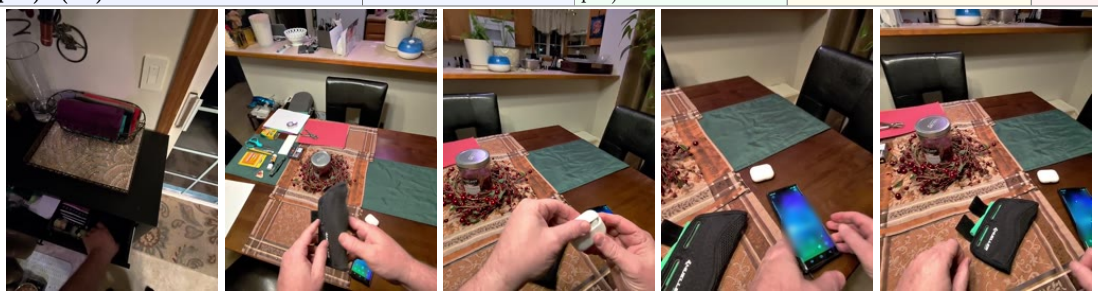
Figure 7. Qualitative examples: out-of-procedure distractions.



Task: How to organize under the bathroom sink. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Open the bathroom sink cabinet doors and remove all items, placing them on the floor or countertop.

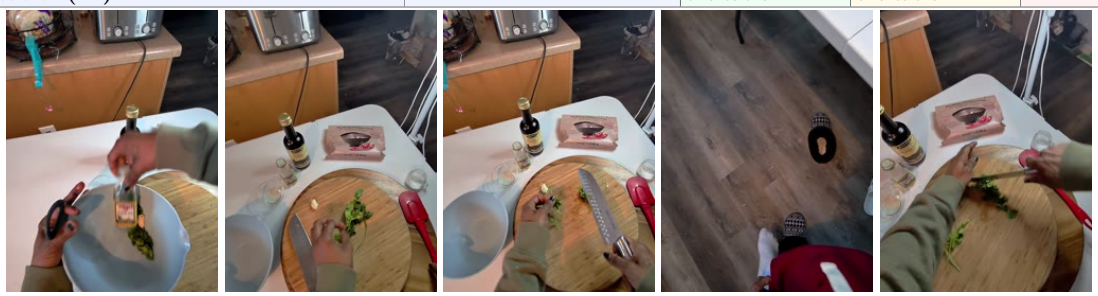
Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	claude ✗
(interruption, Information & Planning) Check the to-do list on the smartphone. → **Continue sorting the items under the sink, grouping similar items together (e.g., toiletries, cleaning supplies, extra toilet paper). (fix)**	Items sorted and grouped in separate containers, cabinet cleared, containers neatly arranged, hands withdrawn.	Continue sorting the items under the sink, grouping similar items together (e.g., toiletries, cleaning supplies, extra toilet paper).	Pick back up sorting the items under the sink, grouping similar ones together such as toiletries, cleaning supplies, and extra toilet paper.	[silent]



Task: How to apply a wrist support brace. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Open the drawer and retrieve the wrist support brace. **2.** Sit at a stable surface such as a table or chair, place the brace on the table, and handle the packaging while preparing.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	claude ✗
(interruption, Media & Entertainment Interruptions) Notice the audio playing from the phone speaker and switch the output to headphones. → **Return your attention to the task and position the wrist support brace directly in front of you on the table. (fix)**	Wrist support placed flat with opening facing up, centered in workspace, hands released and hovering nearby, brief pause indicating placement complete.	Return your attention to the task and position the wrist support brace directly in front of you on the table.	Return your attention to the task and position the wrist support brace directly in front of you on the table.	[silent]



Task: Making fresh chimichurri sauce. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Gather and prepare ingredients: retrieve fresh parsley, cilantro, garlic, red wine vinegar, olive oil, and red pepper flakes from the pantry/fridge and place them on the cutting board. **2.** Transfer the parsley and cilantro to the cutting board, separate leaves from stems, and roughly chop the herbs.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gpt ✓
Put on slippers to warm feet before continuing. (interruption, Sensory & Comfort Adjustments) → **Resume chopping the herbs until they are finely minced, then transfer the chopped herbs to the mixing bowl. (fix)**	Herbs finely and evenly chopped, knife set aside, hands shifting to gather or sweep herbs into mixing bowl, cutting board cleared.	Resume chopping the herbs until they are finely minced, then transfer the chopped herbs to the mixing bowl.	Pick back up chopping the herbs until finely minced, then sweep them into the mixing bowl.	Skip the slippers for now—focus on the sauce. Next step: strip the parsley/cilantro leaves from the thicker stems and pile the leaves on the cutting board. Then mince the garlic clove(s). After that, finely chop the herbs together (keep your fingertips tucked and use a rocking motion).

Figure 7. Out-of-procedure distractions (continued).

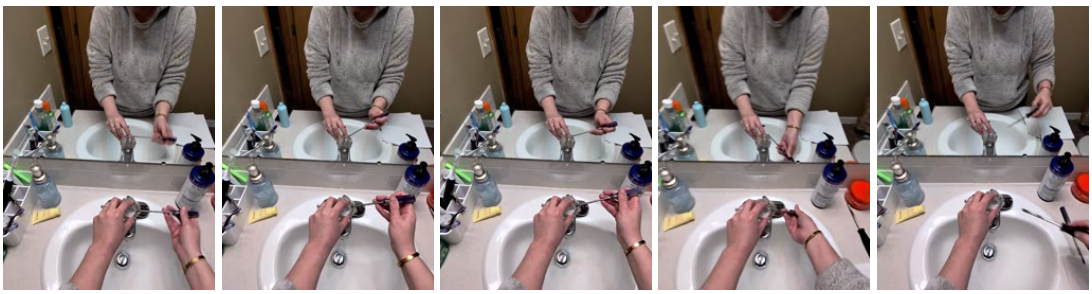


Task: How to remove wrinkles with a spray bottle. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Fill a clean spray bottle with distilled water and add a few drops of white vinegar. **2.** Shake the spray bottle to mix the solution. **3.** (interruption, Media & Entertainment Interruptions) Switch from music to a podcast (or vice versa) while shaking the bottle. **4. (fix)** Resume shaking the spray bottle to ensure the solution is well mixed. **5.** Lay the wrinkled shirt flat on a clean, dry surface such as a table or ironing board. **6.** Lightly mist the wrinkled areas of the shirt with the spray, keeping the bottle about 6 inches away.

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini
(interruption, Body & Health) Pause to adjust a bracelet (or watch/ring) that feels uncomfortable. → **Return to the shirt and continue misting any remaining wrinkled sections (fix).**	Spray bottle idle, no mist, hands making final broad smoothing passes across evenly damp fabric, hands relaxing near garment edges.	Return to the shirt and continue misting any remaining wrinkled sections.	Go back to the shirt and keep misting any sections still wrinkled.	[silent

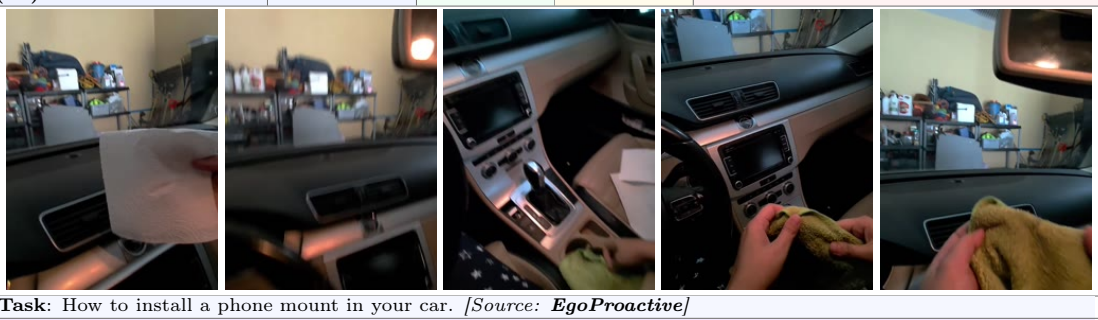
(D) OOP USER MISTAKES: Error detection and Correction



Task: Tightening loose faucet handle. *[Source: **EgoProactive**]*

Dialog history: —

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	claude ✓
Grab a screwdriver without checking if it matches the faucet handle screw head and begin adjusting the knob. (mistake, Tool Selection Error) → **Notice the screwdriver slipping, place it down, and switch to a properly fitting screwdriver before continuing. (fix)**	Ill-fitting tool set aside, hand now gripping a properly sized replacement, posture shifting to align new tool with fastener.	Notice the screwdriver slipping, place it down, and switch to a properly fitting screwdriver before continuing.	Set down the slipping screwdriver and swap to one that fits properly before continuing.	Good — I can see you've set the screwdriver down on the counter and are now working the faucet handle by hand. That's fine for loosening the handle knob. Now, Step 2: Remove the faucet handle — carefully pull or lift the handle off to expose the internal stem and packing nut underneath. Set the handle aside on the counter. Once exposed, we'll tighten the packing nut or stem to fix the looseness.



Task: How to install a phone mount in your car. *[Source: **EgoProactive**]*

Dialog history: —

Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✓
(mistake, Preparation/Setup Error) Wipe the chosen surface with a dry paper towel only, leaving oily residue that reduces adhesive grip. → **Use a slightly damp cloth to wipe the surface again, then dry it thoroughly with a clean paper towel before proceeding. (fix)**	Cloth set aside, surface dry and streak-free, hands free or holding next tool, attention shifting to next component.	Use a slightly damp cloth to wipe the surface again, then dry it thoroughly with a clean paper towel before proceeding.	Wipe the surface again with a slightly damp cloth, then dry it with a clean paper towel.	Next, peel the backing off the phone mount.

Figure 8. Qualitative examples: out-of-procedure user mistakes.

Task: How to inflate a basketball properly. *[Source: **EgoProactive**]*

Dialog history: **1.** Sure! Let's get started. First, Locate a basketball and a hand pump with a needle suitable for inflating sports balls.

INTERVAL 2



Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✗
Locate a basketball and a hand pump with a needle suitable for inflating sports balls. → **(mistake, Tool Selection Error) Grab a bicycle pump that has a different needle type and does not fit the basketball valve.**	Needle failing to seat in valve, size mismatch evident, no inflation visible, pumping stopped.	Tool Selection Error) Grab a bicycle pump that has a different needle type and does not fit the basketball valve.	Grab a bicycle pump that has a different needle type and does not fit the basketball valve.	[silent]

INTERVAL 3



Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✗
***(mistake,** Tool Selection Error) Grab a bicycle pump that has a different needle type and does not fit the basketball valve.* → **(fix) Set the bicycle pump aside and retrieve the correct hand pump with a universal or basketball-specific needle.**	Needle fully inserted with pump engaged, hands stable and ready to pump, hose taut confirming secure connection.	Set the bicycle pump aside and retrieve the correct hand pump with a universal or basketball-specific needle.	Put the bicycle pump aside and grab a hand pump with a universal or basketball-specific needle.	[silent]

INTERVAL 4



Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✗
***(fix)** Set the bicycle pump aside and retrieve the correct hand pump with a universal or basketball-specific needle.* → **Wipe the basketball's valve area with a dry cloth to remove any dirt or debris.**	Valve exposed and clean, hands and cloth withdrawn, person transitioning to inflation tool.	Wipe the basketball's valve area with a dry cloth to remove any dirt or debris.	Use a dry cloth to wipe any dirt or debris from the basketball's valve.	[silent]

INTERVAL 5



Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✗
Wipe the basketball's valve area with a dry cloth to remove any dirt or debris. → **(mistake, Positioning/Alignment Error) Insert the pump needle at a slight angle, causing air to leak out.**	Needle fully seated but tilted off-axis, hands stabilizing, hose stationary and connected.	Insert the pump needle at a slight angle, causing air to leak out.	Insert the pump needle at a slight angle, letting air leak out.	[silent]

INTERVAL 6



Plan (prev → curr)	Cue ($ interrupt$)	Golden	Ours ✓	gemini ✓
***(mistake,** Positioning/Alignment Error) Insert the pump needle at a slight angle, causing air to leak out.* → **(fix) Remove the needle, realign it straight into the valve, and reinsert it firmly to create a tight seal.**	Needle fully seated straight into valve, hose taut after hand release, pressure gauge rising steadily with no visible air leakage."	**(fix)** Remove the needle, realign it straight into the valve, and reinsert it firmly to create a tight seal.	Remove the needle, realign it straight into the valve, and reinsert it firmly to create a tight seal.	**(fix)** Remove the needle and re-insert it straight into the valve.

Figure 9. Ours vs gemini: out-of-procedure capture across a session.

Source domain	Evaluation datasets	Decision points
Arts and Crafts	EgoProactive	2,391
Chef	EgoProactive	2,926
Handyman	EgoProactive	2,241
Tutorial	EgoProactive	2,341
cooking	EPIC-KITCHENS	6,399
general	EgoExo4D, HoloAssist, HowTo100M	10,576

Table 17: Source-reported domain taxonomy in the frozen evaluation rows. Counts use the exact terminal evaluation UUID sets.

H Training Details

We train LoRA interaction models for Gemma 4 31B, Muse Glimmer 30B, and Qwen3.8-VL-27B on EgoExo4D, EPIC-KITCHENS, HowTo100M, and Ego4D; HoloAssist and EgoProactive are excluded from training. The four sources are sampled with fixed square-root-proportional weights of 0.207, 0.225, 0.288, and 0.280, respectively. Each example contains the current visual observation, the preceding dialogue, and, when applicable, the plan state. The loss is computed only on the final assistant turn so that supervision targets the current `interrupt`/`silent` decision and any generated guidance. All three models use LoRA rank 64, scaling factor 128, dropout 0.05, and no bias adaptation. Gemma adapts all linear modules. Muse adapts the language self-attention query, key, value, and output projections and keeps its vision tower frozen. Qwen adapts those attention projections together with the feed-forward gate, up, and down projections. We use AdamW with learning rate 2×10^{-5}, $\beta = (0.9, 0.95)$, weight decay 0.1, gradient clipping at 1.0, and cosine decay after a 3% warmup. Inputs are truncated to 32,768 tokens and at most 30 video frames. The exposure-matched schedules process up to 448,000 training examples, corresponding to 875 optimizer updates for Gemma and 7,000 for Muse and Qwen under their respective distributed batch configurations. Checkpoint selection uses a deterministic, source-stratified 10% validation holdout whose session or procedure groups do not overlap training. We average validation loss equally across the four sources, apply exponential smoothing with coefficient 0.5, and select the checkpoint with the lowest smoothed macro validation loss. Training stops early after 20 validation rounds without an improvement of at least 10^{-3}. PWR uses the selected interaction checkpoint conditioned on the output of its separately trained planner. The plan-conditioned control uses the matched plan context without changing the reported evaluation examples.

I Domain Taxonomy and Domain-Wise Results

Table 17 summarizes the source-reported domain labels in the frozen evaluation rows. The generic label is retained verbatim where a source does not provide a finer category; no semantic relabeling is inferred. Table 18 reports the corresponding learned-planner Turn results. The 14-domain count in the main paper refers to the normalized taxonomy across the full six-source training and evaluation collection. Table 17 instead reports only labels materialized in the five-source frozen evaluation manifest. Three sources expose only the generic label there, so the six rows below are evaluation-manifest groups rather than a replacement for the benchmark-wide 14-domain taxonomy.

J EgoProactive Collection Details

This section provides additional details on EgoProactive, introduced in §2.2. EgoProactive is the only source in Pro^{2}Bench collected specifically for proactive assistance. It contains continuous smart-glasses video, out-of-plan annotations at the procedure level, and recovery guidance.

J.1 Collection Protocol

Device and recording format. Each released procedure is recorded as one continuous video from consumer smart glasses. The first-person viewpoint matches the intended wearable setting and naturally includes head motion, changing illumination, partial

Domain	Model	Decision points	Interrupt F1	Silent F1	Macro-F1
Arts and Crafts	Gemma-31B	2,391	.646	.548	**.597**
Arts and Crafts	Muse-30B	2,391	.363	**.652**	.507
Arts and Crafts	Qwen-27B	2,391	**.720**	.317	.518
Chef	Gemma-31B	2,926	.532	.632	**.582**
Chef	Muse-30B	2,926	.299	**.678**	.488
Chef	Qwen-27B	2,926	**.662**	.501	.581
Handyman	Gemma-31B	2,241	.601	.548	**.574**
Handyman	Muse-30B	2,241	.353	**.657**	.505
Handyman	Qwen-27B	2,241	**.729**	.405	.567
Tutorial	Gemma-31B	2,341	.625	.577	**.601**
Tutorial	Muse-30B	2,341	.336	**.653**	.494
Tutorial	Qwen-27B	2,341	**.728**	.354	.541
cooking	Gemma-31B	6,399	**.757**	.538	**.648**
cooking	Muse-30B	6,399	.490	**.576**	.533
cooking	Qwen-27B	6,399	.748	.333	.540
general	Gemma-31B	10,576	.739	.401	**.570**
general	Muse-30B	10,576	.455	**.504**	.480
general	Qwen-27B	10,576	**.787**	.310	.548

Table 18: Domain-wise Turn results for PWR. Results use the frozen predictions and source-reported evaluation-domain labels. This table analyzes variation within PWR; aggregate matched comparisons are reported in Table 2.

occlusion, and variation in object placement. Participants perform the task at a natural pace without narrating unless speech occurs naturally.

Task assignment. Each recording follows a procedural script containing a goal, an ordered set of subgoals, and executable action steps. Participants may adapt incidental details, such as the particular object or ingredient used, provided that the task goal and procedural structure remain unchanged. The final release covers procedures with 3 to 15 reference steps, with a mean of 7.6 steps.

Recording requirements. All required actions must remain visible from start to finish in a single first-person recording. Written instructions may not appear in view, the camera may not be placed on a surface, and the principal objects and hand actions must remain sufficiently visible for temporal annotation. Videos with an unrecoverable mismatch between the script and the visible execution are not included in the released evaluation manifest.

J.1.1 Released Dataset Accounting

We distinguish *procedures*, *videos*, and *decision points*. A procedure is one recorded execution of a scripted task, represented by one continuous video. A decision point is an observation window at which the assistant must emit either `$interrupt$` with guidance or `$silent$`. One procedure therefore contributes multiple decision points. The median procedure contributes 14 decision points, with a range of 4 to 30. All 700 EgoProactive procedures are held out from planner and interaction-model training in the experiments reported in this paper. The released evaluation manifest does not expose participant identifiers or demographic attributes, so we do not report participant-level counts or demographic distributions.

Activity domains. The four domain labels in the final evaluation manifest are shown in Table 20.

J.2 Out-of-Plan Scenario Design

Of the 700 procedures, 247 contain a scripted departure from the expected execution. Together, these procedures contribute 3,810 of the 9,899 decision points. This procedure-level membership is the canonical EgoProactive-OOP slice used throughout the paper.

Table 19: EgoProactive accounting in the frozen evaluation manifest. OOP decision points are all evaluated points belonging to OOP procedures, not a count of deviation onsets.

Quantity	**Count**
Released procedures	700
Released videos	700
Video duration	31.4 hours
Activity domains	4
Evaluation decision points	9,899
Interrupt	5,338
Silent	4,561
OOP procedures	247
Decision points in OOP procedures	3,810

Table 20: EgoProactive coverage by the domain labels in the frozen evaluation manifest.

Domain	**Procedures**	**Decision points**
Chef	174	2,926
Arts and Crafts	175	2,391
Handyman	175	2,241
Tutorial	176	2,341
Total	**700**	**9,899**

It must not be interpreted as 3,810 distinct mistakes or deviation onsets: each OOP procedure also contains ordinary progress, silence, deviation, redirection, and recovery points. The scripts cover departures such as skipped or reordered actions, incorrect tools or techniques, and attention-shifting interruptions. Each script is designed around a continuous sequence

$$\text{correct lead-in} \rightarrow \text{departure} \rightarrow \text{recovery} \rightarrow \text{correct end state.}$$

The reference plan and guidance identify the corrective action needed to return to the task. For annotation, a departure begins at the first frame at which the observed action can be distinguished from the expected execution. Recovery points identify the action that restores progress toward the task goal.

OOP target audit. We re-audit every OOP interrupt target against the frames and dialogue available at that decision point. A valid target must correct the observed departure or re-engage the user with the procedure. Targets that instead instruct the user to perform the scripted error, reveal a later recovery before the departure is observable, or conflict with the visible state are corrected or excluded before evaluation. Annotation tags and future mistake or recovery text are retained only as scoring metadata and are removed from all model inputs as described in Appendix C.1.

OOP annotation categories. The final manifest assigns one stable category to each procedure. Canonical procedures are marked as ordinary executions; OOP procedures are categorized by whether the record contains an explicit mistake and fix pair, an inferred fix, a fix without a separately retained mistake point, or an interruption with an explicit redirect. This procedure-level category defines the OOP slice. Turn-level fields identify the particular mistake, deviation, interruption, or recovery points inside that procedure.

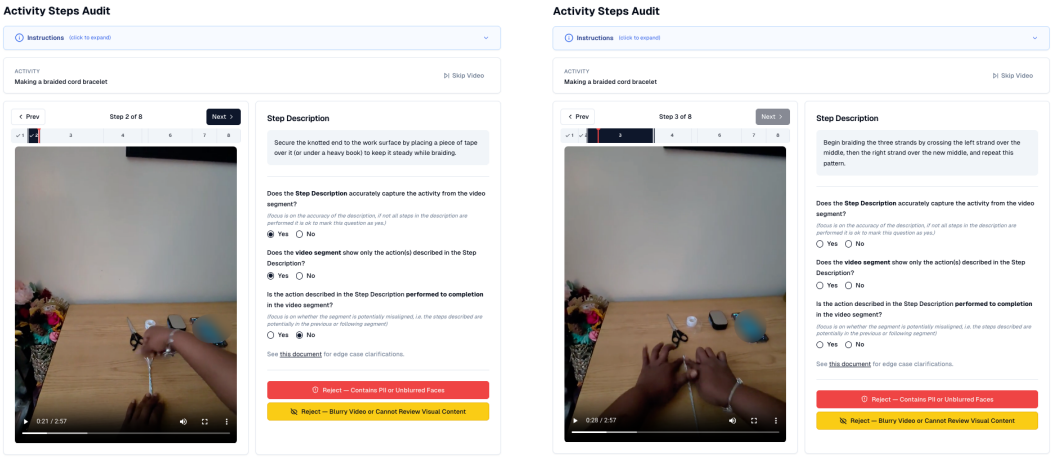
J.3 Annotation and Human Review

The annotation pipeline first proposes procedural steps, temporal boundaries, decision labels, guidance, and visual cues. These are candidates rather than ground truth. EgoProactive then undergoes a two-phase human review that checks the visible execution against the proposed step sequence.

Table 21: Structured OOP support in EgoProactive relative to established egocentric procedural sources. "Partial" denotes naturally occurring instructor corrections without paired onset and recovery annotations.

Feature	EP	HA	EK	Ego4D
Scripted departures	✓	✗	✗	✗
In-video recovery	✓	Partial	✗	✗
Departure onset	✓	✗	✗	✗
Paired departure–recovery guidance	✓	✗	✗	✗

EP = EgoProactive, HA = HoloAssist, and EK = EPIC-KITCHENS.



(a) The described securing action continues beyond the proposed segment, so the step fails the completeness check.

(b) The next segment begins with the preceding action, confirming the boundary error.

Figure 10: **Phase 1 audit.** Adjacent segments are inspected together to identify textual and temporal inconsistencies before correction.

Phase 1: audit. For every candidate step, annotators inspect the corresponding video segment and answer three diagnostic questions: (1) whether the description matches the visible action, (2) whether the segment contains only the described action, and (3) whether the action completes within the proposed boundary. A negative answer routes the video to correction. Figure 10 shows a boundary error detected by comparing adjacent steps.

Phase 2: correction. Annotators adjust temporal boundaries at 0.1-second resolution and edit step descriptions until the three checks pass. Boundary changes propagate to the adjacent segment to preserve a contiguous timeline. The interface rejects gaps and overlaps and presents a final preview of all edits. If a coherent step decomposition cannot be recovered, the video is excluded rather than forced into the release. Figures 11– 13 illustrate this workflow.

Review coverage and agreement. Every released EgoProactive procedure passes the audit above, and every flagged procedure passes a subsequent correction review before release.

Label provenance. Temporal boundaries and step text begin as machine proposals and may be edited during human review. OOP onset and recovery structure are attached to the procedure annotation. The interrupt or silent target is then derived from the verified event type, and every interrupt point carries a reference guidance utterance. Visual completion and incompletion cues are retained when supported by the reviewed

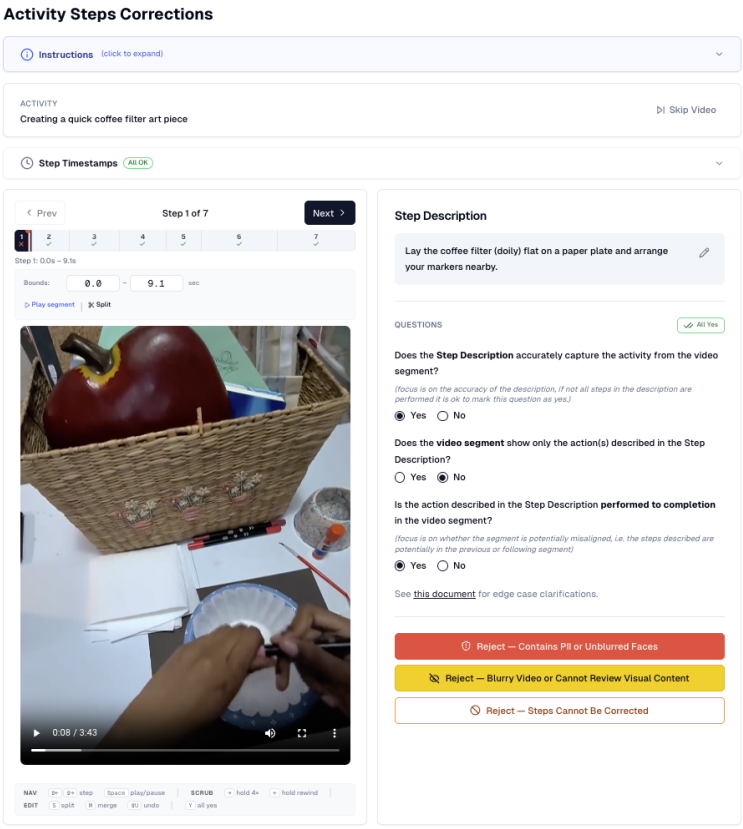


Figure 11: **Correction, initial state.** A segment is flagged because it contains action belonging to an adjacent step.

observation. Although the generation prompt requests multiple cues, the prompt itself is not treated as evidence that every released row has a fixed minimum number.

J.4 Quality Assurance and Release Scope

Quality assurance checks script visibility, first-person viewpoint, visual clarity, goal completion, step coverage, and temporal consistency. OOP procedures receive additional checks that the departure is visible, the recovery occurs in the same continuous recording, and both can be localized in time. The counts in Table 19 describe the accepted frozen evaluation manifest.

Consent and data handling. Participants consented to the research collection and dataset use before recording. Collection instructions prohibited written scripts in view and restricted recording to suitable private task settings. Participants were asked to minimize incidental bystander capture, and the released metadata omits participant identifiers and demographic attributes. Access is governed by the dataset release terms linked from the main paper. The released decision-point metadata contains no participant identity or demographic attributes. This protects contributor privacy but prevents a participant-level fairness analysis from the public metadata alone. More broadly, proactive assistance can reduce procedural errors, but poorly timed or incorrect guidance can distract users or encourage over-reliance. Systems deployed in safety-sensitive settings therefore require additional calibration, human-factors evaluation, and application-specific safeguards beyond the benchmark evaluation reported here.

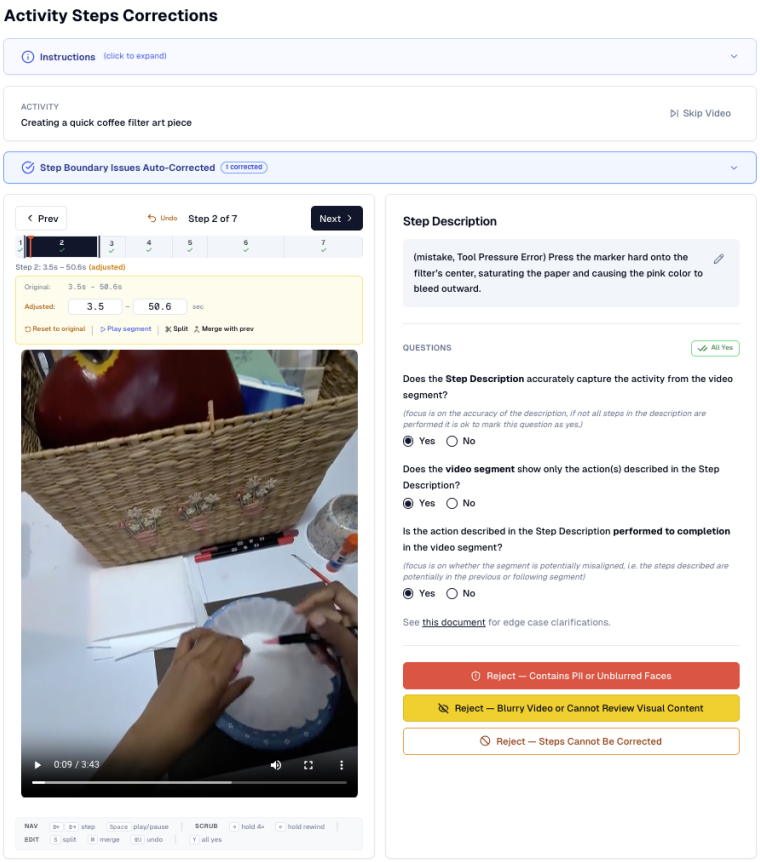


Figure 12: **Correction, boundary edit.** The annotator moves the boundary, replays the segment, and verifies the corrected description and timing.

K Step Alignment Pipeline Details

This section describes the automated step-alignment component used to generate candidate procedural annotations. It builds on the Action100M methodology (Chen et al., 2026). The component does not directly produce the final benchmark. Its candidates enter the source-specific normalization and iterative verification process in §2.3; EgoProactive candidates additionally undergo the human audit and correction described in §J.3.

K.1 Stage 1: Visual Feature Extraction

We encode the video with V-JEPA 2 using a ViT-Giant encoder at 384-pixel resolution. Frames are temporally subsampled and processed in overlapping 64-frame windows with stride 32. A frame covered by more than one window is represented by the mean of its window-specific embeddings:

$$\mathbf{e}_t = \frac{1}{|\mathcal{W}(t)|} \sum_{w \in \mathcal{W}(t)} \mathbf{e}_t^{(w)}, \tag{6}$$

where $\mathcal{W}(t)$ is the set of windows containing frame t. The result is a temporally ordered embedding matrix $\mathbf{F} \in \mathbb{R}^{T \times d}$.

K.2 Stage 2: Temporally Constrained Clustering

We apply hierarchical agglomerative clustering with Ward linkage. A tridiagonal connectivity matrix permits merges only between temporally adjacent clusters, so every node of the resulting dendrogram represents a contiguous time span. The merge distance records the visual change required to join neighboring spans. We retain the full dendro-

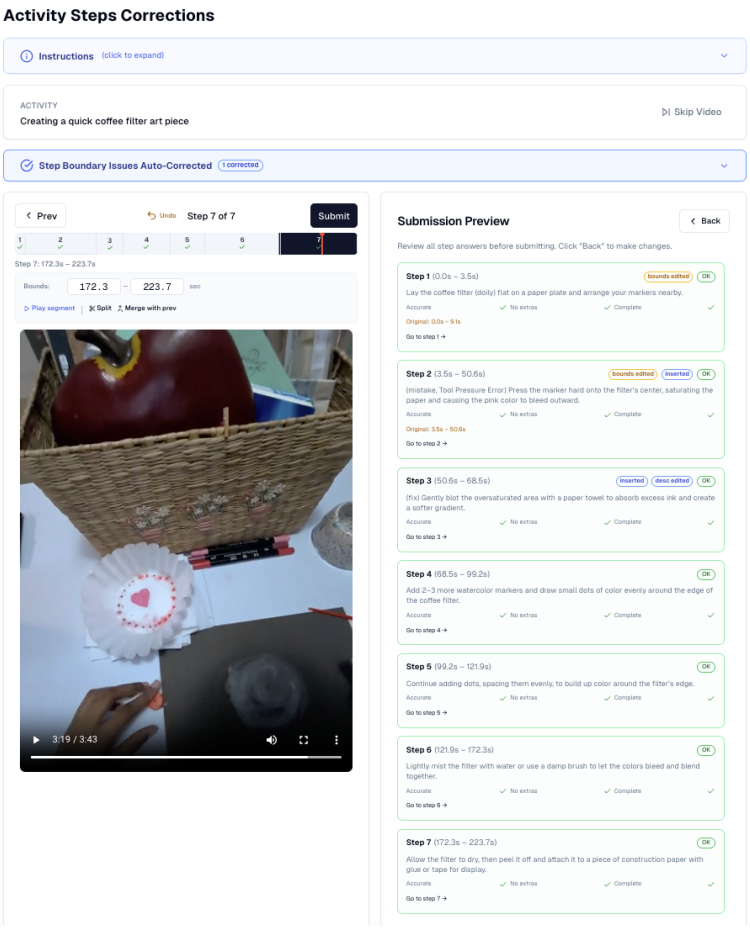


Figure 13: **Correction, final review.** The preview exposes every boundary, insertion, and text edit before submission.

gram because later stages may need to refine a coarse segment without recomputing the embeddings.

K.3 Stage 3: Dendrogram Partitioning

The dendrogram is cut by binary search over its merge-distance threshold. When a source provides an expected step count N, the initial partition targets $\lfloor 1.3N \rfloor$ segments. When no step count is available, the target is estimated from video duration. This deliberate over-segmentation favors recovering action boundaries: a later grouping stage can merge adjacent segments, whereas it cannot reliably recover a transition hidden inside one coarse segment.

K.4 Stage 4: Visual Captioning and Recursive Refinement

For each segment, Qwen3-VL-235B receives up to 16 uniformly spaced frames and describes the visible hand actions, tools, and manipulated objects. If no task action is visible, it returns "No active task." If the response contains multiple distinct actions, the segment is replaced by its two dendrogram children and each child is captioned independently. Recursion stops when the caption describes one coherent activity or the dendrogram reaches a leaf. The captioning prompt is:

```
These frames are from an egocentric (first-person) video of someone
performing a hands-on task.  Describe what you see, with an emphasis on
any action the person is performing.  If no action is visible, say ‘‘No
active task.’’  If there are multiple distinct actions involving a
```

```
different tool, object, or motion, list each on a separate line.  Output
only the activity description or descriptions, one per line.
```

K.5 Stage 5: Procedural Step Grouping

Visual captions are observations of local activity, while a procedural step may span several neighboring segments. A text-only Llama 4 Maverick call receives the chronological captions, their timestamps, the activity goal, and any source-provided reference steps. It merges adjacent observations that describe one action, absorbs inactive transition segments, maps generic object descriptions to task-specific terminology, and rewrites the result as concise instructions. Candidate mistake, interruption, and recovery segments remain separate rather than being merged into canonical steps. These labels support annotation and scoring only; they are stripped from model inputs by the sanitization step in Appendix C.1. The grouping call returns a JSON array of objects with `caption`, `start_ts`, and `end_ts`. The prompt constrains the output to chronological, non-overlapping groups and asks for a number of groups close to the source step count when one exists.

K.6 Stage 6: Reference-Step Alignment

A second text-only call assigns each grouped segment to the best-matching reference step. It considers both semantic similarity and the segment's relative position in the video. Several consecutive segments may map to the same reference step, and unmatched transitions receive index 0. The output is a JSON array containing exactly one integer for each grouped segment. The alignment prompt is:

```
Given numbered procedural steps and chronological video-segment captions,
assign each segment to the best-matching step.  Use both semantic match
and temporal position.  Use 0 when no step matches.  Multiple
consecutive segments may map to one step, and not every step must be
matched.  Return only a JSON array of integers with exactly one entry
per segment.
```

K.7 Stage 7: Automated Candidate Filtering

We first verify that the generated segments form a valid temporal sequence and cover the intended video span. A text-only Llama 4 Maverick call then scores the complete caption sequence for coverage of distinct task phases, logical order, and instructional relevance, each on a 1 to 10 scale. Sequences with an overall score below 5 are removed before downstream annotation. The scoring prompt asks whether a novice with the required materials could complete the activity by following the captions. Coverage rewards distinct hands-on phases rather than repeated descriptions of one phase; order measures procedural coherence; and relevance measures specificity and actionability. The call returns a JSON object containing the overall, coverage, order, and relevance scores together with a short explanation and issue list.

K.8 From Candidates to the Released Benchmark

Stage 7 is an automated prefilter, not the final annotation decision. The remaining candidates pass through four additional layers:

(a) **Source-specific normalization** maps source metadata, timestamps, and task identifiers into the common benchmark schema.
(b) **Field-level verification and correction** checks steps, boundaries, decisions, guidance, and visual cues against the corresponding frames. A separate correction pass revises only the flagged fields, as described in §2.3.
(c) **EgoProactive human review** audits textual fidelity, temporal specificity, and step completeness, and corrects or rejects inconsistent videos using the protocol in §J.3.
(d) **Canonicalization and release checks** assign stable procedure and decision-point identities, remove duplicate or malformed rows, enforce source-specific split constraints, and freeze the manifests used by every matched evaluation condition. This stage also removes annotation-only tags, targets, reference responses, and future deviation or recovery information from model-facing fields.

This separation is important: the seven-stage alignment component proposes a structured transcript, while the later review and release stages determine which annotations enter Pro^{2}Bench. The statistics in Tables 1, 19, and 7 are computed from the final frozen

manifests, not from intermediate alignment output. EgoProactive receives the additional human review in Appendix J.3. The five established sources use the source-specific automated verification and correction process above. The alignment pipeline does not define the evaluation sample. The exact decision-point inclusion rule and per-source denominators are given in Appendix C.5.

K.9 Reproducibility and Limitations

Each final decision point retains stable source, video, procedure, and row identifiers together with its temporal interval and annotation provenance. The prompt contracts used by the model-dependent stages are specified above, and aggregate tables are regenerated deterministically from the frozen manifests. The pipeline has four principal limitations. First, ambiguous or occluded frames can cause visual-captioning errors. Second, a fixed 16-frame captioning budget may undersample long segments. Third, independent processing does not enforce identical language across different videos of the same task. Fourth, text-only alignment can propagate a captioning error into the selected reference step. The iterative verification, EgoProactive human review, and release checks reduce these failure modes but do not eliminate them.