SCHEMA: SCALABLE HARNESS-GUIDED TEMPORAL QUALITY METADATA CONSTRUCTION FOR ROBOTIC MANIPULATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic manipulation trajectories contain temporally heterogeneous behavior, including productive progress, inefficient attempts, recoveries, and failures, yet standard Vision-Language-Action (VLA) model training largely treats these segments alike. We introduce **SCHEMA**, a scalable harness-guided framework that uses a Vision-Language Model (VLM) to construct sparse temporal quality metadata through iterative evidence seeking, contextual quality assessment, and boundary refinement, enabling better use of trajectory quality during VLA training. We further introduce **RTQ-Bench** (Robot Temporal Quality Benchmark), built from multiple public manipulation datasets with human-annotated quality intervals. On RTQ-Bench, SCHEMA substantially improves temporal quality localization over single-pass inference with the same backbone and performs competitively against diverse VLM- and score-based baselines. Pretraining on approximately 1,500 hours of real-world manipulation data with SCHEMA-guided sampling improves downstream VLA performance over uniform sampling, with the training data, model initialization, action-learning objective, and optimization schedule held fixed. Across four real-world tasks, SCHEMA-generated metadata improves both sampling-based and advantage-conditioned learning, with larger gains when failed policy rollouts are included.

1 INTRODUCTION

Vision-language-action (VLA) models combine pretrained vision-language representations with action learning (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024), using robot data across tasks, environments, and embodiments (O'Neill et al., 2024; Khazatsky et al., 2024). Yet successful trajectories can contain ineffective attempts, while failed episodes can contain productive actions or recoveries (Zhang et al., 2025; Zhai et al., 2026). Local quality is therefore distinct from episode outcome. Behavioral cloning without quality weighting or conditioning does not explicitly distinguish these regions, whereas success-only filtering can discard useful behavior. This motivates identifying useful temporal regions for supervision, rather than filtering solely by episode outcome.

Quality-aware robot learning spans data-mixture optimization and demonstration selection (Hejna et al., 2024; 2025), progress and reward estimation (Ma et al., 2025; Chen et al., 2026; Liang et al., 2026), and transition-level curation and rollout segmentation (Zhang et al., 2025; Zhai et al., 2026). Building on these advances, we construct sparse, temporally grounded quality metadata that can be evaluated independently of policy training and reused across learning interfaces. We seek explicit quality intervals with supporting explanations and evidence, without requiring dense scores or a complete episode partition. This requires contextual interpretation and precise boundaries: an ineffective grasp and the productive recovery that follows should be separated, even within a successful episode.

Vision-language models (VLMs) support robot reasoning (Sermanet et al., 2024; Zawalski et al., 2025) and trajectory annotation (Blank et al., 2025; Kim et al., 2026). Offline annotation allows these models to revisit earlier and later observations and investigate uncertain intervals without slowing robot execution. Inspired by adaptive evidence acquisition in video understanding (Wang et al.,

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.

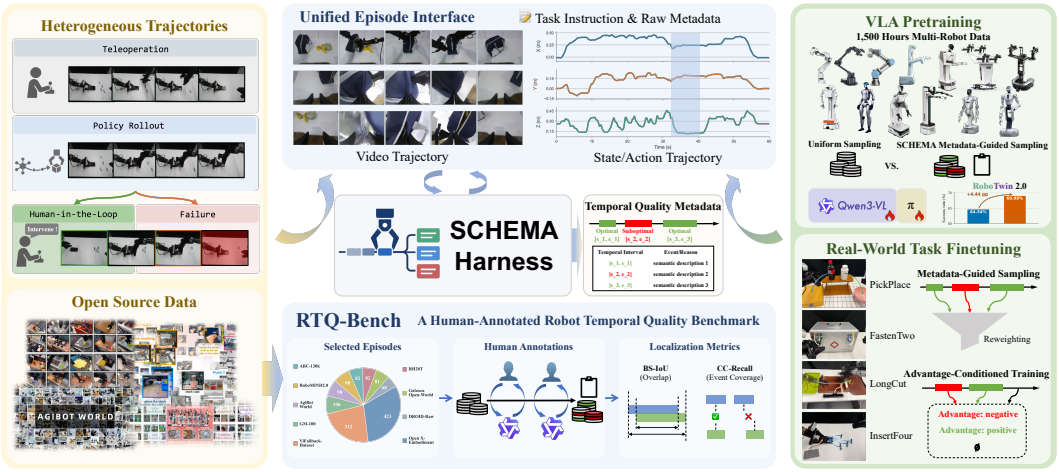


Figure 1: Overview of SCHEMA. Harness-guided VLM reasoning transforms heterogeneous robot trajectories into sparse optimal and suboptimal intervals with explanations and supporting evidence. The metadata supports VLA pretraining and real-world policy learning, while RTQ-Bench independently evaluates temporal localization.

2024; 2025), we organize annotation around targeted temporal inspection rather than a single pass over sparse observations.

We introduce **SCHEMA**, a scalable harness-guided framework for temporal quality metadata construction (Figure 1). A stateful harness manages tool execution, accumulated evidence, and inference budgets for a VLM controller. Through a unified episode interface, the controller inspects a global overview and acquires local evidence with surrounding temporal context. A subsequent refinement stage revisits candidate boundaries using denser observations. SCHEMA exports optimal and suboptimal intervals with explanations and evidence references, leaving uncovered frames unlabeled. Here, optimal denotes visibly productive, task-relevant behavior under a shared rubric, not globally optimal control.

The metadata supports two training interfaces. **Metadata-guided sampling** upweights optimal regions and downweights suboptimal ones without discarding them, leaving unlabeled regions at their base weight. **Advantage-conditioned training** reuses RECAP's interface (Physical Intelligence et al., 2025a), substituting metadata-derived quality conditions for value-derived indicators. Both interfaces use only the quality labels and temporal extents from this metadata; explanations support inspection rather than becoming prediction targets, and the annotation harness is unnecessary during deployment.

We evaluate localization accuracy and downstream learning utility. RTQ-Bench contains 1,350 expert-annotated and reviewed episodes from nine public sources. SCHEMA improves localization over single-pass inference with the same backbone. On approximately 1,500 hours of robot data, metadata-guided pretraining improves downstream performance across 50 RoboTwin 2.0 tasks (Chen et al., 2025) under matched settings. Across four real-world tasks, SCHEMA-guided sampling and SCHEMA-conditioned training improve average success by 6 and 12 percentage points, respectively, and benefit more from adding failed experience than their respective baselines.

Our contributions are threefold:

1. We introduce **SCHEMA**, a scalable framework that constructs sparse temporal quality metadata from offline robot trajectories through harness-guided evidence seeking, contextual assessment, and boundary refinement.

2. We introduce **RTQ-Bench**, a human-annotated benchmark for optimal and suboptimal interval localization, and demonstrate strong localization performance against diverse baselines.

3. We demonstrate the downstream utility of SCHEMA through matched VLA pretraining comparisons on RoboTwin 2.0 and sampling-based and advantage-conditioned learning across four real-world tasks, including improved reuse of failed rollouts.

2 Related Work

Quality-aware robot data curation. Re-Mix optimizes domain mixture weights (Hejna et al., 2024), while DemInf scores demonstrations using state–action mutual information (Hejna et al., 2025). SCIZOR removes suboptimal and redundant transitions using learned progress and state–action similarity (Zhang et al., 2025); HELP's VLAC-CUT segments autonomous rollouts into progress-making, idle, failure-inducing, and recovery regions for iterative post-training (Zhai et al., 2026). RoboCOIN's CoRobot pipeline combines rule-based trajectory validation with hierarchical annotation (Wu et al., 2025). SCHEMA constructs sparse, evidence-linked temporal quality metadata that can be reused across sampling and conditioning.

Progress, reward, and advantage supervision. GVL estimates progress by ordering shuffled observations (Ma et al., 2025), while SARM jointly models task stages and fine-grained progress (Chen et al., 2026). Robometer combines intra-trajectory progress supervision with inter-trajectory comparisons (Liang et al., 2026); Robo-Dopamine 2.0 incorporates history and out-of-distribution awareness into process rewards (Xu et al., 2026), and ProcVLM grounds progress rewards in procedural annotations and reasoning about remaining actions (Feng et al., 2026). RoboReward provides a reward benchmark and general-purpose VLM reward models (Lee et al., 2026). SCHEMA directly annotates contextual quality events rather than deriving quality intervals from a dense progress or value curve. SCHEMA's labels can also replace value-derived indicators in RECAP's advantage-conditioned interface (Physical Intelligence et al., 2025a).

Robot trajectory annotation and evaluation. NILS segments interactions and assigns language instructions (Blank et al., 2025); DeMiAn produces dense, multi-aspect language annotations (Kim et al., 2026). AHA detects and explains manipulation failures (Duan et al., 2025), while ViFailback couples failure diagnosis with textual and visual correction guidance (Zeng et al., 2026). ECoT predicts embodied reasoning before actions (Zawalski et al., 2025); ZR-0 uses dense reasoning supervision with optional reasoning generation at deployment (Li et al., 2026). SCHEMA uses reasoning for offline annotation rather than as an additional policy-prediction target. OpenGVL evaluates visual task progress (Budzianowski et al., 2025), while RoboProcessBench assesses process understanding through diagnostic questions (Xia et al., 2026). RTQ-Bench instead evaluates sparse optimal and suboptimal intervals against human references using temporal-overlap and event-recovery metrics.

Agentic video reasoning and evidence acquisition. Agentic video understanding replaces fixed, one-pass observation with adaptive evidence acquisition. VideoAgent iteratively gathers visual evidence through an LLM-controlled tool loop (Wang et al., 2024), while VideoTree constructs query-adaptive hierarchical representations for long-video reasoning (Wang et al., 2025). SCHEMA adapts this evidence-acquisition principle to offline robot quality annotation. Its budgeted harness maintains findings, revisits candidate intervals with surrounding context, and refines event boundaries without training a separate query policy. The output is reusable temporal quality metadata rather than an answer to an individual video query.

3 SCHEMA

Overview. Figure 2 summarizes SCHEMA. A VLM controller acquires timestamped evidence through a unified episode interface and a budgeted, stateful tool loop. Selected quality events are normalized and boundary-refined, then exported as metadata for sampling-based and advantage-conditioned VLA learning.

3.1 Problem Formulation and Temporal Quality Metadata

Let $\mathcal{D} = \{(\tau^{(n)}, \ell^{(n)})\}_{n=1}^{N}$ denote offline robot trajectories paired with task instructions. An episode provides visual observations and available action/state streams. For an episode with T frames, we

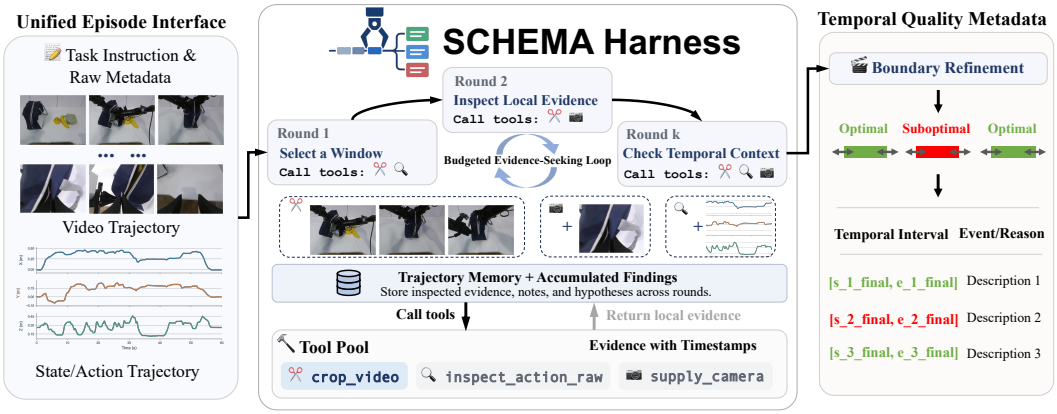


Figure 2: **SCHEMA harness for temporal quality metadata construction.** A stateful VLM controller acquires local evidence through budgeted tool calls, assesses quality using temporal context, and refines event boundaries before exporting sparse quality metadata with explanations and evidence references.

use original frame indices $t \in \{0, \ldots, T-1\}$ and inclusive event intervals. Given (τ, ℓ), SCHEMA produces

$$\mathcal{M}(\tau, \ell) = \{M_i\}_{i=1}^{K_e}, \qquad M_i = \big(y_i, [s_i, e_i], r_i, \mathcal{E}_i, c_i\big), \tag{1}$$

where $y_i \in \{+, -\}$ denotes optimal or suboptimal behavior, $[s_i, e_i]$ specifies its temporal extent, r_i explains the event, $\mathcal{E}_i$ records visual evidence references, and c_i is model-reported confidence, not a calibrated probability. Under our annotation rubric, optimal denotes visibly productive, task-relevant behavior rather than globally optimal control; suboptimal denotes visually supported inefficiency, regression, or failure. The K_e retained events are mutually non-overlapping and need not cover the episode; K_e may be zero. Uncovered frames remain unlabeled, not negative or zero-progress.

3.2 Harness-Guided Evidence Seeking and Quality Assessment

Dataset-specific adapters expose the task instruction, primary video, frame rate, and available auxiliary streams through a unified episode interface while preserving their original temporal coordinates. The interface standardizes observation access rather than robot action spaces. Auxiliary numerical signals are exposed only when available and valid under the corresponding dataset adapter.

SCHEMA begins from a sparse global overview and lets the VLM actively acquire local evidence. A tool request is written as

$$u_k = (I_k, m_k, \eta_k), \qquad m_k \in \mathcal{T}_\tau, \tag{2}$$

where I_k is the target temporal interval, m_k is a tool from the episode-specific tool set $\mathcal{T}_\tau$, and η_k contains tool-specific parameters. The primary operation, `crop_video`, samples observations around I_k while preserving their original frame indices and timestamps. When available, auxiliary tools expose bounded action/state observations or additional camera views. Tool outputs are treated only as evidence: event labels are instantiated through explicit controller finalization rather than directly from tool signals.

During evidence seeking, the harness maintains a compact state b containing trajectory memory, current findings, provenance, and execution counters, and supplies relevant context to the VLM controller at each round. We denote the current returned observation by e, the latest VLM response by z, and the next tool request by u_k. The controller may confirm, revise, or contradict previous findings as new evidence arrives without repeatedly supplying the full trajectory history. For an episode τ and instruction ℓ, the harness operates under

$$\mathcal{B} = (C_{\min}, C_{\max}, A_{\max}, R_{\max}), \tag{3}$$

where $C_{\min}$ specifies the shared minimum for requested crop calls and distinct credited local observations required for normal finalization, $C_{\max}$ bounds crop calls, $A_{\max}$ bounds auxiliary-tool calls, and $R_{\max}$ bounds boundary-refinement attempts. Normal finalization requires the minimum crop

count and distinct credited local evidence. If further search cannot proceed within the configured budget, the harness forces a final response using accumulated evidence. The complete evidence-seeking and metadata-construction procedure is summarized in Algorithm 1.

Here Y denotes the event lists explicitly submitted by the controller, and $\perp$ denotes the absence of a valid final event set or executable request, depending on context. All routines access the episode and task instruction, and VLM calls use p_ϕ. Intermediate findings guide inspection but become candidate labels only through explicit final submission.

Quality assessment uses temporal context beyond the interval being labeled. For a target interval I_k, the controller may inspect a wider window

$$I_k \subseteq \widetilde{I}_k, \tag{4}$$

containing preceding and subsequent observations, while assigning the event only to its supported temporal extent. Optimal events include task-advancing alignment, grasp acquisition, transport, placement, and productive recovery. Suboptimal events include ineffective contact, slipping, regression, unnecessary repetition, and other visually supported failures or inefficiencies. Smooth motion alone does not establish progress, and inactivity alone does not establish poor quality.

Algorithm 1: SCHEMA evidence seeking and metadata construction

Require: (τ, ℓ), VLM p_ϕ, tools $\mathcal{T}_\tau$
Require: $\mathcal{B} = (C_{\min}, C_{\max}, A_{\max}, R_{\max})$
Ensure: metadata $\mathcal{M}$
$(e, b) \leftarrow \mathsf{Init}(\tau, \ell, \mathcal{B})$; $Y \leftarrow \perp$
$(z, b) \leftarrow \mathsf{Decide}(e, b)$
for $k = 1, \ldots, C_{\max} + A_{\max} + 1$ **do**
  if $\mathsf{Stop}(z, b)$ **then**
    $Y \leftarrow \mathsf{Events}(z)$; **break**
  end if
  $(u_k, b) \leftarrow \mathsf{Next}(z, e, b)$
  if $u_k = \perp$ **then break**
  $(e, b) \leftarrow \mathsf{Tool}(u_k, e, b)$
  $(z, b) \leftarrow \mathsf{Decide}(e, b)$
end for
if $Y = \perp$ **then**
  $(z, b) \leftarrow \mathsf{ForceFinal}(e, b)$
  $Y \leftarrow \mathsf{Events}(z)$
end if
$\mathcal{M} \leftarrow \mathsf{Normalize}(Y)$
$\mathcal{M} \leftarrow \mathsf{Refine}(\mathcal{M}, R_{\max})$
$\mathcal{M} \leftarrow \mathsf{Reconcile}(\mathcal{M})$
return $\mathsf{Export}(\mathcal{M}, b)$

We deliberately separate semantic quality assessment from temporal localization. After normalization, Refine revisits at most $R_{\max}$ candidate events using denser local observations. It may update temporal endpoints and evidence references, but keeps the quality label and semantic explanation fixed; invalid refinements retain the original candidate. Reconcile then reapplies temporal and count constraints before export. Tool schemas, execution safeguards, normalization rules, and refinement details are provided in Appendices A.2–A.4.

3.3 Metadata Interfaces for VLA Learning

Both interfaces consume quality labels and temporal extents. Explanations and provenance remain available for inspection but are not additional prediction targets.

Frame-level quality assignment. Let $\mathcal{I}_n^+$ and $\mathcal{I}_n^-$ denote the unions of final optimal and suboptimal frame intervals in trajectory n. A training example starting at frame t receives

$$\bar{y}_{n,t} = \begin{cases} +, & t \in \mathcal{I}_n^+, \\ -, & t \in \mathcal{I}_n^-, \\ \varnothing, & \text{otherwise.} \end{cases} \tag{5}$$

This uses starting-frame membership even for action-chunk targets, rather than assigning quality from overlap with the entire action horizon. Unlabeled examples remain available for action learning.

Metadata-guided learning interfaces. For **metadata-guided sampling**, we reweight the base distribution as $p_{\mathrm{SCHEMA}}(n, t) \propto w_{n,t} p_{\mathrm{base}}(n, t)$, normalized over eligible training examples, where $w_{n,t}$ equals $\gamma_{\mathrm{opt}} > 1$, $\gamma_{\mathrm{sub}} \in (0, 1)$, and 1 for optimal, suboptimal, and unlabeled examples, respectively. This reallocates sampling frequency within the same corpus while preserving each example's observation context, original action target, and the action-learning objective, without adding a metadata-prediction loss. For **advantage-conditioned training**, we reuse RECAP's interface (Physical Intelligence et al., 2025a), replacing indicators derived from learned value-based

advantages with metadata-derived binary quality conditions, not numerical estimates of $A^\pi(o_t, a_t)$. We append `Advantage: positive` or `Advantage: negative` to the original task instruction for $\bar{y}_{n,t} = +$ or $-$, respectively, while unlabeled examples retain the original instruction. The advantage-conditioned objective is retained; at inference, `Advantage: positive` defines the conditional branch, with optional classifier-free guidance (CFG) using the unconditional branch.

4 EXPERIMENTS

4.1 RTQ-BENCH AND EVALUATION PROTOCOL

We introduce RTQ-Bench for temporal quality localization in robotic manipulation. It contains 1,350 episodes from nine public robot-data sources (O'Neill et al., 2024; Zeng et al., 2026; Wang et al., 2026; Bu et al., 2025; Hou et al., 2025; Allshire et al., 2026; Fang et al., 2024; Jiang et al., 2025; Khazatsky et al., 2024) (Table 1), covering productive progress, ineffective or repeated attempts, recoveries, and failures within individual episodes.

Table 1: RTQ-Bench statistics.

Source	Episodes	Hours	Duration (s)	
			Median	[Min, Max]
Open X-Embodiment	423	3.06	20.1	[2.1, 233.8]
ViFailback-Dataset	312	0.84	8.5	[3.7, 30.9]
GM-100	106	1.31	40.1	[9.5, 167.6]
AgiBot World	98	1.39	42.6	[11.1, 185.6]
RoboMind 2.0	98	1.28	37.7	[20.4, 155.1]
ABC-130K	82	1.90	71.3	[11.6, 252.3]
RH20T	82	0.67	24.9	[5.1, 116.7]
Galaxea Open World	81	1.54	55.5	[15.1, 202.2]
DROID-Raw	68	0.50	19.5	[4.4, 190.0]
Total	**1,350**	**12.50**	**21.2**	**[2.1, 252.3]**

Annotation and review. Experts follow a shared training guide with positive and negative video examples. A primary expert annotates event categories and boundaries directly from the task instruction and video; a second expert reviews every episode and refines these annotations as needed. Both experts may use Qwen3.7-Plus (Qwen Team, 2026b) to draft or revise textual reasons, which they verify against the video. Localization scoring uses the finalized human-reviewed categories and boundaries, not the model-assisted reasons. Uncovered regions remain unlabeled. Appendix D details the training guidelines and review protocol.

Metrics. **Boundary-Softened IoU (BS-IoU)** measures temporal overlap using softened boundaries with episode-duration-dependent tolerance. We report BS-IoU over all episodes (All) and category-positive subsets (GT+). All additionally evaluates agreement with empty reference lists, whereas GT+ evaluates overlap on episode–category pairs with retained reference events. Both variants average within each category and then equally across categories. **Center-Constrained Recall (CC-Recall)** measures event recovery under normalized center-distance and temporal IoU constraints, allowing optional same-category grouping across gaps of at most 0.5 seconds on both the prediction and reference sides. Category recall is the fraction of original reference segments recovered; Micro pools recovered and total segment counts across categories. Inference failures receive zero credit. Appendix B provides the formal definitions, tolerance parameters, matching rules, empty-reference conventions, and aggregation details used in Table 2.

Baselines. The primary baseline, **Qwen3.6-35B-A3B (Single-Pass)** (Qwen Team, 2026a), shares SCHEMA's backbone, quality rubric, event representation, and 144-frame cap, with closely matched prompts, but predicts all intervals in one pass. The cap does not imply matched total frame processing across multiple calls.

We also evaluate GPT-5.6 Sol (High) (OpenAI, 2026), Cosmos3-Super (Agarwal et al., 2026), Robo-Dopamine 2.0 (Robo-Dopamine-GRM-2.0-8B-Preview) (Xu et al., 2026), and Robometer (Robometer-4B) (Liang et al., 2026); the latter two use a shared progress-to-label conversion (Appendix C.2).

For RTQ-Bench, we adapt RECAP's value-function component to temporal quality localization (Physical Intelligence et al., 2025a). We pretrain our own value model using the same distributional value-function architecture on the full Open X-Embodiment, AgiBot World, and RoboCOIN datasets, then fine-tune it on benchmark-related episodes with a nominal $10\times$ budget relative to the benchmark episode count. Using the adapted model, we rank nonterminal frames within

Table 2: Temporal quality localization on RTQ-Bench (%; higher is better). All and GT+ include all episodes and category-positive episodes, respectively. For an episode–category pair with an empty reference, the per-pair BS-IoU is 1 for an empty prediction and 0 otherwise; reported table values are percentages after aggregation. Bold and underline indicate the best and second-best results.

Method	Frame Access	BS-IoU (All) ↑			BS-IoU (GT+) ↑			CC-Recall ↑		
		Opt.	Subopt.	Macro	Opt.	Subopt.	Macro	Opt.	Subopt.	Micro
Interval-prediction methods										
SCHEMA (Qwen3.6-35B-A3B)	$\leq$ 144/input	<u>43.11</u>	**57.97**	**50.54**	39.45	**49.83**	<u>44.64</u>	<u>40.16</u>	<u>48.63</u>	<u>41.55</u>
Qwen3.6-35B-A3B (Single-Pass)	$\leq$ 144/input	28.87	44.68	36.78	31.02	12.09	21.55	27.35	10.10	24.52
GPT-5.6 Sol (High)	$\leq$ 256/input	**45.26**	23.51	34.38	<u>46.85</u>	<u>49.03</u>	**47.94**	**52.03**	**54.79**	**52.49**
Cosmos3-Super (65B)	$\leq$ 256/input	33.30	<u>56.76</u>	<u>45.03</u>	36.69	15.68	26.19	28.39	13.87	26.01
Score-based baselines with score-derived intervals										
RECAP (adapted)	Dense	38.51	16.14	27.32	44.20	41.02	42.61	–	–	–
Robo-Dopamine 2.0	Dense	42.01	20.52	31.27	**47.97**	21.48	34.73	–	–	–
Robometer	Dense	39.51	24.42	31.96	43.74	20.65	32.19	–	–	–

Frame access gives per-input image caps or dense per-frame scoring, not cumulative frame processing. Dense denotes scoring every original frame, not supplying the entire video in one input. RECAP (adapted) evaluates a benchmark-adapted value model with per-episode top/bottom-30% advantage selection; its adaptation data overlap benchmark episodes (Appendix C.1).

each episode by estimated advantage and select approximately the top and bottom 30% as optimal and suboptimal, respectively. Adaptation uses source outcome labels and includes benchmark-overlapping episodes, but no human temporal quality annotations. Appendix C.1 details the training and benchmark-specific decoding procedure.

Score-based results characterize each model together with its benchmark-specific adapter (Appendix C). All methods are evaluated against the same human temporal quality annotations.

4.2 Temporal Quality Localization on RTQ-Bench

Table 2 shows that SCHEMA substantially improves temporal quality localization over single-pass inference with the same Qwen3.6-35B-A3B backbone. SCHEMA raises all-episode macro BS-IoU from 36.78% to 50.54%, GT+ macro BS-IoU from 21.55% to 44.64%, and micro CC-Recall from 24.52% to 41.55%, corresponding to gains of 13.76, 23.09, and 17.03 percentage points, respectively. Moreover, SCHEMA achieves the highest all-episode macro BS-IoU among all evaluated methods, exceeding the best-performing alternative on this metric, Cosmos3-Super, by 5.51 points. The gains in GT+ overlap and event recall show that SCHEMA's improvement is not limited to agreement with empty reference lists: the full harness also better localizes annotated quality events.

Against GPT-5.6 Sol (High), SCHEMA achieves substantially higher all-episode macro BS-IoU (50.54% vs. 34.38%), while GPT attains higher GT+ macro BS-IoU (47.94% vs. 44.64%) and micro CC-Recall (52.49% vs. 41.55%). The contrast is most pronounced for suboptimal localization: SCHEMA and GPT achieve comparable GT+ BS-IoU (49.83% vs. 49.03%), whereas SCHEMA reaches 57.97% all-episode BS-IoU compared with 23.51% for GPT. This result profile indicates that SCHEMA retains strong localization of annotated suboptimal events while aligning substantially better with the benchmark's sparse temporal annotations across all episodes.

Inference cost. Using the reference API prices specified in Appendix A.8, SCHEMA (Qwen3.6-35B-A3B) has an API-equivalent cost of $21.63 on RTQ-Bench, compared with $126.80 for GPT-5.6 Sol (High). Qwen inference itself was run locally.

4.3 VLA Pretraining with SCHEMA-Guided Sampling

Pretraining setup. We compare uniform and SCHEMA-guided pretraining of Qwen-PI (`QwenPI_v3` in StarVLA (Ye et al., 2026)), using Qwen3-VL-4B-Instruct (Bai et al., 2025) as the vision-language backbone. Both runs share initialization, corpora, action-learning objectives, and optimization schedules. Stage 1 uses approximately 960 hours from three selected Open X-

Embodiment (O'Neill et al., 2024) subsets spanning 52K distinct task instructions for broad task exposure; Stage 2 continues on 540 hours from 236 selected RoboCOIN (Wu et al., 2025) sub-datasets for embodiment alignment. Each stage runs for 100K updates on 8 NVIDIA H200 GPUs, with per-device batch sizes of 32 and 16 for Stages 1 and 2, respectively. The runs differ only in their frame-sampling distributions. Exact subset IDs and additional settings appear in Appendix E.

SCHEMA-guided sampling. Optimal, suboptimal, and unlabeled frames receive relative sampling weights $(2, 0.5, 1)$, respectively, under the normalized distribution in Section 3.3. The corpus, observation contexts, action targets, and action-learning objective remain unchanged. This comparison evaluates quality-guided reallocation of a fixed policy-training budget without adding trajectories or an auxiliary metadata-prediction loss.

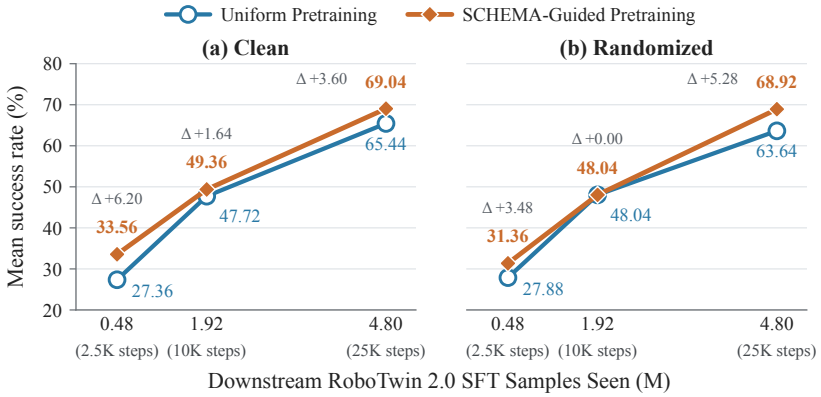


Figure 3: Downstream RoboTwin 2.0 success after uniform or SCHEMA-guided pretraining. Both pretrained models undergo the same joint SFT on clean and randomized episodes and are evaluated under both conditions at three SFT budgets.

Downstream evaluation. We jointly fine-tune each pretrained model on all 50 RoboTwin 2.0 (Chen et al., 2025) tasks, pooling 50 clean and 500 randomized training episodes per task into a single supervised fine-tuning (SFT) corpus. Both runs use identical downstream SFT settings on 16 ZW-810E PPUs with a global batch size of 192. Each resulting checkpoint is evaluated under both clean and randomized conditions on 8 NVIDIA RTX 4090 GPUs, using a fixed evaluation seed of 0 and 50 trials per task per condition. The 2.5K, 10K, and 25K SFT checkpoints correspond to 0.48M, 1.92M, and 4.80M cumulative training examples processed per run, respectively. As shown in Figure 3, SCHEMA-guided pretraining improves mean success at 2.5K steps from 27.36% to 33.56% under clean conditions and from 27.88% to 31.36% under randomized conditions, yielding gains of 6.20 and 3.48 percentage points. These early gains are consistent with a more effective policy initialization after limited downstream adaptation. At 25K steps, SCHEMA-guided pretraining reaches 69.04% and 68.92%, exceeding uniform pretraining by 3.60 and 5.28 percentage points, respectively. Per-task results appear in Table 6.

4.4 Real-World Robot Learning with SCHEMA

Experimental setup. We evaluate SCHEMA on four real-world manipulation tasks, *PickPlace*, *FastenTwo*, *LongCut*, and *InsertFour*, using an ALOHA bimanual robot. Each method–task–data-regime combination is evaluated over 25 trials. For data collection, each task begins with one round of successful teleoperation, followed by 1, 1, 12, and 2 rollout rounds, respectively. Following RECAP (Physical Intelligence et al., 2025a), each rollout policy is obtained using all task data accumulated from preceding rounds, and rollout collection includes human-in-the-loop interventions. After collection, we freeze the resulting task-specific trajectory pool and use the same episodes for all Qwen-PI and $\pi_{0.5}$ comparisons. *Successful Only* contains all successful teleoperation and rollout episodes, whereas *Successful + Failed* additionally includes failed rollouts.

To isolate the effect of metadata-guided fine-tuning, all Qwen-PI runs share the same SCHEMA-guided pretrained initialization from Section 4.3. We compare uniform sampling with SCHEMA-guided fine-tuning using relative weights $(1.5, 0.2, 1)$ for optimal, suboptimal, and unlabeled frames.

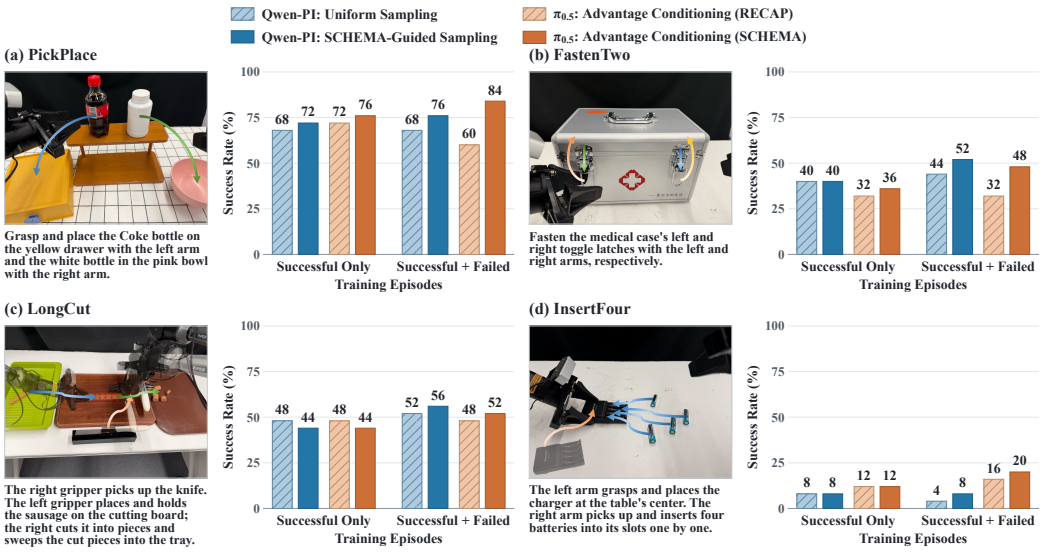


Figure 4: Real-world success rates on four manipulation tasks. SCHEMA metadata guides Qwen-PI sampling and $\pi_{0.5}$ advantage conditioning, compared with uniform sampling and RECAP conditioning, respectively. Both Qwen-PI variants share the same SCHEMA-guided pretrained initialization.

For $\pi_{0.5}$ (Physical Intelligence et al., 2025b), we compare RECAP's value-derived conditioning with SCHEMA's metadata-derived conditioning on the same trajectory subsets. Both variants follow RECAP's advantage-conditioning pipeline and use CFG with a fixed guidance weight of 2.0 at inference. The real-world RECAP baseline starts from our pretrained value-function checkpoint and follows the original RECAP value-function and advantage-conditioned policy-training procedure (Physical Intelligence et al., 2025a); the RTQ-Bench-specific $10\times$ adaptation and top/bottom-30% decoding rule are not used here. Data and training details are provided in Appendix F.

Learning from mixed-quality data. Under *Successful + Failed*, both SCHEMA variants achieve higher observed success rates than their corresponding baselines on all four tasks (Figure 4). Averaged equally across tasks, SCHEMA-guided sampling increases success from 42% to 48%, while SCHEMA-conditioned training improves from 39% to 51%, yielding gains of 6 and 12 percentage points, respectively. The conditioning gains are largest on *PickPlace* ($60\% \rightarrow 84\%$) and *FastenTwo* ($32\% \rightarrow 48\%$).

Utilizing failed experience. Adding failed episodes increases average success by 7 percentage points with SCHEMA-guided sampling ($41\% \rightarrow 48\%$) and 9 points with SCHEMA-conditioned training ($42\% \rightarrow 51\%$), compared with $+1$ and -2 points for uniform sampling and RECAP conditioning, respectively. With successful-only data, the corresponding mean differences are 0 and 1 percentage point, with task-dependent gains and regressions. The larger gains with failed rollouts included suggest more effective reuse of mixed-quality experience, consistent with SCHEMA's local quality formulation: labeled productive regions receive greater sampling weight or positive conditioning, whereas labeled suboptimal regions are downweighted or negatively conditioned.

5 Conclusion

We introduced SCHEMA, a harness-guided framework that constructs sparse temporal quality metadata from offline robot trajectories. RTQ-Bench evaluates localization against human annotations, while pretraining and real-world experiments demonstrate the utility of SCHEMA-generated metadata for sampling and advantage-conditioned learning. Across four real-world tasks, both interfaces yield larger average gains from adding failed rollouts than their respective baselines. These findings support direct temporal quality annotation as a complementary way to extract useful supervision from existing robot experience, without requiring the annotation harness at deployment.

AI USE STATEMENT

Generative models are used within the SCHEMA metadata-construction pipeline and to assist the human annotation workflow for RTQ-Bench. SCHEMA uses Qwen3.6-35B-A3B to construct temporal quality metadata through evidence seeking, quality assessment, and boundary refinement. For RTQ-Bench, both the first-pass annotators and second-pass reviewers may use Qwen3.7-Plus to assist in drafting or revising textual event reasons. Reference interval selection, quality categories, and temporal boundaries are determined and finalized by human experts.

The authors retain responsibility for the scientific claims, reported results, and final manuscript, including any AI-assisted content.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Nils Blank, Moritz Reuss, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Wenzel, Oier Mees, and Rudolf Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4158–4187. PMLR, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

Paweł Budzianowski, Emilia Wiśnios, Michał Tyrolski, Gracjan Góral, Igor Kulakov, Viktor Petrenko, and Krzysztof Walas. Opengvl–benchmarking visual temporal progress for data curation. *arXiv preprint arXiv:2509.17321*, 2025.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 17091–17116, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 653–660. IEEE, 2024.

Youhe Feng, Hansen Shi, Haoyang Li, Xinlei Guo, Yang Wang, Chengyang Zhang, Jinkai Zhang, Xiaohan Zhang, Jie Tang, and Jing Zhang. ProcVLM: Learning procedure-grounded progress rewards for robotic manipulation. *arXiv preprint arXiv:2605.08774*, 2026.

Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. *arXiv preprint arXiv:2502.08623*, 2025.

Chengkai Hou, Kun Wu, Jiaming Liu, Zhengping Che, Di Wu, Fei Liao, Guangrun Li, Jingyang He, Qiuxuan Feng, Zhao Jin, et al. Robomind 2.0: A multimodal, bimanual mobile manipulation dataset for generalizable embodied intelligence. *arXiv preprint arXiv:2512.24653*, 2025.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Bosung Kim, Ruiyi Wang, David Acuna, Jaehun Jung, Alexander Trevithick, Brandon Cui, Yejin Choi, and Prithviraj Ammanabrolu. How to instruct your robot: Dense language annotations power robot policy learning. *arXiv preprint arXiv:2605.17077*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Haoyang Li, Guanlin Li, Youhe Feng, Chen Zhao, Zhuoran Wang, Yang Li, Qizhe Wei, Shifeng Bao, Haitao Shen, Yihan Zhao, Tong Yang, and Jing Zhang. Training vision-language-action models with dense embodied chain-of-thought supervision. *arXiv preprint arXiv:2606.30552*, 2026.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025.

OpenAI. Gpt-5.6 system card. OpenAI Deployment Safety Hub, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026a. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qwen Team. Qwen3.7: The agent frontier, May 2026b. URL https://qwen.ai/blog?id=qwen3.7.

Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J. Joshi, Pete Florence, Wei Han, Robert Baruch, Yao Lu, Suvir Mirchandani, Peng Xu, Pannag Sanketi, Karol Hausman, Izhak Shafran, Brian Ichter, and Yuan Cao. RoboVQA: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 645–652. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610216.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736. PMLR, 2023.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pp. 58–76. Springer, 2024.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3272–3282. IEEE, 2025.

Ziyu Wang, Chenyuan Liu, Yushun Xiang, Runhao Zhang, Qingbo Hao, Hongliang Lu, Houyu Chen, Zhizhong Feng, Kaiyue Zheng, Dehao Ye, et al. The great march 100: 100 detail-oriented tasks for evaluating embodied ai agents. *arXiv preprint arXiv:2601.11421*, 2026.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Dayu Xia, Yue Shi, Yao Mu, Huiting Ji, Chaofan Ma, Yingjie Zhou, Hua Chen, Yang Liu, Jiezhang Cao, and Guangtao Zhai. RoboProcessBench: Benchmarking process-aware understanding in vision-language robotic manipulation. *arXiv preprint arXiv:2606.13040*, 2026.

Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, et al. Robo-dopamine 2.0: History-conditioned and ood-aware process reward modeling for robotic manipulation. *arXiv preprint arXiv:2608.15680*, 2026.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42386–42395, 2026.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Haoran Zhang, Fuxian Huang, Zhanhui Lin, Zijun Xu, and Weinan Zhang. Help: Human-efficient large-scale robot post-training with rollout segmentation. *arXiv preprint arXiv:2607.09776*, 2026.

Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. Scizor: A self-supervised approach to data curation for large-scale imitation learning. *arXiv preprint arXiv:2505.22626*, 2025.

APPENDIX TABLE OF CONTENTS

A SCHEMA Implementation Details **15**

A.1 Episode Interface and Temporal Alignment 15

A.2 Tool Interfaces and Prompt Protocol 16

A.3 Stateful Execution and Search Budgets 17

A.4 Event Normalization and Boundary Refinement 18

A.5 Metadata Export and Training Interfaces 19

A.6 Prompt Templates 20

A.7 Direct VLM Baseline Prompt 36

A.8 Inference Usage Accounting 41

B Temporal Quality Evaluation Metrics **42**

B.1 Notation and Temporal Coordinates 42

B.2 Rationale for BS-IoU and CC-Recall 42

B.3 Boundary-Softened IoU 42

B.4 Center-Constrained Recall 43

B.5 Evaluation Scope and Failure Handling 44

C RTQ-Bench Baseline Implementation Details **45**

C.1 RECAP Value-Based Baseline 45

C.2 Progress-Based Baseline Adaptation 47

D RTQ-Bench Data Sources and Annotation Protocol **49**

E VLA Pretraining and RoboTwin Experimental Details **50**

F Real-World Experimental Details **53**

A SCHEMA IMPLEMENTATION DETAILS

This appendix specifies the episode interface, tool protocol, execution budgets, and output processing used by SCHEMA. Unless otherwise stated, numerical implementation settings refer to the default quality-only pretraining configuration. This configuration performs sparse quality-event annotation directly, without requiring a preceding subtask-decomposition stage. Optimal and suboptimal events share the same controller, evidence tools, and temporal coordinate system.

A.1 EPISODE INTERFACE AND TEMPORAL ALIGNMENT

Episode representation. Dataset adapters expose a common episode record containing the episode identifier, task instruction ℓ, primary video, episode length T, camera identity, and available action/state fields and source metadata. The same reasoning interface also accepts a standalone video paired with a task instruction. Task text is resolved from an explicit task mapping when supplied, followed by episode metadata and dataset task records. The interface standardizes access to heterogeneous sources; it does not map different robots into a shared numerical action space. Auxiliary fields are exposed only when their availability and schema checks permit meaningful interpretation.

Temporal coordinates. All queries and event records use original, zero-based frame indices $t \in \{0, \dots, T-1\}$. Stored intervals $[s, e]$ are inclusive in frame-index space and contain $e - s + 1$ indexed frames; for evaluation, their endpoints are interpreted as boundary-frame timestamps rather than frame-cell edges (Appendix B.1). The primary video's frame rate f is obtained from the video metadata, and its frame timestamps are

$$\mathrm{time}(t) = t/f. \tag{6}$$

The model receives both the frame index and decimal-second timestamp alongside each image. When a response contains both frames and seconds, frame indices are authoritative and timestamps are recomputed from them. Time-only intervals are converted using a floor operation for the start and a ceiling operation for the end, with numerical tolerance, and are clipped to the episode bounds. Reversed endpoints are reordered. Auxiliary streams retain their dataset-provided frame correspondence and timing metadata; this access layer does not estimate an additional cross-modal synchronization model.

Overview and local observations. The initial overview uniformly samples the episode at a target rate of 2 FPS. Ordinary local video observations use a target rate of 4 FPS. The sample count is constrained to 4–144 images when enough source frames exist, rounded down to a multiple of two, and never exceeds the number of available frames. Windows containing at most four frames use all available frames. Consequently, the effective sampling rate can be lower than the target for long windows. Images are supplied individually in temporal order. Images whose pixel count exceeds 320×240 are resized to that resolution; smaller inputs are retained. Sampling preserves the original frame coordinates rather than assigning new indices to the sampled sequence.

Target and context windows. For a requested target $I = [s, e]$ of length $n = e - s + 1$, the desired preceding and following context, measured in frames, is

$$d_{\mathrm{pre}} = \max\{n, \lceil 0.5f \rceil\}, \tag{7}$$

$$d_{\mathrm{post}} = \max\{\mathrm{round}(0.5n), \lceil 0.25f \rceil\}. \tag{8}$$

Both the target and the complete observation window are bounded by $L = \max\{1, \lfloor T/3 \rfloor\}$ frames. The harness first clips the desired context to the episode, then allocates the remaining window budget proportionally between the available preceding and following context. It records the target range, actual observation range, desired and applied context, and whether episode boundaries or the budget limited the context. The VLM is instructed to localize the supported event, rather than label the entire context window. Context availability is therefore explicit, including when no additional context fits the budget.

A.2 Tool Interfaces and Prompt Protocol

Tool requests and observations. The controller returns structured JSON decisions that the harness parses and executes. Requests contain the temporal range, tool-specific parameters, an inspection `reason`, and `expected_evidence`. The latter two fields specify a hypothesis to investigate; they are not treated as observations establishing that hypothesis. Tool results retain temporal references and source metadata so that a finding can be traced to its supporting observation.

- `crop_video` takes start/end frames, optional matching timestamps, and a `sampling_strategy` field. It returns sampled primary-view images with original frame/time labels and the target and context-window metadata described above. In the current primary crop implementation, `coarse`, `medium`, and `fine` record inspection intent; all use the same 4 FPS target and frame-count cap.
- `inspect_action_raw` takes a temporal range, requested numerical fields and views, an output format (`table`, `stats`, or `both`), and a text budget. It returns bounded raw rows and/or statistics with field descriptions, returned ranges, sampling information, and source provenance. Default limits are 300 frames per requested window and 80 returned rows; the requested text budget is constrained to 200–2000 estimated tokens. The tool is available only when suitable numerical data pass the availability and trust checks. Its output accompanies the latest visual observation rather than replacing that visual context.
- `supply_camera` optionally requests named additional views over a temporal range. Returned images carry camera identities and original temporal references and are combined with primary-view evidence, with a default limit of eight returned frames per view. Extra-view access is disabled by default in the pretraining launcher and must be explicitly enabled for datasets with suitable cameras.

Evidence tools return observations, not admitted annotations. Numerical patterns such as gripper changes, repeated corrections, or low motion can guide inspection or help localize a visually identified event, but do not independently determine its quality category.

Prompt construction and response protocol. Each prompt combines the shared quality rubric, task instruction, episode timing metadata, available tools, remaining search budget, current observation, and bounded memory. The overview requests `search_candidate` findings and an initial crop. Later decisions contain a brief `decision_summary`, incremental `objective_findings`, and an `action_type` selecting an available tool or `final_answer`. Tool arguments are placed under the selected tool's name. Findings distinguish observations from contradictions and record their temporal support. Empty finding lists are valid when the current observation adds no useful evidence.

The final response explicitly supplies `optimal_segments` and `suboptimal_segments`. Event entries include frame/time boundaries, a reason, evidence frame/time references, and confidence. They may also contain a subtask description and a progress score; suboptimal entries additionally use severity and a progress-effect field, including `failure`. A `suboptimal_search_audit` records which suspicious ranges were checked and why they did or did not support a negative event. This audit is descriptive metadata, not an independent visual verifier. Responses are requested as JSON with English explanations and decimal-second timestamps.

Quality rubric. The shared semantic instructions apply the following rules:

1. **Task-relevant progress.** Optimal events require a visible contribution to the task, such as completing alignment, acquiring a stable grasp, lifting, target-directed object transport, insertion, placement, release, pressing, cutting, or productive recovery. Modest progress is sufficient when concrete. Smooth motion, broad approach, hovering, holding, and transitions alone do not establish optimal behavior. Boundaries should isolate the decisive progress-producing portion.
2. **Visually supported inefficiency.** Suboptimal events include ineffective contact, unnecessary repeated corrections, inefficient motion, instability, slipping, regression, and rework that visibly affect task execution. The prompt favors recall for mild but visually grounded

Table 3: Default settings for the quality-only pretraining launcher. Frame caps apply to individual model inputs, while boundary refinement has a separate budget from the main crop search.

Setting	Default
VLM controller	Qwen3.6-35B-A3B
Decoding temperature	0
Main completion-token limit per call	4048
Overview / ordinary crop target rate	2 / 4 FPS
Image cap per model input	144
Minimum requested crops / maximum crop calls	4 / 8
Shared auxiliary-tool call budget	3
Recent trajectory-memory entries	12
Boundary-refinement target rate	12 FPS
Boundary-refinement padding per side	0.5 s
Maximum refinement candidates	10

inefficiency, allowing lower severity and confidence rather than silently discarding such evidence. Stationary support-arm behavior and harmless adjustments alone are insufficient.

3. **Failure and recovery.** A failure is represented within the suboptimal category using `progress_effect=failure`. Its interval includes the nearby transition into failure and the complete event with direct consequences, extending to the last relevant frame if unresolved. A subsequent productive recovery is localized separately as optimal rather than merged with the preceding ineffective attempt.
4. **Search before conclusion.** The controller is instructed to inspect productive behavior and plausible negative candidates, use local evidence to check overview hypotheses, and revisit uncertain transitions. It may return an empty event list after sufficient search; neither class is required to appear in every episode. Uncovered frames remain unlabeled.
5. **Consolidation.** Adjacent fragments should be combined when they describe the same continuous event. The controller should preserve distinct events and split at a recovery or other semantic transition. This event-level consolidation is a prompt instruction; deterministic overlap handling is specified separately below.

These instructions guide VLM judgments. Structural checks do not certify that an explanation is visually correct, and model-reported confidence is not a calibrated probability.

A.3 Stateful Execution and Search Budgets

Application-owned state. The same VLM controller performs successive search decisions. The harness stores the current observation, tool history, incremental findings, counters, and termination status. Each new model call receives the current images and compact textual context; it does not resend all historical images. The default workflow retains up to 12 recent trajectory-memory entries, with redundant segment dumps removed. Findings are stored separately, deduplicated by identifier, and serialized under nominal stage-dependent budgets. The serialization selects at most 12 items using recorded priority or confidence and estimates text tokens from serialized length. The overview, ordinary-turn, and final-stage budgets are 1200, 800, and 1800 estimated tokens, respectively. They guide selection rather than impose tokenizer-exact limits; the first selected finding can exceed the nominal budget. These budgets apply to findings context, not the complete model input.

Search and termination. The overview proposes inspection regions but does not count as local evidence. In the normal protocol, the response schema requests another crop until the minimum of four crop calls has been reached. Subsequent decisions may request further crops, use available auxiliary tools, or submit a final answer. The finalization gate checks the number of distinct credited local-observation references against the configured minimum; it is not an exhaustive temporal-coverage test or a simple count of API calls. Invalid model responses do not earn evidence credit. Unresolved search candidates are retained in the search status but are not, by themselves, a separate hard prohibition on finalization.

An insufficient early final answer triggers another crop request, with a runtime crop fallback when needed. Once the budget of eight main crop calls is exhausted, no additional crops are executed; if

a final answer is still required, the harness requests one using accumulated evidence and preserving uncertainty. Raw-action and extra-camera calls share an additional budget of three calls. Overview inference, auxiliary-tool decisions, response repair, forced finalization, and boundary refinement can therefore make the total number of model invocations exceed eight. The 144-image cap similarly does not bound cumulative image processing over an episode.

Response repair and traceability. Malformed or invalid decisions receive at most one model-based repair attempt. The repair completion limit is normally the smaller of the main limit and 2048 tokens; an invalid-action response already containing a final answer retains the main limit to permit a complete repair. The harness records model responses, observations, prompt snapshots, usage, and execution state, including `trajectory.jsonl`, `state.json`, `final_judgment.json`, and `usage_summary.json`. These artifacts support inspection of the actual evidence and stopping behavior. The controller is used at inference time; this procedure does not train a separate query-selection policy.

A.4 Event Normalization and Boundary Refinement

Explicit event submission. The controller's final event lists are the source of candidate labels. Intermediate findings support reasoning but are not automatically backfilled into final annotations; explicitly empty lists remain valid. The harness normalizes temporal coordinates, orders and clips endpoints, and recomputes timestamps from frames. Ordinary events are restricted to at most $L = \max\{1, \lfloor T/3 \rfloor\}$ frames. Overlong intervals are tightened to the span of their internal evidence frames when that span satisfies the limit. Otherwise, a bounded interval is centered on the evidence span, or on the original interval when internal evidence is unavailable. Failure events are exempt from this length restriction so that continuous failure coverage can be retained.

Ranking and overlap resolution. Only candidates with nonempty explanations participate in selection. Let c_i, p_i, and v_i denote a candidate's confidence, progress score, and severity. Writing $[x]_0^1 = \min\{1, \max\{0, x\}\}$, the selection scores are

$$R_i^+ = 0.55[c_i]_0^1 + 0.45[p_i]_0^1, \tag{9}$$

$$R_i^- = 0.45[c_i]_0^1 + 0.35[v_i]_0^1 + 0.20[|p_i|]_0^1. \tag{10}$$

Missing confidence, severity, and progress components receive defaults of 0.5, 0.4, and 0.3, respectively. These scores prioritize candidate retention; they are neither learned value estimates nor additional training targets.

Selection is greedy in descending score order, with temporal ordering restored afterward. Failure intervals are selected first and are not subject to a count cap, although overlapping failure candidates still compete. Next, at most five ordinary suboptimal events are retained, excluding overlaps with selected failures or other retained suboptimal events. Finally, at most six optimal events are retained. An optimal candidate overlapping negative intervals may survive only if the overlap can be removed from its endpoints and the total removed portion is at most 20% of its original inclusive length. Internal conflicts or larger overlaps cause rejection. Surviving optimal intervals must also be mutually non-overlapping. These rules produce a sparse, non-overlapping event set without filling unlabeled gaps.

Local boundary refinement. After provisional normalization, the harness revisits up to ten eligible events using separate model calls. Candidates are taken in stored output order, with optimal entries preceding suboptimal entries in the default configuration; this is a bounded pass rather than a guarantee that every event is revisited. For each event, the inspected range extends 0.5 s before its start and after its end, with at least one padding frame and clipping at episode boundaries. Frames are sampled at a target rate of 12 FPS, subject to the 144-image cap.

The refinement prompt provides the existing category, explanation, boundaries, and local images. The response may change temporal endpoints and evidence references and provide a boundary note and boundary-specific confidence. It cannot change the event category or semantic explanation. After frame-authoritative normalization, the revised interval must lie within the inspected temporal window, and its cited evidence frames must lie within the revised interval. Boundary confidence

is stored separately from the original event confidence. Invalid responses, fields outside the refinement interface, or failed refinement attempts preserve the original candidate at this stage and are recorded in refinement metadata. Candidates beyond the refinement budget are also carried forward unchanged. All resulting candidates remain subject to the selection, count-limit, and overlap rules again before export. This stage adjusts existing events; it does not discover new events or independently reassess their quality categories.

A.5 Metadata Export and Training Interfaces

Full records and compact export. Full annotation artifacts retain event explanations, confidence, evidence references, and execution provenance. For downstream data loading, a compact JSONL export stores one record per episode, identified by `dataset`, `episode`, and `episode_index`, with `optimal` and `suboptimal` interval lists. Each interval contains `start_frame` and `end_frame`; when available, timestamps are included as a pair of `start_time_sec` and `end_time_sec`. Additional source identifiers or paths may be retained. The compact representation omits explanations and confidence because the training interfaces require only quality and temporal extent; the complete records remain available for inspection.

Frame-level lookup. For trajectory n, let $\mathcal{I}_n^+$ and $\mathcal{I}_n^-$ be the unions of its final inclusive frame intervals. As in the main text, the training label is determined by the example's starting frame:

$$\bar{y}_{n,t} = \begin{cases} +, & t \in \mathcal{I}_n^+, \\ -, & t \in \mathcal{I}_n^-, \\ \varnothing, & \text{otherwise.} \end{cases} \tag{11}$$

Here $\varnothing$ denotes the absence of a quality label. This lookup also applies to action-chunk prediction: it uses the starting frame, not majority overlap across the prediction horizon. The action chunk and observation context remain those of the original training example.

Sampling interface. The compact interval lists support the frame weights and normalized sampling distribution defined in the main text. In the order optimal, suboptimal, and unlabeled, the relative weights are $(2, 0.5, 1)$ for VLA pretraining (Section 4.3) and $(1.5, 0.2, 1)$ for real-world Qwen-PI fine-tuning (Section 4.4). All weights are positive, so unlabeled and suboptimal regions remain available for action learning. The interface changes sampling frequency without adding an event-description, confidence-prediction, or other metadata-prediction loss.

Conditioning interface. The LeRobot-format (Cadene et al., 2024) adapter creates two additional task-table entries for each original task instruction by appending the literal strings `Advantage: positive` and `Advantage: negative`, separated from the original text by a space. At labeled frames, the adapter replaces `task_index` with the corresponding augmented task entry. Unlabeled frames retain the original task index and receive no appended indicator. The adapter updates associated task metadata while preserving recorded action and state values and reusing source videos. This supplies the metadata-derived condition through the existing language-input interface. This supplies the metadata-derived condition through the policy interface described in the main text. At inference, `Advantage: positive` defines the conditional branch; the real-world experiments additionally use the classifier-free-guidance procedure described in Appendix F. Neither the annotation harness nor event explanations are needed during robot deployment.

Alignment checks in dataset preparation. Before constructing the conditioned dataset, the adapter checks agreement between annotation and source episode identifiers, consistency between episode lengths and table row counts, and contiguous frame indices $0, \ldots, T-1$. Every labeled interval must lie within its episode, and any positive–negative frame conflict raises an error. These checks prevent ambiguous temporal labels from being silently inserted into the conditioned dataset. They validate data alignment and label consistency, rather than the semantic correctness of the original VLM judgments.

A.6 PROMPT TEMPLATES

We provide the prompt templates used by the quality-only workflow. The templates are extracted from the registered quality objective and the executable prompt builders, including objective-specific overrides. Shared blocks are printed once and referenced by angle-bracketed placeholders. Expanding these references recovers the corresponding prompt text; line wrapping and JSON indentation are changed only for presentation. Other placeholders denote episode-specific inputs, not literal strings sent to the model. A reference ending in `IF_ENABLED` or `IF_PRESENT` expands to the named guidance block when its condition holds, and otherwise to an empty string. Budgets of four minimum and eight maximum crops below correspond to the default configuration.

Message composition. Each invocation contains the following system message and a user message comprising the stage-specific text followed by the current images. Each image is preceded by a caption of the form `Frame image j/N from original video, decimal-second timestamp:` followed by its original frame/time label, e.g., `f=1628 t=54.28 seconds`. Here j is the image position in the current input and does not replace the original frame index.

Listing 1: `SYSTEM_PROMPT`: System message

```
You are a concise robotics video analyst. Return valid JSON only. Do
not reveal chain-of-thought. Use brief decision_summary fields
instead. All free-text fields must be written in English.
```

Shared quality rules. The following block is inserted into overview, iterative, and boundary refinement prompts. It is reproduced without rewriting the semantic instructions. Search and labeling instructions remain model guidance; deterministic output processing is specified in Appendix A.4.

Listing 2: `QUALITY_RUBRIC`: Shared quality rules

```
Video-first evidence hierarchy: task text and video frames are
authoritative for semantic labels. Action cues are navigation and
boundary evidence, lower priority than visual evidence, and must not
alone define a final optimal/suboptimal label. Use action summaries or
raw-action results to choose where to inspect, confirm a visually
suspected motion detail, or tighten a boundary. Pair low-progress
motion, repeated correction, arm dominance, or gripper open/close with
a visible task consequence before calling it suboptimal: failed or
ineffective contact, delayed primary task progress,
collision/interference, object slip/drop, regression, unnecessary
rework, or prolonged task-relevant stall. For dual-arm tasks, infer
the likely primary operator, secondary/support operator, and unrelated
or idle arm from task text plus visual context. Do not penalize
secondary or unrelated arm low progress unless it visibly harms the
primary task. Final reasons should name the visible task consequence,
not only numeric or action-vector behavior.

Prefer high recall for suboptimal discovery while staying visually
evidence-grounded. Mark suboptimal when visible evidence shows
task-relevant hesitation, low-progress motion, repeated correction,
off-target motion, slightly inefficient path choice, failed or
ineffective contact attempts before recovery, unstable contact/grasp
before it is corrected, task regression, or rework. Action context may
identify a candidate window, but it does not establish the final label
without a visible task consequence. Retain mild suboptimal candidates
when visible evidence supports an execution consequence, even if
severity or confidence is low. Slowness, correction, indirect motion,
or instability without a visible task consequence is insufficient for
a final suboptimal label. Borderline slowdowns, short
hover-before-contact, small re-alignment loops, partial
withdraw-and-retry behavior, and minor loss of momentum can all count
when they visibly delay or disrupt the next task step; plausible delay
alone remains uncertain. A suboptimal segment does not need to cause
```

```
final task failure or state regression: a targeted local crop is
sufficient when it shows a visible task delay, disruption, or
instability caused by the slower motion, indirect path, repeated
correction, or stall. Keep such evidence with low severity/confidence
when appropriate. Use uncertain or omit a candidate when no visible
task consequence supports a suboptimal label. Neither smooth motion
nor inactivity alone establishes quality. Do not penalize support-arm
inactivity, harmless adjustment, or a harmless one-off adjustment that
does not visibly delay or destabilize task progress.

Suboptimal search requirement: actively look for at least one
plausible suboptimal candidate before concluding that none exists.
Inspect moments around approach, alignment, grasp/contact,
insertion/cut/place, release, failed-contact attempts and the
following recovery-to-progress boundary, and transitions between
subtasks. Useful negative evidence includes hovering without contact,
approach-then-withdraw, repeated small corrections, object slip or
loss of alignment, low-progress motion while actions change, moving
away from the target, slightly hesitant contact initiation, failed or
unproductive contact before a later recovery, or redoing a previously
achieved state. The failed attempt before recovery is the suboptimal
evidence; the successful recovery after failed contact should be
considered for a separate optimal segment. When a targeted crop
confirms a mild visible delay or disruption, record the evidence in
objective_findings with appropriately lower severity/confidence. If
the task consequence remains ambiguous, keep the finding uncertain.
Findings are evidence for explicit final selection, not mandatory
final labels; retain only qualifying events under the count and
overlap rules. Treat overview-only ambiguous motion as a search
candidate, not a final label, until a targeted crop shows a visible
task consequence. An empty suboptimal list is valid after sufficient
inspection of plausible negative evidence. Report the intervals
actually checked and their outcomes in suboptimal_search_audit; it is
a descriptive audit, not an exhaustive coverage gate. At forced
finalization, report unresolved uncertainty and any search limitation
without inventing a negative event.

Failure-event coverage requirement: use progress_effect='failure' only
for an actual failure event supported by visible evidence. Task text
defines the objective; dataset priors, metadata, and optional
fixed-annotation context may guide inspection but do not establish a
failure label. Explicitly submit supported failure events for
failure-priority selection; overlapping failure candidates still
compete under the final overlap rules. Start each such interval at the
last moment immediately before the failure becomes visible, so the
boundary includes the transition from normal execution into failure.
End only when the failure event and its direct consequences have fully
ended. If the failed state remains unresolved through the end of the
video, end at the last relevant frame. Cover the complete continuous
failure episode rather than only a short symptom or evidence frame. A
failure-coverage suboptimal segment is exempt from the one-third-video
maximum and has no duration limit: never truncate, split, or discard
it solely because it is long. This exception does not justify
unrelated lead-in or trailing context; the boundaries must still be
the complete, evidence-based failure boundaries. Ordinary hesitation
or inefficiency that is not the actual failure remains subject to the
normal tight-segment limit.

Optimal labeling requirement: identify every distinct, visually
confirmed successful alignment and every other distinct task-relevant
action
that visibly advances the task or a named subtask. Progress may be
modest when it is concrete and visually supported; it does not need to
complete the entire subtask. In the final answer, retain at most 6 of
the strongest distinct optimal events, prioritizing the clearest,
```

```
highest-confidence, and most task-relevant progress; never exceed this
cap. Good optimal evidence includes achieving a new task state such
as stable grasp after contact, lifting the target, successfully
inserting, placing, hanging, pressing, or cutting, releasing after
successful placement, recovering from failure or a stall into
effective forward progress, or completing a required subtask. Also
include a
short transport interval when it visibly advances a held object toward
its task-relevant target area; keep only the evidence-bearing
displacement, not the entire transit. Treat a gripper movement or
adjustment from an offset/misaligned pose into a task-relevant pose at
the
target position or object as optimal when it visibly completes
alignment, reaches the contact/grasp/manipulation pose, or directly
enables
the next required task step. This includes the final alignment from a
pose that is already near the expected position but still offset into
the correct expected pose. For a transition to the next subtask, label
the terminal alignment that reaches the correct pose and any separate
evidence-bearing transport that visibly advances a held object toward
its task-relevant target. Exclude unrelated transit and broad approach
context. Do not label a segment optimal merely because motion is
smooth, direct, or error-free. Pure approach, searching, hovering,
waiting,
repositioning, maintaining a hold, moving through free space without
changing the task state or completing target/object alignment, an
initial reach from the task-start pose into the general workspace, or
a normal transition between subtasks should be labeled uncertain or
omitted unless it produces clear task progress. If a smooth segment
only prepares for progress but does not visibly reduce the remaining
task-relevant distance, move a held object toward its target, change
the task state, or resolve a task-relevant gripper-to-target/object
offset, do not include it as optimal. In the final answer, each
optimal reason must explicitly say what progress was achieved and why
it is
more than ordinary motion. Keep every selected optimal segment as
short and boundary-accurate as possible: for alignment, start when the
final task-relevant correction toward the correct pose begins and end
as soon as that alignment is achieved; for another critical progress
event, cover only the decisive progress-producing action. Exclude
unrelated broad approach or transit and later holding or transition
frames. Preserve evidence-bearing, target-directed object transport
when it is the progress event being labeled. Optimal segments must
never
overlap one another or any suboptimal/failure segment; split
boundaries at the semantic transition and keep distinct progress
events as
distinct segments.

Recovery-to-progress rule: a broader interval may contain both
suboptimal and optimal evidence. If the robot first hesitates, slips,
misaligns, searches, repeats corrections, or otherwise loses progress,
mark that earlier portion as suboptimal. But if a later, short portion
of the same local episode visibly resolves the problem and returns the
task to normal progress, that recovery portion should be split out and
labeled optimal. After failed contact, label the failed/ineffective
contact attempt, withdrawal, or repeated correction before the fix as
suboptimal; label the recovery after failed contact as optimal once
contact becomes stable/productive or direct task progress resumes.
Examples include the moment a corrected grasp becomes stable, a
misaligned object becomes aligned, contact finally becomes productive
after failed attempts, the robot exits a correction loop and moves the
object toward the target, or a previously stalled subtask begins
advancing again. Keep the recovery optimal segment short: start at the
first timestamp where the correction clearly succeeds or forward
```

```
progress resumes, and end when that immediate recovery event is
complete. Do not merge the preceding inefficient motion into the
optimal segment.

Final segment consolidation requirement: before returning a
final_answer, review all final and remembered objective_findings in
chronological order. If adjacent or slightly overlapping segments have
the same label, describe the same continuous event, share the same
subtask/progress_effect, and are separated only by a tiny gap or
sampling artifact, merge them into one tight interval with a single
evidence-grounded reason. Do not leave many tiny fragments for one
sustained hesitation, repeated correction loop, unstable contact
episode, or continuous progress-producing action. Do not merge across
different subtasks, across a clear recovery-to-progress transition, or
across a gap where the robot visibly resumes normal productive motion.
The merged segment must still obey the maximum segment length rule
unless it is a complete progress_effect='failure' suboptimal interval.
Never merge distinct optimal progress events, and never merge in a way
that creates overlap. If a non-failure event would become too broad,
keep only the most evidence-bearing representative interval and
explain the specific event.
```

The length block substitutes MAX_EVENT_FRAMES= $\max\{1, \lfloor T/3 \rfloor\}$ and MAX_EVENT_SECONDS= MAX_EVENT_FRAMES$/f$. The seconds display is rounded to two decimal places.

Listing 3: LENGTH_RULE: Shared length rule

```
Segment length requirement: keep every crop_video window and every
final/observed optimal or suboptimal segment as short and
boundary-accurate as possible. A final segment should cover only the
supported event. Inspection context helps interpret that event but is
not automatically part of its label. Do not merge multiple subtasks,
long approach phases, or broad before/after context into one segment.
For this episode, every crop and every optimal or non-failure
suboptimal segment may use at most <MAX_EVENT_FRAMES> frames (about
<MAX_EVENT_SECONDS> seconds), computed as max(1, floor(T/3)) for the
original <T>-frame video. The sole exception is a suboptimal segment
with progress_effect='failure': it must run from the last moment
immediately before failure onset through the complete end of the
failure and has no length limit. Never shorten a complete failure
interval to satisfy the one-third limit. For a non-failure event that
appears longer than the limit, report only the most evidence-bearing
sub-interval. Use inclusive frame intervals: start at the first
supported event frame and end at the last frame belonging to the event
or its direct consequences, not the first subsequent non-event frame.
Failure intervals also include the immediately preceding transition
frame. A single-frame event is valid; unlabeled gaps may remain.
```

Listing 4: FINAL_OUTPUT_GUIDANCE: Shared final-output guidance

```
Explicitly submit selected events with nonempty evidence-grounded
reasons. Consolidate repeated views of the same continuous event.
Final selection retains failure intervals first, with no failure count
cap but with overlapping failures competing; it then retains at most 5
ordinary suboptimal events and at most 6 optimal events. The harness
applies confidence/progress/severity ranking and overlap resolution;
observing an event does not guarantee its retention. Preserve complete
supported failure boundaries without unrelated context and separate
productive recovery. Either class may be empty, and uncovered frames
remain unlabeled. Never create a segment to satisfy a quota.
```

Overview prompt. EPISODE_METADATA_JSON is the filtered episode metadata. The overview can describe candidate regions but must request a local crop. The action-tool guidance is inserted only when action context is enabled; the overview response schema still permits only a crop request.

Listing 5: OVERVIEW_PROMPT: Overview user-message template

```
You are controlling a video-crop tool to analyze a robot manipulation
episode. Your goal is to find distinct visually supported optimal
progress intervals (retaining at most 6), complete visually supported
failure intervals, and representative ordinary suboptimal intervals,
then explicitly select events under the count and overlap rules and
explain why.

Interaction protocol:
1. You first see a coarse full-video overview as a chronological
frame-image sequence.
2. You must request crop_video for targeted local inspection before
finalizing; the overview is only for planning the search.
3. Use the overview to compare a positive-progress candidate with a
suspicious suboptimal candidate. Request the higher-information first
crop, then cover the other evidence direction when it is visible.
4. On a later turn, after the required crop calls and the harness
minimum of distinct credited local-observation references are
satisfied, you may return action_type='final_answer'. The overview
earns no local-evidence credit, and invalid responses do not count.
Unresolved candidates alone do not prevent finalization. If the
harness forces finalization, follow that stage instruction using the
available evidence.
5. In each response, return compact objective_findings that materially
change the search plan; an empty list is valid.
6. For objective_findings and crop_video requests now, and for final
optimal_segments/suboptimal_segments on later eligible turns, return
both original-video frame indices and decimal seconds:
start_frame/end_frame plus start_time_sec/end_time_sec. Copy original
frame indices and timestamps from the shown labels. Compute matching
decimal seconds as frame_index / FPS, using the episode FPS; do not
mix illustrative timestamps with actual frame indices. Do not use
MM:SS timestamps. Frame indices are authoritative and range from 0 to
T-1; [start_frame, end_frame] is inclusive, so its length is
end_frame-start_frame+1. Seconds are for readability and
cross-checking.

<QUALITY_RUBRIC>

<LENGTH_RULE>

<FINAL_OUTPUT_GUIDANCE>

<ACTION_GUIDANCE_IF_ENABLED>

Sampling strategy guidance: coarse, medium, and fine express
inspection intent only; all ordinary crops use the same target rate of
4 FPS and the same 144-image cap. A long window may have lower
effective sampling density under the cap. Use a shorter window for
focused contact/hesitation/progress inspection; selecting fine does
not raise the sampling rate.

Episode: <EPISODE_ID>
Task: <TASK_INSTRUCTION>
Episode length: <T> frames, fps=<FPS>
Maximum crop tool calls: 8
Minimum crop tool calls before final answer: 4
Metadata: <EPISODE_METADATA_JSON>
```

```
Current observation: full-video overview frame sequence. Each image is
preceded by its original-video frame and decimal-second timestamp
text, in the format 'f=FRAME t=SECONDS seconds'.

Return JSON only. Because this is the initial overview, return
crop_video now, following this schema:
<OVERVIEW_RESPONSE_SCHEMA>
```

Iterative reasoning prompt and stage selection. BOUNDED_MEMORY_AND_FINDINGS_JSON contains recent trajectory entries and the selected findings context. CURRENT_OBSERVATION_JSON contains the compact current observation, including its temporal and provenance fields. The common iterative template is followed by the applicable stage instruction and response schema. Auxiliary-tool guidance is empty during required-crop and forced-final turns. Otherwise, it is included only when that tool is available and its budget permits a call. Context guidance is included only when the observation carries a context-window record.

Listing 6: ITERATIVE_PROMPT: Common iterative user-message template

```
You are in a decoupled iterative video reasoning loop. Previous images
are not repeated; use the provided compact context plus the current
observation.

<QUALITY_RUBRIC>

<LENGTH_RULE>

<FINAL_OUTPUT_GUIDANCE>

Episode: <EPISODE_ID>
Task: <TASK_INSTRUCTION>
Episode length: <T> frames, fps=<FPS>
Crop calls used: <CROPS_USED>/8

Minimum crop calls before final answer: 4

Trajectory memory:
<BOUNDED_MEMORY_AND_FINDINGS_JSON>

Current observation:
<CURRENT_OBSERVATION_JSON>

For the current observation, first list objective_findings using both
start_frame/end_frame and decimal-second start_time_sec/end_time_sec.
If current_observation includes action_brief_text or action_findings,
treat them as concise robot action cues for choosing informative crop
windows, not as visual labels; verify labels visually before
finalizing.

<ACTION_GUIDANCE_IF_ENABLED>
<CAMERA_GUIDANCE_IF_ENABLED>
<CONTEXT_GUIDANCE_IF_PRESENT>

Keep overview-only ambiguous slowdowns as search candidates until a
targeted crop shows a visible task consequence. Record a mild slowdown
with lower severity/confidence when a visible task consequence
supports it; otherwise keep it uncertain. Intermediate findings are
not automatically promoted into final labels. Label a segment optimal
when it visibly makes task-relevant progress, including completing a
task-relevant gripper alignment from an offset pose to the target
position or object; identify each such distinct successful alignment,
a successful placement/release/press/insertion, recovery into
effective progress, or a short transport that visibly advances a held
object toward its target. Use progress_score 0.45-0.55 for modest but
```

```
visible task-relevant progress. Keep the interval tight and ensure it
overlaps no other final segment. Retain at most the 6 strongest
optimal events in the final answer. Otherwise use uncertain or omit
it. For crop_video and final answers, return frame ranges and matching
decimal-second time ranges from the original video start, using the
shown original frame labels. Set start_time_sec=start_frame/FPS and
end_time_sec=end_frame/FPS with the episode FPS. Do not use MM:SS
timestamps. If frames and seconds disagree, the system will trust
frames.

<STAGE_INSTRUCTION>

Return JSON only following this schema:
<STAGE_RESPONSE_SCHEMA>
```

Before the minimum crop count, use the required-crop instruction and CROP_RESPONSE_SCHEMA. If an early final answer is rejected after the minimum crop count, use the rejected-final variant with the recorded search-status reason codes and the same crop schema. In an ordinary decision turn, use DECISION_INSTRUCTION and DECISION_RESPONSE_SCHEMA. When finalization is forced, use FORCED_FINAL_INSTRUCTION and FINAL_RESPONSE_SCHEMA.

Listing 7: REQUIRED_CROP_INSTRUCTION: Instruction before the minimum crop count

```
You must request another crop_video before finalizing. The
required-crop stage is active; <REMAINING_REQUIRED_CROPS> more crop
request(s) are required. Normal finalization also requires the harness
minimum of distinct credited local-observation references; invalid
responses earn no credit. Choose the most informative original-video
interval using both start_frame/end_frame and decimal-second
start_time_sec/end_time_sec. Do not use MM:SS timestamps. Use the
required crops to cover both positive-progress and suspicious
suboptimal evidence when the overview provides both. Inspect a
successful alignment, placement/release, short target-directed
transport, or recovery-to-progress candidate as well as a suspicious
slowdown, failed/uncertain contact, repeated correction, brief stall,
or unstable moment. After one direction is checked, prioritize an
uncovered candidate from the other direction or another major
uncertain task phase. Use sampling_strategy='fine' to express a
focused inspection intent for a short suspicious window; all ordinary
crop strategies use 4 FPS subject to the same 144-image cap. Keep the
crop boundary tight and within the one-third-video maximum. Also fill
objective_findings for the current observation before requesting the
next crop.
```

Listing 8: REJECTED_FINAL_INSTRUCTION: Instruction after rejection of an early final answer

```
You must request another crop_video before finalizing. The harness
rejected the early final answer. Use these recorded status reasons to
guide the next crop: <SEARCH_STATUS_REASON_CODES_JSON>. The gate
checks distinct credited local-observation references, not exhaustive
temporal coverage; unresolved candidates alone are not a hard
prohibition on finalization. Choose the most informative
original-video interval using both start_frame/end_frame and
decimal-second start_time_sec/end_time_sec. Do not use MM:SS
timestamps. Use the required crops to cover both positive-progress and
suspicious suboptimal evidence when the overview provides both.
Inspect a successful alignment, placement/release, short
target-directed transport, or recovery-to-progress candidate as well
as a suspicious slowdown, failed/uncertain contact, repeated
correction, brief stall, or unstable moment. After one direction is
checked, prioritize an uncovered candidate from the other direction or
another major uncertain task phase. Use sampling_strategy='fine' to
express a focused inspection intent for a short suspicious window; all
```

```
ordinary crop strategies use 4 FPS subject to the same 144-image cap.
Keep the crop boundary tight and within the one-third-video maximum.
Also fill objective_findings for the current observation before
requesting the next crop.
```

Listing 9: DECISION_INSTRUCTION: Instruction for an ordinary decision turn

```
Choose an available tool action or finalize, subject to the current
stage and remaining tool budgets. Prefer informative inspection when a
material uncertainty can still be resolved. Compare positive-progress
evidence with plausible negative evidence when available, but neither
exhaustive phase coverage nor finding both classes is required.
Finalize when the credited-evidence gate is satisfied and the
available evidence supports the selected sparse events. Explicitly
select qualifying candidates under the count and overlap rules;
findings may remain uncertain, be contradicted, or be excluded during
selection and are never automatically backfilled. Modest but visible
task-relevant progress may remain optimal with progress_score
0.45-0.55 and appropriately lower confidence. If a suboptimal interval
contains a later short recovery-to-progress moment, split that later
moment into an optimal segment rather than labeling the whole local
interval suboptimal. If finalizing with no suboptimal segments,
include a concrete suboptimal_search_audit for all checked suspicious
intervals. When finalizing, first consolidate adjacent same-label
fragments that describe one continuous event, then shorten segment
boundaries to the smallest interval that still supports the reason.
```

Listing 10: FORCED_FINAL_INSTRUCTION: Instruction for forced finalization

```
The harness requires finalization now. Return
action_type='final_answer' using only accumulated evidence; do not
request another tool. This stage takes precedence over further-search
or minimum-crop instructions. Explicitly return the supported event
lists, allowing either or both to be empty. In decision_summary and
suboptimal_search_audit, state relevant evidence limitations and
unresolved uncertainty; do not claim that uninspected candidates were
checked or create labels to compensate for an exhausted budget.
```

Listing 11: CONTEXT_GUIDANCE: Conditional crop-context guidance

```
Crop context contract: target_frame_range is the unverified hypothesis
interval; observation_frame_range includes interpretation-only context
before and after it. Compare the pre-target, target, and post-target
object states; the request reason is not evidence. If the wider visual
context disproves the candidate, return finding_kind='contradiction'.
Do not automatically label buffer frames. For contact, grasp, release,
and placement, distinguish failed contact from a successful release or
placement using visible object displacement and the stable post-event
state. If required transition context is incomplete, keep the finding
uncertain. Request a tighter or better-contextualized crop only when
the current stage and budget permit it; otherwise report the
limitation during finalization.
```

Listing 12: ACTION_GUIDANCE: Conditional raw-action tool guidance

```
inspect_action_raw is optional evidence. Use it only when visual
evidence is ambiguous, a compact action cue points to a suspicious
interval, or motion/gripper details would materially change the
judgment. Its request includes views and prompt_token_budget so the
system can return a compact action table/stats payload. Many scenes
can be judged visually. Request a bounded frame/time window, choose
the smallest useful views, and set prompt_token_budget as a context
```

```
budget hint (recommended 600-1000 tokens; system-enforced hard max
2000). The result will be attached back to the visual crop as
action_agent_result, so continue judging from the images plus this
targeted evidence. Do not request full-episode raw action tables and
do not finalize from action evidence alone.
```

Listing 13: CAMERA_GUIDANCE: Optional extra-camera tool guidance

```
supply_camera is optional visual evidence. Use it only when the
current main-view crop is occluded, too far away, or ambiguous and a
bounded side/wrist camera window would materially change the judgment.
Request only the needed views and a tight frame/time window; the
returned camera images will be attached back to the same visual crop
for final judgment. Prefer left_wrist/right_wrist for gripper,
contact, grasp, or occlusion questions. Keep the main crop as the
primary visual anchor: side-camera evidence should clarify what the
main view cannot show, not replace visible task evidence. Use a tight
interval and a small view set; typical views are left_wrist and
right_wrist.
```

Boundary refinement prompt. CANDIDATE_SEGMENT_JSON contains the selected event, its identity and category, and the existing semantic explanation and temporal references. The observation is the denser local inspection described in Appendix A.4. Its response is evaluated by the refinement interface; unsuccessful refinement retains the candidate at this stage rather than invoking the main-search repair procedure.

Listing 14: REFINEMENT_PROMPT: Boundary refinement user-message template

```
You are refining the temporal boundaries of one already-selected robot
video segment. Do not search for new segments, do not remove this
segment, and do not change its label. Update only temporal endpoints,
matching seconds, evidence frames/timestamps, boundary_note, and
boundary-specific confidence. Keep the category, subtask, semantic
reason, and original event confidence unchanged. Use the shared rubric
below only to interpret the existing event: its search,
event-discovery, and final-list instructions do not apply to this
refinement call. Tighten the interval around the specific event
described by the fixed candidate reason.

<QUALITY_RUBRIC>

<LENGTH_RULE>

Boundary rules:
1. Use only the current high-density local observation and its
decimal-second timestamp labels.
2. Use inclusive frame boundaries. Start at the first shown frame
belonging to the described event, except that a failure includes the
immediately preceding transition frame.
3. End at the last shown frame belonging to the described event or its
direct consequences, not the first subsequent non-event frame. Loss of
visibility alone does not establish event completion.
4. Keep the original label, subtask, and semantic reason; refine only
frame/time boundaries and evidence frames/timestamps.
5. If the original boundary already looks tight, return the same
boundary with a short note.
6. If the shown local window is insufficient, preserve the original
boundary and evidence and explain the limitation in boundary_note.
Report lower boundary-specific confidence without changing the
original event confidence; invalid refinements retain the original
candidate.
7. Return both corrected frame indices and matching decimal seconds.
If they disagree, frames will be trusted.
```

```
8. For an optimal segment, keep the decisive successful alignment or
critical progress action named in its fixed reason, including
target-directed transport that visibly advances a held object. Exclude
unrelated approach, transit, holding, and transition context.
9. For a suboptimal segment with progress_effect='failure', include
the immediately preceding transition and end at the last frame of the
failure and its direct consequences. If the failure remains
unresolved, retain the last relevant boundary supported by the
candidate and observation; do not absorb a separate productive
recovery. Failure intervals have no length limit and must not be
shortened merely to satisfy the one-third rule.
10. Any revised interval must lie within the inspected temporal
window, and revised evidence frames must be shown frames inside that
interval. If a valid supported revision is unavailable, preserve the
original candidate and explain why.

Episode: <EPISODE_ID>
Task: <TASK_INSTRUCTION>
Episode length: <T> frames, fps=<FPS>

Candidate segment:
<CANDIDATE_SEGMENT_JSON>

Current high-density local observation:
<CURRENT_OBSERVATION_JSON>

Return JSON only following this schema:
<REFINEMENT_RESPONSE_SCHEMA>
```

Main-search response repair. A failed main-search response receives at most one repair attempt. The ordinary repair prompt appends an instruction to the original user prompt and reuses the observation. If the response has an invalid action but contains a final answer whose compact serialized candidate is at most 9000 characters, the alternative structural-repair template replaces the original user prompt. Its validation-error text is limited to 2000 characters. The reference to adjacent intervals in this generic repair template does not impose a full-coverage partition requirement on sparse quality annotations.

Listing 15: `REPAIR_PROMPT`: Ordinary response-repair template

```
<ORIGINAL_USER_PROMPT>

RESPONSE REPAIR: The previous response could not be executed. Previous
response status: <VALIDATION_STATUS>. Validation error:
<VALIDATION_ERROR>. Re-evaluate the same observation and return
exactly one compact JSON action allowed by the schema above. Do not
enumerate speculative candidates, repeat the prompt, add markdown, or
include commentary outside JSON.
```

Listing 16: `STRUCTURAL_REPAIR_PROMPT`: Structural final-action repair template

```
STRUCTURAL FINAL-ACTION REPAIR
Repair the previous JSON candidate using the deterministic validation
error below. Preserve semantic labels, goals, evidence, and segment
order unless the error requires a change. These are sparse, inclusive
frame intervals, not a full-video partition. Chronologically
consecutive retained intervals require next.start_frame >
previous.end_frame; equality to previous.end_frame + 1 applies only
when the two intervals actually touch. Preserve valid unlabeled gaps
and single-frame intervals; never extend an event or invent evidence
to fill a gap. Return exactly one complete JSON action and no markdown
or commentary.

VALIDATION ERROR
```

```
<VALIDATION_ERROR>

PREVIOUS CANDIDATE
<PREVIOUS_CANDIDATE_JSON>
```

Response schemas. These are the descriptive JSON templates actually placed in prompts, rather than formal JSON Schema documents. Quoted descriptions specify the expected response values. An angle-bracketed schema reference expands to the entire referenced object before serialization. Overview findings are search hypotheses; later findings are observations or contradictions. The auxiliary-tool entries in the ordinary decision schema are included only when enabled, with the corresponding names added to `action_type`. Thus the displayed all-tools variant is not a claim that extra-camera access is always on.

Listing 17: `SEARCH_FINDING_SCHEMA`: Overview search-candidate fields

```
{
  "finding_kind": "search_candidate",
  "label": "optimal | suboptimal | failure | uncertain",
  "start_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; segment start frame",
  "end_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; segment end frame",
  "start_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; segment start with tight event
  boundary, not broad context",
  "end_time_sec": "number or string; decimal seconds from the original
  video start, matching the associated frame index divided by the
  episode FPS; do not use MM:SS; segment end with a tight event
  boundary; a failure segment may exceed one third of the original
  video only to preserve complete failure coverage",
  "subtask": "English string or null",
  "reason": "English hypothesis grounded in overview cues; state the
  possible progress or task consequence to verify, without treating it
  as a confirmed local event.",
  "evidence_frames": [
    "integer; original video frame index copied from the shown frame
  label, e.g. 1628; evidence frames"
  ],
  "evidence_times_sec": [
    "number or string; decimal seconds from the original video start,
  matching the associated frame index divided by the episode FPS; do
  not use MM:SS; evidence timestamps matching evidence_frames"
  ],
  "progress_effect": "advances_task | slows_task | regresses_task |
  failure | unclear",
  "severity": "0 to 1; provisional hypothesis severity to be checked
  by a local crop, not a final label",
  "progress_score": "-1 to 1; optimal uses 0.45-1; use 0.45-0.55 for
  modest but visible task-relevant progress",
  "confidence": "0 to 1; confidence in the search hypothesis, not a
  verified local label",
  "summary": "Compact English hypothesis explaining why this interval
  may matter to the objective.",
  "priority": "0 to 1; search priority based on expected information
  gain, not final-label confidence",
  "expected_evidence": "Compact English description of what a targeted
  crop should confirm or reject."
}
```

Listing 18: `LOCAL_FINDING_SCHEMA`: Local observation and contradiction fields

```
{
```

```
  "finding_kind": "observation | contradiction",
  "label": "optimal | suboptimal | failure | uncertain",
  "start_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; segment start frame",
  "end_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; segment end frame",
  "start_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; segment start with tight event
  boundary, not broad context",
  "end_time_sec": "number or string; decimal seconds from the original
  video start, matching the associated frame index divided by the
  episode FPS; do not use MM:SS; segment end with a tight event
  boundary; a failure segment may exceed one third of the original
  video only to preserve complete failure coverage",
  "subtask": "English string or null",
  "reason": "English evidence-grounded reason. For optimal, name the
  concrete task progress achieved. For suboptimal, name the
  delayed/regressed task progress.",
  "evidence_frames": [
    "integer; original video frame index copied from the shown frame
  label, e.g. 1628; evidence frames"
  ],
  "evidence_times_sec": [
    "number or string; decimal seconds from the original video start,
  matching the associated frame index divided by the episode FPS; do
  not use MM:SS; evidence timestamps matching evidence_frames"
  ],
  "progress_effect": "advances_task | slows_task | regresses_task |
  failure | unclear",
  "severity": "0 to 1; use 0.10-0.35 for mild visually supported task
  consequences; plausibility alone is not a final label",
  "progress_score": "-1 to 1; optimal uses 0.45-1; use 0.45-0.55 for
  modest but visible task-relevant progress",
  "confidence": "0 to 1; use 0.20-0.55 for mild visually supported
  negative evidence; keep an unsupported or ambiguous task consequence
  uncertain"
}
```

Listing 19: CROP_REQUEST_SCHEMA: Crop-request fields for overview and required-crop turns

```
{
  "start_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; crop start frame",
  "end_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; crop end frame",
  "start_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; crop start time",
  "end_time_sec": "number or string; decimal seconds from the original
  video start, matching the associated frame index divided by the
  episode FPS; do not use MM:SS; crop end time; focused and never
  longer than one third of the original video",
  "sampling_strategy": "coarse | medium | fine; intent only, all
  ordinary crops use 4 FPS subject to the same 144-image cap; prefer a
  short focused window for contact/alignment",
  "reason": "English reason this segment may contain
  optimal/suboptimal evidence.",
  "expected_evidence": "English description of the slowdown,
  correction, success, or transition to inspect."
}
```

Listing 20: ACTION_REQUEST_SCHEMA: Optional raw-action request fields

```
{
  "start_frame": "integer; original video frame index; raw action
  inspection start frame",
  "end_frame": "integer; original video frame index; raw action
  inspection end frame",
  "start_time_sec": "number or string; optional raw action inspection
  start time in decimal seconds",
  "end_time_sec": "number or string; optional raw action inspection
  end time in decimal seconds",
  "fields": [
    "action",
    "Usually keep this as ['action'] unless metadata exposes another
  useful action/state field."
  ],
  "views": [
    "right_arm | right_gripper | left_arm | left_gripper | arm_motion
  | gripper | all",
    "Choose the smallest view set that answers the question."
  ],
  "format": "table | stats | both; use both by default",
  "prompt_token_budget": "Recommended 600-1000; hard maximum is
  enforced by the system.",
  "sampling": "uniform",
  "reason": "English reason this exact action window matters.",
  "expected_evidence": "English description of the action signal
  evidence needed."
}
```

Listing 21: `CAMERA_REQUEST_SCHEMA`: Optional extra-camera request fields

```
{
  "start_frame": "integer; original video frame index; camera
  inspection start frame",
  "end_frame": "integer; original video frame index; camera inspection
  end frame",
  "start_time_sec": "number or string; optional camera inspection
  start time in decimal seconds",
  "end_time_sec": "number or string; optional camera inspection end
  time in decimal seconds",
  "views": [
    "left_wrist | right_wrist | main | third_view | high_left |
  high_right",
    "Choose the smallest view set that can resolve the visual
  ambiguity."
  ],
  "sampling_strategy": "coarse | medium | fine; use fine for short
  contact/gripper ambiguity",
  "prompt_token_budget": "Recommended 500-1000; used as a compactness
  hint for camera evidence.",
  "reason": "English reason the main view is insufficient for this
  exact interval.",
  "expected_evidence": "English description of the side-camera visual
  evidence needed."
}
```

Listing 22: `FINAL_RESPONSE_SCHEMA`: Final event-response schema

```
{
  "action_type": "final_answer",
  "decision_summary": "Brief English explanation of the selected
  events and evidence sufficiency, or the limitations and uncertainty
  when finalization is forced.",
  "optimal_segments": [
```

```
    {
      "start_frame": "integer; original video frame index copied from
the shown frame label, e.g. 1628; segment start frame",
      "end_frame": "integer; original video frame index copied from
the shown frame label, e.g. 1628; segment end frame",
      "start_time_sec": "number or string; decimal seconds from the
original video start, matching the associated frame index divided by
the episode FPS; do not use MM:SS; segment start time",
      "end_time_sec": "number or string; decimal seconds from the
original video start, matching the associated frame index divided by
the episode FPS; do not use MM:SS; segment end time; keep the event
precise, short, and non-overlapping",
      "subtask": "English string or null",
      "reason": "English evidence-grounded reason naming the
successful alignment or other visible, nontrivial task-progress
event.",
      "evidence_frames": [
        "integer; original video frame index copied from the shown
frame label, e.g. 1628; evidence frames"
      ],
      "evidence_times_sec": [
        "number or string; decimal seconds from the original video
start, matching the associated frame index divided by the episode
FPS; do not use MM:SS; evidence timestamps matching evidence_frames"
      ],
      "progress_score": "0.45-1; use 0.45-0.55 for modest but visible
task-relevant progress",
      "confidence": "0 to 1"
    }
  ],
  "suboptimal_segments": [
    {
      "start_frame": "integer; original video frame index copied from
the shown frame label, e.g. 1628; segment start frame",
      "end_frame": "integer; original video frame index copied from
the shown frame label, e.g. 1628; segment end frame",
      "start_time_sec": "number or string; decimal seconds from the
original video start, matching the associated frame index divided by
the episode FPS; do not use MM:SS; segment start time",
      "end_time_sec": "number or string; decimal seconds from the
original video start, matching the associated frame index divided by
the episode FPS; do not use MM:SS; segment end time; ordinary
suboptimal segments stay tight, but progress_effect=failure must
cover from immediately before failure onset through the end of the
failure and has no length limit",
      "subtask": "English string or null",
      "reason": "English evidence-grounded reason explaining slowdown,
regression, instability, or rework.",
      "evidence_frames": [
        "integer; original video frame index copied from the shown
frame label, e.g. 1628; evidence frames"
      ],
      "evidence_times_sec": [
        "number or string; decimal seconds from the original video
start, matching the associated frame index divided by the episode
FPS; do not use MM:SS; evidence timestamps matching evidence_frames"
      ],
      "progress_effect": "slows_task | regresses_task | failure |
unclear",
      "severity": "0 to 1; use low severity for mild locally visible
inefficiency without failure or regression",
      "progress_score": "-1 to 0",
      "confidence": "0 to 1; use lower confidence for mild but
visually grounded local suboptimal evidence"
    }
```

```
  ],
  "suboptimal_search_audit": [
    {
      "checked_frame_range": [
        "start frame index, e.g. 303",
        "end frame index, e.g. 546"
      ],
      "checked_time_range_sec": [
        "start frame index divided by the episode FPS",
        "end frame index divided by the episode FPS"
      ],
      "result": "found_suboptimal | no_clear_issue | uncertain",
      "note": "English note explaining what was checked and why it
  does or does not support suboptimal."
    }
  ],
  "backfilled_from_observed_segments": "boolean; return false. Events
  must be explicitly submitted; intermediate findings are never
  automatically backfilled",
  "summary": "English concise final summary.",
  "confidence": "0 to 1"
}
```

Listing 23: OVERVIEW_RESPONSE_SCHEMA: Overview response envelope

```
{
  "action_type": "crop_video",
  "decision_summary": "Brief English state update without
  chain-of-thought.",
  "objective_findings": [
    <SEARCH_FINDING_SCHEMA>,
    "Return only independent, decision-relevant search candidates from
  the coarse overview. They are hypotheses for choosing crops, not
  final labels; consolidate adjacent views of the same event. An empty
  list is valid when the overview reveals no useful candidate.
  Serialization safety limit: 12 items. This is not a final annotation
  quota."
  ],
  "crop_video": <CROP_REQUEST_SCHEMA>
}
```

Listing 24: CROP_RESPONSE_SCHEMA: Required-crop response envelope

```
{
  "action_type": "crop_video",
  "decision_summary": "Brief English state update without
  chain-of-thought.",
  "objective_findings": [
    <LOCAL_FINDING_SCHEMA>,
    "Return independent evidence-linked findings from the current
  observation, not speculative final labels. Return only
  evidence-linked findings from this observation; consolidate repeated
  views of the same event. An empty list is valid when no
  objective-relevant evidence is present. Serialization safety limit:
  12 items. This is not a final annotation quota."
  ],
  "crop_video": <CROP_REQUEST_SCHEMA>
}
```

Listing 25: DECISION_RESPONSE_SCHEMA: Ordinary decision envelope with both auxiliary tools enabled

```
{
```

```
  "action_type": "crop_video | inspect_action_raw | supply_camera |
  final_answer",
  "decision_summary": "Brief English state update without
  chain-of-thought.",
  "objective_findings": [
    <LOCAL_FINDING_SCHEMA>,
    "Return compact evidence-linked observation or contradiction
  findings before choosing the next action. Return only evidence-linked
  observation or contradiction findings; an empty list is valid."
  ],
  "crop_video": {
    "start_frame": "integer; original video frame index copied from
  the shown frame label, e.g. 1628; crop start frame",
    "end_frame": "integer; original video frame index copied from the
  shown frame label, e.g. 1628; crop end frame",
    "start_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; crop start time",
    "end_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; crop end time; focused and never
  longer than one third of the original video",
    "sampling_strategy": "coarse | medium | fine; intent only, all
  ordinary crops use 4 FPS subject to the same 144-image cap; prefer a
  short focused window for contact/alignment",
    "reason": "English reason this segment may contain
  optimal/suboptimal evidence.",
    "expected_evidence": "English description of what to inspect."
  },
  "final_answer": <FINAL_RESPONSE_SCHEMA>,
  "inspect_action_raw": <ACTION_REQUEST_SCHEMA>,
  "supply_camera": <CAMERA_REQUEST_SCHEMA>
}
```

Listing 26: `REFINEMENT_RESPONSE_SCHEMA`: Boundary-refinement response schema

```
{
  "action_type": "boundary_refinement",
  "segment_id": "copy the provided segment_id",
  "label": "copy the provided label; do not change optimal/suboptimal",
  "refined_segment": {
    "start_frame": "integer; original video frame index within the
  inspected window; inclusive corrected start frame copied from a shown
  label; for failure include the immediately preceding transition",
    "end_frame": "integer; original video frame index within the
  inspected window; inclusive last event frame copied from a shown
  label; for failure preserve the complete direct consequences",
    "start_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; corrected tight start time",
    "end_time_sec": "number or string; decimal seconds from the
  original video start, matching the associated frame index divided by
  the episode FPS; do not use MM:SS; corrected tight end time",
    "evidence_frames": [
      "integer; original video frame index copied from the shown frame
  label, e.g. 1628; shown frames inside the revised interval that
  justify the corrected boundaries"
    ],
    "evidence_times_sec": [
      "number or string; decimal seconds from the original video
  start, matching the associated frame index divided by the episode
  FPS; do not use MM:SS; timestamps matching evidence_frames"
    ],
```

```
    "boundary_note": "English note explaining why the boundary did or
  did not move.",
    "confidence": "0 to 1; confidence in the corrected boundary only"
  }
}
```

An ordinary decision places the final event object under `final_answer`; a forced-final prompt requests the final object directly. The legacy `backfilled_from_observed_segments` field is reproduced as it appears in the model-facing schema, but the finalizer does not automatically promote intermediate findings into labels and writes this field as false. Similarly, the refinement prompt's request to copy the category does not authorize changing it: the returned category identifies the candidate, while only the permitted fields inside `refined_segment` may update the stored event.

Conditional episode-specific insertions. The core templates above omit empty optional blocks. When supplied, dataset priors, a task-specific rubric, and fixed-annotation context are inserted after the length rule in overview and iterative prompts. Initial crop-navigation references and objective runtime context can additionally be inserted in the overview before final-output guidance. Such content depends on the source episode and configuration and is recorded in its prompt snapshots. It must not be interpreted as a universal ground-truth annotation source. Parent–child clarification prompts and task-decomposition prompts belong to optional execution paths and are outside this quality-only template specification.

A.7 Direct VLM Baseline Prompt

Relationship to SCHEMA. The direct VLM baseline reuses SCHEMA's registered quality rubric, event definitions, recovery rule, failure-coverage exception, and event-count and ordinary-duration limits. Tool-dependent instructions are adapted to a single pass over the supplied chronological overview frames. In particular, the baseline is allowed to make final judgments from these frames, rather than being asked to request unavailable local crops. The annotation criteria are shared; the interaction protocol and cumulative inference cost differ between the methods.

The baseline also reuses the overview sampling implementation and the deterministic event-normalization and overlap-selection functions. It does not perform iterative evidence acquisition, model-based response repair, or boundary refinement. Network retries and API-format fallbacks do not introduce a semantic feedback loop. Frame caps and completion budgets are run-specific settings and must be distinguished from the shared prompt policy.

Inputs and placeholders. The following is the implemented prompt with the optional dataset-failure prior disabled, which is the default setting. The model receives the task instruction and sampled video frames with their original indices and timestamps. Reference event annotations are kept outside the model input. The question is assembled from the episode's task and temporal metadata at inference time. Here T is the episode frame count, f (`FPS`) is its frame rate, and `DURATION_SECONDS`$= T/f$. The ordinary-event limit is `MAX_EVENT_FRAMES`$= \max\{1, \lfloor T/3 \rfloor\}$, with `MAX_EVENT_SECONDS`$=$ `MAX_EVENT_FRAMES`$/f$. The implementation displays FPS and duration to six decimal places and the duration limit to two decimal places. These placeholders are replaced with episode-specific values before inference.

Listing 27: Direct VLM system message

```
You are a precise robotics video quality analyst. Analyze only the
supplied task text, chronological video frames, and their timestamps.
Return the requested structured result without chain-of-thought. All
reasons and summaries must be concise English grounded in visible
evidence.
```

Listing 28: Direct VLM question with dataset-failure prior disabled

```
Analyze this robot manipulation episode directly from a chronological,
uniformly sampled full-video overview. Return the final optimal and
```

```
suboptimal time ranges in one response; no additional views or tools
are available. Inspect the complete sequence before deciding.

Goal: Your goal is to find distinct visually supported optimal
progress intervals (retaining at most 6), complete visually supported
failure intervals, and representative ordinary suboptimal intervals,
then explicitly select events under the count and overlap rules and
explain why.

Evidence policy:
Video-first evidence hierarchy: task text and video frames are
authoritative for semantic labels. Only task text and visual frames
are supplied; base navigation, boundary, and semantic decisions on
them. Pair low-progress motion, repeated correction, arm dominance, or
gripper open/close with a visible task consequence before calling it
suboptimal: failed or ineffective contact, delayed primary task
progress, collision/interference, object slip/drop, regression,
unnecessary rework, or prolonged task-relevant stall. For dual-arm
tasks, infer the likely primary operator, secondary/support operator,
and unrelated or idle arm from task text plus visual context. Do not
penalize secondary or unrelated arm low progress unless it visibly
harms the primary task. Final reasons should name the visible task
consequence rather than unsupported latent signals.

Prefer high recall for suboptimal discovery while staying visually
evidence-grounded. Mark suboptimal when visible evidence shows
task-relevant hesitation, low-progress motion, repeated correction,
off-target motion, slightly inefficient path choice, failed or
ineffective contact attempts before recovery, unstable contact/grasp
before it is corrected, task regression, or rework. Temporal context
may identify a candidate window, but it does not establish the final
label without a visible task consequence. Retain mild suboptimal
candidates when visible evidence supports an execution consequence,
even if severity or confidence is low. Slowness, correction, indirect
motion, or instability without a visible task consequence is
insufficient for a final suboptimal label. Borderline slowdowns, short
hover-before-contact, small re-alignment loops, partial
withdraw-and-retry behavior, and minor loss of momentum can all count
when they visibly delay or disrupt the next task step; plausible delay
alone remains uncertain. A suboptimal segment does not need to cause
final task failure or state regression: the supplied chronological
overview frames are sufficient when they show a visible task delay,
disruption, or instability caused by the slower motion, indirect path,
repeated correction, or stall. Keep such evidence with low
severity/confidence when appropriate. Use uncertain or omit a
candidate when no visible task consequence supports a suboptimal
label. Neither smooth motion nor inactivity alone establishes quality.
Do not penalize support-arm inactivity, harmless adjustment, or a
harmless one-off adjustment that does not visibly delay or destabilize
task progress.

Suboptimal search requirement: actively look for at least one
plausible suboptimal candidate before concluding that none exists.
Inspect moments around approach, alignment, grasp/contact,
insertion/cut/place, release, failed-contact attempts and the
following recovery-to-progress boundary, and transitions between
subtasks. Useful negative evidence includes hovering without contact,
approach-then-withdraw, repeated small corrections, object slip or
loss of alignment, low-progress motion despite ongoing robot movement,
moving away from the target, slightly hesitant contact initiation,
failed or unproductive contact before a later recovery, or redoing a
previously achieved state. The failed attempt before recovery is the
suboptimal evidence; the successful recovery after failed contact
should be considered for a separate optimal segment. Retain a mild
slowdown with appropriately lower severity/confidence when the
```

```
supplied frames show a visible task delay or disruption. If the
consequence remains ambiguous, use uncertainty or omit the event.
Select qualifying events under the final count and overlap rules; a
plausible candidate is not a mandatory final label. Treat ambiguous
sparsely sampled motion as uncertain rather than forcing a final label
without a visible task consequence. Either final list may be empty
after inspecting the supplied frames. In suboptimal_search_audit,
describe suspicious intervals actually inspected and whether they
support a negative label or remain uncertain; disclose sparse-sampling
limitations. No additional crop, exhaustive evidence gate, or
negative-event quota is required.

Failure-event coverage requirement: use progress_effect='failure' for
actual failure events supported by the supplied visual evidence,
including local failures in an otherwise successful episode.
Explicitly submit supported failure events for failure-priority
selection; overlapping failure candidates still compete under the
final overlap rules. Start each such interval at the last moment
immediately before the failure becomes visible, so the boundary
includes the transition from normal execution into failure. End only
when the failure event and its direct consequences have fully ended.
If the failed state remains unresolved through the end of the video,
end at the last relevant frame. Cover the complete continuous failure
episode rather than only a short symptom or evidence frame. A
failure-coverage suboptimal segment is exempt from the one-third-video
maximum and has no duration limit: never truncate, split, or discard
it solely because it is long. This exception does not justify
unrelated lead-in or trailing context; the boundaries must still be
the complete, evidence-based failure boundaries. Ordinary hesitation
or inefficiency that is not the actual failure remains subject to the
normal tight-segment limit.

Optimal labeling requirement: identify every distinct, visually
confirmed successful alignment and every other distinct task-relevant
action
that visibly advances the task or a named subtask. Progress may be
modest when it is concrete and visually supported; it does not need to
complete the entire subtask. In the final answer, retain at most 6 of
the strongest distinct optimal events, prioritizing the clearest,
highest-confidence, and most task-relevant progress; never exceed this
cap. Good optimal evidence includes achieving a new task state such
as stable grasp after contact, lifting the target, successfully
inserting, placing, hanging, pressing, or cutting, releasing after
successful placement, recovering from failure or a stall into
effective forward progress, or completing a required subtask. Also
include a
short transport interval when it visibly advances a held object toward
its task-relevant target area; keep only the evidence-bearing
displacement, not the entire transit. Treat a gripper movement or
adjustment from an offset/misaligned pose into a task-relevant pose at
the
target position or object as optimal when it visibly completes
alignment, reaches the contact/grasp/manipulation pose, or directly
enables
the next required task step. This includes the final alignment from a
pose that is already near the expected position but still offset into
the correct expected pose. For a transition to the next subtask, label
the terminal alignment that reaches the correct pose and any separate
evidence-bearing transport that visibly advances a held object toward
its task-relevant target. Exclude unrelated transit and broad approach
context. Do not label a segment optimal merely because motion is
smooth, direct, or error-free. Pure approach, searching, hovering,
waiting,
repositioning, maintaining a hold, moving through free space without
changing the task state or completing target/object alignment, an
```

```
initial reach from the task-start pose into the general workspace, or
a normal transition between subtasks should be labeled uncertain or
omitted unless it produces clear task progress. If a smooth segment
only prepares for progress but does not visibly reduce the remaining
task-relevant distance, move a held object toward its target, change
the task state, or resolve a task-relevant gripper-to-target/object
offset, do not include it as optimal. In the final answer, each
optimal reason must explicitly say what progress was achieved and why
it is
more than ordinary motion. Keep every selected optimal segment as
short and boundary-accurate as possible: for alignment, start when the
final task-relevant correction toward the correct pose begins and end
as soon as that alignment is achieved; for another critical progress
event, cover only the decisive progress-producing action. Exclude
unrelated broad approach or transit and later holding or transition
frames. Preserve evidence-bearing, target-directed object transport
when it is the progress event being labeled. Optimal segments must
never
overlap one another or any suboptimal/failure segment; split
boundaries at the semantic transition and keep distinct progress
events as
distinct segments.

Recovery-to-progress rule: a broader interval may contain both
suboptimal and optimal evidence. If the robot first hesitates, slips,
misaligns, searches, repeats corrections, or otherwise loses progress,
mark that earlier portion as suboptimal. But if a later, short portion
of the same local episode visibly resolves the problem and returns the
task to normal progress, that recovery portion should be split out and
labeled optimal. After failed contact, label the failed/ineffective
contact attempt, withdrawal, or repeated correction before the fix as
suboptimal; label the recovery after failed contact as optimal once
contact becomes stable/productive or direct task progress resumes.
Examples include the moment a corrected grasp becomes stable, a
misaligned object becomes aligned, contact finally becomes productive
after failed attempts, the robot exits a correction loop and moves the
object toward the target, or a previously stalled subtask begins
advancing again. Keep the recovery optimal segment short: start at the
first timestamp where the correction clearly succeeds or forward
progress resumes, and end when that immediate recovery event is
complete. Do not merge the preceding inefficient motion into the
optimal segment.

Final segment consolidation requirement: before returning the final
structured result, review all candidate intervals in chronological
order. If adjacent or slightly overlapping segments have the same
label, describe the same continuous event, share the same
subtask/progress_effect, and are separated only by a tiny gap or
sampling artifact, merge them into one tight interval with a single
evidence-grounded reason. Do not leave many tiny fragments for one
sustained hesitation, repeated correction loop, unstable contact
episode, or continuous progress-producing action. Do not merge across
different subtasks, across a clear recovery-to-progress transition, or
across a gap where the robot visibly resumes normal productive motion.
The merged segment must still obey the maximum segment length rule
unless it is a complete progress_effect='failure' suboptimal interval.
Never merge distinct optimal progress events, and never merge in a way
that creates overlap. If a non-failure event would become too broad,
keep only the most evidence-bearing representative interval and
explain the specific event.

Segment length requirement: keep every final optimal or suboptimal
segment as short and boundary-accurate as possible. A final segment
should cover only the supported event. Inspection context helps
interpret that event but is not automatically part of its label. Do
```

```
not merge multiple subtasks, long approach phases, or broad
before/after context into one segment. For this episode, every optimal
or non-failure suboptimal segment may use at most <MAX_EVENT_FRAMES>
frames (about <MAX_EVENT_SECONDS> seconds), computed as max(1,
floor(T/3)) for the original <T>-frame video. The sole exception is a
suboptimal segment with progress_effect='failure': it must run from
the last moment immediately before failure onset through the complete
end of the failure and has no length limit. Never shorten a complete
failure interval to satisfy the one-third limit. For a non-failure
event that appears longer than the limit, report only the most
evidence-bearing sub-interval. Use inclusive frame intervals: start at
the first supported event frame and end at the last frame belonging to
the event or its direct consequences, not the first subsequent
non-event frame. Failure intervals also include the immediately
preceding transition frame. A single-frame event is valid; unlabeled
gaps may remain.

Direct-decision requirements:
- Compare clear task-progress events against suspicious slowdowns,
failed contacts, corrections, stalls, instability, regressions, and
rework across all major task phases.
- Treat the shown original frame indices (0 to T-1) as authoritative.
Use inclusive intervals with length end_frame-start_frame+1. Copy
boundary and evidence frames from the timestamp labels, and provide
matching decimal seconds as frame_index/FPS.
- Optimal and suboptimal intervals must be tight, chronological, and
mutually non-overlapping. Split a later recovery-to-progress interval
from preceding suboptimal behavior.
- Retain at most 6 distinct optimal events and at most 5 ordinary
suboptimal events, subject to failure-priority ranking and overlap
selection. Failure intervals have no count cap, but overlapping
failure candidates still compete.
- Either class may be empty. Audit suspicious intervals using only the
supplied frames, preserving uncertainty where sampling is
insufficient; do not invent a segment to fill a quota.
- Explicitly submit selected events with nonempty evidence-grounded
reasons. Consolidate repeated views of the same continuous event.
Final selection retains failure intervals first, with no failure count
cap but with overlapping failures competing; it then retains at most 5
ordinary suboptimal events and at most 6 optimal events. The harness
applies confidence/progress/severity ranking and overlap resolution;
observing an event does not guarantee its retention. Preserve complete
supported failure boundaries without unrelated context and separate
productive recovery. Either class may be empty, and uncovered frames
remain unlabeled. Never create a segment to satisfy a quota.

Task: <TASK_INSTRUCTION>
Episode length: <T> frames
FPS: <FPS>
Approximate duration: <DURATION_SECONDS> seconds

Use the structured output fields to return the final answer.
```

API-specific message suffixes. The Chat Completions backend used by the local Qwen baseline appends the following text to the question. The output-field instructions are retained even when an endpoint requires a fallback from strict JSON Schema to JSON-object mode or unconstrained JSON prompting.

Listing 29: Chat Completions suffix and output-field contract

```
The chronological video frames follow. Every image is preceded by its
authoritative original-video frame index and decimal-second timestamp.
Return only the JSON object requested by the schema.
```

```
Required JSON object keys: decision_summary, optimal_segments,
suboptimal_segments, suboptimal_search_audit, summary, confidence.
Every optimal segment must contain start_frame, end_frame,
start_time_sec, end_time_sec, subtask (string or null), reason,
evidence_frames, evidence_times_sec, progress_score, and confidence.
Every suboptimal segment must contain the same fields plus
progress_effect (slows_task, regresses_task, failure, or unclear) and
severity. Every suboptimal_search_audit item must contain
checked_frame_range, checked_time_range_sec, result (found_suboptimal,
no_clear_issue, or uncertain), and note. Use no additional keys.
```

The Responses backend uses the same question and system instruction but appends the shorter suffix below; its structured output schema is supplied through the API's format field.

Listing 30: Responses backend suffix

```
The chronological video frames follow. Every image is preceded by its
authoritative original-video frame index and decimal-second timestamp.
```

For either backend, each image is preceded by `Frame image j/N from original video: f=FRAME t=SECONDS seconds`, where the timestamp is displayed to six decimal places. The images follow in chronological order.

Structured output and postprocessing. In strict-schema mode, the API schema requires the six top-level fields listed above and disallows additional keys. Optimal events contain frame/-time endpoints, a nullable subtask, a nonempty reason, evidence frames/timestamps, a progress score, and confidence. Suboptimal events additionally contain severity and a progress effect from `slows_task`, `regresses_task`, `failure`, or `unclear`. Confidence and severity lie in $[0, 1]$; the baseline schema permits progress scores in $[-1, 1]$ for either category. It caps the optimal list at six entries. Failure exceptions and the five-event cap for ordinary suboptimal events are handled by the shared deterministic normalizer rather than by a cap on the entire suboptimal array.

The event information is aligned with SCHEMA, but the API's formal schema is not identical to SCHEMA's descriptive prompt schema. In particular, SCHEMA additionally supplies category-specific numerical guidance for progress scores and mild-event confidence or severity. The direct output does not include tool-selection or intermediate-finding fields. After parsing, the same frame-authoritative normalization, failure-priority selection, and overlap rules described in Appendix A.4 are applied, without the subsequent model-based boundary-refinement stage.

A.8 Inference Usage Accounting

We record cumulative input and output token usage during RTQ-Bench evaluation for transparency. SCHEMA (Qwen3.6-35B-A3B) uses 108.9M input tokens and 5.4M output tokens, compared with 12.0M/1.4M for single-pass Qwen3.6-35B-A3B, 13.7M/3.6M for GPT-5.6 Sol (High), and 13.1M/1.9M for Cosmos3-Super (65B).

The open-weight models are evaluated with local deployment; therefore, we report token usage rather than measured monetary cost, which depends on hardware configuration and serving utilization. For reference, we estimate API-equivalent costs from the recorded token counts using the OpenRouter rate for Qwen-3.6-35B-A3B and the official OpenAI API rate for GPT-5.6 Sol (High), as accessed on September 26, 2026.[1] Over the full RTQ-Bench, these rates yield reference costs of $21.63 for SCHEMA and $126.80 for GPT-5.6 Sol. Qwen-based inference was performed locally, so the former is an API-equivalent estimate rather than a measured self-hosting cost.

[1] Qwen-3.6-35B-A3B pricing: `https://openrouter.ai/qwen/qwen3.6-35b-a3b`. GPT-5.6 Sol pricing: `https://developers.openai.com/api/docs/pricing`.

B TEMPORAL QUALITY EVALUATION METRICS

This appendix specifies the evaluation protocol used in Table 2. We evaluate optimal and suboptimal intervals separately. Boundary-Softened IoU (BS-IoU) measures temporal overlap, while Center-Constrained Recall (CC-Recall) measures recovery of annotated events under center and overlap constraints.

B.1 NOTATION AND TEMPORAL COORDINATES

For interval-format predictions and references stored with inclusive frame endpoints $[s, e]$, the indices identify the first and last boundary frames. For evaluation, these endpoints are interpreted as frame timestamps, mapping $[s, e]$ to $[s/f_i, e/f_i]$; no additional frame period is added to the end. Thus, inclusivity refers to the discrete frame-index representation rather than temporal-cell coverage. Episode duration is $T_i = F_i/f_i$, where F_i is the frame count. Predictions are matched to references by episode identity, and their frame rates and frame counts must agree. For frame-selection outputs, a selected frame k instead represents the temporal cell $[k/f_i, (k+1)/f_i)$, with adjacent selected cells joined into intervals before scoring. All reported metrics are percentages.

B.2 RATIONALE FOR BS-IOU AND CC-RECALL

RTQ-Bench evaluates sparse quality events intended to support policy learning, rather than an exhaustive frame-level partition. Annotators may agree on an ineffective attempt while placing its onset at the initial contact or a later visible consequence, and its endpoint at different points in the transition to recovery. Hard IoU is sensitive to such differences: shifting a one-second interval by 0.2 seconds reduces IoU to $0.8/1.2 \approx 0.67$. BS-IoU softens both reference and predicted boundaries while retaining penalties for missing or excessive temporal coverage. Its duration-dependent tolerances are fixed across methods and are not tuned per model.

Under metadata-guided sampling, missed suboptimal regions retain their base weight, whereas productive regions mistakenly labeled suboptimal are downweighted but remain available with nonzero weight. This motivates explicitly measuring suboptimal-event recovery, without assuming that false positives are harmless or always less costly. CC-Recall measures recovery under center and overlap constraints, allowing same-category grouping across bounded gaps on both sides to accommodate fragmentation. It counts recovered original reference segments, not matched groups. Category-specific CC-Recall keeps suboptimal-event recovery explicit; Micro pools both categories. The gains from adding failed rollouts in Section 4.4 support the importance of using failed experience effectively, but do not isolate localization errors or establish their relative costs.

BS-IoU retains IoU's overlap-based structure, and CC-Recall also uses temporal IoU for matching. For binary temporal masks on the same domain with nonzero denominators, frame-level F1 and hard IoU J satisfy $\mathrm{F1} = 2J/(1+J)$: at the episode–category level, F1 is a monotonic transformation of hard overlap, rather than a separate measure of event recovery. This does not imply identical rankings after averaging. Event-level F1 additionally requires split/merge matching and counting conventions. Reporting recovery separately keeps missed events explicit. Standard IoU and explicitly specified F1 variants remain valid complementary measures; our choice concerns boundary treatment, event matching, and aggregation, rather than universal superiority.

CC-Recall does not penalize unmatched predictions and must be read alongside BS-IoU, which penalizes disagreement in temporal coverage. All and GT+ distinguish agreement across all episodes, including empty reference lists, from overlap where reference events are present. Because unlabeled regions are not verified negatives, an unmatched prediction need not be a semantic error. BS-IoU still penalizes coverage outside retained reference intervals and does not resolve reference incompleteness. Together, the metrics assess agreement with the sparse annotation protocol and recovery of its selected events.

B.3 BOUNDARY-SOFTENED IOU

Soft temporal masks. For BS-IoU, intervals are clipped to $[0, T_i]$, and overlapping or touching intervals of the same category are replaced by their union. The positive-gap grouping introduced for CC-Recall below is not applied to BS-IoU. For an interval $I = [a, b]$, define its distance from time t

as

$$d(t, I) = \max\{a - t,\ 0,\ t - b\}. \tag{12}$$

The softened membership of I is

$$\phi_i(t; I) = \begin{cases} 1, & d(t, I) \le \delta_i, \\ 1 - \dfrac{d(t, I) - \delta_i}{\lambda_i}, & \delta_i < d(t, I) < \delta_i + \lambda_i, \\ 0, & d(t, I) \ge \delta_i + \lambda_i, \end{cases} \qquad t \in [0, T_i]. \tag{13}$$

Here δ_i is the full-credit padding outside each boundary, and λ_i is the additional linear-decay width. Both are in seconds and depend only on episode duration:

$$(\delta_i, \lambda_i) = \begin{cases} (0.2, 0.3), & 0 < T_i \le 10, \\ (0.3, 0.6), & 10 < T_i \le 20, \\ (0.5, 1.0), & T_i > 20. \end{cases} \tag{14}$$

For an interval collection S, its soft union is

$$\mu_i(t; S) = \max_{I \in S} \phi_i(t; I), \qquad \mu_i(t; \varnothing) = 0. \tag{15}$$

The same softening is applied to predictions and references. Taking a maximum, rather than a sum, prevents overlapping intervals from receiving repeated temporal credit.

Episode–category score. For a successfully processed episode, define

$$J_{i,c} = \begin{cases} 1, & P_{i,c} = G_{i,c} = \varnothing, \\ 0, & \text{exactly one of } P_{i,c}, G_{i,c} \text{ is empty}, \\ \dfrac{\int_0^{T_i} \min\{\mu_i(t; P_{i,c}), \mu_i(t; G_{i,c})\}\, dt}{\int_0^{T_i} \max\{\mu_i(t; P_{i,c}), \mu_i(t; G_{i,c})\}\, dt}, & \text{otherwise}. \end{cases} \tag{16}$$

In particular, an empty reference with a nonempty prediction receives zero, not partial credit. The integrals are evaluated over the piecewise-linear masks using their breakpoints and intersections, rather than by sampling a fixed temporal grid.

All-episode and category-positive aggregation. Let N be the number of evaluation episodes, and define $\mathcal{I}_c^+ = \{i : n_{i,c} > 0\}$ and $N_c^+ = |\mathcal{I}_c^+|$. Then

$$\mathrm{BSIoU}_c^{\mathrm{All}} = \frac{100}{N} \sum_{i=1}^{N} J_{i,c}, \tag{17}$$

$$\mathrm{BSIoU}_c^{\mathrm{GT+}} = \frac{100}{N_c^+} \sum_{i \in \mathcal{I}_c^+} J_{i,c}. \tag{18}$$

For either scope $s \in \{\mathrm{All}, \mathrm{GT+}\}$,

$$\mathrm{BSIoU}_{\mathrm{Macro}}^{s} = \frac{1}{2} \sum_{c \in \mathcal{C}} \mathrm{BSIoU}_c^{s}. \tag{19}$$

Thus, GT+ first averages within each category and then weights the two categories equally; it does not pool all category-positive pairs into one mean. GT+ membership depends only on the reference annotations and is identical across methods. Episodes are not weighted by duration or by the number of annotated segments.

B.4 Center-Constrained Recall

Admissible interval matches. For $I = [a, b]$, write $\mathrm{ctr}(I) = (a + b)/2$ and $\ell(I) = b - a$. A predicted interval p and a positive-length reference interval g satisfy the matching criterion if

$$\frac{|\,\mathrm{ctr}(p) - \mathrm{ctr}(g)|}{\ell(g)/2} \le \alpha, \qquad \mathrm{tIoU}(p, g) = \frac{|p \cap g|}{|p \cup g|} \ge \tau, \quad \alpha = 0.5, \quad \tau = 0.1. \tag{20}$$

Interval-set sizes in tIoU denote temporal lengths. The center distance is normalized by the reference half-length; with $\alpha = 0.5$, the allowed center displacement is at most one quarter of the reference length. These tests use hard intervals, not the softened masks used by BS-IoU.

Table 4: Evaluation denominators for Table 2.

Category	All episodes	GT+ episodes	Empty-GT episodes	Original GT segments
Optimal	1,350	1,176	174	2,973
Suboptimal	1,350	531	819	584

Optional grouping on both sides. To accommodate temporally fragmented annotations and predictions, we consider both singleton intervals and contiguous groups of same-category intervals on each side. Within an episode–category pair, sort intervals by start time, then end time, and finally original index to break ties. A group can be extended by the next interval if its start is at most $\gamma = 0.5$ seconds after the maximum end of the intervals already in the group. Thus, overlapping and touching intervals are also eligible. Each group is represented by its temporal hull, from the earliest start to the latest end, including any intervening gaps. Grouping is optional: singleton intervals remain candidates, and no maximal merge is imposed in advance.

Let U and V denote groups of original reference and prediction indices, respectively, and let $h_G(U)$ and $h_P(V)$ denote their hulls. A pair (U, V) is admissible when its hulls satisfy Equation 20. Among all sets of admissible pairs, choose a matching maximizing the number of covered original reference intervals:

$$M_{i,c} = \max_{\mathcal{A}_{i,c}} \sum_{(U,V)\in\mathcal{A}_{i,c}} |U|, \tag{21}$$

subject to $U \cap U' = \varnothing$ and $V \cap V' = \varnothing$ for every pair of distinct matches $(U, V), (U', V')$. These constraints prevent reuse of an original reference or prediction segment; they concern segment indices rather than temporal overlap. Matched pairs need not preserve temporal order across the two sides. The matching is computed exactly by dynamic programming over reference groups and used prediction indices. It maximizes original-reference coverage rather than the number of matched groups.

Under the endpoint-timestamp convention in Appendix B.1, a single-frame reference $[k, k]$ has zero temporal duration. Such entries are retained in the original count. They cannot form a valid singleton reference match because center normalization requires positive length, but can be covered within an admissible positive-length group. Their softened masks remain defined by Equation 13 for BS-IoU.

For example, a reference interval $[2, 6]$ can match the group formed by predictions $[2, 3.8]$ and $[4.2, 6]$, whose gap is 0.4 seconds. This recovers one reference interval. Conversely, if those two short intervals are references and $[2, 6]$ is the prediction, matching the reference group recovers two original reference intervals.

Segment-level aggregation. The reported category recall and overall micro recall are

$$\mathrm{CCR}_c = 100\frac{\sum_i M_{i,c}}{\sum_i n_{i,c}}, \tag{22}$$

$$\mathrm{CCR}_{\mathrm{Micro}} = 100\frac{\sum_{c\in\mathcal{C}}\sum_i M_{i,c}}{\sum_{c\in\mathcal{C}}\sum_i n_{i,c}}. \tag{23}$$

The denominator always counts original reference intervals, not merged groups. These quantities are not episode-level averages, and Micro is not an equal average of the two category recalls. An empty-reference pair contributes neither hits nor denominator, regardless of its predictions. Unmatched predictions do not directly reduce recall; BS-IoU supplies the complementary overlap assessment.

B.5 Evaluation Scope and Failure Handling

All methods are evaluated on the same final set of 1,350 reference episodes. Table 4 gives the evaluation denominators used in Table 2. The training-manifest count in Appendix C.1 describes value-model adaptation data and does not change these evaluation denominators.

The score-based baselines are converted into temporal selections before BS-IoU evaluation. For Robo-Dopamine 2.0 and Robometer, the shared progress-to-label conversion is specified in Appendix C.2. GT+ selection is solely an evaluation restriction; it does not modify predictions using

reference labels. We do not report CC-Recall for the score-based baselines under the current protocol; the dashes in Table 2 indicate an unreported metric, not a zero score or a mathematical inability to evaluate converted intervals.

C RTQ-BENCH BASELINE IMPLEMENTATION DETAILS

This section describes the value- and progress-based baselines evaluated on RTQ-Bench. We first specify the RECAP value-model adaptation and advantage-based frame selection, then describe Robo-Dopamine 2.0 and Robometer with their shared progress-to-label conversion.

C.1 RECAP VALUE-BASED BASELINE

For RTQ-Bench, we adapt the value-function component of RECAP (Physical Intelligence et al., 2025a), the learning framework used by $\pi^*_{0.6}$, to temporal quality localization. The baseline comprises value-model pretraining, additional training on benchmark-related tasks, and offline advantage estimation followed by frame selection. Table 2 reports this localization baseline as **RECAP (adapted)**. This benchmark experiment evaluates the adapted value model and its frame-selection decoder, not a RECAP policy. Real-world RECAP policy learning is described separately in Appendix F.

C.1.1 VALUE MODEL AND PRETRAINING

We pretrain our own value model on the full Open X-Embodiment, AgiBot-World, and RoboCOIN datasets using the same distributional vision-language value-function architecture as RECAP (Physical Intelligence et al., 2025a). We denote the resulting pretrained checkpoint by V_{pre}. This checkpoint initializes two separate downstream branches: benchmark-task adaptation (Appendix C.1.2) and real-world RECAP learning (Appendix F). The implementation combines a SigLIP2-Base (Tschannen et al., 2025) visual encoder, a learned visual projector, a Gemma-3-270M (Team et al., 2025) language backbone, and a distributional value head. Images are resized to 224×224 pixels. Inputs contain the task instruction and the available main and wrist camera views at the current time. Missing wrist views are masked; robot-state prompts and visual history are disabled.

The model predicts a distribution $p_\phi(b \mid o_i, \ell)$ over $B = 201$ uniformly spaced values $z_b \in [-1, 0]$, where o_i is the visual observation and ℓ is the task instruction. The scalar prediction is

$$V_i = \sum_{b=1}^{B} p_\phi(b \mid o_i, \ell) z_b. \tag{24}$$

Value training minimizes cross-entropy against a two-hot encoding of the normalized return target: probability mass is linearly distributed between its two neighboring support values. The visual and language backbones are updated during benchmark adaptation.

C.1.2 BENCHMARK-TASK ADAPTATION

Starting from V_{pre}, we fine-tune the value model on the benchmark-related corpus specified below.

Episode selection. For each source dataset and referenced task ℓ, let n_ℓ denote its reference episode count. The selector targets $10n_\ell$ training episodes, prioritizing matches to the normalized task identity or instruction. Sampling uses seed 42 and selects distinct episodes within each source dataset. If task-matched episodes are insufficient, the remaining budget is filled from unused episodes in the same dataset. A dataset can remain below its target when its inventory is exhausted. Thus, $10\times$ specifies a requested expansion, rather than a guarantee of ten additional same-task episodes for every evaluation episode.

The recorded manifest was constructed from $1{,}350$ reference episodes as specified in Appendix B.5 and contains $12{,}788$ selected episodes.

Return targets and training. Adaptation videos are sampled at $f_{\mathrm{tr}} = 5$ Hz. For a sampled episode of length L, let $d \in \{0, 1\}$ indicate failure and let M_ℓ be the maximum sampled episode length for the normalized task instruction in the training manifest. With failure cost $C_\ell = M_\ell$, the target at sampled frame j is

$$G_j = \mathrm{clip}_{[-1,0]}\left(-\frac{L - 1 - j + C_\ell d}{M_\ell + C_\ell}\right). \tag{25}$$

This target combines a remaining-step cost with an episode-level failure offset. It requires no temporal quality annotations. Training uses episode-balanced sampling and AdamW on eight GPUs, with 32 frames per GPU and one gradient-accumulation step, giving a global batch size of 256. We denote the 80K-step checkpoint from this adaptation run by V_{bench} and use it for RTQ-Bench evaluation with the advantage estimator and frame-selection decoder below.

C.1.3 Native-Frame Advantage Estimation

At inference, every frame of the benchmark's primary video is decoded and scored at its native rate f; available wrist streams provide time-aligned observations. The primary video determines the output frame indices. For an episode of F frames, the model produces $V_0, \ldots, V_{F-1}$. Both the training task-length normalizer and the configured lookahead are converted to native-frame units:

$$\widetilde{M}_\ell = \max\{1, \mathrm{round}(M_\ell f / f_{\mathrm{tr}})\}, \qquad n = \max\{1, \mathrm{round}(50 f / f_{\mathrm{tr}})\}. \tag{26}$$

Thus, the configured 50-step lookahead corresponds to approximately 10 seconds, rather than 50 native frames for every video. The rounding operation follows Python's nearest-integer convention, including ties to even. If a task instruction is absent from the training task-length index, its source episode's length at the training sampling rate supplies the normalizer before conversion to native-frame units.

We recompute G_i using Equation 25 with $L = F$, $M_\ell = \widetilde{M}_\ell$, and $C_\ell = \widetilde{M}_\ell$. Rewards are obtained by differencing these targets, $r_i = G_i - G_{i+1}$ for $i < F - 1$. For each nonterminal frame, define $k_i = \min\{n, F - 1 - i\}$ and compute

$$A_i = \frac{n}{k_i}\left(\sum_{j=i}^{i+k_i-1} r_j + V_{i+k_i} - V_i\right), \qquad 0 \le i < F - 1. \tag{27}$$

There is no discount factor. Full windows use an n-step Bellman residual; shorter windows near the end are multiplied by n/k_i. The bootstrap is V_{i+k_i}, including the predicted terminal-frame value when the window reaches the episode end. The terminal frame has no future transition, is assigned zero advantage internally, and is excluded from rank selection. Its terminal reward is not included in the preceding residual's reward sum. These boundary conventions specify the estimator implemented by our adapter.

C.1.4 Advantage-to-Frame Conversion and Evaluation

Let $m = F - 1$ denote the number of eligible nonterminal frames and set

$$q = \min\{\lfloor m/2 \rfloor, \max\{1, \lceil 0.30m \rceil\}\}. \tag{28}$$

Episodes with fewer than two eligible frames are rejected. Sort eligible frames in ascending lexicographic order of (A_i, i). The first q frames are suboptimal selections and the last q frames are optimal selections. The frame index breaks ties deterministically, so earlier tied frames enter the lower group and later tied frames enter the upper group. The two selections are disjoint; all remaining frames, including the terminal frame, are unselected.

This is an episode-relative ranking rule, not a test of whether A_i is positive or negative. It selects approximately 30% of eligible frames in each category even when all advantages share the same sign. The decoder applies no temporal smoothing, gap filling, minimum-duration filtering, or SCHEMA event-count cap. It therefore differs from both RECAP's original policy-conditioning rule, which we use for the real-world RECAP baseline (Appendix F), and the progress-rate decoder in Appendix C.2.

Table 5: Shared progress-to-label conversion parameters.

Parameter	Symbol	Value
Centered smoothing width	w_s	$0.5\,\mathrm{s}$
Centered change width	w_d	$2.0\,\mathrm{s}$
Positive rate threshold	θ_+	$0.02\,\mathrm{s}^{-1}$
Negative rate magnitude	θ_-	$0.02\,\mathrm{s}^{-1}$
Maximum neutral gap	$g_{\max}$	$0.2\,\mathrm{s}$
Minimum event duration	$d_{\min}$	$0.3\,\mathrm{s}$

Selected original frame indices are exported as `optimal_frames` and `suboptimal_frames`. For evaluation, frame i represents $[i/f, (i+1)/f)$, with adjacent same-category cells forming intervals, following Appendix B.1. BS-IoU uses the same references, All/GT+ subsets, aggregation, and failure handling as the other methods. CC-Recall is left unreported in Table 2. The results characterize the adapted value model together with this relative-ranking decoder; selected frames need not constitute complete semantic quality events.

C.2 Progress-Based Baseline Adaptation

Robo-Dopamine 2.0 and Robometer produce progress estimates rather than optimal/suboptimal event annotations. We adapt these outputs to RTQ-Bench using the same deterministic progress-to-label conversion and parameters. This appendix describes these two progress-based adapters. RECAP's value-based adaptation and per-episode advantage ranking are specified separately in Appendix C.1.

C.2.1 Per-Frame Progress Estimation

For an episode with F frames, let t_i and p_i denote the timestamp and progress value of original frame $i \in \{0, \ldots, F-1\}$. Timestamps are measured from the video start. The adapters produce a dense progress sequence rather than estimating only a fixed number of uniformly sampled target frames.

Robo-Dopamine 2.0. We use the Robo-Dopamine-GRM-2.0-8B-Preview model in forward mode. For each frame $i > 0$, the request contains the task instruction, the initial frame, a blank goal image, three copies of the initial frame, and three copies of the current frame. The repeated images fill the three-view slots of the single-camera protocol; they are not additional camera observations. The returned percentage s_i is parsed as $p_i = \mathrm{clip}(s_i, -100, 100)/100$. Frame zero is an initialization anchor with $p_0 = 0$, not a model query. The conversion uses the resulting progress curve, rather than directly thresholding the stored adjacent-frame difference.

Robometer. We use the Robometer-4B model. For each target frame i, four prefix frames are selected at indices $\lfloor ji/3 \rfloor$, $j \in \{0, 1, 2, 3\}$, including repeated indices for short prefixes. The task instruction and these frames are submitted to the progress model, and the progress output at the last prefix position becomes p_i. Every original frame, including frame zero, receives a model prediction. Success probabilities are saved separately and are not used to assign quality labels. Thus, dense per-frame scoring does not mean that the entire video is supplied in a single request.

C.2.2 Shared Progress-to-Label Conversion

Table 5 lists the parameters shared by both adapters. They are settings of our benchmark conversion, not native quality-class definitions of either model. The conversion does not read reference annotations or estimate episode-specific rank thresholds.

Time-weighted smoothing. Let $p(t)$ be the piecewise-linear interpolant of the original progress samples. For each frame, define

$$a_i = \max\{t_0, t_i - w_s/2\}, \qquad b_i = \min\{t_{F-1}, t_i + w_s/2\}, \tag{29}$$

$$\bar{p}_i = \frac{1}{b_i - a_i} \int_{a_i}^{b_i} p(t)\, dt. \tag{30}$$

When $b_i = a_i$, set $\bar{p}_i = p_i$. This is a centered time-weighted mean, not a fixed-frame-count average. Windows are clipped at the video boundaries, and nonuniform frame timestamps are supported.

Local progress rate. Let $\bar{p}(t)$ be the piecewise-linear interpolant of the smoothed samples $\bar{p}_i$. Define

$$L_i = \max\{t_0, t_i - w_d/2\}, \qquad R_i = \min\{t_{F-1}, t_i + w_d/2\}, \tag{31}$$

$$v_i = \frac{\bar{p}(R_i) - \bar{p}(L_i)}{R_i - L_i}. \tag{32}$$

Set $v_i = 0$ for a zero-width window. The denominator is the actual clipped time span, including at episode boundaries. Rates are measured in progress fractions per second; a threshold of $0.02\,\mathrm{s}^{-1}$ corresponds to two percentage points per second.

Thresholding. With numerical tolerance $\epsilon = 10^{-12}$, the preliminary frame label is

$$\hat{y}_i = \begin{cases} +, & v_i > \theta_+ + \epsilon, \\ -, & v_i < -\theta_- - \epsilon, \\ \varnothing, & \text{otherwise.} \end{cases} \tag{33}$$

Here $+$ and $-$ denote optimal and suboptimal candidates, respectively; $\varnothing$ corresponds to the implementation's `normal` label and means unselected. Equality at either threshold remains unselected. Frame zero and any explicitly marked initialization anchors are forced to $\varnothing$, including Robometer's model-predicted first frame.

Temporal cleanup. First, fill an unselected gap of duration at most $g_{\max}$ only when both neighboring runs have the same nonempty label and the gap contains no anchor. Opposite-category runs are never bridged. Next, discard nonempty runs shorter than $d_{\min}$ by setting their labels to $\varnothing$. This order is fixed: gap filling precedes short-run removal. Anchors remain unselected after cleanup. For these duration tests, define frame edges $\xi_i = t_i$ for $i < F$ and $\xi_F = t_{F-1} + 1/f$, where f is the nominal frame rate. A run containing frames $a, \ldots, b$ has duration $\xi_{b+1} - \xi_a$; duration comparisons use the same 10^{-12} tolerance.

The classifier does not apply per-episode min–max normalization, monotonic correction, fixed positive/negative quotas, or SCHEMA's event-count caps. Interpolation is used only in postprocessing already obtained progress values; missing model predictions are not synthesized. The centered windows use future context, so this conversion is an offline localization procedure rather than a causal online reward.

C.2.3 Export, Evaluation, and Interpretation

The final positive and negative frame indices are exported as `optimal_frames` and `suboptimal_frames`. For evaluation, each selected frame i represents the temporal cell $[i/f, (i+1)/f)$; adjacent selected cells of the same category form an interval, following Appendix B.1. This frame-cell conversion is distinct from the timestamp-based duration checks during label cleanup. Predictions are aligned to the same reference episode identities as all other methods, without using the reference labels to alter predictions.

The exporter requires complete, finite progress values with contiguous frame indices and strictly increasing timestamps. Missing or invalid predictions are recorded as failures, not valid all-unselected outputs. Evaluation follows the failure handling in Appendix B.5; Table 2 reports BS-IoU for these baselines and leaves CC-Recall unreported.

The resulting labels are progress-based proxies for behavioral quality. A sufficiently negative progress rate produces a suboptimal candidate, whereas a stall or mild slowdown can remain unselected without a measurable decrease. Likewise, a positive rate does not itself identify a complete semantic progress event. The comparison therefore evaluates each progress model together with this explicit adaptation to RTQ-Bench.

D RTQ-BENCH DATA SOURCES AND ANNOTATION PROTOCOL

This appendix documents the data sources, expert training materials, and annotation-and-review workflow used to construct RTQ-Bench.

Data sources. RTQ-Bench contains selected episodes from nine public sources: Open X-Embodiment (O'Neill et al., 2024), ViFailback-Dataset (Zeng et al., 2026), GM-100 (Wang et al., 2026), AgiBot World (Bu et al., 2025), RoboMind 2.0 (Hou et al., 2025), ABC-130K (Allshire et al., 2026), RH20T (Fang et al., 2024), Galaxea Open World (Jiang et al., 2025), and DROID-Raw (Khazatsky et al., 2024). Table 1 reports statistics for the selected benchmark episodes rather than the complete source datasets.

Annotation workflow and model assistance. The annotation interface presents the task instruction and episode video and supports timeline editing with explicit selection of start and end frames. For each episode, a primary expert creates the first-pass annotations directly from the task instruction and video, identifying optimal and suboptimal events and determining their quality categories and temporal boundaries. No model-generated candidate intervals are prefilled in the annotation interface.

Qwen3.7-Plus is available during annotation to assist with drafting textual reasons for the selected events. The expert checks each assisted reason against the observed behavior and may revise it or provide an event-specific hint for regeneration. A different expert then reviews every episode and refines the first-pass annotations, adding or removing events and adjusting categories or temporal boundaries when needed. The reviewing expert likewise has access to Qwen3.7-Plus for drafting or revising textual reasons. All interval selection, category assignment, and boundary refinement remain human decisions.

This second pass covers all episodes, not a sampled subset, and is a review and refinement of the first-pass human annotations rather than a second independent annotation.

Training materials and quality categories. The shared training guide includes positive and negative example videos, instructions for using the annotation interface, and an explicit emphasis on annotation quality over speed. Experts judge behavior relative to the episode's task instruction. An optimal interval contains actions that clearly advance the task, such as moving the manipulator to acquire and lift an object. A suboptimal interval contains task regression, hesitation, backtracking, repeated attempts, unusually slow execution, incorrect actions, or task failure. These are operational categories of local behavioral quality, not claims of globally optimal control.

Temporal extent and annotation density. The guide requests short intervals containing the key actions that support the annotation, rather than broad spans of surrounding activity. Failure intervals are an exception to this short-interval preference and should cover the complete failure process. The guide specifies at most five intervals in total for videos shorter than 30 seconds and at most ten for videos longer than one minute; these totals combine the optimal and suboptimal categories. It does not specify a separate numerical cap for intermediate durations. The resulting reference annotations are sparse and event-selective, not an exhaustive partition of every frame into a quality class.

Failures and recoveries. Recoverable local failures are annotated separately from subsequent productive recovery. For an unresolved failure that persists to the end of an episode, the suboptimal interval extends from its identified onset to the final relevant frame. Episode-level failure or incompleteness alone does not determine the quality of every subsequent action.

Empty annotations and unannotatable cases. The guide permits saving an episode with no qualifying optimal or suboptimal events; an empty annotation is not, by itself, a reason to skip the

episode. A separate interface option handles cases that cannot be annotated. Uncovered regions remain unlabeled, and an empty category list means that no event of that category was retained under the annotation protocol, not that all frames belong to the other category.

E VLA PRETRAINING AND ROBOTWIN EXPERIMENTAL DETAILS

Pretraining stages and corpus composition. We use a two-stage schedule that first emphasizes broad task exposure and then embodiment alignment. Stage 1 comprises 100K updates on approximately 960 hours drawn from Open X-Embodiment (O'Neill et al., 2024) subsets: `fractal20220817_data_lerobot` (Brohan et al., 2022), `droid_lerobot` (Khazatsky et al., 2024), and `bridge_orig_lerobot` (Walke et al., 2023). This first-stage corpus spans 52K distinct task instructions, providing broad task exposure before embodiment alignment. Stage 2 continues training for another 100K updates on approximately 540 hours from 236 selected RoboCOIN (Wu et al., 2025) subdatasets. Both stages use 8 NVIDIA H200 GPUs, with per-device batch sizes of 32 and 16 for Stages 1 and 2, respectively. Uniform and SCHEMA-guided runs share the initial model, corpora, action-learning objectives, and optimization schedules, differing only in the frame-sampling distribution. The exact Stage 2 subset IDs are listed below.

Stage 2 RoboCOIN subset manifest. The Stage 2 corpus contains the following 236 selected RoboCOIN (Wu et al., 2025) subdatasets. Dataset identifiers are reproduced verbatim from the training manifest.

```
AgiBot-g1_battery_storage_b
AgiBot-g1_battery_storage_c
AgiBot-g1_box_storage_a
AgiBot-g1_box_storage_b
AgiBot-g1_box_storage_c
AgiBot-g1_box_storage_cardboard_box_a
AgiBot-g1_box_storage_cardboard_box_b
AgiBot-g1_box_storage_cardboard_box_c
AgiBot-g1_box_storage_e
AgiBot-g1_box_storage_part_a
AgiBot-g1_box_storage_tool
AgiBot-g1_left_capture_part
AgiBot-g1_mobile_accessory_storage_box_a
AgiBot-g1_mobile_accessory_storage_box_c
AgiBot-g1_mobile_accessory_storage_box_d
AgiBot-g1_mobile_accessory_storage_box_e
AgiBot-g1_mobile_accessory_storage_box_f
AgiBot-g1_pick_up_battery
AgiBot-g1_picks_up_battery_a
AgiBot-g1_picks_up_battery_b
AgiBot-g1_picks_up_cardboard_box
AgiBot-g1_picks_up_parts_a
AgiBot-g1_picks_up_parts_b
AgiBot-g1_remove_the_accessory
AgiBot-g1_right_capture_part
AgiBot-g1_robotic_arm_picks_up_battery
AgiBot-g1_robotic_arm_picks_up_parts
AgiBot-g1_storage_item_b
AgiBot-g1_storage_item_d
AgiBot-g1_storage_item_e
AgiBot-g1_tool_storage
Cobot_Magic_box_storage_chopsticks
Cobot_Magic_cap_the_pen_a
Cobot_Magic_catch_the_ball
Cobot_Magic_classification_of_fruits_and_vegetables
Cobot_Magic_classification_of_fruits_and_vegetables_a
Cobot_Magic_classification_of_tableware
Cobot_Magic_clean_blackboard
Cobot_Magic_clean_up_the_tableware
Cobot_Magic_clear_the_desktop
Cobot_Magic_close_book
Cobot_Magic_close_button
Cobot_Magic_cube_reset
Cobot_Magic_cut_banana
Cobot_Magic_desktop_organization
Cobot_Magic_drawer_storage_mineral_water
Cobot_Magic_fold_clothes
Cobot_Magic_fold_the_towel
Cobot_Magic_fold_towel_a
Cobot_Magic_food_packaging
Cobot_Magic_make_fruit_salad
Cobot_Magic_make_hamburger
Cobot_Magic_mobile_cube
Cobot_Magic_mobile_cube_blackboard
Cobot_Magic_move_beverage
Cobot_Magic_move_plate
Cobot_Magic_move_the_ball
Cobot_Magic_move_the_ball_and_the_cube_block
Cobot_Magic_move_the_ball_interference
Cobot_Magic_move_the_bread
Cobot_Magic_move_the_cup
Cobot_Magic_move_the_plate
Cobot_Magic_move_the_small_ball
Cobot_Magic_movethe_position_of_the_bluetooth
Cobot_Magic_open_the_shoebox
Cobot_Magic_place_square_pyramid
Cobot_Magic_place_the_cube_block
Cobot_Magic_place_the_test_tube
Cobot_Magic_plate_storage_apple
Cobot_Magic_plate_storage_bread
Cobot_Magic_plate_storaje_baozi
Cobot_Magic_pot_storage_steamer
Cobot_Magic_pour_drink
Cobot_Magic_pour_water_a
Cobot_Magic_pour_water_bottle
Cobot_Magic_prepare_breakfast
Cobot_Magic_pull_zipper
Cobot_Magic_pushing_magnet
Cobot_Magic_put_in_the_pear
Cobot_Magic_put_the_building_block_on_the_table
Cobot_Magic_steamer_storage_dumpling
Cobot_Magic_storage_plate
Cobot_Magic_take_out_a_pen_from_the_pen_holder
Cobot_Magic_take_out_the_bread
Cobot_Magic_take_the_shoes_off_the_shelf
Cobot_Magic_the_box_stores_table_tennis_balls
Cobot_Magic_the_plate_holds_the_fruit
Cobot_Magic_the_plate_holds_the_vegetables
Cobot_Magic_turn_off_the_desk_lamp
Cobot_Magic_turn_on_the_bulb
Cobot_Magic_turn_on_the_desk_lamp
Cobot_Magic_twist_bottle_cap
Cobot_Magic_vase_storage_flower
Cobot_Magic_water_bottle_storage
Galbot_g1_fold_clothe_b
Galbot_g1_fold_clothe_c
Galbot_g1_fold_clothe_e
Galbot_g1_steamer_storage_baozi_a
Galbot_g1_steamer_storage_baozi_b
Galbot_g1_steamer_storage_baozi_c
Galbot_g1_steamer_storage_baozi_d
Galbot_g1_steamer_storage_baozi_e
Galbot_g1_steamer_storage_baozi_f
Galbot_g1_steamer_storage_baozi_g
Galbot_g1_steamer_storage_baozi_h
Galbot_g1_steamer_storage_baozi_i
Galbot_g1_steamer_storage_baozi_j
R1_Lite_boil_water_in_a_kettle
R1_Lite_build_blocks
R1_Lite_catch_the_water
R1_Lite_clean_the_floor
R1_Lite_clean_the_sink
R1_Lite_clean_toilet
R1_Lite_connect_the_router_cable
R1_Lite_cook_a_meal
R1_Lite_cover_the_pot_lid
R1_Lite_dispose_of_leftover_food
R1_Lite_drawer_storage_hair_dryer
R1_Lite_fold_clothes
R1_Lite_garbage_disposal
R1_Lite_hang_clothes
R1_Lite_make_a_landline_call
R1_Lite_make_breakfast
R1_Lite_make_tea
```

R1_Lite_make_the_bed
R1_Lite_move_the_position_of_the_apple
R1_Lite_move_the_position_of_the_black_marker
R1_Lite_move_the_position_of_the_brush
R1_Lite_move_the_position_of_the_coffee_capsule
R1_Lite_move_the_position_of_the_cookie
R1_Lite_move_the_position_of_the_duck
R1_Lite_move_the_position_of_the_glass
R1_Lite_move_the_position_of_the_long_bread
R1_Lite_move_the_position_of_the_milk
R1_Lite_move_the_position_of_the_orange
R1_Lite_move_the_position_of_the_peeler
R1_Lite_move_the_position_of_the_pen
R1_Lite_move_the_position_of_the_rubiks_cube
R1_Lite_move_the_position_of_the_soda
R1_Lite_move_the_position_of_the_spoon
R1_Lite_move_the_position_of_the_triangle_bread
R1_Lite_open_and_close_curtains
R1_Lite_open_and_close_microwave_oven
R1_Lite_open_and_close_nightstand_door
R1_Lite_open_and_close_nightstand_drawer
R1_Lite_open_and_close_the_freezer_door
R1_Lite_open_the_food_pan
R1_Lite_opening_and_closing_aalcony_sliding_doors
R1_Lite_peach_storage
R1_Lite_pick_up_and_store_items
R1_Lite_place_the_dress_shirt_on_the_hanger
R1_Lite_plug_the_socket
R1_Lite_pour_water
R1_Lite_put_on_a_garbage_bag
R1_Lite_put_slippers_into_floor_standing_shoe_cabinet
R1_Lite_put_the_pillow_on_the_bed
R1_Lite_put_the_shoes_into_the_shoe_box
R1_Lite_put_the_tableware_into_the_cupboard
R1_Lite_sliding_chair
R1_Lite_stack_baskets
R1_Lite_storage_of_toiletries
R1_Lite_switch_labels
R1_Lite_switch_on_and_off_the_central_air_conditioning
R1_Lite_tableware_arrangement
R1_Lite_tableware_cleaning
R1_Lite_take_and_place_the_portable_power_bank
R1_Lite_take_and_put_away_garden_stuff
R1_Lite_take_and_put_away_garden_stuff_a
R1_Lite_take_and_put_away_items
R1_Lite_take_and_put_the_bowl
R1_Lite_take_clothes_out_of_the_washing_machine
R1_Lite_take_or_store_plates
R1_Lite_tea_service_table_setting
R1_Lite_throw_out_the_trash
R1_Lite_tidy_up_toiletries
R1_Lite_wash_the_tableware
R1_Lite_washing_board
R1_Lite_wipe_the_table
RMC-AIDA-L_basket_storage_banana
RMC-AIDA-L_basket_storage_egg_yolk_pastry
RMC-AIDA-L_basket_storage_long_bread
RMC-AIDA-L_basket_storage_orange
RMC-AIDA-L_basket_storage_peach
RMC-AIDA-L_box_up_down
RMC-AIDA-L_build_blocks
RMC-AIDA-L_clean_table
RMC-AIDA-L_desktop_organization
RMC-AIDA-L_fold_shirt
RMC-AIDA-L_fold_shorts
RMC-AIDA-L_fold_towel
RMC-AIDA-L_food_packaging
RMC-AIDA-L_food_storage
RMC-AIDA-L_fruit_storage
RMC-AIDA-L_get_water
RMC-AIDA-L_glasses_storage
RMC-AIDA-L_organise_the_document_bag
RMC-AIDA-L_place_test_tube
RMC-AIDA-L_place_the_fruits_repeatedly
RMC-AIDA-L_place_towel
RMC-AIDA-L_plate_storage
RMC-AIDA-L_pour_coffee_beans
RMC-AIDA-L_pour_rice
RMC-AIDA-L_pour_tea
RMC-AIDA-L_pull_open_bag
RMC-AIDA-L_stack_baskets
RMC-AIDA-L_stir_coffee
RMC-AIDA-L_storage_bin_storage
Split_aloha_basket_storage_banana
Split_aloha_basket_storage_bread
Split_aloha_basket_storage_egg_yolk_pastry
Split_aloha_basket_storage_long_bread
Split_aloha_basket_storage_orange
Split_aloha_basket_storage_peach
Split_aloha_fold_the_pants
Split_aloha_plate_storage
Split_aloha_pour_rice
Split_aloha_pour_tea
Split_aloha_scoop_coffee_beans
Split_aloha_stack_baskets
Split_aloha_stir_coffee
Split_aloha_wipe_table
Split_aloha_wipe_the_table
Split_aloha_zip_up_the_document_bag
Tianqin_A2_box_storage_part
Tianqin_A2_container_storage_graphics_card
Tianqin_A2_place_the_paper_box
alpha_bot_2_carry_the_clothes_basket
alpha_bot_2_item_reversal
alpha_bot_2_move_the_table
alpha_bot_2_operate_the_microwave_oven
alpha_bot_2_pass_the_sandbag
alpha_bot_2_press_the_button_a
alpha_bot_2_press_the_button_b
alpha_bot_2_recover_after_touching_an_obstacle
alpha_bot_2_stack_building_blocks
alpha_bot_2_sticker

Downstream SFT and evaluation. Each pretrained model is jointly fine-tuned on a pooled corpus covering all 50 RoboTwin 2.0 (Chen et al., 2025) tasks, with 50 clean and 500 randomized episodes per task (27,500 episodes in total). Both runs use identical SFT settings, and each checkpoint is evaluated under both clean and randomized conditions. Downstream SFT uses 16 ZW-810E PPUs with a global batch size of 192, while policy evaluation uses 8 NVIDIA RTX 4090 GPUs, a fixed evaluation seed of 0, and 50 trials per task for each evaluation setting. The 2.5K, 10K, and 25K SFT checkpoints correspond to 0.48M, 1.92M, and 4.80M cumulative training examples processed, respectively. Table 6 reports the per-task success rates.

Table 6: Per-task success rates (%) on RoboTwin 2.0 after downstream SFT at three optimization checkpoints. Each task is evaluated over 50 trials for each evaluation setting. U denotes Uniform Pretraining, and S denotes SCHEMA-Guided Pretraining. The Average row reports the mean success rate across all tasks.

	2.5K steps				10K steps				25K steps			
	Clean		Randomized		Clean		Randomized		Clean		Randomized	
Task	U	S	U	S	U	S	U	S	U	S	U	S
Average	27.36	**33.56**	27.88	**31.36**	47.72	**49.36**	48.04	48.04	65.44	**69.04**	63.64	**68.92**
adjust_bottle	60	92	66	90	88	94	94	100	98	100	94	100
beat_block_hammer	26	38	22	26	44	50	44	48	76	80	74	82
blocks_ranking_rgb	6	10	2	14	22	20	18	24	72	74	68	78
blocks_ranking_size	0	4	2	0	16	8	6	14	42	48	46	52
click_alarmclock	94	54	76	48	20	18	32	14	8	22	14	38
click_bell	82	62	74	56	68	32	78	38	40	44	36	66
dump_bin_bigbin	56	82	42	72	90	78	88	80	96	98	96	94
grab_roller	90	78	88	84	100	100	100	100	98	100	100	100
handover_block	0	0	0	0	2	12	0	4	48	54	50	54
handover_mic	6	24	12	38	40	50	42	62	78	82	80	86
hanging_mug	4	0	0	2	18	16	24	18	12	20	8	26
lift_pot	0	18	0	8	72	78	68	72	68	86	62	82
move_can_pot	0	4	4	10	28	26	40	30	54	38	50	48
move_pillbottle_pad	12	30	14	10	34	42	32	32	74	68	66	70
move_playingcard_away	64	70	68	70	80	76	86	86	96	98	90	94
move_stapler_pad	4	2	2	2	20	22	16	20	28	22	24	30
open_laptop	76	84	72	80	88	84	96	94	96	98	100	100
open_microwave	14	28	32	42	0	0	0	2	16	24	22	24
pick_diverse_bottles	0	2	0	0	28	16	32	14	38	46	30	38
pick_dual_bottles	0	0	0	0	26	22	28	10	42	46	36	50
place_a2b_left	38	58	32	44	72	70	74	66	82	94	78	82
place_a2b_right	32	46	44	44	50	76	78	80	78	88	80	96
place_bread_basket	14	38	16	20	56	60	34	46	72	76	60	64
place_bread_skillet	6	4	0	8	34	30	30	28	46	48	36	36
place_burger_fries	20	38	20	40	72	68	86	58	90	92	92	88
place_can_basket	8	0	6	6	28	30	16	32	36	54	40	56
place_cans_plasticbox	12	20	10	16	42	22	34	18	68	58	50	58
place_container_plate	86	70	80	78	96	94	92	96	100	98	98	100
place_dual_shoes	0	4	0	8	28	32	34	36	52	58	44	50
place_empty_cup	76	68	64	68	90	90	92	92	100	100	96	100
place_fan	12	0	14	16	28	38	30	40	52	60	54	70
place_mouse_pad	0	10	8	10	26	42	34	34	48	52	58	60
place_object_basket	6	22	18	14	56	60	66	62	76	84	72	72
place_object_scale	22	36	28	26	64	64	40	40	78	80	64	78
place_object_stand	34	42	38	48	72	62	74	66	82	86	88	78
place_phone_stand	14	28	20	28	54	68	58	62	88	92	94	92
place_shoe	24	36	38	36	66	90	74	82	92	92	90	86
press_stapler	76	94	82	78	86	86	80	90	96	96	96	98
put_bottles_dustbin	0	0	0	0	2	4	0	2	8	12	12	18
put_object_cabinet	6	8	0	4	10	20	6	12	62	58	64	56
rotate_qrcode	38	50	30	32	56	50	32	56	76	82	76	84
scan_object	0	8	0	6	26	42	24	24	54	62	48	58
shake_bottle_horizontally	80	94	86	92	98	98	96	98	100	100	98	100
shake_bottle	90	92	90	94	94	100	96	98	96	100	100	98
stack_blocks_three	0	0	0	0	0	2	2	0	10	18	10	12
stack_blocks_two	6	6	2	4	40	30	32	20	86	90	86	76
stack_bowls_three	6	10	6	8	40	52	30	38	56	58	54	52
stack_bowls_two	38	46	38	40	66	76	78	80	82	76	80	80
stamp_seal	8	28	18	16	18	26	30	36	64	62	56	66
turn_switch	22	40	30	32	32	42	26	48	62	78	62	70

F Real-World Experimental Details

This appendix provides the dataset composition and policy-training settings for the real-world experiments in Section 4.4.

Data collection and training subsets. We construct one shared task-specific trajectory pool before running the reported policy comparisons. Each task begins with one round of successful teleoperation. We then perform 1 rollout round for *PickPlace*, 1 for *FastenTwo*, 12 for *LongCut*, and 2 for *InsertFour*.

Rollout collection follows the iterative RECAP procedure (Physical Intelligence et al., 2025a). At each rollout round, all trajectories collected for the task up to that point are pooled, and the RECAP value function and collection policy are trained on this accumulated dataset before generating the next round of rollouts. The rollout episodes include human-in-the-loop (HITL) interventions. Following the original RECAP procedure, the value function and policy for each update are fine-tuned from their respective pretrained checkpoints using the accumulated task data rather than warm-started from the preceding round's weights.

After the final collection round, the trajectory pool is frozen and reused for all reported Qwen-PI and $\pi_{0.5}$ experiments; no method receives additional method-specific robot trajectories. For each task, *Successful Only* contains all successful teleoperation and rollout episodes, whereas *Successful + Failed* additionally includes all failed rollouts.

Table 7 reports the resulting episode counts.

Table 7: Real-world data-collection schedule and episode counts. Each task uses one initial teleoperation round, followed by the listed number of RECAP-style rollout rounds. Rollout (+HITL) denotes policy-rollout data including human-in-the-loop interventions. *Successful Only* combines successful teleoperation and rollout episodes; *Successful + Failed* additionally includes failed rollouts.

	Data collection				Training regimes	
Task	Teleoperation Successful	Rollout Rounds	Rollout (+HITL) Successful	Rollout (+HITL) Failed	Successful Only	Successful + Failed
PickPlace	20	1	9	11	29	40
FastenTwo	20	1	13	7	33	40
LongCut	117	12	283	136	400	536
InsertFour	20	2	23	7	43	50

Matched policy-training settings. All Qwen-PI runs are initialized from the same SCHEMA-guided pretrained checkpoint from Section 4.3. Uniform and SCHEMA-guided fine-tuning differ only in the frame-level sampling distribution. For SCHEMA-guided fine-tuning, optimal, suboptimal, and unlabeled examples receive relative weights 1.5, 0.2, and 1, respectively, using the normalized sampling distribution defined in Section 3.3.

For $\pi_{0.5}$, the RECAP baseline follows the original iterative learning procedure (Physical Intelligence et al., 2025a), using V_{pre} from Appendix C.1.1 as its value-model initialization. This is a separate branch from RTQ-Bench adaptation: V_{bench} is not used for real-world learning.

After each data-collection round, the value function is fine-tuned on the accumulated task data, and its updated advantage estimates provide the indicators for policy training. Following the original procedure, each round fits the value function and policy from their respective pretrained checkpoints, rather than the preceding round's weights. The resulting policy is evaluated on the robot and used for subsequent data collection. Real-world reward construction, advantage estimation, and indicator assignment follow RECAP, rather than the benchmark-specific top/bottom-30% decoder in Appendix C.1.4.

The SCHEMA variant uses the same advantage-conditioned $\pi_{0.5}$ policy interface but supplies the metadata-derived quality conditions defined in Section 3.3. Both variants use the same RECAP-style classifier-free-guidance inference procedure with a fixed guidance weight of 2.0 for all tasks and data regimes.

Within each task, policy family, and data regime, the training episodes and all remaining policy-training settings are identical between the baseline and SCHEMA. The hardware, global batch size, and task-specific training step counts are listed in Table 8. For each task and policy family, the same batch size and step count are used under both *Successful Only* and *Successful + Failed*. Training budgets are matched within each policy family, not between Qwen-PI and $\pi_{0.5}$.

Table 8: Real-world policy-training settings. Each task-specific step count applies to both training data regimes and to the baseline and SCHEMA variants of the corresponding policy.

Setting / Task	**Qwen-PI**	$\boldsymbol{\pi}_{0.5}$
Training hardware	16× ZW-810E	4× H200
Global batch size	192	32
Policy-training steps by task		
PickPlace	7,000	40,000
FastenTwo	7,000	40,000
LongCut	15,000	100,000
InsertFour	8,000	50,000

Result aggregation. Each method–task–data-regime combination is evaluated over 25 trials on the ALOHA robot. The average success rates reported in Section 4.4 are unweighted means of the four per-task success rates. Tasks therefore contribute equally to the reported averages, irrespective of their training-set sizes.