ARS: AGENTIC REWARD SYSTEM FOR ROBOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Progress reward modeling is the problem of estimating how a robot's behavior changes task progress over time. Reliable estimation requires distinguishing meaningful state changes from failed attempts and task-irrelevant actions. We introduce the **Agentic Reward System (ARS)**, an inference framework for progress reward modeling with general-purpose vision-language models (VLMs), without additional reward-model training. Given an offline trajectory and a task instruction, ARS uses adaptive visual inspection for both event proposal and verification. A subagent proposes a task-relevant event timeline, which a primary agent verifies and revises before estimating per-frame progress. ARS can incorporate optional terminal outcome labels and visual references to inform its judgments. It can also audit progress estimates from external reward models. We evaluate ARS with a 27B VLM on a controlled semantic-mismatch benchmark and downstream policy learning in simulation and on a real robot. The benchmark reveals that several evaluated reward baselines assign spurious progress to wrong-object manipulation even in simple pick-and-place scenes. ARS better suppresses these errors and outperforms these baselines in simulation policy learning. We further demonstrate that ARS supports long-horizon policy learning from mixed-quality offline experience on real-robot multi-screw fastening in a full-scale laboratory replica of an industrial washing-machine assembly line. These results suggest that structured inference and verification can improve the usefulness of general-purpose VLMs for robot reward modeling. Our code will be publicly available.

1 INTRODUCTION

Reliable rewards are central to learning robot policies from experience. Sparse terminal feedback offers limited guidance about individual actions, while hand-designed dense rewards require task-specific engineering and often privileged simulator state. In long-horizon manipulation, successful trajectories can contain failed attempts, while failed trajectories can retain useful intermediate actions. Estimating task progress from visual observations can identify these local differences and turn mixed-quality experience into training supervision.

Vision-language models (VLMs) can interpret task instructions together with visual trajectories. Existing approaches elicit progress from pretrained models (Ma et al., 2025; Chen et al., 2026b) or train dedicated reward models using trajectory supervision (Liang et al., 2026; Wu et al., 2026; Tan et al., 2026). Prior studies show that specialized robotic reward models can outperform directly prompted frontier flagship VLMs of the time, including GPT and Gemini (Zhang et al., 2026d; Schroeder et al., 2026; Lee et al., 2026). These findings raise the question of how much a general-purpose VLM's reward capabilities depend on the inference procedure used to obtain its judgments.

Accurate estimation requires checking the evidence behind an interpretation: smooth manipulation of the wrong object does not advance the instructed task, and an apparent grasp may require later observations for confirmation. Relevant evidence varies in temporal extent and spatial location. Recent methods structure estimation through spatiotemporal, recursive, and recurrent reasoning (Schroeder et al., 2026; 2025; Zhang et al., 2026d). We build on this direction by letting unresolved event claims guide additional visual inspection and revisions to an explicit event timeline.

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.

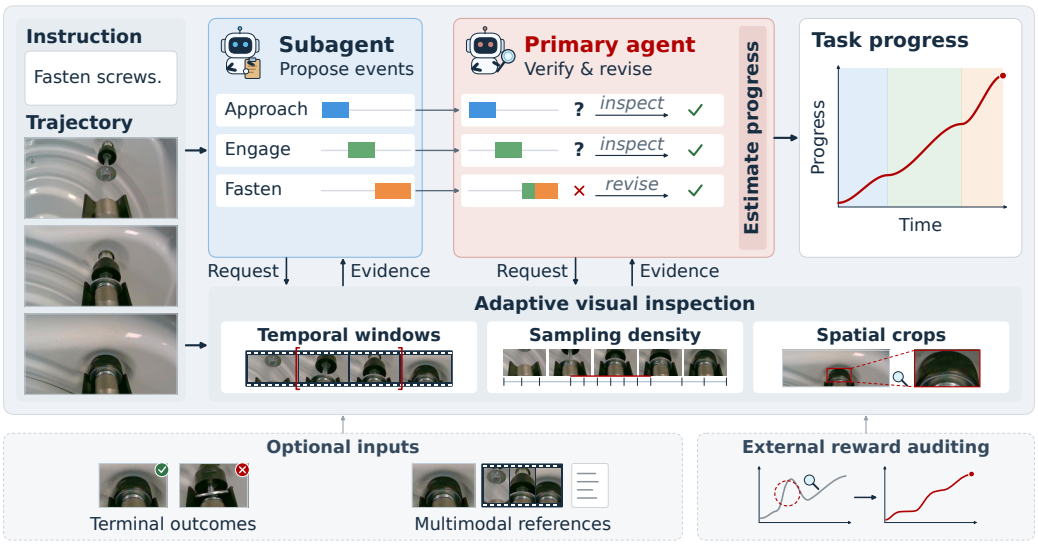


Figure 1: Overview of ARS. ARS uses adaptive visual inspection for both event proposal and verification. A subagent proposes task-relevant events from an instruction and offline trajectory. The primary agent verifies and refines these proposals, then estimates per-frame task progress. ARS supports optional terminal outcomes and multimodal references, plus external reward auditing.

We introduce the **Agentic Reward System (ARS)** (Figure 1). A subagent proposes task-relevant events and their intervals; a primary agent verifies and revises them. Both agents adaptively select temporal windows, sampling densities, and spatial crops to inspect visual evidence. The primary agent then constructs per-frame progress from verified state changes, assigning no additional credit to failed attempts and allowing regressions after setbacks (Figure 2). ARS requires no additional reward-model training and operates on offline trajectories, allowing later evidence to resolve earlier ambiguities. The workflow can optionally use terminal outcomes and visual references, or audit external reward predictions.

We evaluate ARS using a 27B general-purpose VLM. We construct a controlled semantic-mismatch benchmark in ManiSkill (Tao et al., 2025), revealing that several evaluated reward systems assign spurious progress under wrong-object instructions even in simple pick-and-place scenes with only two candidate objects. ARS better suppresses these errors. On the long-horizon manipulation tasks of LIBERO-Long (Liu et al., 2023), selecting action chunks by estimated progress improves policy success by 13.7 percentage points over unfiltered behavior cloning (BC) and 10.3 points over the strongest compared reward baseline. On real-robot multi-screw fastening in a full-scale laboratory replica of an industrial washing-machine assembly line, ARS selection with visual references raises success from 58.0% for demonstration-only behavior cloning to 81.5%. Together, these results highlight the importance of the inference workflow used to elicit and verify reward judgments.

Our contributions are:

- We introduce ARS, combining adaptive visual inspection, an editable event timeline, and cross-agent verification for progress estimation without reward-model training.
- We construct a controlled semantic-mismatch benchmark that exposes spurious credit for wrong-object manipulation in simple pick-and-place scenes.
- We show that ARS with general-purpose VLMs outperforms the evaluated specialized reward models and supports long-horizon policy learning in simulation and on an industrially relevant real-robot task.

2 Related Work

Robotic reward and progress estimation. Robotic reward estimation draws on pretrained visual representations (Ma et al., 2023b;a), video–language similarity (Sontakke et al., 2023), tem-

poral supervision (Yang et al., 2024; Zhang et al., 2025; Liu et al., 2026), and VLM-generated preferences (Wang et al., 2024b). Dedicated models learn task-conditioned progress and completion signals (Lee et al., 2026; Tan et al., 2026; Liang et al., 2026; Wu et al., 2026; Zhang et al., 2026c; Fang et al., 2026), while GVL and TOPReward elicit progress without reward-model training through shuffled-frame inference and token probabilities, respectively (Ma et al., 2025; Chen et al., 2026b). Other approaches exploit stage structure (Chen et al., 2026a; Hung et al., 2025; Yang et al., 2026b) and relative progress (Mao et al., 2026; Yu et al., 2026), or improve reliability through label correction, online refinement, confidence gating, and prompt optimization (Zhao et al., 2026a; Ayalew et al., 2025; Yang et al., 2026a; Gumbsch et al., 2026). Explicit reasoning further structures estimation. Schroeder et al. introduced ROVER (Schroeder et al., 2025) to structure training-free progress estimation through recursive subtask reasoning. They later proposed SOLE-R1 (Schroeder et al., 2026), a video-language reward model trained for spatiotemporal reasoning and dense progress prediction. ProcVLM uses procedural reasoning (Feng et al., 2026), while ProgressLM combines demonstration-conditioned episodic retrieval with mental simulation (Zhang et al., 2026a). R^2VLM recurrently updates task decomposition and completion status (Zhang et al., 2026d). ARS builds on these directions by letting unresolved event claims guide additional visual inspection and revisions to an event timeline.

Agentic inference and reward verification. Reward models support action selection, execution auditing, and world-model planning (Dai et al., 2025; Zhao et al., 2026b; Ji et al., 2026; Wang et al., 2026). Agentic inference provides mechanisms for acquiring and checking the evidence underlying such judgments. ReAct interleaves reasoning and tool use (Yao et al., 2023), while VideoAgent adaptively retrieves visual information (Wang et al., 2024a). Agentic reward methods apply related ideas to evaluation: ARM-Thinker invokes multimodal tools (Ding et al., 2026); ProRe coordinates a reasoner with evaluator agents (Dai et al., 2026), while the Agentic Verifier framework introduced with AgentV-RL combines forward and backward verification agents (Zhang et al., 2026b); VAGEN inspects trajectories and probes GUI environments (Cui et al., 2026). Reward as an Agent uses structured evaluation and reflection for generated embodied videos (Li et al., 2026). ARS applies active verification to executed robot trajectories, coupling event proposals with verification in separate reasoning contexts. Its harness requires evidence coverage and renewed confirmation after event revisions before dense progress is constructed.

3 Agentic Reward System

3.1 Problem setup

Let $\mathcal{D}$ be an offline dataset of robot trajectories. For a trajectory $\tau \in \mathcal{D}$, we write $\tau = (\mathbf{o}_{0:T}, \mathbf{a}_{0:T-1}, l)$. Here, $\mathbf{o}_t$ and $\mathbf{a}_t$ denote the observation and action at time t, while l is the task instruction and T is the terminal index.

A reward system f typically takes an observation sequence and a task instruction as input and predicts per-frame task progress:

$$p_{0:T} = f(\mathbf{o}_{0:T}, l) \in [0, 1]^{T+1}. \tag{1}$$

Estimating progress at time t may require observations beyond $\mathbf{o}_{0:t}$ to resolve ambiguity about the task state. For example, whether an apparent grasp has secured an object may only become clear when the object is subsequently lifted or transported. The temporal location and extent of the evidence needed for verification vary across events, motivating adaptive visual inspection guided by unresolved questions about the trajectory.

3.2 Adaptive visual inspection

We observe that agents given a complete trajectory video often produce coarse or incorrect interpretations, particularly for long-horizon tasks. Brief state transitions can be overlooked, and unsuccessful attempts can be conflated with later successful interactions. Access to all frames therefore does not ensure close examination of the evidence relevant to task progress.

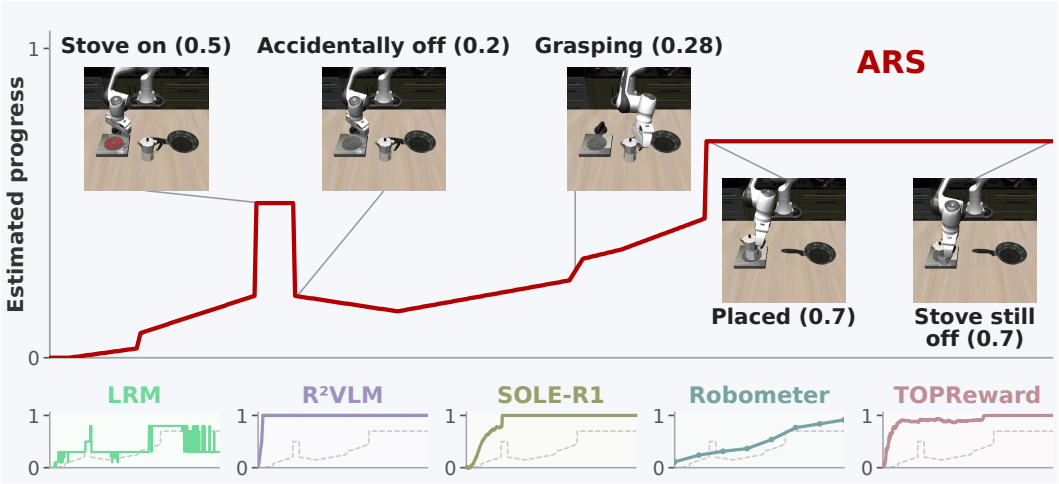


Figure 2: Progress estimates for a partially successful LIBERO trajectory under the instruction "turn on the stove and put the moka pot on it." Insets show selected frames and ARS progress values. ARS captures the regression when the stove is unintentionally switched off and credits subsequent progress toward pot placement. Baseline curves appear below (Appendix C), with ARS repeated as a light gray dashed line for reference. The event proposal and primary-agent verification for this trajectory are detailed in Appendix B.

ARS supports adaptive visual inspection through tool calls. An agent selects a temporal window, adjusts its sampling density, and optionally requests a spatial crop. The request is guided by the current question and can extend to later observations when an earlier event requires retrospective verification. The returned visual evidence informs subsequent reasoning and inspection requests. Our workflow ablations show that targeted inspection provides limited additional gains on the short semantic-mismatch benchmark, while improving downstream policy performance on long-horizon tasks even when the complete video is provided initially (Section 4.3).

3.3 Event-grounded reasoning

ARS represents the trajectory as a timeline of task-relevant events. Each event associates a temporal interval with a description grounded in cited visual evidence. The harness maintains the event timeline and its supporting evidence across reasoning contexts. Agent-authored state updates are executed through tool calls. When an event is created or revised, its cited evidence must cover the full event interval. Before proposal generation can finish, the event collection must cover the complete trajectory. Failed checks return the missing intervals for correction, making incomplete coverage an actionable part of inference.

3.4 Cross-agent verification

A single agent can complete the workflow by inspecting and revising events before estimating progress. However, ARS delegates event proposal to a subagent, which constructs a task-relevant event timeline through adaptive visual inspection. In a separate reasoning context, the primary agent gathers additional visual evidence to verify and revise these proposals.

Event confirmation requires evidence acquired during primary-agent verification. The harness checks that sampled frames include the event interior and the preceding and following context available within the trajectory. Revising an event invalidates its confirmation, requiring the updated claim to be checked again. These checks enforce evidence requirements; the primary agent judges whether the visual content supports the event description and its timing.

The primary agent then estimates per-frame progress from the verified events and their supporting evidence. The two agents may share the same underlying VLM.

This organization gives the primary agent explicit claims to challenge. For long-horizon tasks, this separation also reduces the primary agent's context usage by retaining event proposals without carrying over the subagent's full interaction history, leaving more capacity for targeted verification and progress estimation. See Section 4.3 for further analysis.

3.5 PROGRESS ESTIMATION

A central goal of progress-based reward modeling is to translate task understanding into useful numerical estimates of progress. Training-free methods approach this through numerical prediction over shuffled frames (GVL), completion-token probabilities (TOPReward), or recursive subtask reasoning (ROVER) (Ma et al., 2025; Chen et al., 2026b; Schroeder et al., 2025). ARS grounds progress estimation in an explicit timeline of task events whose descriptions and timing undergo targeted visual inspection and independent verification. The primary agent constructs and revises the progress curve using these verified events and their supporting evidence. Figure 2 illustrates how the resulting estimates capture local regression while preserving partial task achievement.

Through tool calls, the agent chooses between interpolated segments and local pointwise edits. Progress readback and summary tools expose the resulting curve's shape and explicitly flag jump locations and magnitudes, allowing the agent to check its estimates against visual evidence and revise them. Appendix A summarizes the agent prompts and available tools.

3.6 FLEXIBLE INTEGRATION OF AUXILIARY INFORMATION

As an agentic system, ARS can incorporate auxiliary context and audit external reward proposals within the same reasoning and verification workflow. Terminal outcomes and references help clarify task criteria that may be ambiguous from visual observations and brief instructions.

- **Terminal outcomes.** Trajectory-level success/failure labels provide low-cost supervision that may already be available from data collection. ARS uses them to anchor terminal completion while estimating intermediate progress from verified events, preserving local regressions in successful trajectories and partial progress in failed ones. We evaluate outcome conditioning in downstream policy learning (Section 4.2, Table 2).
- **In-context references.** Building on in-context progress estimation (Ma et al., 2025; Zhang et al., 2026c), ARS accepts images, video clips, textual descriptions, and positive or negative examples. Progress annotations are optional, allowing references to clarify task criteria without numerical labels. Section 4.4 examines their use in real-world policy learning.
- **External reward auditing.** ARS audits progress curves from external reward models, using questionable predictions to guide visual inspection and revising the curves according to verified task events. Section 4.1 evaluates this capability. The corrected curves and accompanying audit traces could also be explored as potential supervision for reward-model training.

3.7 PROGRESS-GUIDED POLICY LEARNING

ARS is compatible with value learning from progress-derived rewards (Liang et al., 2026; Tan et al., 2026). However, to evaluate the contribution of its progress estimates more directly, we adopt progress-guided policy learning following prior work on weighted behavior cloning (Ma et al., 2023b; 2025; Chen et al., 2026a; Yu et al., 2026; Mao et al., 2026). We use local progress changes to determine whether an action chunk contributes to policy training. This makes the mapping from progress estimates to training supervision explicit and avoids introducing an additional critic's estimation errors and optimization choices into the comparison.

We measure progress over chunks of H consecutive actions. For a chunk starting at t, we consider the trajectory segment $(\mathbf{o}_{t:t+H}, \mathbf{a}_{t:t+H-1}, l)$ and define its progress-derived score as

$$\Delta_t = p_{t+H} - p_t. \tag{2}$$

We introduce an indicator function I_t for chunk selection, defined as

$$I_t = \mathbb{1}[\Delta_t > \epsilon(p_t)] \in \{0, 1\}, \tag{3}$$

where $\epsilon(p_t)$ is a threshold that depends on the current progress value p_t.

Over the dataset $\mathcal{D}$, we minimize the following binary-weighted behavior-cloning objective:

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \frac{1}{\sum_{\tau\in\mathcal{D}}\sum_t I_t} \sum_{\tau\in\mathcal{D}}\sum_t I_t\, \ell_{\mathrm{BC}}(\theta; \mathbf{o}_t, l, \mathbf{a}_{t:t+H-1})\,. \tag{4}$$

Here, θ denotes the policy parameters, and $\mathbf{a}_{t:t+H-1}$ is the recorded action sequence. The function ℓ_{BC} denotes the policy's per-chunk behavior-cloning loss, conditioned on the current observation $\mathbf{o}_t$ and instruction l. The sums range over trajectories $\tau \in \mathcal{D}$ and their candidate chunk starting positions t, with the weights and action sequences defined for the corresponding trajectory.

We choose one of two alternative rules for the threshold function $\epsilon(p_t)$:

- **Zero threshold.** In the simplest setting, we set $\epsilon(p_t) = 0$ for all chunks, so that only chunks with positive estimated progress contribute to policy training.
- **Stage-dependent threshold.** Inspired by the advantage thresholding in $\pi^*_{0.6}$ (Physical Intelligence et al., 2025), we choose thresholds to retain the top fraction α of chunks ranked by Δ_t. Global ranking can favor early stages or stages with larger progress changes. We divide the progress range $[0, 1]$ into B equal-width bins and assign each chunk to a bin according to its current progress value p_t. Within each bin, we rank chunks by their progress changes Δ_t and set $\epsilon(p_t)$ to a shared threshold that retains the top fraction α.

4 Experiments

We evaluate task alignment through controlled semantic mismatches and progress-guided policy learning on long-horizon simulation and real-robot tasks. Ablations examine base-model choice and inference components.

Compared methods and experimental settings. We compare ARS against LRM (Wu et al., 2026), R^{2}VLM (Zhang et al., 2026d), SOLE-R1 (Schroeder et al., 2026), Robometer (Liang et al., 2026), and TOPReward (Chen et al., 2026b). Implementation details of these methods are provided in Appendix C. Although ARS can run with proprietary models, we use open-weight Qwen3.6-27B (Qwen Team, 2026b) in our main experiments for reproducibility. The main reward-system comparisons use only the trajectory and instruction as inputs. Configurations using auxiliary information or privileged simulator state are identified separately.

4.1 Semantic-mismatch benchmark

Related task-alignment evaluations use cross-task video–instruction pairing (Sontakke et al., 2023; Zhang et al., 2025; Liang et al., 2026; Chen et al., 2026b), counterfactual relabeling (Lee et al., 2026), and distractor instructions (Zhang et al., 2026d). We isolate wrong-object credit assignment in simple pick-and-place tasks, measuring the temporal extent of spurious progress increases and peak false progress.

Task and data. Inspired by prior work (Liu et al., 2025), we generate motion-planner expert trajectories in ManiSkill (Tao et al., 2025), with two candidate objects per tabletop scene and randomized backgrounds. Each video is paired with the correct pick-and-place instruction and a mismatch referring to the other visible object. Across 25 object types, the benchmark contains 500 pairs (1,000 samples), averaging 31.7 frames per trajectory. Expert trajectories advance the correct task at every recorded step; privileged simulator states verify that the gripper never approaches the mismatch object, even when camera projection suggests proximity. Training-data exposure is summarized in Appendix D.1.

Metrics. We evaluate using three complementary metrics:

- **Value-Order Correlation (VOC)** (Ma et al., 2025) reports the mean per-trajectory Spearman rank correlation between predicted progress and timestep under correct instructions.
- **False Progress Suppression Ratio (FPSR)** measures the fraction of mismatch-trajectory duration not covered by strictly positive progress-increase intervals. See Appendix D.2 for details.

Table 1: Results on the semantic-mismatch benchmark. The Audit configuration uses Robometer predictions as an external proposal, audited and revised by ARS with Qwen3.6-27B.

Group	Method	VOC ↑	FPSR ↑	CMPG ↑
Reward-trained	LRM	0.828	0.893	0.373
	R^2VLM	0.916	0.184	-0.050
	SOLE-R1	0.902	0.361	0.266
	Robometer	0.983	0.240	0.366
Audit	ARS w/ Robometer proposal	0.988	0.915	0.891
Reward-training-free	TOPReward + Qwen3-VL-8B	0.930	0.368	0.685
	TOPReward + Qwen3-VL-32B	0.945	0.383	0.756
	ARS + Qwen3.6-27B	**0.995**	**0.918**	**0.904**

- **Correct–Mismatch Progress Gap (CMPG)** is the difference between the mean terminal progress under correct instructions and the mean per-trajectory maximum progress under mismatch instructions. See Appendix D.3 for details.

Table 1 shows that temporal consistency does not ensure instruction alignment: Robometer achieves 0.983 VOC but only 0.240 FPSR, while R^2VLM assigns higher mean peak progress to mismatched instructions than mean terminal progress to correct ones (negative CMPG). ARS leads all three metrics without reward-model training, combining accurate ordering with suppression of spurious progress. Additional TOPReward base-model results appear in Appendix F.1.1.

Auditing Robometer predictions with ARS raises FPSR from 0.240 to 0.915 and CMPG from 0.366 to 0.891 while maintaining high VOC (0.988). These results approach ARS without proposals, supporting its use to correct external reward predictions.

4.2 Policy learning in simulation

Offline rollout dataset. A fixed behavior policy collects 4,096 trajectories across the complete LIBERO-Long suite (Liu et al., 2023): 2,204 successful and 1,892 failed trajectories, averaging 390.3 frames. All reward systems score this shared dataset, whose successive object interactions provide mixed-quality chunks within both successful and failed trajectories. Training-data exposure and task instruction clarifications are detailed in Appendices E.1 and E.2.

Training protocol. All downstream policies start from the same pretrained $\pi_{0.5}$ checkpoint (Black et al., 2025), with chunk horizon $H = 10$ and a shared training and evaluation schedule. The behavior policy serves only data collection. We apply the zero-threshold rule (Section 3.7) to each reward system's estimates, retaining chunks with positive progress differences at unit weight; Vanilla BC retains all chunks. The retained ratio is the percentage of candidate chunks kept. Appendix E.4 details the protocol and selection rationale.

ARS reaches 57.9% success, the highest among systems using only trajectories and instructions, exceeding Vanilla BC by 13.7 points (Table 2). Without reward-model training, it also surpasses LRM, SOLE-R1, and Robometer, which have documented LIBERO training exposure. ARS and TOPReward-32B retain similar proportions of chunks (66.9% and 67.9%), yet differ by 10.3 points in success, supporting the importance of selection quality.

LRM and R^2VLM retain fewer than 14% of chunks and underperform Vanilla BC. Their quantized or repeated progress predictions yield many zero increments, limiting temporal information for selection (Appendix E.5).

Outcome conditioning. ARS + outcome additionally receives a terminal success/failure label for each trajectory during progress inference. It reaches 62.2% success, 9.6 points above success-only BC (which retains all chunks from trajectories labeled successful) and 4.3 points below the privileged-state reference (66.5%), capturing 80.7% of the reference's success-rate gain over Vanilla BC (44.2%). Appendix E.3 describes how we construct the privileged progress signal from simulator

Table 2: Policy-learning results on LIBERO-Long. Retained is the micro-averaged percentage of action chunks kept from the original dataset; gains are in percentage points.

Group	Method	Success (%) ↑	vs. Vanilla ↑	Retained (%)
Baseline	Vanilla BC	44.2 ± 0.3	—	100.0
Reward-trained	LRM	38.6 ± 0.5	-5.6	13.7
	R^2VLM	41.5 ± 0.4	-2.7	13.6
	SOLE-R1	44.8 ± 0.4	+0.6	52.6
	Robometer	46.8 ± 0.9	+2.6	89.5
Reward-training-free	TOPReward + Qwen3-VL-8B	43.1 ± 0.5	-1.1	77.0
	TOPReward + Qwen3-VL-32B	47.6 ± 0.7	+3.4	67.9
	ARS + Qwen3.6-27B	**57.9** ± 0.4	**+13.7**	66.9
Outcome labels	Success-only BC	52.6 ± 0.7	+8.4	38.5
	ARS + outcome	62.2 ± 0.6	+18.0	60.3
Privileged state	Privileged progress	66.5 ± 0.8	+22.3	57.2

Table 3: Base-model and inference-procedure comparisons on the semantic-mismatch benchmark.

Configuration	VOC ↑	FPSR ↑	CMPG ↑
Robometer-4B	0.983	0.240	0.366
ARS + Qwen3.5-4B	0.986	0.625	0.633
GVL-style + Qwen3.6-27B	0.637	0.900	0.354
ARS + Qwen3.6-27B	**0.995**	**0.918**	**0.904**

state. Even with outcome conditioning, trajectories judged successful can retain flat or decreasing progress segments, while those judged unsuccessful can retain increasing segments (Appendix E.5).

4.3 Ablation Study

Base model and inference procedure. Under the same ARS workflow, Qwen3.6-27B substantially improves FPSR and CMPG over Qwen3.5-4B (Qwen Team, 2026a), showing that progress estimation benefits from the capabilities of the base model (Table 3). Nevertheless, ARS with Qwen3.5-4B already exceeds Robometer-4B on all three metrics without additional reward-model training, despite using a different base model at the same nominal parameter scale. With Qwen3.6-27B held fixed, GVL-style single-prompt inference achieves substantially lower VOC and CMPG than ARS (Appendix F.2.2). Thus, the gains depend on how the model is used, beyond the choice of base model alone. Additional base-model execution details are provided in Appendices F.1.2–F.1.3.

Inference usage and workflow ablations. ARS achieves higher downstream success than frame-wise TOPReward with substantially fewer model queries and cumulative input tokens, while introducing generative reasoning overhead (Table 4). Compared with TOPReward using Qwen3-VL-32B, full ARS improves success from 47.6% to 57.9%, with 93.6% fewer model queries and 65.8% fewer cumulative input tokens.

With Qwen3.6-27B fixed, visual inspection improves success from 45.4% to 52.3%. Adding same-context self-verification yields a slightly lower mean success rate of 51.8%, despite substantially higher inference usage. Full ARS verifies proposals in a separate reasoning context and reaches 57.9%, outperforming the same-context control by 6.1 percentage points. These results support the benefit of separating proposal generation from verification in this setting. Appendix F.2 details the workflow variants, inference usage, and additional semantic-mismatch results.

We further examine the role of explicit language chain-of-thought (CoT) in intermediate reasoning by disabling the base model's thinking mode; the results and analysis are provided in Appendix F.3.

Table 4: Workflow ablations on LIBERO-Long. Input/output token counts are in thousands (K).

Configuration	Success (%) ↑	Retained (%)	Queries	Input (K)	Output (K)
TOPReward + Qwen3-VL-32B	47.6 ± 0.7	67.9	390.3	2,866.3	0.0
ARS w/o verification & inspection	45.4 ± 0.3	78.1	7.3	242.7	10.1
ARS w/o verification	52.3 ± 0.8	70.3	10.2	413.2	9.9
ARS w/o context separation	51.8 ± 0.5	68.7	19.0	1,097.2	15.2
Full ARS	**57.9** ± 0.4	66.9	24.9	980.6	21.1

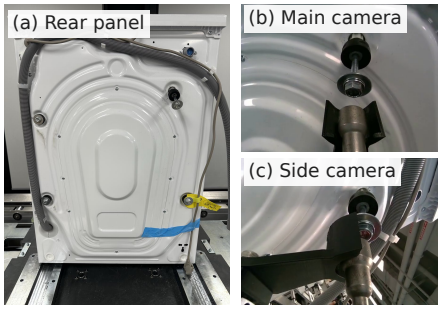


Figure 3: Screw-fastening setup.

Table 5: Real-world policy learning.

Configuration	Success (%) ↑
Demo + all rollouts BC	50.5
Demo-only BC	58.0
ARS	72.0
ARS + visual reference	**81.5**

4.4 Real-world policy learning

We evaluate sequential fastening of three screws on a washing machine's rear panel in a full-scale laboratory replica of an industrial assembly line (Figure 3; Appendix G).

Experimental setup. We use 2,574 demonstration trajectories and a shared pool of 3,092 offline rollout trajectories to train $\pi_{0.5}$ policies (Black et al., 2025). Table 5 compares demonstration-only BC, BC with all rollouts, and ARS-based rollout selection with or without visual references. All demonstration chunks are retained. For ARS-based selection, we use the stage-dependent threshold rule in Section 3.7, with $B = 8$ bins and $\alpha = 30\%$. Configurations incorporating rollout data use stratified sampling with a fixed 1:1 ratio between demonstration and eligible rollout chunks. Each configuration uses one training seed and is evaluated over 200 complete-task trials.

The visual-reference condition provides three image–text examples of task-relevant states during ARS progress inference, without numerical progress annotations. Additional training, reference, and evaluation details are provided in Appendix G.

Adding all rollouts reduces success from 58.0% to 50.5%, whereas ARS-selected rollouts raise it to 72.0% (Table 5), demonstrating the value of selective data incorporation. Visual references further raise observed success to 81.5%, a total gain of 23.5 points over demonstration-only BC. Appendix G.3 reports additional outcome conditioning.

5 Discussion

ARS improves downstream policy performance while using fewer model queries and cumulative input tokens than frame-wise TOPReward. However, its iterative visual inspection and generative verification introduce additional inference overhead compared with single-forward-pass reward models such as Robometer. This tradeoff makes the current workflow particularly suited to offline data curation. Our progress estimates can also use later observations to resolve earlier ambiguities, and extending the workflow to causal online reward estimation remains an open direction. A promising direction is to distill ARS-generated supervision into lightweight reward models, combining the benefits of explicit verification with efficient deployment.

AI USE STATEMENT

We used generative AI tools to assist with research ideation and execution, including code development, literature retrieval and discovery, and drafting and polishing portions of the manuscript. We reviewed and approved all code, compiled and verified all experimental results, and revised and finalized the manuscript. We take full responsibility for the entire work, including all AI-assisted content and artifacts.

REPRODUCIBILITY STATEMENT

The main text describes the ARS inference workflow and experimental protocols. Additional details on agent prompts and tools, baseline implementations, policy training and evaluation, and ablation configurations are provided in Appendices. We will provide reviewers with anonymized source code for ARS and for generating the semantic-mismatch benchmark, and will subsequently make both publicly available.

REFERENCES

Tewodros W. Ayalew, Xiao Zhang, Kevin Yuanbo Wu, Tianchong Jiang, Michael Maire, and Matthew R. Walter. PROGRESSOR: A perceptually guided reward estimator with self-supervised online refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2026a.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. TOPReward: Token probabilities as hidden zero-shot rewards for robotics. In *Conference on Robot Learning (CoRL)*, 2026b. Accepted for publication.

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic reward modeling: Verifying GUI agent via progressive trajectory-grounded interaction. *arXiv preprint arXiv:2602.00575*, 2026.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuqing Yang, Yuanchun Li, Rui Tan, Mo Li, and Lili Qiu. ProRe: A proactive reward system for GUI agents via reasoner-actor collaboration. In *International Conference on Learning Representations (ICLR)*, 2026.

Mingtong Dai, Lingbo Liu, Yongjie Bai, Yang Liu, Zhouxia Wang, Rui Su, Chunjie Chen, Liang Lin, and Xinyu Wu. RoVer: Robot reward model as test-time verifier for vision-language-action model. *arXiv preprint arXiv:2510.10975*, 2025.

Shengyuan Ding, Xinyu Fang, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiangyu Zhao, Haodong Duan, Xiaoyi Dong, Jianze Liang, Bin Wang, Conghui He, Dahua Lin, and Jiaqi Wang. ARM-Thinker: Reinforcing multimodal generative reward models with agentic tool use and visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22195–22205, 2026.

Yu Fang, Wanxi Dong, Jiaqi Liu, Yue Yang, Mingxiao Huo, Yao Mu, Huaxiu Yao, Li Erran Li, Daniel Szafir, and Mingyu Ding. DenseReward: Dense reward learning via failure synthesis for robotic manipulation. *arXiv preprint arXiv:2607.13033*, 2026.

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.

Christian Gumbsch, Leonardo Barcellona, Lennard Schuenemann, Platon Karageorgis, Andrii Zadaianchuk, Zehao Wang, Sergey Zakharov, Fabien Despinoy, Rahaf Aljundi, and Stratis Gavves. From demonstrations to rewards: Test-time prompt optimization for VLM reward models. *Reinforcement Learning Journal*, 7, 2026. RLC 2026 pre-proceedings.

Kuo-Han Hung, Pang-Chi Lo, Jia-Fong Yeh, Han-Yuan Hsu, Yi-Ting Chen, and Winston H. Hsu. VICtoR: Learning hierarchical vision-instruction correlation rewards for long-horizon manipulation. In *International Conference on Learning Representations (ICLR)*, 2025.

Yuheng Ji, Yuyang Liu, Huajie Tan, Xuchuan Huang, Fanding Huang, Yijie Xu, Cheng Chi, Yuting Zhao, Huaihai Lyu, Peterson Co, Mingyu Cao, Qiongyu Zhang, Zhe Li, Enshen Zhou, Pengwei Wang, Zhongyuan Wang, Shanghang Zhang, and Xiaolong Zheng. PRM-as-a-Judge: A dense evaluation paradigm for fine-grained robotic auditing. *arXiv preprint arXiv:2603.21669*, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

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.

Pu Li, Zhigang Lin, Qiang Wu, Yongxuan Lv, Fei Wang, and Shan You. Reward as an agent for embodied world models. *arXiv preprint arXiv:2606.19990*, 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. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.140.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can RL bring to VLA generalization? an empirical study. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. TimeRewarder: Learning dense reward from passive videos via frame-wise temporal distance. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23301–23320, 2023a.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023b.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *International Conference on Learning Representations (ICLR)*, 2025.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690. PMLR, 2022.

Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. ARM: Advantage reward modeling for long-horizon manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4468–4477, 2026.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen blog, 2026a.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model. Qwen blog, 2026b.

Philip Schroeder, Ondrej Biza, Thomas Weng, Hongyin Luo, and James Glass. ROVER: Recursive reasoning over videos with vision-language models for embodied tasks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Philip Schroeder, Thomas Weng, Karl Schmeckpeper, Eric Rosen, Stephen Hart, and Ondrej Biza. SOLE-R1: Video-language reasoning as the sole reward for on-robot reinforcement learning. *arXiv preprint arXiv:2603.28730*, 2026.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Cheng Chi, Yuheng Ji, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. General process reward modeling for robotic reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav

Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Demonstrating GPU parallelized robot simulation and rendering for generalizable embodied AI with ManiSkill3. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.021.

Kailin Wang, Haoxiang Jie, Yaoyuan Yan, Jiacheng Zhou, and Zhiyou Heng. EV-WM: Event-verified world models for long-horizon robotic manipulation. *arXiv preprint arXiv:2606.13053*, 2026.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *Computer Vision – ECCV 2024*, volume 15138 of *Lecture Notes in Computer Science*, pp. 58–76, 2024a. doi: 10.1007/978-3-031-72989-8_4.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement learning from vision language foundation model feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51484–51501, 2024b.

Yanru Wu, Weiduo Yuan, Ang Qi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Large reward models: Generalizable online robot reward generation with vision-language models. *arXiv preprint arXiv:2603.16065*, 2026.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2Reward: Learning shaped reward functions from passive video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Pengzhi Yang, Xinyu Wang, Pengyu Jing, Kehan Wen, Yiduo Qu, Zhenhao Huang, Minghao Fu, Xin Liu, Yaheng Shen, and Fan Shi. RARM: Confidence-gated progress reward modeling for RL in manipulation. In *Conference on Robot Learning (CoRL)*, 2026a. Accepted for publication.

Yang Yang, Bingjie Chen, Zihan Wang, Yizhe Li, Guoping Pan, Yi Cheng, and Houde Liu. Stage-transition dense reward modeling for reinforcement learning. *arXiv preprint arXiv:2606.31377*, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Justin Yu, Andrew Goldberg, Kavish Kondap, Karim El-Refai, Ethan Ransing, Qianzhong Chen, Mac Schwager, Fred Shentu, Philipp Wu, and Ken Goldberg. WARP-RM: A warp-augmented relative progress reward model for data curation. *arXiv preprint arXiv:2606.28320*, 2026.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 460–488, 2025.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. ProgressLM: Towards progress reasoning in vision-language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11243–11271, 2026a. doi: 10.18653/v1/2026.acl-long.516.

Jiazheng Zhang, Ziche Fu, Zhiheng Xi, Wenqing Jing, Mingxu Chai, Wei He, Guoqiang Zhang, Chenghao Fan, Chenxin An, Wenxiang Chen, Zhicheng Liu, Haojie Pan, Dingwei Zhu, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentV-RL: Scaling reward modeling with agentic verifier. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23078–23100, 2026b. doi: 10.18653/v1/2026.findings-acl.1156.

Qi Zhang, Shaopeng Zhai, Shengzhe Zhang, Litao Liu, Tianyi Zhang, Fuxian Huang, and Ming Zhou. A generalist pair-wise progress critic model for vision-language-action robots. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026c.

Yuelin Zhang, Sijie Cheng, Chen Li, Zongzhao Li, Yuxin Huang, Yang Liu, and Wenbing Huang. Recurrent reasoning with vision-language models for estimating long-horizon embodied task progress. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41150–41159, 2026d.

Lirui Zhao, Modi Shi, Li Chen, Qi Liu, Ping Luo, and Hongyang Li. UR-VC: Unsupervised robotic value correction for time-derived progress proxies. *arXiv preprint arXiv:2607.12892*, 2026a.

Sizhe Zhao, Shengping Zhang, Shuo Yang, Weiyu Zhao, Shuigen Wang, and Xiangyang Ji. Tap-Sampling: Inference-time sampling with a task-progress-understanding verifier for robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

A AGENT PROMPTS AND TOOL SET

We summarize the main instructions and tools used by ARS. The prompt summaries retain the core reasoning and evidence requirements while omitting repetitive instructions and interface examples. The default workflow consists of event proposal, independent verification, and progress estimation. A subagent performs proposal generation; the primary agent performs verification and progress estimation in a separate reasoning context.

The proposal stage receives the full trajectory video when it fits within the configured input limit; otherwise, the agent inspects the trajectory in segments.

A.1 PROMPT SUMMARIES

Table 6 summarizes the shared instructions, stage-specific instructions, and optional auxiliary context.

Table 6: Main instructions used by ARS, summarized by component.

Component	Main instructions
Shared instructions	Ground visual claims in inspected trajectory media. Use the task instruction to identify relevant entities and goal relations. Describe visible state changes and localize events using original frame indices. Request focused temporal windows and spatial crops when additional evidence could change the judgment. Express uncertainty through confidence scores and concise explanations.
Event proposal	Construct an event timeline covering the entire trajectory. Organize intervals around meaningful changes in task state, including preparation, control acquisition, sustained manipulation, completion, recovery, and periods without progress. Associate each event with its frame interval, supporting visual evidence, and confidence.
Independent verification	Treat proposed events as provisional claims. Acquire visual evidence during verification to examine each event's description, temporal boundaries, and confidence. Inspect neighboring frames to assess boundary placement. Revise, split, or merge events when necessary, and confirm each resulting event before proceeding.
Progress estimation	For sequential tasks, infer subgoals and estimate total progress from their currently achieved contributions, including partial completion. Assign limited credit to useful preparation, localized increases to verified achievements, and gradual increases to continued advancement. Use plateaus for preserved states and decreases for deterioration. Later observations may verify an earlier event; assign credit to the supported event boundary. Preserve meaningful partial achievement in incomplete trajectories.
Candidate-curve auditing	When an external progress curve is supplied, inspect its increases, decreases, and terminal values against the visual evidence. Retain supported estimates and revise unsupported intervals. Without a candidate curve, generate progress values for the complete trajectory.
Optional auxiliary context	Use terminal outcome labels to constrain completion judgments while determining event timing from visual evidence. Use reference examples to interpret task states and goal relations. Apply reference progress values only when the corresponding state is visibly supported in the evaluated trajectory.

A.2 TOOL SET

The harness exposes tools according to the current stage. In Table 7, E denotes event proposal, V denotes verification, and P denotes progress estimation. Tools use the term *observation* for an event record in the timeline.

Table 7: Tools available in the default ARS workflow.

Tool	Stage	Function
`read_robot_instruction`	E, V, P	Retrieve the task instruction and any explicitly supplied task clarification.
`read_metadata`	E, V, P	Retrieve technical video metadata, including frame count, frame dimensions, presentation frame rate, and frame-indexing conventions. This tool provides no additional task annotations or simulator state.
`inspect_visual`	E, V, P	Retrieve a selected frame interval as a video clip or image tile, with configurable temporal sampling and an optional spatial crop. Return the media with its evidence identifier and frame indices.
`read_observations`	E, V, P	Retrieve the current event timeline and coverage diagnostics.
`add_observations` `revise_observations`	E, V	Create or revise event intervals, descriptions, evidence links, and confidence scores.
`merge_observations`	V	Combine events that describe one continuous phase.
`confirm_observations`	V	Confirm an event using evidence acquired during verification.
`read_progress` `summarize_progress`	P	Inspect candidate and current progress values, curve statistics, and jump locations and magnitudes.
`edit_progress_segments`	P	Generate or revise local intervals by interpolating between specified endpoint values, supporting increases, plateaus, and decreases.
`calibrate_progress_curve`	P	Apply a global affine calibration using two frame–value anchors when the curve's relative trend is supported by the evidence.
`validate_progress`	P	Check frame assignments, numerical bounds, and configured curve constraints.
`write_summary`	P	Record the final judgment, supporting evidence, progress edits, and need for external review.
`complete_stage`	E, V, P	Check stage requirements and advance the workflow; completion of the final stage exports the results.

The agent represents progress using integer points in $[0, 1000]$, corresponding to a normalized progress scale of $[0, 1]$.

Observation creation, revision, and merging, as well as progress editing and calibration, use a shared confidence scale from 1 (very uncertain) to 5 (very high confidence). Scores reflect the clarity of the supporting evidence. Progress edits and calibration also include an evidence-grounded explanation.

Cited evidence must cover each event's recorded interval, and the event collection must cover the entire trajectory. Verification considers context before and after each interval when available. Revisions reset confirmation status.

To finalize progress estimates, the agent validates the curve, writes the summary, and completes the stage, in that order. Blocked tool calls report unmet requirements and corrective actions.

B ARS Reasoning Trace on a LIBERO Trajectory

We examine the ARS reasoning trace for the partially successful LIBERO trajectory in Figure 2. The trajectory contains 520 frames, indexed from 0 to 519, under the instruction "turn on the stove and put the moka pot on it." This case illustrates how verification changes event boundaries, how tool feedback guides retries, and where errors remain after verification.

Subagent inspection and event proposal. The subagent receives the complete trajectory video and makes nine additional `inspect_visual` calls to examine knob manipulation, moka pot grasping and placement, the final state, and transitions between these phases. It then proposes the eight events listed in Table 8.

Table 8: Subagent-proposed event timeline for the LIBERO trajectory in Figure 2, before primary-agent verification.

Proposed event	Frame interval
Initial state and approach to the stove knob	$[0, 94]$
Knob grasping and turning	$[95, 127]$
Burner switching off and withdrawal	$[128, 180]$
Repositioning and approach to the moka pot	$[181, 269]$
Moka pot grasp acquisition	$[270, 295]$
Lifting and transporting the moka pot	$[296, 339]$
Placement on the stove burner	$[340, 370]$
Final state: pot on the burner, burner off	$[371, 519]$

Primary-agent verification and boundary refinement. The primary agent receives this timeline in a separate reasoning context and makes eight additional `inspect_visual` calls. The main revision concerns the boundary between *Initial state and approach* and *Knob grasping and turning*. After inspecting frames $[0, 100]$, the primary agent judges that knob grasping begins substantially earlier than the proposed frame 95. It shortens the approach event to $[0, 45]$ and moves the start of knob manipulation to frame 46.

Manual inspection identifies frame 56 as the onset of actual control over the knob. Relative to this annotation, verification reduces the boundary error from 39 frames late to 10 frames early. The revised boundary is therefore substantially closer to the annotated transition, although it still assigns control acquisition prematurely. This case illustrates both the benefit of independent verification and the underlying VLM's remaining difficulty in precisely localizing subtle manipulation events.

Blocked calls and corrective retries. The initial boundary-revision call is blocked because the cited clip $[90, 140]$ does not cover the expanded event interval $[46, 126]$. The agent adds an already inspected clip covering $[0, 100]$ and retries successfully. This illustrates how evidence-coverage feedback guides correction of tool calls.

The primary agent subsequently uses the verified timeline to assign progress, capturing the regression when the stove is unintentionally switched off and crediting the later manipulation and placement of the moka pot.

C Implementation Details of Compared Methods

We evaluate the compared methods using the implementations and input protocols described below:

- **LRM** (Wu et al., 2026): we use its Absolute Progress model, which estimates absolute task progress for each frame and outputs one of 11 discrete values, $\{0.0, 0.1, \ldots, 1.0\}$.
- **R^{2}VLM** (Zhang et al., 2026d) recurrently processes video snippets, carrying forward a CoT that tracks subtask completion to estimate current task progress. We use its best available checkpoint.
- **SOLE-R1** (Schroeder et al., 2026) performs per-timestep spatiotemporal CoT reasoning to estimate dense task progress.
- **Robometer** (Liang et al., 2026) was trained on sequences of 8 sampled video frames to predict per-frame task progress. We also tested sliding-window inference and larger input frame counts on long-horizon tasks, but observed no improvement over the original 8-frame setting. We therefore retain 8 sampled frames per trajectory.
- **TOPReward** (Chen et al., 2026b) is a reward-training-free method that scores video prefixes conditioned on the task instruction using the log-probability of the `True` token in a task-completion query. Following the model choices in the original work, we evaluate TOPReward with Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct (Bai et al., 2025). At each timestep, we run inference using the task instruction and all video frames up to and including the current frame.

SOLE-R1 uses both the main and wrist cameras as inputs, while all other methods, including ARS, use only the main camera. We provide all trajectory frames as input to LRM, R^{2}VLM, SOLE-R1, and TOPReward. For Robometer, we follow the frame-sampling protocol specified in its paper, including both the first and last frames. We linearly interpolate between adjacent predictions to construct dense progress curves. Metrics in Table 1 are computed using each method's native model outputs, excluding interpolated values.

Output normalization. Let p_t^{raw} denote a method's raw output and p_t its evaluated progress. We use the following fixed output conventions; they specify numerical scales without assuming a common calibration across methods.

- **LRM** and **Robometer** output progress in $[0, 1]$ and require no rescaling.
- **R^{2}VLM** outputs values in $[0, 100]$; we use $p_t = p_t^{\text{raw}}/100$.
- **SOLE-R1** outputs signed values in $[-100, 100]$. The main results use $p_t = p_t^{\text{raw}}/100 \in [-1, 1]$, preserving its protocol-defined initial zero. We additionally report the affine mapping $p_t^{\text{aff}} = (p_t^{\text{raw}} + 100)/200 = (p_t + 1)/2 \in [0, 1]$ as a diagnostic.
- **TOPReward** outputs the log-probability of the `True` token, $p_t^{\text{raw}} \in (-\infty, 0]$. Following its original reward definition, our primary output convention uses the fixed linear normalization $p_t^{\text{lin}} = (\max(p_t^{\text{raw}}, -30) + 30)/30$. The minimum raw output observed across all our experiments was -28.5, so no output was clipped by the -30 lower bound. We additionally evaluate the corresponding token probability, $p_t^{\text{prob}} = \exp(p_t^{\text{raw}})$; Appendix D compares the two conventions. Exponentiation compresses absolute increments in low-probability regions, making them more susceptible to filtering by an absolute tolerance.
- **ARS** outputs integer progress points in $[0, 1000]$; dividing by 1000 gives progress in $[0, 1]$.

For the TOPReward curve in Figure 2, we use Qwen3-VL-32B-Instruct and follow the original paper's within-trajectory min–max normalization (Chen et al., 2026b), using $p_t^{\text{vis}} = (p_t^{\text{raw}} - p_{\min}^{\text{raw}})/(p_{\max}^{\text{raw}} - p_{\min}^{\text{raw}})$. The raw log-probabilities over this trajectory have a minimum of -12.9 and a maximum of -0.0014. This normalization applies only to this visualization.

D Additional Details on the Semantic-Mismatch Benchmark

D.1 Training-data exposure to ManiSkill

We summarize the training-data exposure of the evaluated reward models to ManiSkill based on the original papers and model documentation. $\mathbb{E}$ indicates documented use of ManiSkill data during

training, including reward learning and other supervised fine-tuning. N indicates that no evidence of such use was found. These labels describe benchmark-family exposure and do not establish overlap with our specific evaluation trajectories. N does not rule out exposure during training of the underlying VLM.

Table 9: Training-data exposure of reward models to ManiSkill.

Method	Exposure	Evidence
LRM	N	ManiSkill is described as zero-shot to the reward models (Wu et al., 2026).
R^2VLM	N	No ManiSkill training is reported for the evaluated checkpoint (Zhang et al., 2026d).
SOLE-R1	N	No ManiSkill training source is identified in the published training mixture (Schroeder et al., 2026).
Robometer	$\mathbb{E}$	RBM-1M includes FAILSafe trajectories collected in ManiSkill (Liang et al., 2026).
TOPReward	N	No ManiSkill training is reported in the Qwen model cards; no additional reward-model training is performed (Chen et al., 2026b).
ARS	N	No ManiSkill training is reported in the Qwen model cards; no additional reward-model training is performed.

Qwen model cards: Qwen3-VL-8B-Instruct, Qwen3-VL-32B-Instruct, Qwen3.6-27B, and Qwen3.5-4B.

D.2 False Progress Suppression Ratio (FPSR)

Definition. For mismatch trajectory i, let $0 = t_0 < \cdots < t_K = T_i$ be the frame indices of its native predictions $p_{t_0}, \ldots, p_{t_K}$, where T_i is the terminal frame index. Omitting the trajectory index on the prediction times and values for readability, we define, for a tolerance $\eta \geq 0$,

$$\mathrm{FPSR}_i(\eta) = 1 - \frac{\sum_{k=0}^{K-1} (t_{k+1} - t_k)\, \mathbb{1}[p_{t_{k+1}} - p_{t_k} > \eta]}{T_i}. \tag{5}$$

The dataset score is the unweighted mean over mismatch trajectories. We compute the metric from adjacent native predictions without interpolation. Its value remains sensitive to the temporal spacing of these predictions.

Expert progress coverage ratio (EPCR). Increasing η can suppress useful progress estimates as well as spurious increases. We measure the fraction of each correctly instructed expert trajectory's duration covered by intervals whose predicted progress increase exceeds η, and report its unweighted mean across trajectories as $\mathrm{EPCR}(\eta)$. This uses the fraction in Equation 5, without the leading $1-$, evaluated under correct instructions. The expert trajectories in this benchmark advance the correctly instructed task at every recorded step, making EPCR informative about the availability of the intended dense signal. A predictor that is flat on both instruction groups has FPSR equal to one but EPCR equal to zero.

Tolerance sensitivity. Table 10 reports FPSR together with EPCR for $\eta \in \{0, 10^{-4}, 10^{-3}, 0.005, 0.01\}$. The main metric uses $\eta = 0$. Only increments exceeding η are counted as progress increases. For integer-valued outputs, we compare raw integer differences against the appropriately scaled threshold to avoid floating-point subtraction errors at the boundary. The output conventions are specified in Appendix C.

Table 10 shows that ARS combines high false-progress suppression with high expert progress coverage. For $0 \leq \eta \leq 10^{-3}$, its FPSR remains 0.918–0.921 and EPCR remains 0.902–0.908. By comparison, LRM achieves FPSR of 0.893 but covers only 0.159 of expert-trajectory duration.

Increasing the tolerance can raise FPSR while removing useful progress signals. For TOPReward-8B probability outputs, increasing η from zero to 10^{-4} raises FPSR from 0.368 to 0.853 while

Table 10: Tolerance sensitivity on the semantic-mismatch benchmark. Each entry is FPSR / EPCR, both on a $[0, 1]$ scale, using native predictions. TOPReward-8B and -32B use Qwen3-VL. ARS and Audit use Qwen3.6-27B. TOPReward probability outputs are reported in the diagnostics section.

Method	0	10^{-4}	10^{-3}	0.005	0.01
LRM	0.893 / 0.159	0.893 / 0.159	0.893 / 0.159	0.893 / 0.159	0.893 / 0.159
R^2VLM	0.184 / 0.791	0.184 / 0.791	0.184 / 0.791	0.184 / 0.791	0.190 / 0.786
SOLE-R1, $[-1, 1]$	0.361 / 0.731	0.361 / 0.731	0.361 / 0.731	0.361 / 0.731	0.371 / 0.729
Robometer	0.240 / 0.978	0.242 / 0.977	0.246 / 0.976	0.273 / 0.971	0.306 / 0.965
TOPReward-8B, linear	0.368 / 0.656	0.383 / 0.642	0.390 / 0.622	0.452 / 0.560	0.508 / 0.509
TOPReward-32B, linear	0.383 / 0.723	0.412 / 0.698	0.447 / 0.660	0.585 / 0.526	0.693 / 0.406
ARS Audit	0.915 / 0.899	0.915 / 0.899	0.919 / 0.885	0.942 / 0.691	0.960 / 0.578
ARS-27B	0.918 / 0.908	0.918 / 0.908	0.921 / 0.902	0.940 / 0.717	0.960 / 0.588
Additional output-scale diagnostics					
SOLE-R1, $[0, 1]$	0.361 / 0.731	0.361 / 0.731	0.361 / 0.731	0.371 / 0.729	0.500 / 0.641
TOPReward-8B, prob.	0.368 / 0.656	0.853 / 0.354	0.947 / 0.277	0.976 / 0.213	0.982 / 0.183
TOPReward-32B, prob.	0.383 / 0.723	0.583 / 0.583	0.792 / 0.490	0.927 / 0.371	0.955 / 0.296

reducing EPCR from 0.656 to 0.354. Even ARS loses substantial coverage at larger tolerances, with EPCR falling to 0.588 at $\eta = 0.01$. These gains in FPSR therefore require joint interpretation with the remaining expert progress coverage.

D.3 Correct–Mismatch Progress Gap (CMPG)

Definition and motivation. Robometer (Liang et al., 2026) evaluates the difference in terminal rewards between successful and failed trajectories of the same task. Our semantic-mismatch benchmark imposes a stronger condition on the failure cases: the robot never approaches the object specified by the mismatch instruction or advances the corresponding task. Progress should therefore remain low throughout each mismatch trajectory. We accordingly replace terminal mismatch progress with the maximum predicted progress over the trajectory, so that transient false progress is penalized even if the prediction later decreases.

For each of the $N = 500$ trajectory pairs, let $p^+_{i,t}$ and $p^-_{i,t}$ denote progress under the correct and mismatch instructions, respectively. Let T_i denote the terminal timestep and $\mathcal{T}_i^-$ the timesteps with native model predictions under the mismatch instruction. We define

$$\mathrm{CMPG} = \frac{1}{N}\sum_{i=1}^{N} p^+_{i,T_i} - \frac{1}{N}\sum_{i=1}^{N} \max_{t\in\mathcal{T}_i^-} p^-_{i,t}. \tag{6}$$

Each trajectory contributes equally, irrespective of its length or number of predictions. Higher CMPG indicates greater separation between correct terminal progress and peak false progress.

CMPG complements VOC and FPSR by measuring numerical separation between correct terminal progress and peak mismatch progress. For example, a model that predicts a perfectly increasing progress curve from 0 to 1 under the correct instruction, but a constant progress of 0.99 under the mismatch instruction, achieves perfect VOC and FPSR. Nevertheless, it assigns nearly complete progress to an unrelated task, which is reflected in its low CMPG of 0.01. Although CMPG is sensitive to the output scale, we retain it to capture this complementary aspect of task alignment and interpret it jointly with VOC and FPSR under the normalization conventions in Appendix C.

Analysis under different output conventions. For SOLE-R1, the main $[-1, 1]$ convention preserves its protocol-defined initial zero. Mapping the same predictions to $[0, 1]$ via $p_t^{\mathrm{aff}} = (p_t + 1)/2$ halves CMPG from 0.266 to 0.133: the common offset cancels, while the gap scales by one half. VOC and zero-threshold FPSR and EPCR remain unchanged.

For TOPReward, replacing linearly normalized log-probabilities with token probabilities increases CMPG from 0.190 to 0.685 with Qwen3-VL-8B and from 0.129 to 0.756 with Qwen3-VL-32B.

This transformation preserves VOC and zero-threshold FPSR and EPCR; the larger CMPG values reflect the nonlinear change in output scale. We retain linearly normalized log-probabilities as our primary output convention, following TOPReward's original reward definition (Chen et al., 2026b). In Table 1, we report the probability-based CMPG values to present the more favorable results for TOPReward among the two evaluated conventions. Both values remain below ARS's CMPG of 0.904. Table 10 further shows that, across the tested positive tolerances, probability outputs exhibit larger FPSR increases and EPCR reductions than the linear convention.

E Additional Details on LIBERO-Long

E.1 Training-data exposure to LIBERO

Table 11 summarizes training-data exposure to LIBERO based on the original papers and model documentation. $\mathbb{E}$ indicates documented use of LIBERO data during training, and N indicates that no evidence of such use was found. The labels describe benchmark-family exposure without establishing overlap with our specific policy-training trajectories. N does not rule out exposure during training of the underlying VLM.

Table 11: Training-data exposure of reward models to LIBERO.

Method	Exposure	Evidence
LRM	$\mathbb{E}$	LIBERO is an explicit source in the training mixture (Wu et al., 2026).
R^2VLM	N	No LIBERO training is reported for the evaluated checkpoint (Zhang et al., 2026d).
SOLE-R1	$\mathbb{E}$	Supervised fine-tuning includes 504,544 `ecot_libero_all` examples (Schroeder et al., 2026).
Robometer	$\mathbb{E}$	RBM-1M includes LIBERO-{Long, Object, Spatial, Goal} and generated failures (Liang et al., 2026).
TOPReward	N	No LIBERO training is reported in the Qwen model cards; no additional reward-model training is performed (Chen et al., 2026b).
ARS	N	No LIBERO training is reported in the Qwen model cards; no additional reward-model training is performed.

E.2 Task instruction clarifications

For four of the LIBERO-Long tasks, we append brief clarifications to the original task instructions to clarify object identities and disambiguate spatial references. We provide identical clarifications to all reward systems, including ARS and every compared baseline, ensuring that all VLM-based methods receive the same additional semantic information.

For the three basket-placement tasks, the object names in the original instructions can be difficult to match to the visible entities. At the relatively low default resolution of 256×256 pixels, packaging text and graphics can be difficult to discern. Without background information about the objects, even human viewers can struggle to establish this correspondence. We therefore append short descriptions of object color, shape, and packaging.

For the mug-placement task, the original instruction specifies "left" and "right" without identifying the reference frame. We clarify that these directions are robot-relative and provide the corresponding image directions. Table 12 lists the original instructions and the appended clarifications.

E.3 Hand-designed privileged progress

We construct a progress signal from simulator-provided goal predicates and manually specified scoring rules. All scoring parameters were fixed before downstream policy training. For placement

Table 12: Task instruction clarifications used on LIBERO-Long. The same additions are provided to ARS and every compared baseline. Each addition is appended to the original instruction under the heading "Task-specific hints:".

Original instruction	**Appended clarification**
put both the alphabet soup and the tomato sauce in the basket	alphabet soup: blue-and-orange can with alphabet-letter soup graphics. tomato sauce: red/green/white can with tomato pictures.
put both the alphabet soup and the cream cheese box in the basket	alphabet soup: blue-and-orange can with alphabet-letter soup graphics. cream cheese: white rectangular box with a large purple-and-blue label.
put both the cream cheese box and the butter in the basket	cream cheese: white rectangular box with a large purple-and-blue label. butter: red-orange rectangular box with a blue-and-white panel.
put the white mug on the left plate and put the yellow and white mug on the right plate	Plate sides are robot-relative: white mug goes to robot-left (image right); yellow-and-white mug goes to robot-right (image left).

goals, the signal combines reaching the designated object, lifting it, and reducing its distance to the target. For articulated goals, such as opening or closing an object, progress is based on changes in joint distance to the target range. This configuration uses privileged task information and simulator state during offline data selection and follows the same chunk-selection, policy-training, and evaluation protocol as the other progress-based configurations.

For example, the task "put both the alphabet soup and the tomato sauce in the basket" contains two object-placement goals. For each object i, we compute normalized reaching, lifting, and placement scores $R_i, L_i, D_i \in [0, 1]$. Reaching depends on the gripper–object distance, while lifting measures the object's height increase relative to reset, modulated by gripper proximity. Placement measures the reduction in distance to the basket's containment region relative to reset. This distance is computed in three dimensions, with a 1 cm downward tolerance at the region's lower boundary. Before the corresponding containment predicate is satisfied, its score is

$$u_i = \min(0.98,\ \max\{0.15R_i + 0.35L_i + 0.50D_i,\ 0.35L_i + 0.65D_i\})\,. \tag{7}$$

For these initially unsatisfied goals, we set $q_i = 1$ when the environment's corresponding containment predicate is satisfied and $q_i = u_i$ otherwise. Overall progress is $p_t = (q_1 + q_2)/2$, with the environment's task-success check explicitly setting $p_t = 1$. We apply the same selection rule as in the other progress-based configurations, retaining chunks with $p_{t+H} - p_t > 0$, where $H = 10$ for LIBERO-Long.

E.4 Policy training and evaluation

Protocol for comparing reward systems. Our simulation experiments evaluate the utility of different reward systems for downstream policy learning under a shared protocol. Following prior work on weighted behavior cloning (Ma et al., 2023b; 2025; Chen et al., 2026a; Yu et al., 2026; Mao et al., 2026), we use predicted progress changes to determine the contribution of each action chunk to policy training. All reward systems operate on the same fixed rollout dataset and share the policy initialization, training budget, and evaluation procedure. This setup provides a direct connection between progress estimates and policy supervision without introducing an additional learned critic.

We also considered continuous weighting based on progress increments. However, different reward systems produce progress curves with different temporal profiles and distributions of increment magnitudes, making it difficult to choose shared weighting hyperparameters that yield comparable weighting behavior across methods. Tuning these hyperparameters separately would introduce additional optimization choices into the comparison. We therefore retain chunks with $\Delta_t > 0$ and assign them equal weight. Although this is a coarse criterion that discards progress magnitude, it provides a common comparison protocol with no additional selection hyperparameters and reduces dependence on the numerical calibration of progress estimates.

The zero-threshold rule also retains slow but potentially necessary transitions whenever their predicted progress change is positive. Selecting only high-gain chunks within each progress stage may reduce coverage of these actions. Adding unfiltered demonstrations can mitigate this issue but introduces another source of supervision, making downstream performance a less direct measure of rollout selection quality. We therefore use a zero threshold for the simulation comparison and stage-dependent thresholds for the real-robot experiments, where all demonstration chunks are retained for training.

Training and evaluation schedule. We train all policies for 30k gradient steps with three independent training seeds, using the same global batch size and optimization schedule. Following the LIBERO evaluation setup of OpenVLA-OFT (Kim et al., 2025), we evaluate all methods every 2k steps on all ten LIBERO-Long tasks, increasing the budget to 100 rollouts per task (1,000 per checkpoint). Each rollout allows up to 520 control steps, with the same benchmark-provided initial states and evaluation seed across methods. Following the best-checkpoint reporting convention in LIBERO and robomimic (Liu et al., 2023; Mandlekar et al., 2022), we select the checkpoint with the highest mean success rate across the ten tasks for each training seed and report the mean and sample standard deviation of these peak rates across the three seeds. Checkpoint selection and reporting use the same benchmark evaluation set.

Different reward systems retain different subsets of the shared rollout dataset, producing policy-training datasets that vary in both size and composition. We therefore fix the number of gradient updates rather than the number of epochs to maintain a matched optimization budget across methods; methods retaining fewer chunks revisit their training samples more frequently. Under this training configuration, we observe that evaluation success rates plateau for all methods, with no sustained improvement toward the end of the 30k-step training budget. As a diagnostic of performance at a fixed training step, we additionally report results at the final 30k-step checkpoint for the main method comparison (Table 13).

Results at the final checkpoint. ARS achieves 51.6% success at the final checkpoint, exceeding Vanilla BC by 11.5 percentage points and the strongest baseline, Robometer, by 9.5 points.

Table 13: Policy-learning results on LIBERO-Long at the final 30k-step checkpoint. Entries report mean success rate ± standard deviation over the same three training runs per method as in Table 2.

Method	Success (%) ↑
Vanilla BC	40.1 ± 1.8
LRM	36.8 ± 1.5
R^2VLM	34.1 ± 1.1
SOLE-R1	40.4 ± 2.8
Robometer	42.1 ± 2.3
TOPReward + Qwen3-VL-8B	37.9 ± 2.5
TOPReward + Qwen3-VL-32B	40.5 ± 2.9
ARS + Qwen3.6-27B	**51.6** ± 1.4

E.5 Progress-curve diagnostics

Zero progress increments. In Table 2, LRM and R^2VLM retain only 13.7% and 13.6% of the chunks, respectively, and yield the two lowest downstream success rates, 38.6% and 41.5%. For

LRM, the 11-level absolute-progress output produces zero progress differences for 76.7% of candidate chunks. R^2VLM assigns $\Delta_t = 0$ to 79.5% of chunks because it often repeats an early progress estimate across subsequent frames, leaving little temporal information for chunk selection.

Local progress under outcome conditioning. We examine adjacent-frame progress changes on the 3,098 trajectories where the conditioned and unconditioned runs agree on terminal outcome. With outcome conditioning, non-increasing intervals on success-predicted trajectories remain at 21.45% (21.70% without conditioning), while failure-predicted trajectories retain 23.38% increasing intervals (27.03% without conditioning), micro-averaged within each group. These patterns are consistent with the terminal label serving as an endpoint anchor while intermediate progress remains guided by visual evidence, rather than collapsing into a uniformly increasing ramp for success or a flat zero curve for failure.

F Additional Ablation Details

F.1 Base-model evaluation and compatibility

F.1.1 Additional base-model evaluation for TOPReward

Following the authors' public inference implementation (Chen et al., 2026b), we evaluate TOPReward with Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct and obtain reasonable results on both evaluations reported in Tables 1 and 2. To match ARS's default base model, we additionally evaluate TOPReward with Qwen3.6-27B. Starting from the reference settings, we test alternative prompt and chat-template formats, as well as enabling and disabling thinking mode. These attempts do not yield competitive results in our setup. The reported Qwen3.6-27B configuration achieves 0.452 VOC, 0.471 FPSR, and -0.049 CMPG under the linear log-probability convention described in Appendix C.

We hypothesize that this difficulty may partly reflect TOPReward's sensitivity to prompt formatting, as documented by the original authors (Chen et al., 2026b). Qwen3.6-27B uses a different chat template from Qwen3-VL-Instruct, including different handling of thinking prefixes (Bai et al., 2025; Qwen Team, 2026b), which may affect completion-token probabilities. However, our experiments do not isolate these differences as the cause of the performance degradation. We retain the original Qwen3-VL model family in the main comparisons and report the Qwen3.6-27B adaptation here.

F.1.2 Base-model execution requirements

To compare ARS with Robometer using the same base model, we attempted to run ARS with Qwen3-VL-4B-Instruct, the base model from which Robometer was trained (Liang et al., 2026). Despite its visual understanding capabilities, the model failed to complete the ARS workflow in any of our runs. Failures involved tool use and compliance with harness constraints. We then evaluated Qwen3.5-4B, a model of the same nominal parameter size from the Qwen family, which completed all 1,000 semantic-mismatch evaluations. We therefore use Qwen3.5-4B for the comparison with Robometer, matching the parameter scale while using a different base model.

F.1.3 Small-model execution on long-horizon trajectories

On LIBERO-Long, Qwen3.5-4B produced valid ARS outputs for 4,063 of 4,096 trajectories, an inference completion rate of 99.19%. The remaining 33 runs were marked as failures after extended attempts, including cases of repeated violations of harness constraints. These observations indicate greater difficulty sustaining reliable agent execution on long-horizon trajectories, even when the same model completes the simpler semantic-mismatch benchmark.

F.2 Workflow ablations: implementation details and additional experiments

F.2.1 ARS workflow variants

We compare full ARS with three ablated workflows. The first omits both visual inspection and explicit event verification, relying on the full video supplied initially. The second retains visual

inspection but omits explicit event verification. The third retains the backbone, tools, and verification procedure of full ARS but performs proposal generation and verification within a single reasoning context. Progress estimation continues in that same context. Full ARS instead generates proposals in one context and performs verification and progress estimation in a separate context. Across variants, we use Qwen3.6-27B with the same inference settings and progress-scoring criteria, and provide all video frames at their original resolution. A workflow completes when the agent requests completion and passes the applicable harness checks.

Actual query and token counts vary with workflow execution and the agents' stopping decisions. A query is one model scoring or generation request. We sum usage across all agents and report the mean per trajectory over all 4,096 trajectories. Input and output tokens are taken from the reported prompt and completion usage, respectively. Input counts include repeated context and visual tokens; completion counts already include reasoning tokens.

During verification and progress estimation, the primary agent averages 41.4K input tokens per query, 28.1% below the single-context control's average of 57.6K across the complete workflow (Table 14). This is consistent with the benefit of context separation described in Section 3.4: retaining event proposals without carrying over the subagent's full interaction history.

Table 14: Per-agent inference usage of full ARS on LIBERO-Long, with the single-context control for comparison. Queries and token counts are means per trajectory. Input per query is aggregate input tokens divided by model queries.

Configuration	Queries	Input (K)	Output (K)	Input / query (K)
Full ARS: subagent	5.8	187.0	3.9	32.5
Full ARS: primary agent	19.2	793.6	17.1	41.4
Full ARS: total	24.9	980.6	21.1	39.3
ARS w/o context separation	19.0	1,097.2	15.2	57.6

We also evaluate all three ablated workflows on the semantic-mismatch benchmark. On these short trajectories (31.7 frames on average), all three variants achieve scores close to those of full ARS across all three metrics (Table 15). This simple benchmark therefore provides limited separation among the evaluated workflows. On LIBERO-Long, all three ablations yield lower downstream policy success than full ARS, supporting the utility of the complete workflow in this setting.

Table 15: Additional workflow ablations on the semantic-mismatch benchmark using Qwen3.6-27B.

Configuration	VOC ↑	FPSR ↑	CMPG ↑
ARS w/o verification & inspection	0.991	0.915	0.895
ARS w/o verification	0.995	0.916	0.910
ARS w/o context separation	0.993	0.913	0.910
Full ARS	0.995	0.918	0.904

F.2.2 GVL-STYLE SINGLE-PROMPT EVALUATION

Purpose and protocol. We evaluate Qwen3.6-27B with a GVL-style single prompt (Ma et al., 2025) to assess the progress-estimation capabilities of the base model under direct prompting. The no-inspection ARS variant still permits multi-turn reasoning and harness checks, whereas GVL-style inference produces its progress estimates in a single model response. We provide the task instruction and all recorded trajectory frames, without temporal subsampling. The initial frame serves as the reference scene, and the model predicts task-completion percentages for the remaining frames. We evaluate both chronological presentation and the shuffled presentation used by GVL. In the shuffled variant, the initial reference frame remains fixed and the remaining frames are randomly permuted. Predictions are mapped back to their original temporal order before evaluation and action-chunk selection.

Semantic-mismatch benchmark. Chronological presentation performs better than shuffled presentation in our experiments. We therefore report the chronological variant in Table 3, which achieves 0.637 VOC, 0.900 FPSR, and 0.354 CMPG. Although its FPSR approaches that of ARS (0.918), its lower VOC and CMPG indicate weaker progress ordering and separation between correct and mismatched instructions. Thus, suppressing progress increases under mismatched instructions alone does not establish the overall quality of the estimated progress signal.

LIBERO-Long. We apply the same action-chunk selection rule as in the main simulation experiments, retaining chunks with $\Delta_t = p_{t+H} - p_t > 0$, where $H = 10$. Table 16 reports policy success and the percentage of action chunks retained for training. With chronological presentation, 95.20% of chunks have zero progress change and only 3.58% are retained. Policy training completes, but the resulting policy achieves 0.0% success.

Shuffling reduces the zero-change fraction to 70.50% and increases the retained fraction to 14.80%. The resulting policy achieves 34.1% success, 10.1 percentage points below Vanilla BC and 23.8 points below ARS with the same Qwen3.6-27B base model. These results show that frame ordering substantially affects the progress signal produced by single-prompt inference. However, the additional variation induced by shuffling does not translate into effective data selection relative to unfiltered behavior cloning in this setting.

Table 16: GVL-style single-prompt inference with Qwen3.6-27B on LIBERO-Long, using all trajectory frames. Retained is the percentage of action chunks kept for policy training.

Frame order	Success (%) ↑	Retained (%)
Chronological	0.0	3.58
Shuffled	34.1 ± 1.1	14.80

F.3 The form of intermediate reasoning

R^{2}VLM and SOLE-R1 organize intermediate reasoning primarily through explicit language CoT. R^{2}VLM recurrently maintains task decomposition, key steps, and their completion status through CoT over ordered video snippets (Zhang et al., 2026d). SOLE-R1 generates spatiotemporal CoT before each progress prediction; removing CoT from its progress training produces flatter, noisier rewards and weakens downstream learning (Schroeder et al., 2026).

Table 17: Effect of the base model's thinking mode under the same ARS workflow with Qwen3.6-27B. LIBERO-Long uses no outcome conditioning; Retained is the percentage of chunks kept from the original dataset.

Thinking mode	Semantic-mismatch			LIBERO-Long	
	VOC ↑	FPSR ↑	CMPG ↑	Success (%) ↑	Retained (%)
Disabled	0.990	0.915	0.877	55.4 ± 0.7	62.0
Enabled	**0.995**	**0.918**	**0.904**	**57.9** ± 0.4	66.9

We share their emphasis on intermediate reasoning, but examine whether it must take this form. Disabling the base model's thinking mode largely preserves ARS's semantic-mismatch performance and yields 55.4% downstream success, 11.2 points above Vanilla BC (44.2%). Enabling thinking further improves success by 2.5 points to 57.9% (Table 17). The harness still maintains and revises reasoning state through its event timeline, evidence links, and progress curve. These results suggest that a standalone language CoT trace is not the only effective way to organize intermediate reasoning for reward estimation.

G Additional Details on Real-World Policy Learning

G.1 Task and evaluation protocol

The robot must complete three screw-fastening operations on the washing machine's rear panel. Success is evaluated at the complete-task level: all three screws must be successfully fastened, as confirmed by a human evaluator. The system monitors end-effector rotation time and torque, and automatically marks timed-out trials as failures. Avoiding a timeout does not by itself establish success; final completion is determined by human assessment.

Real-robot evaluation is substantially more time-consuming than simulation. We therefore use one training seed per configuration and evaluate each policy over 200 complete-task trials. These results do not quantify variability across training seeds.

G.2 Policy training and data sampling

We initialize all policies in Table 5 from the same pretrained $\pi_{0.5}$ checkpoint (Black et al., 2025). We use an action-chunk horizon of $H = 50$ and evaluate the checkpoint saved at 200,000 gradient steps for all configurations.

The dataset contains 2,574 demonstration trajectories and 3,092 offline rollout trajectories. Demonstration-only BC uses all demonstration chunks. The remaining configurations combine the demonstrations with either all rollout chunks or the subset selected using ARS progress estimates. ARS-based configurations use the stage-dependent selection rule specified in Section 3.7.

For every configuration that incorporates rollout data, we use stratified sampling with a fixed 1:1 ratio between demonstration and rollout chunks. Within each source, eligible chunks are sampled uniformly. For ARS-based configurations, the rollout pool consists of the retained chunks. Thus, filtering changes the size and composition of the rollout pool while preserving the relative sampling weights of demonstrations and rollouts. We apply the BC loss in Equation 4 separately to each source, with equal expected weights under this sampling scheme.

Visual references. We provide two positive examples showing initial end-effector engagement with the screw head and full tightening with end-effector withdrawal, and one negative example showing stalled fastening. Each reference consists of a single image and a brief description, without numerical progress annotations. The references are supplied during ARS progress inference.

G.3 Outcome conditioning on the real robot

Adding terminal outcome labels to ARS with visual references increases observed success from 81.5% to 84.0%, but the incremental gain is not statistically significant (two-sided Fisher's exact test, $p = 0.597$).