

Robo-Dopamine 2.0: History-Conditioned and OOD-Aware Process Reward Modeling for Robotic Manipulation

Yijie Xu^{1,2*}, Haopeng Jin^{3,4*}, Run Zhou^{5*}, Shengbang Liu^{6*},
Sixiang Chen^{1,2}, Hongyang Cheng², Sicheng Hu^{1,2}, Peterson Co^{1,2},
Jinwen Luo⁴, Huajie Tan^{1,2}, Shanghang Zhang^{1†}

¹State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University

²EvoPhys AI

³Beijing University of Posts and Telecommunications

⁴Tencent

⁵Renmin University of China

⁶Sun Yat-sen University

Abstract

Vision-language-action (VLA) models have improved robotic manipulation, yet their policies remain vulnerable to compounding errors, unseen scene changes, and off-trajectory states. Reinforcement learning (RL) offers a promising route to refine pretrained VLA policies, but its effectiveness depends on informative rewards. Sparse success signals make exploration inefficient, whereas manually engineered dense rewards are costly, brittle, and task-specific. Learned visual reward models provide dense feedback, but existing robotic reward models often rely on static before–after observations and struggle to capture execution context and failure semantics. This leads to *temporal ambiguity*, where similar observations correspond to different latent progress due to hidden contact events, repeated phases, or reordered subgoals, and limits their ability to distinguish robustness-preserving variations from task-invalid failures under out-of-distribution (OOD) execution. We introduce **Robo-Dopamine 2.0**, a history- and OOD-aware process reward model that retains a pairwise prediction interface. Robo-Dopamine 2.0 improves progress estimation under these conditions through two complementary designs: (1) a history-conditioned pairwise reward formulation that uses source-aligned reference panels for synthetic OOD queries and observed rollout history for on-line queries, while preserving the actual queried endpoints; and (2) an OOD-aware signed progress space that organizes positive, robustness-preserving, and negative states into a unified progress representation. To improve optimization of fine-grained progress ordering, we further develop a **Signed-Hop Curriculum** with transition-aware replay that progressively learns coarse execution ordering before fine-grained progress calibration. We also construct an OOD trajectory dataset and a five-family benchmark for comprehensive evaluation. Reference panels improve mean visual order consistency (VOC) from 0.967 to 0.986 and OOD-robust VOC from 0.906 to 0.958. Under the same 400K pairwise-reward budget, Signed-Hop training with 25% replay reaches 0.9872 mean VOC, compared with 0.9858 for a matched-pool shuffled control. In downstream RL, the full model reaches 86.8% mean RoboTwin success and 71/80 successful real-world insertions.

*These authors contributed equally.

†Corresponding author.

Introduction

Large-scale imitation learning and vision-language-action models have substantially improved visuomotor policy generalization from diverse demonstrations (Brohan et al. 2023; Zitkovich et al. 2023; Kim et al. 2025). However, long-horizon, contact-rich manipulation remains challenging due to compounding errors, unseen scene changes, and off-trajectory states that require recovery. Reinforcement learning can improve policies through interaction, but its practical deployment on real robots critically depends on rewards that are dense, task-relevant, and reliable beyond the demonstration distribution. Sparse terminal rewards make exploration inefficient, whereas handcrafted dense rewards are brittle, task-specific, and expensive to engineer (Lee, Smith, and Abbeel 2021; Wang et al. 2024; Sontakke et al. 2023).

Recent visual and language-conditioned reward models address this bottleneck by learning dense feedback from observations and task descriptions (Ma et al. 2023b,a; Baumli et al. 2023; Wang et al. 2024); robot-specific methods further introduce stage- or process-aware progress estimation (Hung et al. 2025; Tan et al. 2026; Chen et al. 2026a; Zhang et al. 2026). Nevertheless, existing approaches remain limited in accurately evaluating execution progress under challenging manipulation scenarios. **First**, static endpoint pairs suffer from *temporal aliasing*: visually similar states may correspond to different latent progress due to hidden contact events, repeated phases, or reordered subgoals (Ni, Eysenbach, and Salakhutdinov 2022; Guhur et al. 2023; Fu et al. 2024). **Second**, robust execution requires distinguishing semantics-preserving variations from true failures: occlusions or distractors may preserve task progress, whereas failed grasps, wrong-object interactions, and incomplete releases should receive negative feedback. However, successful in-distribution trajectories provide limited supervision for learning such failure-aware distinctions.

To address these challenges, we introduce **Robo-Dopamine 2.0**, a history- and OOD-aware pairwise reward learning framework that retains the pairwise progress formulation used in prior process reward modeling (Tan

et al. 2026). Robo-Dopamine 2.0 improves phase-aware and failure-aware progress estimation through two complementary designs: (1) *a history-conditioned pairwise reward formulation* that uses same-episode expert history for standard queries and observed rollout history for online queries, while synthetic OOD queries use phase-aligned successful-reference panels without replacing their edited endpoints; and (2) *an OOD-aware signed progress space* that organizes execution states into positive, robustness-preserving, and negative branches, enabling the model to represent valid progress, semantics-preserving variations, failures, and recovery within a unified pairwise training interface.

Furthermore, we introduce a *Signed-Hop Curriculum* with transition-aware replay to progressively learn execution ordering. The curriculum first captures coarse-grained ordering between successful, robust, and failed states, and then refines fine-grained progress differences. The replay strategy preserves informative large-transition pairs during training, improving progress calibration under diverse execution scenarios. We evaluate Robo-Dopamine 2.0 through a trajectory-level visual order consistency (VOC) evaluation across five benchmark families covering in-distribution, temporal, and OOD scenarios.

Our contributions are summarized as follows:

- We propose **Robo-Dopamine 2.0**, a history- and OOD-aware pairwise reward framework that supports source-reference context for synthetic OOD queries and observed rollout context for online reward inference, together with a signed progress space for valid progress, robustness, failure, and recovery.
- We develop a *Signed-Hop Curriculum* with transition-aware replay that improves learning of global execution ordering and fine-grained progress calibration.
- We present a trajectory-level visual order consistency (VOC) evaluation across five benchmark families covering in-distribution, temporal, and OOD scenarios, including controlled studies of static versus history-conditioned reward modeling.

Related Work

Generalist Robot Policies. Generalist robot policies and vision-language-action models scale visuomotor learning and semantic generalization across tasks and embodiments. RT-1 studies large-scale real-world robot control, RT-2 transfers web-scale semantic knowledge into action prediction, and OpenVLA provides an open model for adaptation to new manipulation domains (Brohan et al. 2023; Zitkovich et al. 2023; Kim et al. 2025). These works establish the foundation for general robot policies, but do not address process reward modeling for off-trajectory recovery. Robo-Dopamine 2.0 instead focuses on the reward layer for refining policies under long-horizon and out-of-distribution execution.

Visual Reward Learning. Reward learning explores preference supervision (Lee, Smith, and Abbeel 2021), language-conditioned state changes (Nair et al. 2022), and feedback derived from visual or video-language representations (Sontakke et al. 2023; Ma et al. 2023b,a; Alakuijala et al.

2024). Recent foundation-model approaches learn pairwise visual preferences, general-purpose rewards, or failure-aware feedback (Wang et al. 2024; Baumli et al. 2023; Yang et al. 2024), while VICtoR introduces stage-aware vision-instruction rewards (Hung et al. 2025). Despite these advances, existing methods remain largely endpoint-oriented or success-centric and do not combine query-dependent temporal context with explicit OOD failure/recovery supervision. Robo-Dopamine 2.0 complements these approaches by jointly modeling temporal-context-conditioned pairs and signed OOD progress within a unified reward formulation.

Temporal and Process-aware Reward Modeling. Temporal context has been explored in recurrent RL, history-aware manipulation policies, and order-sensitive proxy rewards (Ni, Eysenbach, and Salakhutdinov 2022; Guhur et al. 2023; Fu et al. 2024). These methods primarily introduce memory into policies or trajectory matching, whereas Robo-Dopamine 2.0 targets temporal ambiguity within pairwise reward estimation. Meanwhile, robotic process reward models investigate multi-view relative progress, large-scale reward data, stage-aware estimation, and critic-like architectures (Tan et al. 2026; Lee et al. 2026; Zhang et al. 2026; Chen et al. 2026a; Mao et al. 2026); failure- and stage-reasoning models provide complementary diagnostics (Duan et al. 2025; Zhou et al. 2026). Robo-Dopamine 2.0 bridges these directions by using query-dependent temporal context—same-episode expert history for standard queries, observed rollout history for online queries, and source-aligned successful-reference panels for synthetic OOD queries—and by unifying semantics-preserving variations, failures, and recovery within a signed pairwise progress space.

Method

Figure 1 summarizes Robo-Dopamine 2.0: ordered temporal context addresses ambiguous endpoints, signed OOD supervision models valid progress, failure, and recovery, and a Signed-Hop curriculum improves global-to-local ordering. Closed-loop inference converts pairwise predictions into an online potential for reinforcement learning.

History-Conditioned Pairwise Reward Formulation

Robo-Dopamine 2.0 preserves the pairwise interface of prior process reward modeling (Tan et al. 2026). Given an instruction c , task-reference anchors $\mathcal{R} = (r_s, r_g)$, ordered temporal context $\mathcal{C}_{ba}$, and an ordered pair (X_b, X_a) , the General Reward Model (GRM) predicts signed relative progress:

$$\hat{y}_{ba} = f_{\theta}(c, \mathcal{R}, \mathcal{C}_{ba}, X_b, X_a). \quad (1)$$

Here X_b and X_a are the actual queried multi-view states. The reference anchors specify the task start and goal but are not treated as observed execution history. The static baseline sets $\mathcal{C}_{ba} = \emptyset$.

Let $X_b = (b^h, b^l, b^r)$ and $X_a = (a^h, a^l, a^r)$, where h, l, r denote the canonical head/third-person, left-wrist, and right-wrist view slots. For a pair with temporal indices t_b and t_a , we define

$$\begin{aligned} \mathcal{W}_{\text{pre}} &= \{t : 0 \leq t < \min(t_b, t_a)\}, \\ \mathcal{W}_{\text{mid}} &= \{t : \min(t_b, t_a) < t < \max(t_b, t_a)\}. \end{aligned} \quad (2)$$

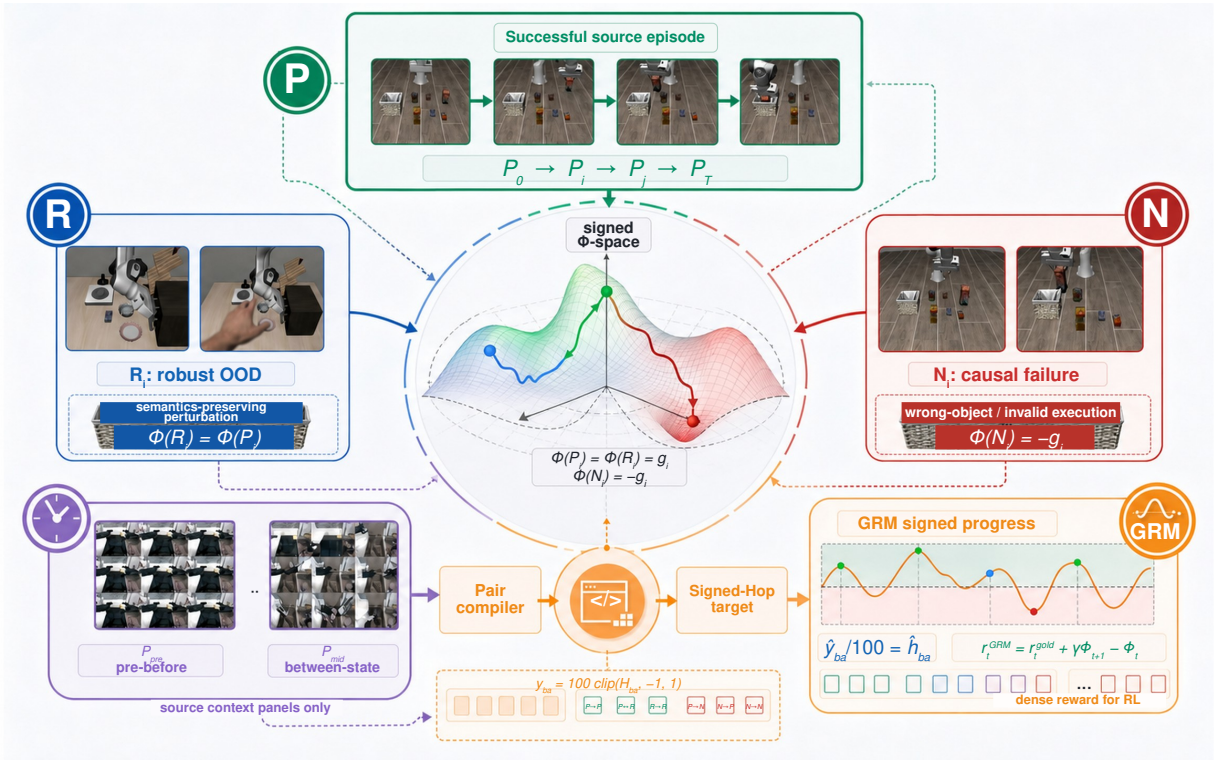


Figure 1: Overview of Robo-Dopamine 2.0. Standard queries use same-episode expert history, online queries use the observed rollout history, and synthetic OOD queries use source-aligned successful-reference panels while preserving their edited end-points. All context modes share the same pairwise GRM interface.

For standard and online queries, these are indices in the queried episode or rollout; for synthetic OOD pairs, they are indices in the aligned source episode. Pair order determines the target direction, while panel frames remain chronological. Each window is uniformly sampled into panels $M_{\text{pre}}^{1:K}$ and $M_{\text{mid}}^{1:L}$. The two input layouts are

$$\begin{aligned} \mathcal{I}_{\text{static}} &= [r_s, r_g, X_b, X_a], \\ \mathcal{I}_{\text{ref}} &= [r_s, r_g, M_{\text{pre}}^{1:K}, X_b, \\ &\quad M_{\text{mid}}^{1:L}, X_a]. \end{aligned} \quad (3)$$

Frames in each panel are ordered left-to-right and top-to-bottom. Empty cells use a fixed padding image, and the prompt records the actual panel count so that image and text positions remain aligned. Panels may be rendered offline or assembled and cached by the data loader under the same ordering and padding contract. For same-episode queries, pre-query and between-state panels summarize the actual events preceding and separating the endpoints. For synthetic OOD pairs, they instead provide nominal successful progression at the aligned source indices.

Context provenance follows the query type. Standard training pairs use temporal panels from the same expert episode as the queried states. Synthetic OOD pairs use panels exclusively from the aligned successful source episode, while X_b and X_a remain the edited observations; we refer to these as *source-only reference panels*. Online inference instead con-

structs panels from the observed rollout history, as described in the *Closed-Loop Reward Framework* subsection.

Structured OOD Data Construction

We use OOD to denote counterfactual states outside the distribution of successful source executions. Successful trajectories are drawn from the RoboCasa (Nasiriany et al. 2024) and LIBERO (Liu et al. 2023) simulation benchmarks and from real-robot AgiBot World trajectories (AgiBot-World Contributors et al. 2025). Dataset-specific camera streams are mapped to common view slots; unavailable auxiliary views are padded and masked according to their availability. Data splits are formed at the source-episode level before OOD generation, and every edited descendant remains in the same split as its source episode.

Given a successful source trajectory τ and instruction c , we apply a task- and phase-aware edit operator to construct an aligned counterfactual trajectory:

$$\begin{aligned} \tilde{\tau}_z &= \mathcal{E}_z(\tau, c), \\ z &\in \mathcal{Z}_{\text{rob}} \cup \mathcal{Z}_{\text{neg}}. \end{aligned} \quad (4)$$

Here z denotes the edit type, $\mathcal{E}_z$ applies it over a phase-valid interval, and $\tilde{\tau}_z$ contains source-index-aligned edited endpoints. The sets $\mathcal{Z}_{\text{rob}}$ and $\mathcal{Z}_{\text{neg}}$ contain semantics-preserving and task-invalid operators, respectively.

Negative operators include wrong-object interaction, empty grasp, non-release, and target replacement, whereas

robust operators include partial occlusion, background variation, irrelevant distractors, and non-target color changes. Operator availability is primitive-dependent, and each edited sequence uses one operator specification over a phase-valid interval and across all available camera views. Each edited frame retains its source episode, frame index, view, and operator metadata for aligned pair compilation. Full prompt, temporal-localization, and view-specific editing details are provided in the supplementary.

For synthetic OOD pairs, the task-reference anchors and all temporal panels are drawn from the aligned successful source episode, while the intervention affects only the queried endpoints. Edited observations never enter these source-reference panels. Online deployment follows the rollout-history construction described below and therefore does not require source selection or source-index alignment.

OOD-aware Signed Progress Space

For a successful source trajectory $\tau = \{s_0, \dots, s_{T-1}\}$, define

$$g_i = \frac{i}{T-1}. \quad (5)$$

Here P_i and R_i denote the original positive state and its aligned semantics-preserving robust state, respectively; N_i denotes an aligned task-invalid state instantiated only at phase-valid indices $i > 0$. For each instantiated state, we assign

$$\Phi(P_i) = \Phi(R_i) = g_i, \quad \Phi(N_i) = -g_i. \quad (6)$$

The sign separates valid execution from task-invalid failure, while $|\Phi(N_i)| = g_i$ indexes failure depth by the aligned source phase. Thus, $-g_i$ is a designed supervision coordinate rather than a calibrated failure probability or a literal percentage of negative completion. Since every instantiated negative state satisfies $i > 0$, we have $\Phi(N_i) < 0$; such states represent task-invalid execution rather than arbitrary difficulty or visual novelty.

Define $b(P_i) = b(R_i) = 0$ and $b(N_i) = 1$, where $b(\cdot)$ is the binary negative-branch indicator. For a pair (X_b, X_a) , let

$$\begin{aligned} \phi_b &= \Phi(X_b), & \phi_a &= \Phi(X_a), \\ \Delta_{ba} &= \phi_a - \phi_b, & L_{ba} &= -\max\{b(X_b), b(X_a)\}. \end{aligned} \quad (7)$$

Here L_{ba} selects the lower endpoint of the normalization interval: $L_{ba} = 0$ for valid-only pairs and $L_{ba} = -1$ whenever either endpoint belongs to the negative branch. The signed Hop target is

$$\mathcal{H}_{ba} = \begin{cases} 0, & \Delta_{ba} = 0, \\ \frac{\Delta_{ba}}{1 - \phi_b}, & \Delta_{ba} > 0, \\ \frac{\Delta_{ba}}{\phi_b - L_{ba}}, & \Delta_{ba} < 0, \end{cases} \quad (8)$$

The corresponding regression label is

$$y_{ba} = 100 \text{ clip}(\mathcal{H}_{ba}, -1, 1), \quad (9)$$

where the factor 100 preserves the output scale of the original GRM interface; it does not turn the continuous Hop into

a three-class target. Zero denotes same-progress invariance rather than uncertainty or a generic neutral class. Following the Hop normalization of Tan et al. (2026), $\mathcal{H}_{ba}$ measures the fraction of the available potential range in the queried direction: positive transitions are normalized toward the goal 1, whereas negative transitions are normalized toward the branch floor L_{ba} . Replacing the original floor 0 with the pair-dependent L_{ba} extends the same invertible mapping to failure and recovery; clipping is used only as a numerical safeguard.

Training includes valid pairs $U_i \rightarrow V_j$ with $U, V \in \{P, R\}$, failure pairs $U_i \rightarrow N_j$, recovery pairs $N_i \rightarrow U_j$, and within-failure pairs $N_i \rightarrow N_j$. A same-index pair $P_i \leftrightarrow R_i$ has zero Hop and explicitly supervises invariance, whereas cross-index robust pairs such as $R_i \rightarrow R_j$ retain nonzero progress or regression according to $g_j - g_i$.

We train the GRM with mean-squared regression:

$$\mathcal{L}_{\text{pair}}(\theta) = \mathbb{E}_{(X_b, X_a) \sim \mathcal{D}} [(\hat{y}_{ba} - y_{ba})^2]. \quad (10)$$

We retain start- and goal-anchored comparisons and use fixed family quotas so that abundant positive and within-branch pairs do not dominate robust, failure, and recovery pairs.

Signed-Hop Curriculum

The signed Hop also provides a task-aligned measure of training difficulty. We rank examples by $|\mathcal{H}_{ba}|$ within each pair family, so that frequent families do not dominate selection. The large-Hop pool $\mathcal{D}_{\text{large}}$ contains the top 40% of examples by $|\mathcal{H}_{ba}|$ within each family, while $\mathcal{D}_{\text{fine}}$ contains the remaining small-Hop and zero-Hop calibration examples.

Training uses two 200K-example pairwise-reward stages, totaling 400K pairwise examples. Stage 1 samples from $\mathcal{D}_{\text{large}}$ to learn global progress, failure, and recovery geometry. Stage 2 primarily samples from $\mathcal{D}_{\text{fine}}$ to improve local calibration and robust invariance while replaying a fraction ρ of large-Hop examples:

$$\begin{aligned} \mathcal{D}_2 &= (1 - \rho)\mathcal{D}_{\text{fine}} \\ &\quad + \rho\mathcal{D}_{\text{large}}. \end{aligned} \quad (11)$$

We use $\rho = 0.25$, so 25% of Stage 2 examples are replayed large-Hop examples and 75% are fine or zero-Hop examples. The replay ratio changes only the Stage 2 mixture; it does not alter the Stage 1 composition or the 40% large-Hop selection rule.

Closed-Loop Reward Framework

At rollout step t , let X_0 , $X_G = r_g$, X_{t-1} , and X_t denote the initial state, goal reference, previous observed state, and current observed state. Online inference constructs temporal panels directly from observations available in the current rollout and therefore requires neither a successful source episode nor an endpoint-to-source alignment. To avoid repeatedly writing the full GRM input, define the normalized query

$$\begin{aligned} h_\theta(X_b, X_a; \mathcal{C}) \\ = \text{clip}\left(\frac{f_\theta(c, \mathcal{R}, \mathcal{C}, X_b, X_a)}{100}, -1, 1\right). \end{aligned} \quad (12)$$

The online context is constructed separately for each query from the observed rollout prefix. For the forward-anchored query $X_0 \rightarrow X_t$, the pre-query panel is empty and observations strictly between X_0 and X_t populate the between-state panels, yielding $\mathcal{C}_t^F$. For the incremental query $X_{t-1} \rightarrow X_t$, observations before X_{t-1} populate the pre-query panels, while any observations between X_{t-1} and X_t populate the between-state panels, yielding $\mathcal{C}_t^I$. The goal observation X_G is a non-temporal task anchor, so the backward-anchored query is evaluated without temporal panels. When an explicitly aligned expert episode is available, the same slots may optionally use its source-reference frames.

The three normalized predictions are therefore

$$\begin{aligned}\hat{h}_t^F &= h_\theta(X_0, X_t; \mathcal{C}_t^F), \\ \hat{h}_t^B &= h_\theta(X_G, X_t; \emptyset), \\ \hat{h}_t^I &= h_\theta(X_{t-1}, X_t; \mathcal{C}_t^I).\end{aligned}\quad (13)$$

All online temporal panels contain only observations available before or at rollout step t ; no query uses future rollout observations. We denote the resulting augmented online state by $\bar{s}_t = (X_t, \mathcal{C}_t, m_t)$, where $\mathcal{C}_t$ collects the temporal query context and m_t denotes the recursive estimator memory.

Because the initial state has potential zero, a negative forward-anchored score operationally indicates the negative branch. For $t \geq 1$, we define

$$\begin{aligned}\hat{b}_t &= \mathbb{I}[\hat{h}_t^F < 0], \quad \hat{b}_0 = 0, \\ L_t^F &= L_t^B = -\hat{b}_t, \quad L_t^I = -\max\{\hat{b}_{t-1}, \hat{b}_t\}.\end{aligned}\quad (14)$$

Here $\hat{b}_t$ is the online branch gate: L_t^F and L_t^B depend on the current branch, while L_t^I depends on both adjacent branches so that a negative-to-valid recovery pair retains the floor -1 .

We invert a normalized Hop using

$$\Gamma(h; \phi, L) = \begin{cases} \phi + h(1 - \phi), & h \geq 0, \\ \phi + h(\phi - L), & h < 0. \end{cases}\quad (15)$$

The three recovered potential estimates are

$$\begin{aligned}\hat{\Phi}_t^F &= \Gamma(\hat{h}_t^F; 0, L_t^F), \\ \hat{\Phi}_t^B &= \Gamma(\hat{h}_t^B; 1, L_t^B), \\ \hat{\Phi}_t^I &= \Gamma(\hat{h}_t^I; \hat{\Phi}_{t-1}^I, L_t^I),\end{aligned}\quad (16)$$

and are fused as

$$\hat{\Phi}_t = \text{clip}\left(\frac{\hat{\Phi}_t^F + \hat{\Phi}_t^B + \hat{\Phi}_t^I}{3}, -1, 1\right), \quad \hat{\Phi}_0 = 0. \quad (17)$$

After inverse normalization, the three terms provide complementary estimates of the context-conditioned online potential, denoted $\hat{\Phi}_t \equiv \hat{\Phi}(\bar{s}_t)$. We retain the fixed unweighted multi-anchor aggregation used by Tan et al. (2026); the fusion weights are neither learned nor selected on the evaluation set.

Finally, the reward supplied to the downstream RL algorithm is

$$r_t^{\text{GRM}} = r_t^{\text{task}} + \gamma_{\text{RL}} \hat{\Phi}_{t+1} - \hat{\Phi}_t, \quad (18)$$

where r_t^{task} is the original environment reward, γ_{RL} is the discount factor used by policy optimization, and $\hat{\Phi}_t = \hat{\Phi}(\bar{s}_t)$.

With the GRM fixed during policy optimization, using the same discount and assigning zero potential to the absorbing terminal state retains the standard potential-based shaping form on the augmented process (Ng, Harada, and Russell 1999). The resulting potential is designed to be insensitive to semantics-preserving appearance changes, responsive to task-invalid failures, and positive for valid recovery transitions.

Experiments and Results

Experimental Setup

Training matrix. GRM-8B-Pro, released by Tan et al. (2026), serves as a prior baseline; Robo-Dopamine 2.0-8B uses Qwen3-VL-8B. We vary OOD supervision, disabled or $2 \times 2/3 \times 3$ temporal context panels, prebuilt or runtime construction, and visual-QA mixing. The main model uses 400K signed pairs with OOD supervision and prebuilt 3×3 panels: standard pairs use same-episode expert history, whereas synthetic OOD pairs use source-aligned successful-reference history. The RM is not trained on downstream rollout trajectories. A fixed 200K visual-QA mix (QA200K) is excluded from the signed-pair budget. Runtime variants construct the same ordered panels at data-loading time and pad missing views. QA200K is included identically across runs to preserve broad visual-language understanding and instruction following.

Curriculum controls. All Signed-Hop variants fix the Qwen3-VL-8B backbone (Bai et al. 2025), OOD supervision, prebuilt 3×3 panels with one between-state panel, token budget, QA200K mix, and 400K signed pairs split evenly across two stages. We compare random mixing (E0), the matched E4 pool without ordering (E1), reverse fine-to-large training (E2), large-to-fine training without replay (E3), and large-to-fine training with $\rho = 0.25$ replay (E4). E1 isolates ordering; E3 isolates replay. The same QA200K auxiliary mix is fixed across curriculum variants and excluded from the signed-pair budget; each curriculum stage contains 200K examples.

Evaluation protocol. We report trajectory-level visual order consistency (VOC): pairwise progress predictions are accumulated along each trajectory and correlated with the ground-truth order using Spearman correlation. Five families cover in-distribution success (ID), temporal memory, OOD-negative, OOD-robust, and OOD-temporal cases, evaluated with `static8` and `history_panel` inputs. Temporal cases include multi-stage, multi-object, and mid-episode trajectories; OOD-temporal cases add edits such as non-release. Temporal-memory examples come from dedicated temporal manifests and trajectories explicitly marked as history-dependent. We report aggregate VOC for each family rather than a per-cause breakdown. Together, these families probe event-, phase-, and order-dependent progress instead of end-point success alone. Offline synthetic-OOD evaluation uses source-reference panels aligned by dataset metadata, whereas simulation and real-robot RL use panels constructed from the observed rollout history.

Model	Input	ID	Temp.	Neg.	Rob.	OOD-T	Avg.	Base	Training condition	Static	Panel	Gain
GRM	static	0.974	0.977	0.951	0.939	0.948	0.958	GRM	baseline	0.958	0.965	+0.007
Robo-Dopamine 2.0	static	0.981	0.984	0.985	0.906	0.979	0.967	Robo-Dopamine 2.0	3×3+QA	0.967	0.986	+0.019
Robo-Dopamine 2.0	panel	0.991	0.994	0.994	0.958	0.991	0.986	GRM	OOD+no panel+QA	0.967	0.977	+0.010
								GRM	OOD+2×2	0.957	0.985	+0.028
								GRM	OOD+3×3	0.955	0.988	+0.033
								Robo-Dopamine 2.0	OOD+no panel	0.965	0.974	+0.009
								Robo-Dopamine 2.0	OOD+2×2	0.962	0.985	+0.023
								Robo-Dopamine 2.0	OOD+3×3	0.960	0.987	+0.027

Table 1: Trajectory-level VOC across five benchmark families. GRM denotes the GRM-8B-Pro baseline; Robo-Dopamine 2.0 is one QA200K-mixed Qwen3-VL-8B checkpoint evaluated with static or prebuilt 3×3 panel inputs. ID, Temp., Neg., Rob., and OOD-T denote in-distribution, temporal-memory, OOD-negative, OOD-robust, and OOD-temporal evaluation. Avg. is the unweighted mean; higher is better.

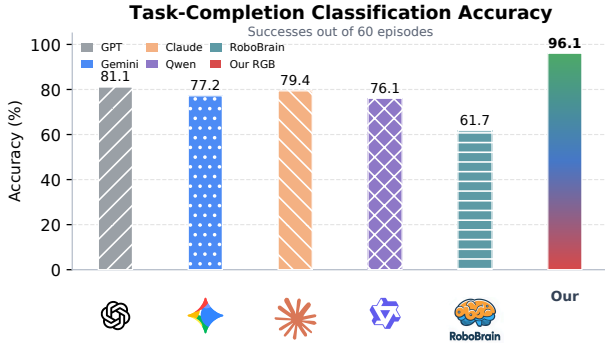


Figure 2: Task-completion classification accuracy on 60 episodes. We compare GPT-5.5, Gemini-3-Pro, Claude-Opus-4.8, Qwen3.5-397B-A17B-fp8, RoboBrain 2.0, and Robo-Dopamine 2.0-8B. This discrete label metric is distinct from VOC.

Offline Results

Panels raise Robo-Dopamine 2.0 average VOC from 0.967 to 0.986 and OOD-robust VOC from 0.906 to 0.958; GRM-8B-Pro reaches 0.958 average static VOC. The same-checkpoint comparison isolates the benefit of training-aligned temporal context from model capacity and training data.

Robo-Dopamine 2.0-8B obtains 96.1% task-completion accuracy, versus GPT-5.5 (81.1%) (OpenAI 2026), Claude-Opus-4.8 (79.4%) (Anthropic 2026), Gemini-3-Pro (77.2%) (Google 2025), Qwen3.5-397B-A17B-fp8 (76.1%) (Qwen Team 2026), and RoboBrain 2.0 (61.7%) (BAI RoboBrain Team et al. 2025).

Temporal Context and OOD Ablations Structured panels provide the largest gain: moving from OOD no-panel training to 3×3 panels raises panel VOC from 0.977 to 0.988 for GRM and from 0.974 to 0.987 for Robo-Dopamine 2.0. OOD supervision also improves panel VOC over no-OOD, no-panel training, but the panel contribution is larger. QA200K acts as a weak regularizer for no-panel models, with a smaller marginal effect once structured temporal context is available. Across backbones, one or two between-state panels perform best; larger panels are not monotonic under a fixed visual-token budget, and prebuilt panels out-

Table 2: Temporal-context and OOD ablations in average VOC. Static and Panel query the same checkpoint; +QA denotes QA200K. Full results are supplementary.

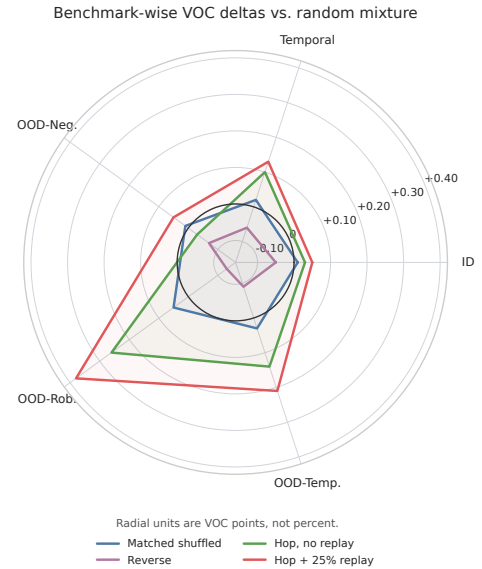


Figure 3: Per-family VOC gains of Signed-Hop over its matched-pool shuffled control. Replay sensitivity is provided in the supplementary material.

perform runtime construction. The non-monotonic trend reflects a trade-off between additional temporal evidence and the visual-token budget. Runtime construction also changes panel provenance, image order, and prompt tokens relative to the training distribution. Full panel-size, panel-count, and construction factorials are provided in the supplementary material.

Signed-Hop Curriculum Signed-Hop first learns global geometry from large-Hop pairs, then refines local calibration with small and zero-Hop pairs while replaying large-Hop examples. Under the fixed 400K budget, 25% replay reaches 0.9872 average VOC, exceeding the matched-pool shuffled control (0.9858) and no replay (0.9866). The shuffled control uses exactly the same sample pool as the replay curriculum, isolating ordering from data composition; the no-replay variant isolates retention of large-Hop geometry during local refinement.

Figure 3 shows gains in every family, especially OOD-

Reward condition	OOD	Hist.	Cup	Plate	Hand.	Avg.
RLinf Sparse	—	—	94.2	95.0	68.2	85.8
C00 Controlled	×	×	77.5	82.1	57.0	72.2
C01 History	×	✓	79.5	88.6	57.1	75.1
C10 OOD	✓	×	85.2	89.5	67.1	80.6
C11 Full	✓	✓	96.2	94.6	69.7	86.8

Table 3: RoboTwin success rate (%). Sparse uses 1,000 epochs; C00–C11 use 300. Avg. is the unweighted task mean.

robust and OOD-temporal cases. Curriculum gains complement the larger gains from temporal context panels. Signed-Hop is training-only: large-Hop pairs establish global progress, failure, and recovery geometry before small and zero-Hop pairs refine local calibration. The full replay sweep and branch-wise diagnostics are provided in the supplementary material.

Reinforcement Learning

In all downstream RL experiments, the RM remains frozen and the history-enabled conditions populate the same panel slots causally from the observed rollout prefix; no source alignment or repetition labels are used.

Simulation on RoboTwin We optimize task-specific OpenVLA-OFT (Kim, Finn, and Liang 2025) policies with GRPO (Shao et al. 2024) in RLinf (Yu et al. 2025) on three RoboTwin 2.0 tasks (Chen et al. 2026b): `place_empty_cup`, `place_container_plate`, and `handover_block`. Conditions share policy initialization, environment, optimizer, and evaluation protocol, so the controlled factorial tests whether OOD supervision and temporal context improve policy learning. RLinf Sparse uses only a termination-triggered success reward (scale 5) for 1,000 epochs; C00–C11 use the shaping interface in Eq. (18) for 300 epochs. C00 enables neither OOD supervision nor rollout-history panels, C01 adds rollout-history panels, C10 adds OOD supervision, and C11 uses both. Complete policy, GRPO, evaluation, and task-specific hyperparameters are supplementary.

Within 300 epochs, rollout-history panels raise mean success from 72.2% (C00) to 75.1% (C01), OOD supervision raises it to 80.6% (C10), and their combination reaches 86.8% (C11). C11 exceeds the 1,000-epoch sparse baseline (85.8%) overall and on two of three tasks, supporting complementary gains from rollout history and OOD supervision. Relative to C10 and C01, C11 improves the mean by 6.2 and 11.7 points, respectively. It is best on `place_empty_cup` and `handover_block`; the sparse baseline remains 0.4 points higher on `place_container_plate`.

Real-World Repeated Insertions We evaluate ConRFT (Chen et al. 2025) on dual Franka arms with multi-view RGB observations. In *Insert Square*, one arm aligns a four-hole block with four pegs for $K = 4$ consecutive attempts. Event Sparse uses a reward after each insertion; C00–C11 add GRM shaping. A terminal-only pilot did not learn. C00–C11 use 200 online episodes and Event Sparse uses 300;

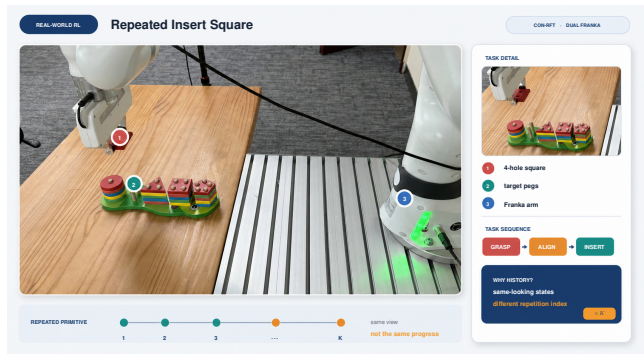


Figure 4: Dual-Franka repeated *Insert Square* setup with ConRFT. The numbered callouts identify the four-hole square, target pegs, and acting arm. Each trial has $K = 4$ attempts, creating visually similar states at different cumulative progress.

Reward	OOD	Hist.	Inserts / 80	Full / 20
Event Sparse	—	—	48	8
C00 Controlled	×	×	46	7
C01 History	×	✓	67	13
C10 OOD	✓	×	50	9
C11 Full	✓	✓	71	15

Table 4: Repeated *Insert Square*. Event Sparse uses only the per-insertion event reward; C00–C11 additionally use GRM shaping. Inserts counts successes over 80 attempts; Full counts 20 trials completing all four attempts.

all conditions use 20 evaluation trials and shared controls. Details are supplementary.

Repeated insertion produces visually similar endpoints at different cumulative phases. The observed rollout history supplies event and phase information for interpreting recurring endpoint observations. Matched real-robot traces show non-monotonic progress jumps with static queries and more order-consistent estimates with history panels; these traces are provided in the supplementary material.

Rollout-history panels provide the largest gain: relative to C00 and C10, C01 and C11 each add 21 successful insertions and 6 full-sequence successes. C11 reaches 71/80 insertions and 15/20 full successes, versus 48/80 and 8/20 for Event Sparse, using 200 rather than 300 online episodes.

Conclusion

Robo-Dopamine 2.0 combines history-conditioned pairwise prediction, an OOD-aware signed progress space, and a Signed-Hop curriculum in one reward interface. Standard queries use same-episode expert context, synthetic OOD queries retain edited endpoints with aligned successful-reference panels, and online queries use observed rollout history. The signed space models valid progress, robustness-preserving variations, task-invalid failures, and recovery, while the curriculum learns large transitions before fine-grained calibration. Offline and downstream RL results show

complementary gains from temporal context, OOD supervision, curriculum ordering, and replay across offline ordering, simulation, and real-world manipulation settings. The same pairwise interface supports offline progress estimation and closed-loop reward shaping. Future work will broaden task, OOD, and real-world evaluation.

References

- AgiBot-World Contributors; et al. 2025. AgiBot World Colosseo: A Large-Scale Manipulation Platform for Scalable and Intelligent Embodied Systems. arXiv:2503.06669.
- Alakuijala, M.; McLean, R.; Woungang, I.; Farsad, N.; Kaski, S.; Marttinen, P.; and Yuan, K. 2024. Video-Language Critic: Transferable Reward Functions for Language-Conditioned Robotics. arXiv:2405.19988.
- Anthropic. 2026. Introducing Claude Opus 4.8. <https://www.anthropic.com/research/claude-opus-4-8>. Model ID: claude-opus-4-8. Accessed: 2026-07-24.
- BAAI RoboBrain Team; Cao, M.; Tan, H.; Ji, Y.; Lin, M.; Li, Z.; Cao, Z.; Wang, P.; Zhou, E.; Han, Y.; Tang, Y.; Xu, X.; Guo, W.; Lyu, Y.; Xu, Y.; Shi, J.; Du, M.; Chi, C.; Zhao, M.; Hao, X.; Zhao, J.; Zhang, X.; Rong, S.; Lyu, H.; Cai, Z.; Fu, Y.; Chen, N.; Zhang, B.; Zhang, L.; Zhang, S.; Liu, D.; Feng, X.; Wang, S.; Liu, X.; Jiao, Y.; Lyu, M.; Chen, Z.; He, C.; Ao, Y.; Sun, X.; He, Z.; Zheng, J.; Yang, X.; Shi, D.; Xie, K.; Zhang, B.; Nie, S.; Men, C.; Lin, Y.; Wang, Z.; Huang, T.; and Zhang, S. 2025. RoboBrain 2.0 Technical Report. arXiv:2507.02029.
- Bai, S.; Cai, Y.; Chen, R.; Chen, K.; Chen, X.; Cheng, Z.; Deng, L.; Ding, W.; Gao, C.; Ge, C.; Ge, W.; Guo, Z.; Huang, Q.; Huang, J.; Huang, F.; Hui, B.; Jiang, S.; Li, Z.; Li, M.; Li, M.; Li, K.; Lin, Z.; Lin, J.; Liu, X.; Liu, J.; Liu, C.; Liu, Y.; Liu, D.; Liu, S.; Lu, D.; Luo, R.; Lv, C.; Men, R.; Meng, L.; Ren, X.; Ren, X.; Song, S.; Sun, Y.; Tang, J.; Tu, J.; Wan, J.; Wang, P.; Wang, P.; Wang, Q.; Wang, Y.; Xie, T.; Xu, Y.; Xu, H.; Xu, J.; Yang, Z.; Yang, M.; Yang, J.; Yang, A.; Yu, B.; Zhang, F.; Zhang, H.; Zhang, X.; Zheng, B.; Zhong, H.; Zhou, J.; Zhou, F.; Zhou, J.; Zhu, Y.; and Zhu, K. 2025. Qwen3-VL Technical Report. arXiv:2511.21631.
- Baumli, K.; Baveja, S.; Behbahani, F.; Chan, H.; Comanici, G.; Flennerhag, S.; Gazeau, M.; Holsheimer, K.; Horgan, D.; Laskin, M.; Lyle, C.; Masoom, H.; McKinney, K.; Mnih, V.; Neitz, A.; Nikulin, D.; Pardo, F.; Parker-Holder, J.; Quan, J.; Rocktäschel, T.; Sahni, H.; Schaul, T.; Schroecker, Y.; Spencer, S.; Steigerwald, R.; Wang, L.; and Zhang, L. 2023. Vision-Language Models as a Source of Rewards. arXiv:2312.09187.
- Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jackson, T.; Jesmonth, S.; Joshi, N.; Julian, R.; Kalashnikov, D.; Kuang, Y.; Leal, I.; Lee, K.-H.; Levine, S.; Lu, Y.; Malla, U.; Manjunath, D.; Mordatch, I.; Nachum, O.; Parada, C.; Peralta, J.; Perez, E.; Pertsch, K.; Quiambao, J.; Rao, K.; Ryoo, M. S.; Salazar, G.; Sanketi, P. R.; Sayed, K.; Singh, J.; Sontakke, S.; Stone, A.; Tan, C.; Tran, H.; Vanhoucke, V.; Vega, S.; Vuong, Q. H.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yu, T.; and Zitkovich, B. 2023. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*. Daegu, Republic of Korea.
- Chen, Q.; Yu, J.; Schwager, M.; Abbeel, P.; Shentu, F.; and Wu, P. 2026a. SARM: Stage-Aware Reward Modeling for Long Horizon Robot Manipulation. In *The Fourteenth International Conference on Learning Representations*.
- Chen, T.; Chen, Z.; Chen, B.; Cai, Z.; Liu, Y.; Li, Z.; Liang, Q.; Lin, X.; Ge, Y.; Gu, Z.; Deng, W.; Guo, Y.; Nian, T.; Xie, X.; Chen, Q.; Su, K.; Xu, T.; Liu, G.; Hu, M.; Gao, H.-a.; Wang, K.; Liang, Z.; Qin, Y.; Yang, X.; Luo, P.; and Mu, Y. 2026b. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*.
- Chen, Y.; Tian, S.; Liu, S.; Zhou, Y.; Li, H.; and Zhao, D. 2025. ConRFT: A Reinforced Fine-Tuning Method for VLA Models via Consistency Policy. arXiv:2502.05450.
- Duan, J.; Pumacay, W.; Kumar, N.; Wang, Y. R.; Tian, S.; Yuan, W.; Krishna, R.; Fox, D.; Mandlkar, A.; and Guo, Y. 2025. AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation. In *The Thirteenth International Conference on Learning Representations*.
- Fu, Y.; Zhang, H.; Wu, D.; Xu, W.; and Boulet, B. 2024. Robot Policy Learning with Temporal Optimal Transport Reward. In *Advances in Neural Information Processing Systems*, volume 37, 122078–122103.
- Google. 2025. Gemini 3 Pro Preview. <https://ai.google.dev/gemini-api/docs/models/gemini-3-pro-preview>. Model ID: gemini-3-pro-preview. Accessed: 2026-07-24.
- Guhur, P.-L.; Chen, S.; Pinel, R. G.; Tapaswi, M.; Laptev, I.; and Schmid, C. 2023. Instruction-Driven History-Aware Policies for Robotic Manipulations. In Liu, K.; Kulic, D.; and Ichnowski, J., eds., *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, 175–187. PMLR.
- Hung, K.-H.; Lo, P.-C.; Yeh, J.-F.; Hsu, H.-Y.; Chen, Y.-T.; and Hsu, W. H. 2025. VICtoR: Learning Hierarchical Vision-Instruction Correlation Rewards for Long-horizon Manipulation. In *The Thirteenth International Conference on Learning Representations*.
- Kim, M. J.; Finn, C.; and Liang, P. 2025. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*. Los Angeles, CA, USA.
- Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E. P.; Sanketi, P. R.; Vuong, Q.; Kollar, T.; Burchfiel, B.; Tedrake, R.; Sadigh, D.; Levine, S.; Liang, P.; and Finn, C. 2025. OpenVLA: An Open-Source Vision-Language-Action Model. In Agrawal, P.; Kroemer, O.; and Burgard, W., eds., *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 2679–2713. PMLR.
- Lee, K.; Smith, L. M.; and Abbeel, P. 2021. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-Training. In

- Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 6152–6163. PMLR.
- Lee, T.; Wagenmaker, A.; Pertsch, K.; Liang, P.; Levine, S.; and Finn, C. 2026. RoboReward: General-Purpose Vision-Language Reward Models for Robotics. arXiv:2601.00675.
- Liu, B.; Zhu, Y.; Gao, C.; Feng, Y.; Liu, Q.; Zhu, Y.; and Stone, P. 2023. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, volume 36.
- Ma, Y. J.; Kumar, V.; Zhang, A.; Bastani, O.; and Jayaraman, D. 2023a. LIV: Language-Image Representations and Rewards for Robotic Control. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 23301–23320. PMLR.
- Ma, Y. J.; Sodhani, S.; Jayaraman, D.; Bastani, O.; Kumar, V.; and Zhang, A. 2023b. VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training. In *The Eleventh International Conference on Learning Representations*.
- Mao, Y.; Yu, Z.; Mao, W.; Li, Y.; Hu, Q.; Lan, Z.; Zhu, M.; and Chen, H. 2026. ARM: Advantage Reward Modeling for Long-Horizon Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4468–4477.
- Nair, S.; Mitchell, E.; Chen, K.; Ichter, B.; Savarese, S.; and Finn, C. 2022. Learning Language-Conditioned Robot Behavior from Offline Data and Crowd-Sourced Annotation. In Faust, A.; Hsu, D.; and Neumann, G., eds., *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, 1303–1315. PMLR.
- Nasiriany, S.; Maddukuri, A.; Zhang, L.; Parikh, A.; Lo, A.; Joshi, A.; Mandlekar, A.; and Zhu, Y. 2024. RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots. In *Proceedings of Robotics: Science and Systems*.
- Ng, A. Y.; Harada, D.; and Russell, S. J. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 278–287. San Francisco, CA, USA: Morgan Kaufmann.
- Ni, T.; Eysenbach, B.; and Salakhutdinov, R. 2022. Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 16691–16723. PMLR.
- OpenAI. 2026. GPT-5.5 System Card. <https://openai.com/index/gpt-5-5-system-card/>. Accessed: 2026-07-24.
- Qwen Team. 2026. Qwen3.5-397B-A17B-FP8. <https://huggingface.co/Qwen/Qwen3.5-397B-A17B-FP8>. Accessed: 2026-07-24.
- Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.
- Sontakke, S.; Zhang, J.; Arnold, S.; Pertsch, K.; Biyik, E.; Sadigh, D.; Finn, C.; and Itti, L. 2023. RoboCLIP: One Demonstration is Enough to Learn Robot Policies. In *Advances in Neural Information Processing Systems*, volume 36, 55681–55693.
- Tan, H.; Chen, S.; Xu, Y.; Wang, Z.; Chi, C.; Ji, Y.; Lyu, Y.; Zhao, Z.; Chen, X.; Co, P.; Xie, S.; Yao, G.; Wang, P.; Wang, Z.; and Zhang, S. 2026. General Process Reward Modeling for Robotic Reinforcement Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 22412–22422.
- Wang, Y.; Sun, Z.; Zhang, J.; Xian, Z.; Biyik, E.; Held, D.; and Erickson, Z. 2024. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 51484–51501. PMLR.
- Yang, Y.; Chen, M.; Qiu, Q.; Wu, J.; Wang, W.; Lin, B.; Guan, Z.; and He, X. 2024. Adapt2Reward: Adapting Video-Language Models to Generalizable Robotic Rewards via Failure Prompts. In *Computer Vision – ECCV 2024*, 163–180. Springer Nature Switzerland.
- Yu, C.; Wang, Y.; Guo, Z.; Lin, H.; Xu, S.; Zang, H.; et al. 2025. RLinf: Flexible and Efficient Large-Scale Reinforcement Learning via Macro-to-Micro Flow Transformation. arXiv:2509.15965.
- Zhang, Q.; Zhai, S.; Zhang, S.; Liu, L.; Zhang, T.; Huang, F.; and Zhou, M. 2026. A Generalist Pair-wise Progress Critic Model for Vision-Language-Action Robots. In *Proceedings of the 43rd International Conference on Machine Learning*.
- Zhou, X.; Chen, X.; Zhang, S.; Li, X.; Wan, S.; Hu, X.; Yuan, L.; Gan, L.; and Zhan, D.-c. 2026. MARVL: Multi-Stage Guidance for Robotic Manipulation via Vision-Language Models. arXiv:2602.15872.
- Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohllhart, P.; Welker, S.; Wahid, A.; Vuong, Q.; Vanhoucke, V.; Tran, H.; Soricut, R.; Singh, A.; Singh, J.; Sermanet, P.; Sanketi, P. R.; Salazar, G.; Ryoo, M. S.; Reymann, K.; Rao, K.; Pertsch, K.; Mordatch, I.; Michalewski, H.; Lu, Y.; Levine, S.; Lee, L.; Lee, T.-W. E.; Leal, I.; Kuang, Y.; Kalashnikov, D.; Julian, R.; Joshi, N. J.; Irpan, A.; Ichter, B.; Hsu, J.; Herzog, A.; Hausman, K.; Gopalakrishnan, K.; Fu, C.; Florence, P.; Finn, C.; Dubey, K. A.; Driess, D.; Ding, T.; Choromanski, K. M.; Chen, X.; Chebotar, Y.; Carbajal, J.; Brown, N.; Brohan, A.; Arenas, M. G.; and Han, K. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In Tan, J.; Toussaint, M.; and Darvish, K., eds., *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 2165–2183. PMLR.

Supplementary Material

OOD Data Construction Pipeline

We construct synthetic OOD data from successful trajectories in the RoboCasa, LIBERO, and RoboTwin 2.0 simulation benchmarks, together with real-robot AgiBot World data. Task- and phase-aware edits produce source-aligned counterfactual observations rather than arbitrary visual variation. Negative edits change a task-relevant execution condition, whereas robust edits change nuisance appearance while preserving the intended task state.

Figure 5 illustrates the synthetic-OOD provenance contract. Task-reference frames and temporal panels are drawn from the aligned successful source episode, while edited before-after views remain the actual query evidence. Standard training instead uses same-episode expert history, and online inference uses the observed rollout history, following the context contract in the main paper.

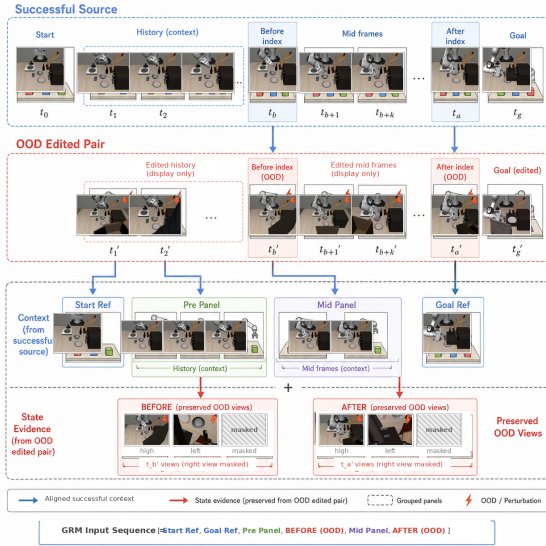


Figure 5: OOD provenance case study. All task-reference and temporal-context images are drawn from the aligned successful source episode, while only the queried before-after views contain the OOD edits.

Given a successful source trajectory from dataset d ,

$$\begin{aligned} \tau &= \{o_i^v \mid i \in \{0, \dots, T-1\}, v \in \mathcal{V}_d\}, \\ \tilde{\tau}_z &= \{\tilde{o}_i^v \mid i \in \mathcal{S}_z(\tau), v \in \mathcal{V}_d\}, \end{aligned} \quad (19)$$

where $\mathcal{V}_d \subseteq \{h, l, r\}$ is the available camera set after mapping dataset-specific streams to canonical high/third-person, left-wrist, and right-wrist slots. Unavailable slots are padded and masked by the data loader. The variable z denotes one OOD operator, and $\mathcal{S}_z(\tau)$ is its phase-valid interval. The same operator specification is applied throughout the interval, using references from the corresponding source camera.

Pipeline overview. The construction proceeds in six steps: (i) normalize the instruction, identify the task-relevant target, and map the task to manipulation primitives; (ii) assign a

semantically compatible negative or robustness operator; (iii) localize the trajectory phase in which the intervention is meaningful; (iv) construct a multi-reference editing request for every selected frame and camera; (v) generate the edited observation under operator-specific, geometric, and view-specific constraints; and (vi) retain source-episode, frame, camera, prompt, and reference provenance for downstream reward-pair construction.

Task Parsing and Operator Assignment

The task instruction is first normalized and mapped to one or more manipulation primitives. We use seven primitive categories: *grasp-lift*, *place-transfer*, *open*, *close*, *toggle-on*, *toggle-off*, and *push-move*. A pick-and-place instruction, for example, activates both *grasp-lift* and *place-transfer*, whereas opening a drawer, toggling a device, or pushing an object activates the corresponding interaction primitive.

We additionally extract a target phrase from the instruction. The parser removes the action prefix and truncates the remaining phrase at spatial or relational markers such as “into,” “onto,” “from,” or “next to.” The resulting phrase is inserted into the generation prompt and used to distinguish the task object from non-target scene objects.

Primitive-to-operator mapping. Grasp-lift tasks admit the negative operators *wrong object*, *empty grasp*, and *target replacement*, together with all robust operators. Place-transfer tasks admit *wrong object*, *non-release*, and *target replacement*, again together with all robust operators. Open, close, toggle, and push tasks use the robust operators *foreground occlusion*, *background change*, *distractor addition or removal*, and *non-target color change*, which preserve the relevant articulated, on/off, or displacement state.

We apply two semantic guards before assigning an operator. First, for instructions involving multiple legitimate task objects, identity-changing edits such as *wrong object* and *target replacement* are avoided when another valid negative operator is available. This prevents ambiguous correspondences among multiple admissible targets. Second, when the instruction explicitly specifies a target color, the *non-target color change* operator is restricted to non-target objects. Each generated OOD trajectory is assigned a single operator, so its edited frames form a temporally coherent counterfactual trajectory rather than a sequence of unrelated perturbations.

Phase-Aware Temporal Localization

Applying a failure edit to an arbitrary frame may produce a visibly different image without changing the task interpretation. We therefore localize each interaction-related operator to a primitive-dependent temporal window. For sampled frame index i , normalized source-trajectory progress is defined as

$$p_i = \frac{i}{T-1}. \quad (20)$$

Each frame is assigned an active primitive π_i . For common composite instructions, the active primitive changes according to source-trajectory progress. Grasp-to-place and open-to-place tasks switch to place-transfer at $p_i = 0.45$; place-to-

close tasks switch to close at $p_i = 0.65$; and toggle-to-place tasks switch to place-transfer at $p_i = 0.35$.

The valid progress intervals are $[0.18, 0.80]$ for grasp-lift, $[0.28, 0.95]$ for place-transfer, $[0.15, 0.85]$ for open and close, $[0.10, 0.75]$ for toggle-on and toggle-off, and $[0.12, 0.85]$ for push-move. These windows cover the principal interaction phases—approach, contact, transport, articulation, release, or displacement—while excluding early frames whose appearance is not yet informative for the selected intervention.

Let $G_z(\pi, p) \in \{\text{KEEP}, \text{EDIT}\}$ denote the fixed operator-specific eligibility rule. For *non-release*, the rule activates only in the release subphase; the remaining interaction-related operators activate at their first eligible task phase. The intervention onset is

$$i_{\text{start}} = \min \{i \mid G_z(\pi_i, p_i) = \text{EDIT}\}. \quad (21)$$

The corresponding operator-specific rule is used for *wrong object*, *empty grasp*, *non-release*, *target replacement*, and *foreground occlusion*. In contrast, *background change*, *distractor addition or removal*, and *non-target color change* begin at the first sampled state because they represent persistent scene-level nuisance variation. Applying these operators throughout the trajectory avoids an artificial appearance discontinuity at the interaction boundary.

For a standard single-stage or pick-and-place trajectory, the intervention is propagated over the selected suffix:

$$\mathcal{S}_z(\tau) = \{i_{\text{start}}, \dots, T - 1\}. \quad (22)$$

The active primitive supplied to the prompt may still change from grasping to transport or release as the trajectory progresses. For explicitly multi-stage instructions containing distinct subgoals, the selected interval is restricted to the contiguous segment governed by the primitive active at i_{start} . This prevents an edit defined for one subgoal from being propagated into a later subgoal with a different task meaning.

Multi-Reference Edit Construction

A single queried frame may not reveal the identity or original location of the target. This occurs when the target is already held, partially occluded by the gripper, visually similar to nearby objects, or absent from the current field of view. We therefore condition each edit on an ordered set of observations from the same successful source trajectory.

For queried frame i , the pipeline selects $K_i \leq 13$ reference indices:

$$\mathcal{J}_i = \{r_1, \dots, r_{K_i}\}, \quad r_k \neq i. \quad (23)$$

The indices are distributed over the complete sampled trajectory and retained in chronological order. When fewer than 13 alternative frames are available, all available source frames are used. For camera v , the visual input is

$$\mathcal{Q}_i^v = [o_{r_1}^v, \dots, o_{r_{K_i}}^v, o_i^v]. \quad (24)$$

Thus, each request contains K_i read-only references followed by one editable image; $K_i = 13$ whenever at least 13 alternative frames are available. The queried frame is excluded

from the reference set, and the final image in the sequence is always the only image that may be modified.

All references satisfy three alignment constraints: they originate from the same successful episode as the queried frame, use exactly the same camera, and are passed to the editor in temporal order. A head-camera query therefore receives only head-camera references, while a left- or right-wrist query uses references from the corresponding wrist camera. This prevents the editor from transferring the projection, visible object surfaces, or gripper geometry of one camera to another.

The ordered reference stack serves complementary roles. Early frames provide the initial target identity, target location, object inventory, and unmodified scene layout. Frames around the interaction phase reveal how the target approaches or enters the gripper. Later frames provide evidence about successful transport, release, or the final articulated state. The references are used to interpret the task and reconstruct local content; their global camera pose or complete image layout may not be copied into the queried frame.

Each available camera view is edited through a separate view-specific request:

$$\tilde{o}_i^v = \mathcal{G}(\mathcal{Q}_i^v; \mathcal{P}(c, \pi_i, z, v)), \quad (25)$$

where $\mathcal{G}$ denotes the instruction-following image editor and $\mathcal{P}$ denotes the structured prompt. The source index, OOD type, and task semantics are shared across the three requests, whereas geometric constraints are specialized to each camera.

Structured Prompt Design

The generation prompt is composed from a fixed set of functional blocks:

$$\mathcal{P} = \mathcal{P}_{\text{input}} \oplus \mathcal{P}_{\text{scene}} \oplus \mathcal{P}_{\text{task}} \oplus \mathcal{P}_{\text{target}} \oplus \mathcal{P}_{\text{operator}} \oplus \mathcal{P}_{\text{view}} \oplus \mathcal{P}_{\text{physical}} \oplus \mathcal{P}_{\text{output}}. \quad (26)$$

This decomposition separates the required intervention from the quantities that must remain invariant.

Input contract. The prompt first states that the request is an image-editing task rather than text-to-image generation. Images 1 through K_i are declared read-only references, and image $K_i + 1$ is declared the only editable image. The editor is prohibited from returning a reference image or reconstructing the scene from a different camera pose.

The output must preserve the queried resolution of 320×240 , field of view, crop, perspective, exposure, white balance, color temperature, sharpness, noise level, depth of field, and compression characteristics. The prompt explicitly prohibits global enhancement, denoising, restyling, or unnecessary full-image redrawing.

Scene and target binding. The prompt contains the complete task instruction, current frame ID, ordered reference-frame IDs, and camera identity. It states that all references come from the same successful trajectory and should be interpreted jointly to recover the original task process. When a target phrase can be extracted, it is included explicitly and used together with the trajectory references to identify the

task object. If several visually similar target candidates exist, a rule that removes, preserves, or moves the target is applied consistently to all relevant candidates unless the operator is explicitly defined for one instruction-referenced instance.

Primitive-specific task block. A task block specifies whether the current frame belongs to grasping, placement, opening, closing, toggling, or pushing. The grasp block, for example, identifies the object originally intended for acquisition, whereas the placement block emphasizes the transported object, support surface, and target placement region. This block prevents a visually plausible edit from becoming semantically inconsistent with the active manipulation stage.

Operator contract. The operator block defines the exact counterfactual intervention. For negative states, it specifies the task-relevant relation to invalidate, such as object occupancy inside the gripper, target identity, or release completion. For robust states, it specifies the nuisance factor that may change and explicitly states that the target, robot state, and task outcome must remain unchanged.

Global physical constraints. Across all OOD types, the prompt prohibits moving, rotating, reshaping, or repositioning the robot arm. Joint configuration, link geometry, and end-effector pose must remain identical to those in the queried source image. The gripper position, orientation, and opening are also preserved, except for the minimal local reconstruction required when removing or inserting held content.

Every moved, removed, added, replaced, or occluding object must respect the source perspective, support relation, contact geometry, illumination, shadow direction, material response, depth ordering, and occlusion pattern. Removed regions must be completed using surrounding scene texture without duplicated boundaries, smearing, holes, or inconsistent gripper geometry.

View-specific constraints. The prompt contains an additional block specialized to the queried camera. Head-camera prompts emphasize global scene consistency and the projected grasp corridor. Wrist-camera prompts restrict editing to the gripper, target, and immediately adjacent region while preserving visible finger geometry and local occlusion ordering.

Output contract. The editor is instructed to return exactly one image: the edited version of the final input image. No text, explanation, alternative image, or intermediate result is permitted. The requested intervention must remain visually identifiable, but realism and local consistency take priority over exaggerated change.

In abbreviated form, every prompt specifies (i) which images are read-only references; (ii) which final image is editable; (iii) the instruction, target, and active primitive; (iv) the OOD intervention and its state-dependent branch; (v) camera, robot, object, geometry, lighting, and image-quality invariants; and (vi) the single-image output requirement.

Negative Counterfactual Operators

Negative operators change a task-relevant execution condition while keeping the source robot pose and trajectory phase fixed.

Wrong object. If the gripper is empty or approaching the target, the task target in the grasp corridor is replaced by a visually distinct non-target object, while the original target is relocated to a plausible position outside the corridor. This makes the impending interaction correspond to the wrong object rather than an empty grasp. If the gripper is occupied, only the held content is replaced by an object with a different category, dominant color, and silhouette. The original target remains naturally situated elsewhere whenever visible.

Empty grasp. The defining condition is an empty region between the gripper fingers. If the gripper is approaching an object, the target and any object intersecting the opening, closing region, or forward grasp corridor are moved laterally outside the grasp path. If the gripper already contains an object, that object is removed without changing the gripper pose or opening. Only the minimal local region is reconstructed to recover the visible fingers, interior background, shadows, and reflections. The target is returned to its source-scene location when this location is supported by the trajectory references.

Non-release. The *non-release* operator converts a successful release or near-release state into a continued-holding state. If the target has already left the gripper, the same target is inserted back into the current gripper and its released instance is removed to avoid duplication. The gripper pose is not replaced by an earlier source pose. If the target is already held, its identity and appearance are preserved while the continued-holding configuration remains unchanged.

Target replacement. Only the instruction-referenced target is replaced. The replacement preserves the original target’s approximate image footprint, location, local support or grasp relation, and occlusion order, but differs clearly in category and silhouette. All surrounding objects, articulated states, and robot geometry remain fixed. The resulting state therefore preserves the local spatial arrangement while violating task-object identity. Unlike *wrong object*, this operator changes the identity of the instruction-referenced scene target rather than the identity of the object currently approached or held by the gripper.

Robustness-Preserving Operators

Robustness operators change image appearance without changing task semantics and therefore retain the same progress value as their aligned source states.

Foreground occlusion. A plausible close-range occluder, such as a hand, board, box, or cloth region, is inserted in front of the critical interaction area. The occluder may partially hide the target or gripper, but it may not cover the complete image. The underlying robot-object state, target identity, contact state, task outcome, and all pre-existing scene objects remain unchanged. The occluder is placed according to the queried camera’s depth ordering rather than used to redraw the hidden interaction into a different state.

Background change. One or two small modifications are applied to secondary background regions, such as a minor layout adjustment or local appearance variation away from the interaction area. Target identity and location, robot and gripper state, articulated-object state, and the geometry of the critical contact region remain unchanged.

Distractor addition or removal. A small number of non-target objects are added to or removed from background or secondary regions. Added objects must have plausible scale, support, material, illumination, and shadow; removed regions are completed using neighboring scene texture. New objects are not placed adjacent to the gripper or target, and no task-relevant object or receptacle may be removed.

Non-target color change. A small, natural change in color, saturation, or brightness is applied to one or two non-target objects. No object is added, removed, moved, rotated, or re-placed. The target color and identity, object geometry, scene layout, robot state, contact relations, and task outcome remain fixed. When color is part of the instruction, the target color is protected by both operator selection and the prompt.

Camera-Specific Geometric Constraints

The same semantic edit requires different geometric instructions under the head and wrist cameras. We therefore append a camera-specific block to each prompt.

Head camera. The head view exposes the global scene layout and the projected grasp corridor. For *empty grasp*, the prompt gives highest priority to clearing the region between the fingers, directly below the gripper, and along its projected approach path. For *wrong object*, it verifies that no valid target remains within the visually reachable region around an empty gripper. For *non-release*, it verifies that the final edited gripper is visibly non-empty.

Wrist cameras. Wrist views are treated as local close-up observations. Editing is restricted to the gripper, target, and immediately adjacent image region. The prompt prohibits expanding an object surface that is occluded in the queried wrist view, copying geometry from the head camera, or producing the opposite wrist’s observation. The left-wrist prompt preserves the visible left finger, contact boundary, and local reflection pattern; the right-wrist prompt applies the analogous constraint to the visible right-side geometry.

When an object is inserted, removed, or replaced inside the gripper, its boundary must remain behind the visible metal finger edges according to source depth ordering. If the robot itself occludes the interaction region, that occlusion is preserved rather than removed merely to make the edit more explicit. These constraints encourage the available edited views to express the same trajectory-level OOD semantics while respecting their distinct projections and visible local geometry.

Generation and Provenance

The OOD construction uses Gemini 3.1 Flash Image as the instruction-following image-editing backend in batch mode. Each frame-camera pair is submitted as one editing request

containing the ordered reference images, the final queried image, and the fully instantiated prompt.

Every request receives a unique identifier. Its record contains the source dataset, task instruction, episode ID, frame ID, normalized frame index, camera, active primitive, OOD type, OOD group, reference-frame IDs, generation prompt, backend identifier, generation timestamp, and source path. The identifier is subsequently used to associate each returned image with the exact source frame and camera. At the trajectory level, a manifest records the selected source-frame interval, operator, OOD group, onset mode, primary primitive, multi-stage indicator, multi-object indicator, reference configuration, and complete list of edited source-frame IDs.

This metadata preserves the alignment

$$\tilde{o}_i^v \longleftrightarrow o_i^v \longleftrightarrow (\tau, i, v, z). \quad (27)$$

The downstream loader can therefore reconstruct both the original successful observation and its edited counterpart at the same source progress index:

$$\begin{aligned} X_i &= \{o_i^v \mid v \in \mathcal{V}_d\}, \\ \tilde{X}_i^z &= \{\tilde{o}_i^v \mid v \in \mathcal{V}_d\}. \end{aligned} \quad (28)$$

For every OOD pair, all task-reference and temporal-context images are reconstructed exclusively from the aligned successful source episode. The generated trajectory contributes only the queried before-after observations. This source-only context contract preserves the source phase and panel ordering without replacing the wrong-object, empty-grasp, non-release, target-replacement, occlusion, or background-perturbation evidence contained in the OOD query.

The recorded provenance keeps the successful source state, edited observation, camera projection, and temporal context separately traceable. Negative and robust states are therefore compiled into the signed progress space using their shared source index rather than independently sampled trajectories.

Computing Infrastructure and Training Cost

Reward-model fine-tuning and RoboTwin simulation policy learning use eight NVIDIA H100 GPUs with 80 GB of memory per GPU (640 GB aggregate accelerator memory) for every training run. Real-world ConRFT training instead uses one NVIDIA RTX 4090. The retained experiment records do not provide verified host CPU, system RAM, operating-system, or CUDA details, so we report only the accelerator allocations rather than inferring unverified model or version information. A complete reward-model training run takes approximately 20 hours in this eight-GPU setting. Reward-model training uses 400K signed pairs across two 200K stages, together with the fixed QA200K auxiliary mix excluded from this pairwise budget. Each trainable reward-model configuration and policy-learning condition is trained once, and the GRM remains frozen in all downstream RL experiments.

Simulation Reinforcement Learning Details

We optimize task-specific OpenVLA-OFT SFT checkpoints with GRPO in RLinf on the RoboTwin 2.0 tasks

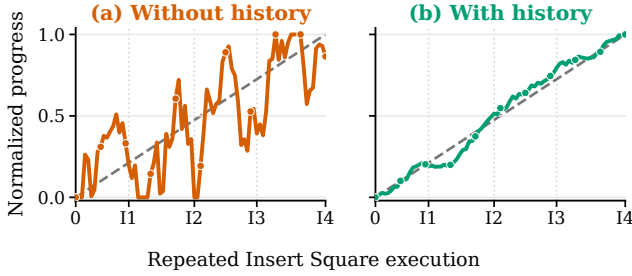


Figure 6: Logged progress predictions from matched static and rollout-history-conditioned executions of repeated *Insert Square*. Tick I_k , $k \in \{1, \dots, 4\}$, denotes the k -th insertion attempt, and the gray dashed line denotes execution order. Static predictions exhibit non-monotonic jumps at recurring states, whereas rollout-history conditioning produces more order-consistent estimates.

`place_empty_cup`, `place_container_plate`, and `handover_block`. The GRM remains frozen, and history-enabled conditions construct panels from the observed rollout prefix. The policy receives head-camera RGB and 14-D proprioception, predicts 14-D actions in 25-action chunks, and is LoRA-tuned in BF16 with FSDP. GRPO uses 128 parallel environments, groups of eight rollouts, normalized group-relative advantages, and $\gamma = 1$.

The sparse baseline is trained for 1,000 epochs and C00–C11 for 300 epochs. We run deterministic 128-episode evaluations every 20 epochs over fixed reset states. For `place_empty_cup`, `place_container_plate`, and `handover_block`, respectively, the episode horizons are 200, 150, and 400 steps and the learning rates are 10^{-4} , 2×10^{-4} , and 2×10^{-4} .

Real-World Experiments

Platform, Observations, and Task

The physical workcell contains two Franka manipulators, a Pika teleoperation interface, and calibrated ZED cameras. The camera system supplies synchronized wrist and third-person RGB observations to both the reward model and the policy. The GRM remains frozen during online policy learning. History-enabled conditions construct temporal panels from the observed rollout prefix; no successful source alignment or repetition label is provided online.

We evaluate the single-arm *Insert Square* task. The acting arm grasps a square block containing four holes, adjusts its position and orientation, and inserts it onto four upright pegs on the target board. Successful insertion requires millimeter-scale alignment. To expose temporal aliasing rather than only one-shot manipulation accuracy, each trial contains $K = 4$ consecutive back-and-forth insertion attempts. Consequently, similar pickup, alignment, and insertion observations recur at different cumulative progress levels. Figure 6 compares representative static and rollout-history-conditioned progress traces.

Hyperparameter	Symbol	Value
Global batch size	–	256
Learning rate	–	3×10^{-4}
Reward discount	γ	0.98
Demonstrations	–	40
Insertion attempts / trial	K	4
Online episodes (C00–C11)	–	200
Online episodes (Sparse)	–	300
Evaluation trials / condition	–	20
Insertion attempts / condition	–	80
Offline BC weight	β	1.0
Offline Q weight	η	0.1
Conservative penalty	α	0.1
Online BC weight	β'	0.5
Online Q weight	η'	1.0
Compute	–	$1 \times \text{RTX 4090}$

Table 5: Optimization settings for the real-world ConRFT experiments.

Offline-to-Online ConRFT Protocol

We use ConRFT as the policy-optimization procedure. Its policy retains a frozen Octo-Small observation backbone and replaces the original action head with a consistency policy; a learned critic supplies the Q objective. Offline initialization uses 40 teleoperated trajectories in a demonstration buffer $\mathcal{D}$. The policy is optimized with behavior cloning and calibrated Q-learning, while the critic includes a conservative penalty to stabilize value estimates outside the demonstration distribution.

During online adaptation, autonomous transitions are appended to replay buffer $\mathcal{R}$. A human operator intervenes when an action is unsafe or likely to cause failure, and corrected trajectories are returned to demonstration buffer $\mathcal{D}$. Updates sample equally from $\mathcal{D}$ and $\mathcal{R}$. The online policy retains the behavior-cloning and Q terms with weights shifted toward reward-driven improvement, while the critic uses the Bellman error without the offline conservative penalty. Intervention and data-routing rules are identical across reward conditions.

Table 5 summarizes the real-world optimization settings.

Controlled Comparison and Metrics

Every reported condition uses the same event-sparse task reward, emitted once after each successful insertion. The Event Sparse baseline uses only this reward, whereas C00–C11 add GRM shaping. A terminal-only pilot rewarding only completion of all four attempts did not learn and is excluded from the reported comparison. All reported variants start from the same policy initialization and use identical demonstration data, observations, resets, interventions, and evaluation. C00–C11 use 200 online episodes and Event Sparse uses 300. Each condition is then evaluated in 20 trials under the shared evaluation protocol. This design isolates the effects of OOD supervision and ordered history while making the larger sparse-reward interaction budget explicit. We report the number of successful insertions among the resulting 80 attempts and the number of full-sequence successes among

Strategy	Step	Temp.	Neg.	Rob.	OOD-T
No replay	200K	0.9860	0.9945	0.9510	0.9870
No replay	400K	0.9950	0.9937	0.9606	0.9924
25% replay	200K	0.9860	0.9945	0.9510	0.9870
25% replay	400K	0.9953	0.9945	0.9618	0.9931

Table 6: Stage 2 VOC for the two large-to-fine curricula. Moderate replay yields slightly higher final scores on the reported temporal and OOD families.

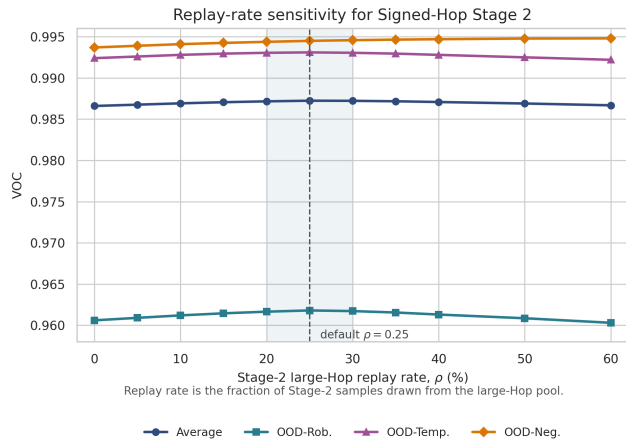


Figure 7: Sensitivity to the fraction ρ of Stage 2 samples replayed from the large-Hop pool. The shaded neighborhood marks the moderate-replay regime around the selected $\rho = 0.25$ setting.

20 trials. A trial contributes to the latter count only when all four insertion attempts succeed.

Additional Ablation Details

Additional Signed-Hop Diagnostics

The main paper reports the curriculum controls and the per-family improvement of the selected schedule. Here we separate the Stage 2 movement for the two large-to-fine variants and provide the complete replay-rate sensitivity sweep. The first 200K large-Hop examples are shared by E3 and E4; their second stages differ only in whether large-Hop pairs are replayed.

The replay rate changes only the Stage 2 mixture; it does not redefine the large-Hop pool used in Stage 1. Average VOC increases from 0.9866 without replay to 0.98724 at $\rho = 0.25$, remains nearly unchanged at $\rho = 0.30$ (0.98722), and decreases to 0.98668 at $\rho = 0.60$ (Figure 7). The shallow maximum near $\rho = 0.25$ is consistent with moderate replay retaining large-transition examples while preserving most of the fine- and zero-Hop calibration budget.

OOD-negative VOC increases from 0.9937 at $\rho = 0$ to 0.9948 at $\rho = 0.60$, whereas OOD-robust and OOD-temporal VOC peak near moderate replay and decline at larger ratios. The selected $\rho = 0.25$ achieves the highest observed overall average rather than the best score on every

branch. Replay is used only during reward-model training and is unrelated to inference-time memory or the RL replay buffer.

Trajectory-Level Case Studies

Figure 8 complements the aggregate VOC results with representative trajectory-level progress curves.

On the foreground-occlusion case, the static GRM remains close to zero through the middle of the episode and then jumps late. OOD supervision without history recovers much of the trend, while the panel-conditioned model follows the reference ordering most closely in the early and middle phases. The remaining late-episode gap shows that history does not eliminate all sensitivity to a strong foreground edit.

The in-distribution drawer case serves as a calibration check rather than an OOD stress test. All three learned curves preserve the episode order and reach completion. The same-episode panel-conditioned curve follows the reference closely over most of the rollout but saturates early at the final evaluated states. We retain this behavior rather than smoothing or selecting only favorable points, exposing both the benefit of temporal context and the remaining endpoint-calibration error.

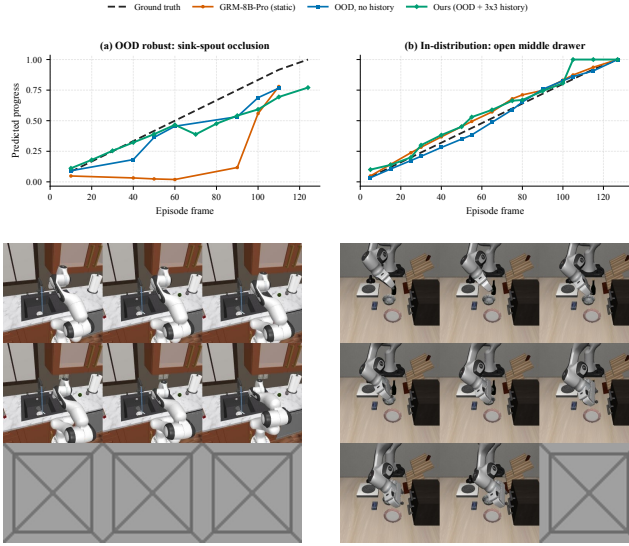


Figure 8: Episode-level progress traces on two held-out cases. Top: progress curves for a RoboCasa sink-spout episode with foreground occlusion and an in-distribution LIBERO middle-drawer episode. Bottom left: the source-reference pre-query panel for the sink-spout case. Bottom right: the same-episode pre-query panel for the drawer case. Gray cells denote fixed padding.

Compact Ablation Tables

Table 7 reports the complete checkpoint and configuration comparisons underlying the main ablation summary.

Base	Cond.	Type	Stat.	Hist.	Gain
GRM	base	Obs.	0.958	0.965	+0.007
Robo-Dopamine 2.0	3×3+QA	Obs.	0.967	0.986	+0.019
GRM	no OOD	Swp.	0.954	0.962	+0.008
GRM	OOD+QA	Swp.	0.967	0.977	+0.010
GRM	OOD+2×2	Swp.	0.957	0.985	+0.028
GRM	OOD+3×3	Swp.	0.955	0.988	+0.033
GRM	OOD+3×3+QA	Swp.	0.958	0.988	+0.030
Robo-Dopamine 2.0	OOD	Swp.	0.965	0.974	+0.009
Robo-Dopamine 2.0	OOD+QA	Swp.	0.967	0.977	+0.010
Robo-Dopamine 2.0	OOD+2×2	Swp.	0.962	0.985	+0.023
Robo-Dopamine 2.0	OOD+3×3	Swp.	0.960	0.987	+0.027
Robo-Dopamine 2.0	OOD+3×3+QA	Swp.	0.967	0.986	+0.019

Table 7: Compact ablation grid supporting the main summary. Obs. denotes the main reported checkpoint comparison, and Swp. denotes the controlled configuration sweep. Stat. and Hist. denote `static8` and `history_panel VOC`, respectively.

The controlled sweep shows complementary effects from OOD supervision and structured temporal panels. The QA mix provides a smaller additional effect once history panels are enabled.

Table 8 further examines the interaction between panel resolution and the number of between-state panels.

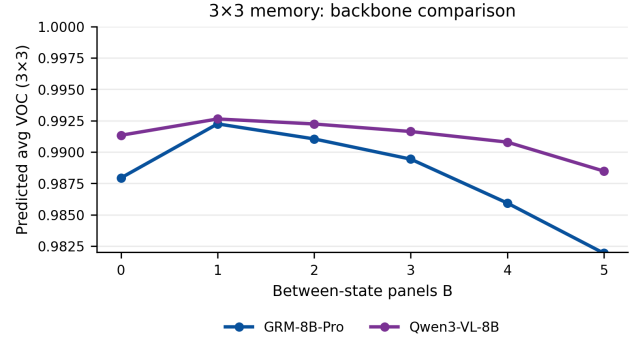


Figure 9: 3×3 memory-panel sweep over between-state panel count. Both backbones peak with one or two between-state panels; additional panels do not monotonically improve VOC.

	Grid	$B=0$	$B=1$	$B=2$	$B=3$	$B=4$	$B=5$
<i>GRM</i>							
1×1	0.965	0.971	0.969	0.967	0.963	0.959	
2×2	0.980	0.986	0.984	0.981	0.978	0.974	
3×3	0.988	0.992	0.991	0.989	0.986	0.982	
4×4	0.981	0.987	0.985	0.983	0.979	0.975	
<i>Robo-Dopamine 2.0</i>							
1×1	0.966	0.972	0.970	0.967	0.964	0.960	
2×2	0.981	0.987	0.985	0.982	0.979	0.975	
3×3	0.991	0.993	0.992	0.992	0.991	0.988	
4×4	0.982	0.988	0.986	0.984	0.980	0.976	

Table 8: Memory-layout sweep. B denotes the number of between-state panels. Both backbones perform best with 3×3 panels and one or two between-state panels.

Increasing panel capacity is not uniformly beneficial. Both backbones favor 3×3 panels with one or two between-state panels, whereas larger grids or panel counts provide no consistent improvement.

Figure 9 visualizes the same panel-count trend for the 3×3 configuration.

The curve confirms that the main benefit comes from adding a limited amount of between-state evidence rather than continuously increasing the visual-token load. Runtime panel construction is retained as a diagnostic because it also tests data-loading-time assembly and prompt alignment.

Overall, the ablations consistently favor structured 3×3 panels with one or two between-state panels. OOD supervision supplies the main robustness gain, while QA mixing and additional panel capacity have smaller effects once structured temporal context is available.