RAISE: LEARNING FROM FAILURES BY REWEIGHTING SUCCESSFUL ROBOT DEMONSTRATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning from demonstrations is a cornerstone of general-purpose robotic manipulation. Yet success-centered imitation often underuses failure experience, while monotonic progress supervision overlooks local setbacks and retries within successful executions. We introduce RAISE (Risk-Aware Imitation from Successful Experience), a data-centric framework that turns failures into learning priorities rather than imitation targets. RAISE adapts a stage-aware progress model using only trajectories that reach stable completion, with retry supervision capturing local degradation and recovery. The frozen model grounds failure localization in stage- and progress-matched successful experience through temporally corroborated action deviations, changes in success support, and progress regression. Successful retries and localized failure onsets yield risk anchors that guide offline reweighting of existing demonstrations, emphasizing segments with similar pre-action states and positive predicted task advancement. Simulation and real-robot experiments show that RAISE substantially improves overall task success across diverse imitation learners, including conventional behavior cloning, diffusion policies, and vision–language–action models, without changing policy architectures or imitation targets. Project page: here.

1 INTRODUCTION

A central challenge in embodied intelligence is to turn experience into reliable action in the physical world (Osa et al., 2018). Large-scale demonstration learning advances this goal by sharing manipulation knowledge across tasks, environments, and robot embodiments (O'Neill et al., 2024; Team et al., 2024; Kim et al., 2024; Black et al., 2024). Yet policy performance depends not only on model capacity and data volume, but also on how experience is selected, organized, and weighted (Hejna et al., 2025; Mandlekar et al., 2021; Lin et al., 2025; Hejna et al., 2024). Successful demonstrations show how tasks can be accomplished; failures reveal where this capability breaks down. Their complementarity suggests a broader role for data curation: not merely retaining better behavior, but identifying where learning the right behavior deserves greater emphasis.

Realizing this opportunity requires distinguishing local execution quality from trajectory outcomes: successful demonstrations can contain setbacks and retries, while failed executions may remain locally correct before a decisive deviation. Neither monotonic progress assumptions nor terminal labels alone adequately identify where learning should concentrate. As Fig. 1 illustrates, local risk can refocus learning on task-advancing behavior already present in successful demonstrations. Existing data-curation methods assess behavioral quality and filter suboptimal or redundant data (Zhang et al., 2025b; Dass et al., 2026). Stage- and retry-aware supervision incorporates task structure and local mistakes into progress assessment (Chen et al., 2026a;b; Qin et al., 2026), while outcome-based approaches use successful and failed executions for reward learning, policy optimization, and demonstration selection (Liang et al., 2026; Fei et al., 2026; Chen et al., 2025a; Xu et al., 2025). However, recognizing poor behavior does not fully exploit failure: errors do not make an entire rollout uninformative (Liu et al., 2025). The challenge is to strengthen imitation from recoverable mistakes and terminal failures without being misled by erroneous actions or unreliable progress estimates. Beyond states covered by successful experience, genuine deterioration can be confused with prediction error (Liang et al., 2026; Agia et al., 2024).

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.

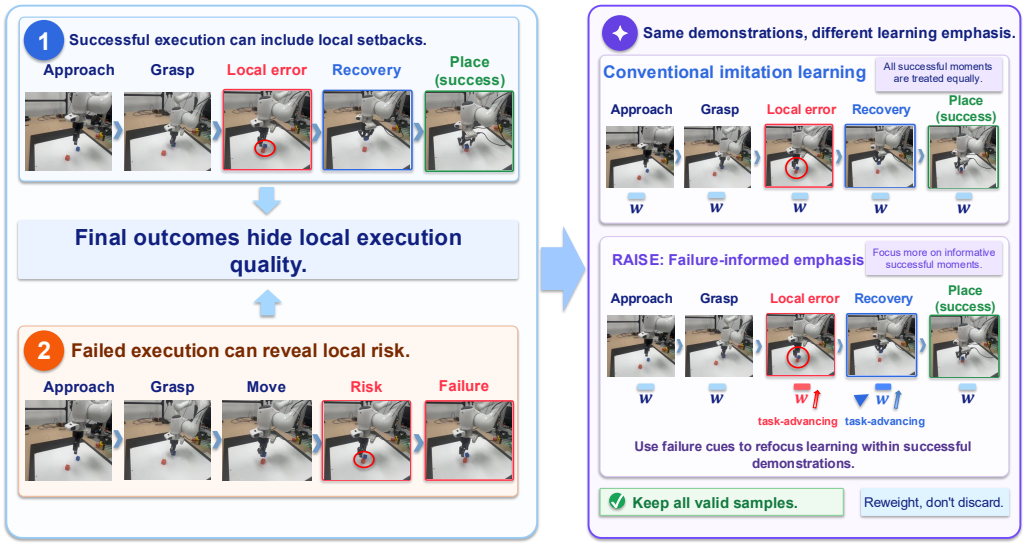

Figure 1: **Failures guide learning emphasis, not action targets.** Successful executions can contain setbacks and recovery; failed executions expose vulnerable contexts. RAISE reweights task-advancing segments in the same successful demonstrations while retaining all valid samples. Frames and event labels are conceptual illustrations, not temporal ground truth.

To address these challenges, we introduce RAISE (Risk-Aware Imitation from Successful Experience), a data-centric framework that learns more effectively from successes by reasoning about failures. RAISE adapts a stage-aware progress model only on stable successful trajectories, combining global task structure with local degradation–recovery dynamics. The frozen model localizes failures against task-, stage-, and progress-matched successful references through temporally corroborated action deviations, success-support changes, and progress regression. Successful retries and localized failures yield risk anchors that emphasize successful segments with similar pre-action states and positive predicted task advancement. All valid samples and their original action targets are retained; failed rollouts neither update the adapted progress model nor supply policy imitation targets. Reweighting is offline and leaves policy architectures unchanged. Simulation and real-robot experiments demonstrate substantial task-success gains for existing behavior-cloning methods, with analyses of progress accuracy, failure localization, and risk-guided reweighting.

Our contributions are threefold:

Success-only, retry-sensitive progress learning. We combine task-stage structure with local degradation–recovery preferences to learn a progress reference from stable successes.

Success-referenced failure localization. We identify failure onsets through temporally corroborated action deviations, success-support changes, and progress regression under stage- and progress-matched comparisons.

Risk-guided success reweighting. We use retry- and failure-derived risk to emphasize segments with positive predicted task advancement, improving imitation through reweighting rather than architectural changes.

2 Related Work

Data curation and weighted imitation. Robot data curation optimizes dataset mixtures (Hejna et al., 2024), estimates trajectory quality (Hejna et al., 2025; Kuhar et al., 2023), filters transitions (Zhang et al., 2025b), and retrieves demonstrations using task similarity, training influence, or execution outcomes (Du et al., 2023; Dass et al., 2026; Chen et al., 2025a). Weighted imitation uses discriminator scores or value estimates to favor desirable actions (Xu et al., 2022; Peng et al.,

2019; Wang et al., 2020). These approaches motivate distinguishing action quality from the need for additional supervision: unpredictable actions can still be essential (Hejna et al., 2025). RAISE emphasizes successful segments whose pre-action contexts resemble those implicated in retries or failures, without rejecting valid samples.

Progress and reward models. Temporal alignment (Sermanet et al., 2018; Dwibedi et al., 2019), visual–language representations (Ma et al., 2022; 2023; Alakuijala et al., 2024; Ma et al., 2025; Sontakke et al., 2023), and synthetic or comparative supervision (Zhang et al., 2025a; Lee et al., 2026; Liang et al., 2026) support progress and reward estimation. SARM/SARM2 align progress with semantic stages (Chen et al., 2026a;b). ReTVL learns retry-sensitive values and weights imitation chunks by predicted progress improvement (Qin et al., 2026). RAISE builds on these ideas but uses a distinct weighting signal: retry- and failure-derived risk is transferred to successful pre-action contexts and gated by predicted advancement. Unlike ReTVL's use of failed-terminal labels in value learning, RAISE analyzes failed rollouts only after task-specific progress adaptation is complete. All valid policy-training samples retain positive weight.

Failure analysis and policy improvement. Interactive imitation obtains expert corrections at policy-induced states (Ross et al., 2011; Kelly et al., 2019; Hoque et al., 2021; Liu et al., 2025); monitoring and diagnosis identify unreliable execution (Agia et al., 2024; Sagar et al., 2026). Offline-to-online and preference-based methods improve policies through values, rewards, or preferences (Nair et al., 2020; Chen et al., 2025c; Zhang et al., 2024; Xu et al., 2025). SRPO uses whole-trajectory successful references (Fei et al., 2026), while RePO-VLA retains useful failure prefixes and collects corrective rollouts (Liufu et al., 2026). GTP-FA (Grasp-Then-Plan with Failure Attribution) routes optimization between grasping and downstream execution (Xu et al., 2026). RAISE instead transfers localized risk to weights on existing successful actions, requiring neither new corrective targets nor online policy reinforcement learning (RL). It can complement a fixed GTP-FA system rather than replace its grasping pipeline.

3 METHOD

3.1 PROBLEM SETUP

An imitation policy $\pi_\theta(\cdot \mid x_t, \ell)$ maps observation context x_t and instruction ℓ to an action chunk $\mathbf{a}_t = (a_t, \ldots, a_{t+H_a-1})$. Single-step policies use $H_a = 1$; language-free policies omit ℓ. The fixed dataset $\mathcal{D}_E = \{(x_i, \ell_i, \mathbf{a}_i)\}_{i=1}^N$ contains valid samples from successful demonstrations, including any locally imperfect actions within them.

Progress supervision uses a separately annotated corpus of stable successes, with and without retries:

$$\mathcal{D}_V = \mathcal{D}^+_{\text{clean}} \cup \mathcal{D}^+_{\text{retry}}. \tag{1}$$

A causal visual-history model $V_\phi(h_t, \ell) \in [0, 1]$ estimates task advancement, not discounted return, and may decrease during mistakes. It is frozen before analyzing failed rollouts $\mathcal{D}^-$, which never achieve stable completion within the execution horizon. These rollouts provide risk evidence but neither update the adapted progress model nor supply policy action targets.

For a training batch $\mathcal{B}$, RAISE changes only the loss weights:

$$\mathcal{L}_{\text{RAISE}}(\theta; \mathcal{B}) = \frac{\sum_{i \in \mathcal{B}} w_i \mathcal{L}_i^{\text{base}}(\theta)}{\sum_{i \in \mathcal{B}} w_i}, \qquad 1 \le w_i \le 2. \tag{2}$$

Here $\mathcal{L}_i^{\text{base}}$ is the learner's original per-sample objective. Weights are computed offline and fixed during training. Dataset membership, action targets, and sampling probabilities are preserved; unit weights recover the baseline. Figure 2 summarizes the pipeline.

3.2 RETRY-SENSITIVE PROGRESS LEARNING

Stage-aware representation. We initialize from ROBOMETER-4B (Liang et al., 2026) and adapt the progress model exclusively on the stable-success corpus $\mathcal{D}_V$, using low-rank updates to its language component while keeping pretrained weights frozen. From $z_t = f_\phi(h_t, \ell)$, two linear softmax

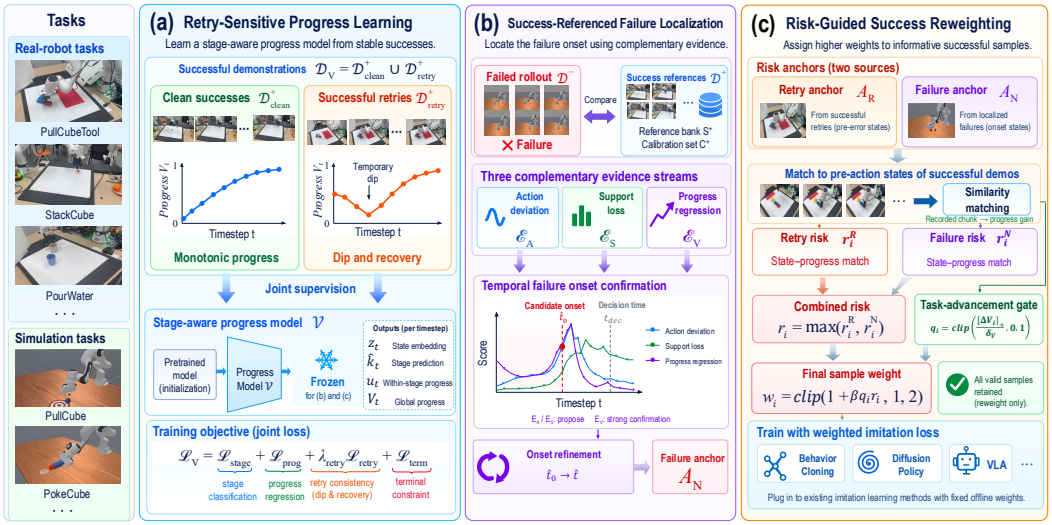

Figure 2: **RAISE overview.** (a) Stable successes train a stage-aware, retry-sensitive progress model. (b) The frozen model supports local successful-reference comparisons and temporal failure-onset confirmation. (c) Retry and failure anchors are matched independently to successful pre-action states; the larger risk score is gated by predicted advancement to produce offline weights. All valid imitation targets remain unchanged. Curves are schematic.

heads predict stage probabilities $p_{t,k}$ and a shared ten-bin within-stage distribution $q_{t,j}$. For K non-terminal stages and completion stage K,

$$u_t = \sum_{j=0}^{9} q_{t,j} \frac{j}{9}, \qquad V_t = p_{t,K} + \sum_{k=0}^{K-1} p_{t,k}(b_k + \alpha_k u_t). \tag{3}$$

The nonnegative stage proportions α_k sum to one and are estimated from clean training trajectories; $b_k = \sum_{m<k} \alpha_m$. Both are then fixed. The within-stage prediction u_t is shared, not separately conditioned on each stage.

Global calibration and local retries. Clean successes provide stage classification, segment-wise progress targets, and unit-progress completion anchors (Chen et al., 2026a). For a contiguous non-terminal segment $[s, e]$, $e > s$, with stage label k_t^*,

$$u_t^* = \frac{t-s}{e-s}, \qquad g_t^* = b_{k_t^*} + \alpha_{k_t^*} u_t^*, \qquad j_t^* = \mathrm{round}(9u_t^*). \tag{4}$$

Returning to a stage starts a new segment. Singleton segments and retry neighborhoods are excluded from absolute-progress supervision; the first stable-completion window anchors completion, and later frames are excluded from progress-model training. With ρ denoting unit-transition SmoothL1, the clean stream minimizes

$$\begin{aligned} \mathcal{L}_{\text{stage}} &= -\langle \log p_{t,k_t^*} \rangle_{\text{stage}}, \\ \mathcal{L}_{\text{prog}} &= \langle -\log q_{t,j_t^*} + \rho(V_t - g_t^*) \rangle_{\text{prog}}, \\ \mathcal{L}_{\text{term}} &= \langle \rho(V_t - 1) \rangle_{\text{term}}. \end{aligned} \tag{5}$$

Each mean uses its own validity mask. Stage/retry labels are constructed offline and are not model inputs; annotation and reduction details appear in Appendix A.3.

Successful retries provide pre-error, degraded, and recovered windows $W_r^{\text{pre}}, W_r^{\text{near}}, W_r^{\text{post}}$, clipped at trajectory and neighboring-event boundaries. With $\mu_r^v = |W_r^v|^{-1} \sum_{t \in W_r^v} V_\phi(h_{r,t}, \ell_r)$, we minimize

$$\mathcal{L}_{\text{retry}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \left[\mathrm{softplus}(\mu_r^{\text{near}} - \mu_r^{\text{pre}}) + \mathrm{softplus}(\mu_r^{\text{near}} - \mu_r^{\text{post}}) \right]. \tag{6}$$

Here $\mathcal{R}$ is a batch of retry events from distinct successful trajectories, and $\mathrm{softplus}(x) = \log(1 + \exp x)$. Equally weighted events encourage a local valley without ordering pre-error and recovered progress (Qin et al., 2026). Retry trajectories contribute only through this auxiliary stream, not the clean calibration losses. Joint training uses

$$\mathcal{L}_V = \mathcal{L}_{\mathrm{stage}} + \mathcal{L}_{\mathrm{prog}} + \lambda_{\mathrm{retry}}\mathcal{L}_{\mathrm{retry}} + \mathcal{L}_{\mathrm{term}}. \tag{7}$$

We use $\lambda_{\mathrm{retry}} = 2$. After adaptation, z_t, $\hat{k}_t = \arg\max_k p_{t,k}$, u_t, and V_t form the frozen interface for localization and reweighting. Training settings and retry-strength analyses appear in Appendices A.3 and B.

3.3 Success-Referenced Failure Localization

Local successful references. Outside regions supported by successful trajectories, progress estimates alone may be unreliable. We therefore compare action, support, and progress evidence against local successful references (Fig. 2(b)). The reference bank $\mathcal{S}^+$ and calibration set $\mathcal{C}^+$ are trajectory-disjoint, exclude evaluation rollouts, and retain natural retries. Using frozen representations $\bar{z}_t = z_t/\|z_t\|_2$, each query retrieves $K_{\mathrm{nn}} = 5$ neighbors with the same task and predicted stage from nearby within-stage progress bins. Sparse bins expand symmetrically within the stage. For this neighbor set $\mathcal{N}_t$, state-support distance and action deviation are

$$d_S(t) = \frac{1}{K_{\mathrm{nn}}} \sum_{j \in \mathcal{N}_t} \|\bar{z}_t - \bar{z}_j^+\|_2, \tag{8}$$

$$D_A(t) = \frac{1}{K_{\mathrm{nn}}} \sum_{j \in \mathcal{N}_t} \mathrm{RMS}(\mathbf{A}_t - \mathbf{A}_j^+). \tag{9}$$

Here superscript $+$ denotes successful references, and $\mathbf{A}_t = \mathrm{Std}_{\mathcal{C}^+}([a_{t-2}^{\mathrm{appl}}; a_{t-1}^{\mathrm{appl}}; a_t^{\mathrm{appl}}])$ concatenates three actually applied controls, standardized using fixed calibration statistics; $\mathrm{RMS}(v) = \|v\|_2/\sqrt{\dim(v)}$. Larger d_S indicates weaker support; larger D_A indicates unusual control among comparable states. This history differs from the policy prediction horizon H_a. Progress regression compares adjacent three-state windows:

$$g_t = \mathrm{median}(V_{t+1}, V_{t+2}, V_{t+3}) - \mathrm{median}(V_{t-2}, V_{t-1}, V_t). \tag{10}$$

Both windows must be available. Negative g_t indicates predicted regression; medians reduce sensitivity to isolated fluctuations.

Calibrated evidence and temporal confirmation. Queries from $\mathcal{C}^+$ against $\mathcal{S}^+$ yield stage- and progress-conditioned 99th-percentile thresholds $q_S(\hat{k}_t, u_t)$ and $q_A(\hat{k}_t, u_t)$ for the state and action scores, and a stage-conditioned 1st-percentile threshold $q_V(\hat{k}_t)$ for g_t. Sparse statistics use fixed stage-level and global nonterminal fallbacks. With $r_S(t) = d_S(t)/q_S(\hat{k}_t, u_t)$ and indicator $\mathbb{1}[\cdot]$,

$$\begin{aligned} E_A(t) &= \mathbb{1}[D_A(t) > q_A(\hat{k}_t, u_t)], \\ E_S(t) &= \mathbb{1}[r_S(t-1) \le 1]\mathbb{1}[r_S(t) > 1], \\ E_V(t) &= \mathbb{1}[g_t < q_V(\hat{k}_t)]. \end{aligned} \tag{11}$$

The indicators mark atypical control, departure from successful-state support, and unusually low progress change relative to successful calibration trajectories, respectively. E_V indicates predicted regression only when $g_t < 0$. Calibration excludes failure-test onset labels.

A candidate onset $\hat{t}_0$ begins at an E_A or E_S event and requires a different strong-evidence type within five steps, not repetition of its own type. E_V confirms but does not initiate candidates, because progress loss can lag the initiating action or state change. An E_A candidate may also use near-boundary support $0.95 \le r_S(s) < 1$ within its confirmation window when $D(s) = 1$, where

$$D(s) = \mathbb{1}\left[r_S(s+1) \ge 0.90 \ \vee \max_{\tau \in \{s+1,\ldots,s+5\}} r_S(\tau) > 1\right]. \tag{12}$$

This check rejects immediate recovery deep inside support unless a later boundary crossing corroborates the deviation. This weaker route cannot initiate candidates, and confirmation preserves $\hat{t}_0$.

Offline onset refinement. After confirmation, a five-step look-back using the successful action-calibration distribution's 95th percentile refines $\hat{t}_0$ to $\hat{t}$. Only the reported location changes, not detection, the evidence route, or the logged decision index $t_{\text{dec}}^{\log}$. Future observations enter progress and persistence checks, so this index is not a measured online alarm time. Confirmed failed rollouts supply candidate risk anchors, not causal ground truth; unconfirmed rollouts supply none. Appendix C details the detector and evaluation.

3.4 Risk-Guided Success Reweighting

Risk anchors and matching. Risk identifies where extra supervision is relevant; predicted advancement selects which successful segments receive it (Fig. 2(c)). The retry pool $\mathcal{A}_R$ stores the last valid pre-error state in W_r^{pre} for each successful retry; $\mathcal{A}_N$ stores confirmed, refined natural-failure onsets. Anchors store $(\bar{z}, \hat{k}, u, \ell)$: normalized representation, predicted stage, within-stage progress, and instruction, not action targets. For demonstration sample i, we encode its pre-action state with the same frozen model. Each source $\sigma \in \{R, N\}$ independently returns the nearest same-task, same-stage anchor by cosine distance:

$$j_\sigma(i) = \underset{\substack{j \in \mathcal{A}_\sigma \\ \ell_j = \ell_i,\ \hat{k}_j = \hat{k}_i}}{\arg\min}\ d_{ij}, \qquad d_{ij} = 1 - \bar{z}_i^\top \bar{z}_j. \tag{13}$$

This uses one anchor per source, not the detector's five neighbors. Let $d_i^\sigma = d_{i,j_\sigma(i)}$ and $\Delta u_i^\sigma = |u_i - u_{j_\sigma(i)}|$. The matched risk is

$$r_i^\sigma = \mathbb{1}[d_i^\sigma \le \delta_z,\ \Delta u_i^\sigma \le \delta_u] \left[1 - \frac{d_i^\sigma}{\delta_z}\right]_+ \left[1 - \frac{\Delta u_i^\sigma}{\delta_u}\right]_+ . \tag{14}$$

Here $[x]_+ = \max(x, 0)$; δ_z and δ_u bound representation distance and within-stage progress mismatch. Missing or rejected anchors give zero risk without another search; retrieval precedes progress filtering. Risk correspondence is not failure probability; neither failed-action similarity nor an extra global-progress constraint is used.

Advancement-gated weights. Risk proximity alone may emphasize local mistakes within successful trajectories. For sample i starting at t_i, let T_i be its trajectory's last recorded state index and $V_{i,t} = V_\phi(h_{i,t}, \ell_i)$. We compute

$$\begin{aligned} t_i^+ &= \min(t_i + H_a, T_i), \qquad \Delta V_i = V_{i,t_i^+} - V_{i,t_i}, \\ q_i &= \text{clip}\left(\frac{[\Delta V_i]_+}{\delta_V}, 0, 1\right). \end{aligned} \tag{15}$$

Here $\delta_V > 0$ controls saturation; clip restricts values to the specified interval. Endpoint clipping excludes artificial transitions from action padding. Advancement is a proxy, not an action-correctness label, and gates only additional emphasis. For fixed nonempty active sources $\mathcal{U} \subseteq \{R, N\}$ and weighting strength β, we define

$$r_i = \max_{\sigma \in \mathcal{U}} r_i^\sigma, \qquad w_i = \text{clip}(1 + \beta q_i r_i, 1, 2). \tag{16}$$

Single-source choices yield retry-only or failure-only weighting; both sources combine by maximum, not addition or pooled retrieval. All ten main simulation tasks use $\mathcal{U} = \{R, N\}$, $\delta_z = \delta_u = 0.15$, $\delta_V = 0.12$, and $\beta = 1$, fixed before policy training without task-specific adjustment. Appendix A.5 details loss integration; Appendix E.1 gives the StackCube data and weighting protocol.

The weights enter Eq. (2) while preserving all valid samples, action targets, and sampling probabilities, including samples from successful retries. Unmatched or non-advancing samples retain unit weight. Risk annotations and subsequent observations affect only offline weights, not policy inputs. Deployment adds no progress-model or risk-matching calls; risk collection and offline computation still incur costs.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate ten ManiSkill3 tasks (Tao et al., 2025) spanning picking, stacking, pushing/pulling, pose adjustment, tool use, insertion, and transport (Table 1). Each task supplies 300 successful demonstrations without an imposed retry ratio. We compare Behavior Cloning (BC), Diffusion Policy (DP) (Chi et al., 2025), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and GTP-FA-$\pi_{0.5}$ (Xu et al., 2026) with their RAISE variants. Within each pair, demonstrations, initialization, architecture, action targets, and optimization budget are fixed; GTP-FA's grasp front-end is also fixed. The 300-demonstration budget counts policy action-supervision trajectories only; RAISE additionally uses successful reference and calibration trajectories and retry/failure evidence for offline weight construction. Progress models are adapted per task, including PushCube, which retains GTP-FA's base task setting. Data roles and training configurations are given in Appendix A.

Each final model is evaluated on 200 fixed environment seeds with a 250-environment-step horizon, without stopping at transient success. We report equal-weight means over three training seeds: $\{0, 1, 2\}$ for BC/DP and $\{42, 43, 44\}$ for VLA systems. Baseline and RAISE share evaluation initializations within each pair. Entries report *success-once / Stable Success@3*: any successful state versus any three consecutive evaluation states satisfying the completion predicate. Unlike terminal-only `success_at_end`, Stable Success@3 rejects isolated successes without requiring completion at the rollout endpoint; it does not guarantee lasting stability. Full definitions appear in Appendix A.1.

Table 1: Cross-policy evaluation on ten ManiSkill3 tasks: success-once / Stable Success@3 (%). Each baseline is followed by its RAISE variant, trained on the same 300 demonstrations. Bold task values mark within-pair stable-success gains. Avg. and Δ average all ten tasks; Δ is in percentage points. All rows average three training seeds, with 200 episodes per task and seed.

Method	Stack	Push	Pull	Pick	Lift	Poke	Tool	Peg	Apple	Box	Avg.	Δ (pp)
BC	0/0	3.5/2.3	100/99.8	94.2/93.8	61.5/61.2	68.7/64.5	30.8/30.8	19.2/10.3	20.7/19.8	36.5/32.2	43.5/41.5	–
+ RAISE	15.3/**13.7**	23.2/**21**	100/99.8	96.2/**95.2**	73.2/**71.2**	70.3/**66.5**	45/**44.7**	33.2/**32.7**	36.8/**35.7**	48.5/**48.2**	54.2/52.9	**+10.7/+11.4**
DP	39.7/39.2	0/0	42/41.7	72/72	26.7/24.7	69.3/68.8	32.8/32.7	0.17/0.17	56.2/52.2	32.2/27.7	37.1/35.9	–
+ RAISE	50.2/**49.5**	12.2/**12**	52.3/**51.7**	79.2/**79**	41.8/**41.7**	76.2/**76**	41.5/**41.3**	13.5/**13.5**	65/**65**	42.3/**41.2**	47.4/47.1	**+10.3/+11.2**
$\pi_{0.5}$	96.2/96.2	93.3/92	94.5/94	1.2/1.2	13.5/11.3	44.2/44	38/38	1.3/1	56/53.5	34.2/32	47.2/46.3	–
+ RAISE	96.2/96	95.7/**94.5**	96.5/**96**	21.5/**21.2**	30/**29.5**	57/**56.2**	49.5/**49.2**	13.5/**13.2**	65/**64.5**	46.3/**44.5**	57.1/56.5	**+9.9/+10.2**
GTP-FA-$\pi_{0.5}$	97.5/97.2	95.7/94.5	95.5/94.3	26.5/26.2	69.7/69.2	71/70.3	54.3/53.5	16.5/16.2	69.2/64.3	51/46.3	64.7/63.2	–
+ RAISE	98.3/**98.2**	96.2/**95.3**	96.3/94.2	45.2/**45**	81/**80.3**	82.5/**81.2**	66.5/**65.2**	29.5/**29.2**	82.5/**82.2**	64.7/**62.5**	74.3/73.3	**+9.6/+10.1**

Stack, Push, Pull, Pick, and Poke denote the corresponding Cube tasks; Lift: LiftPegUpright; Tool: PullCubeTool; Peg: PegInsertionSide; Apple: PlaceApple; Box: TransportBox. Means and gains are rounded independently; DP on Peg is $1/600$ under both metrics (0.17%).

4.2 PERFORMANCE ACROSS POLICY FAMILIES AND TASKS

Table 1 shows that RAISE improves ten-task mean Stable Success@3 from 41.5% to 52.9% for BC (+11.4 pp), 35.9% to 47.1% for DP (+11.2 pp), 46.3% to 56.5% for $\pi_{0.5}$ (+10.2 pp), and 63.2% to 73.3% for GTP-FA-$\pi_{0.5}$ (+10.1 pp). The four-pair mean improvement is **10.7 percentage points in stable success** and 10.1 points in success-once, computed before rounding. Stable success increases in 37 of 40 task–base-system settings, is unchanged in one, and decreases in two. Gains remain task-dependent: GTP-FA-$\pi_{0.5}$ improves from 26.2% to 45.0% on PickCube and 64.3% to 82.2% on PlaceApple, but decreases slightly from 94.3% to 94.2% on PullCube. The standard $\pi_{0.5}$ pair also decreases slightly on StackCube, from 96.2% to 96.0%.

These comparisons measure reweighting on fixed base systems, not the contribution of GTP-FA itself. Mean gains alone do not establish statistical significance on every task. The main results average three training seeds; the earlier seed-42 Expert300 study and its role in V3 selection are reported separately in Appendix E.3.

Table 2: Published VLA adaptation systems and our GTP-FA combinations. Published rows retain source-defined success rates; our rows report Stable Success@3 (%). Online policy RL is separated explicitly. Different data budgets and evaluation protocols make these system-level comparisons, not controlled estimates of RAISE's effect.

Panel A: Shared tasks from StARe-VLA (Xu et al., 2025)

Method	Data/task	Online RL	Stack	Push	Pull	Lift	Avg.
Other backbones: published reference results							
Octo (FT) (Team et al., 2024)	1000 D	No	1.0	67.0	90.0	0.0	39.5
SmolVLA (FT) (Shukor et al., 2025)	1000 D	No	12.7	86.3	90.7	16.3	51.5
RoboFAC-7B (Ye et al., 2025)	N/S	N/S	85.5	80.4	80.7	84.0	82.7
OpenVLA-7B: without online policy RL							
SFT	100 D	No	12.0	11.7	31.0	5.3	15.0
GRAPE (Zhang et al., 2024)	100 D + 50 P	No	15.7	13.3	35.3	7.7	18.0
StA-TPO	100 D + 50 P	No	19.3	16.0	35.7	12.3	20.8
RAISE (GTP-FA)*†	300 D	**No**	87.5	89.2	90.2	71.3	**84.5**
OpenVLA-7B: with online policy RL							
RL4VLA (Liu et al., 2026)	100 D	Yes	64.0	95.7	90.3	32.0	70.5
StA-PPO	100 D	Yes	92.7	96.0	95.3	89.7	93.4
IPI	100 D + 50 P	Yes	94.3	97.3	98.5	95.5	96.4
$\pi_{0.5}$**: without online policy RL**							
SFT	100 D	No	26.3	18.3	43.0	10.7	24.6‡
GRAPE (Zhang et al., 2024)	100 D + 50 P	No	22.7	16.3	45.0	6.3	22.6
StA-TPO	100 D + 50 P	No	28.0	22.3	44.7	15.7	27.7
GTP-FA*	300 D	No	97.2	94.5	94.3	69.2	88.8
RAISE (GTP-FA)*†	300 D	**No**	98.2	95.3	94.2	80.3	**92.0**
$\pi_{0.5}$*: with online policy RL*							
π_{RL} (Chen et al., 2025b)	100 D	Yes	72.3	96.7	93.3	58.0	80.1
StA-PPO	100 D	Yes	80.7	98.0	92.7	75.3	86.7
IPI	100 D + 50 P	Yes	84.3	99.3	95.0	80.7	89.9

D: successful demonstrations; P: preference pairs; N/S: not specified. *GTP-FA uses the backbone named in its group. Our results average training seeds $\{42, 43, 44\}$, with 200 episodes/task/seed and 250 environment steps/episode; StARe retains its own success criterion and 300-episode/30-step protocol. †RAISE also uses auxiliary retry/failure evidence, not additional action targets. ‡The $\pi_{0.5}$ SFT average is recomputed from the four published task rates; the source reports 25.6%. Other published averages are retained as reported.

Panel B: Failure-informed systems

Method	Evaluation	Stack	Pull	Lift	Avg.
DAG-VLA + AFIL (Zheng et al., 2026)	Published; 50×3 runs	84.7	98.0	90.7	91.1
GTP-FA-$\pi_{0.5}$	Stable@3; 3×200 episodes	97.2	94.3	69.2	86.9
RAISE (GTP-FA-$\pi_{0.5}$)	Stable@3; 3×200 episodes	98.2	94.2	80.3	90.9

AFIL uses 1500 successful, 1000 failure-corrected, and 1500 raw-failure trajectories. Our 300-demonstration $\pi_{0.5}$ results reuse Table 1; no matched-protocol re-evaluation or metric conversion is implied.

Progress, localization, and risk-source analyses. The paired input study changes camera, resolution, and history jointly: render-camera 512×512/H8 increases retry-preference satisfaction from 45.5% to 69.5% over base-camera 128×128/H16, with clean-progress MAE changing from 0.05288 to 0.05449 (Appendix B.4). On Dev2, comprising 1000 controlled failures and 1000 clean successes, the same change raises detection recall from 60.2% to 69.0% and all-failure localization accuracy within ten steps from 40.8% to 49.4%, while adding eight false positives. Dev2 informed development; these are not independent final-test results (Appendix C).

In seven matched policy–task settings, full RAISE raises mean Stable Success@3 from 37.2% to 49.4% relative to uniform training. Removing retry supervision, using progress-only localization, or removing advancement gating reduces the full-method mean by 9.1, 14.6, and 3.5 points, respectively (Appendix E.5). The separate seed-42 StackCube study shows composition-dependent gains but no gain on original Expert300 (Appendices E.3 and E.4).

4.3 Comparison with Published VLA Adaptation Systems

Table 2 positions RAISE on GTP-FA-based systems, separately from the matched comparisons above. Without online policy RL, the OpenVLA and $\pi_{0.5}$ combinations achieve 84.5% and 92.0% mean Stable Success@3, respectively; published StA-TPO means are 20.8% and 27.7% (Xu et al., 2025). IPI includes online policy RL and reports 96.4% for OpenVLA. Different demonstration budgets, grasp front-ends, metrics, and horizons prevent attributing these cross-study gaps to RAISE alone. Avoiding online policy RL also does not eliminate auxiliary rollout or offline computation costs.

Within our GTP-FA-$\pi_{0.5}$ pair, RAISE increases the four-task mean from 88.8% to 92.0% (+3.2 pp). No unweighted GTP-FA-OpenVLA baseline is evaluated, so this comparison does not isolate RAISE's contribution on that backbone. On AFIL's shared tasks, our system reports 90.9% versus its published 91.1%; AFIL uses distinct trial construction and a failure-informed action-generation mechanism (Zheng et al., 2026). This is system-level context, not a matched-protocol comparison. Additional RDT/PhysGen references appear in Appendix F.

4.4 Real-Robot Evaluation

Five Franka Research 3 tasks use 300 demonstrations and 50 physical trials per method and task: Orange-to-Tray, StackCube, PokeCube, PullCubeTool, and PourWater (cup-to-cup transfer of small balls). We retain GTP-FA's task-level *end-of-execution* completion criteria, not simulation Stable Success@3. The two baseline rows in Table 3 are reused from Xu et al. (2026); RAISE variants are evaluated under the same protocol. The grasp front-end is fixed within the GTP-FA pair. We separately adapt the progress model on successful real-robot trajectories using base-camera RGB; platform and annotation details appear in Appendices A.6 and D.

RAISE increases mean task success from 11.2% to 26.4% on the standard $\pi_{0.5}$ policy (+15.2 pp), and from 76.8% to 84.0% on GTP-FA-$\pi_{0.5}$ (+7.2 pp). Both comparisons improve on all five tasks. The GTP-FA system gains eight points on StackCube and PullCubeTool and fourteen on PourWater. These are task-level comparisons with reused baseline measurements, not paired-trial estimates; absolute performance remains limited for the standard VLA policy.

Discussion. These results support two complementary observations: RAISE improves a standard VLA policy without changing its architecture or adding failed-rollout actions as imitation targets, and remains beneficial with a stronger, fixed grasp front-end. Reweighting thus acts at the data-supervision level and can complement improvements elsewhere in the manipulation pipeline.

Table 3: Real-robot task success (%): 50 trials per method/task. Baseline measurements are reused from Xu et al. (2026); RAISE rows are evaluated here under the same task-specific demonstrations and protocol. Δ is the macro-average gain in percentage points.

Method	Orange	Stack	Poke	PullTool	Pour	Avg.	Δ (pp)
$\pi_{0.5}$	10.0	4.0	24.0	16.0	2.0	11.2	–
RAISE ($\pi_{0.5}$)	28.0	20.0	36.0	32.0	16.0	26.4	**+15.2**
GTP-FA-$\pi_{0.5}$	92.0	74.0	86.0	78.0	54.0	76.8	–
RAISE (GTP-FA-$\pi_{0.5}$)	**94.0**	**82.0**	**90.0**	**86.0**	**68.0**	**84.0**	**+7.2**

5 Conclusion

We introduced RAISE, a data-centric framework for learning from failures without treating failed actions as imitation targets. RAISE first learns a retry-sensitive progress reference from stable successful executions, then localizes failures through success-referenced temporal evidence, and finally transfers retry- and failure-derived risk back to the original successful demonstrations through bounded, task-advancement-aware reweighting. This design preserves the policy architecture, action targets, and valid training samples, while changing only where learning places greater emphasis. Future work will explore using RAISE's risk-guided weights to focus reinforcement-learning fine-tuning on failure-prone contexts.

AI USE STATEMENT

AI tools assisted with language editing, manuscript organization, LaTeX preparation, and local scripts for analysis and figure assembly. The authors are responsible for the scientific content, reported measurements, and interpretation of the results.

ETHICS STATEMENT

This work studies tabletop robotic manipulation and the use of execution failures to improve imitation learning. Controlled failure interventions are used for simulation analysis, while recorded physical executions support real-robot inspection. Improved task success should not be interpreted as a safety certification: partial observability, hardware faults, and unmodeled conditions remain relevant deployment limitations. Failure-derived information changes offline learning weights rather than introducing failed actions as imitation targets.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024.

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *arXiv preprint arXiv:2405.19988*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Annie S Chen, Alec M Lessing, Yuejiang Liu, and Chelsea Finn. Curating demonstrations using online experience. *arXiv preprint arXiv:2503.03707*, 2025a.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online RL Fine-Tuning for Flow-Based Vision-Language-Action Models. *arXiv preprint arXiv:2510.25889*, 2025b.

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

Qianzhong Chen, Hau Zheng, Justin Yu, Suning Huang, Jiankai Sun, Ken Goldberg, Chuan Wen, Pieter Abbeel, Yide Shentu, Philipp Wu, et al. Sarm2: Multi-task stage aware reward modeling for self improving robotic manipulation. *arXiv preprint arXiv:2606.10305*, 2026b.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025c.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations*, volume 2026, pp. 1311–1336, 2026.

Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1801–1810. IEEE, 2019.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. Srpo: Self-referential policy optimization for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6718–6728, 2026.

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

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

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

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

Sachit Kuhar, Shuo Cheng, Shivang Chopra, Matthew Bronars, and Danfei Xu. Learning to discern: Imitating heterogeneous human demonstrations with preference and representation learning. In *Conference on Robot Learning*, pp. 1437–1449. PMLR, 2023.

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.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 54877–54910, 2025.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *The International Journal of Robotics Research*, 44(10-11):1727–1742, 2025.

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. *Advances in Neural Information Processing Systems*, 38:97121–97151, 2026.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Weijia Liufu, Xiaoyu Guo, Ruiyi Chen, Jingzhi Liu, Kaidong Zhang, Xiwen Liang, Jianqi Lin, Dawei Sun, Yuze Wang, Rongtao Xu, et al. Repo-vla: Recovery-driven policy optimization for vision-language-action models. *arXiv preprint arXiv:2605.09410*, 2026.

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. *arXiv preprint arXiv:2210.00030*, 2022.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pp. 23301–23320. PMLR, 2023.

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

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. *arXiv preprint arXiv:2108.03298*, 2021.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

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

Xinyao Qin, Junjie Lu, Kaixin Wang, Chuheng Zhang, Sinjae Kang, Kimin Lee, Min Xu, Bin Liang, Jun Yang, and Li Zhao. Beyond monotonic progress: Retry-supervised value learning for robot imitation. *arXiv preprint arXiv:2606.24633*, 2026.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Som Sagar, Jiafei Duan, Sreevishakh Vasudevan, Yifan Zhou, Heni Ben Amor, Dieter Fox, and Ransalu Senanayake. Robomd: Uncovering robot vulnerabilities through semantic potential fields. In *International Conference on Learning Representations*, volume 2026, pp. 110599–110624, 2026.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Zijian Song, Qichang Li, Sihan Qin, Yuhao Chen, Tianshui Chen, Liang Lin, and Guangrun Wang. Learning physics from pretrained video models: A multimodal continuous and sequential world interaction models for robotic manipulation. In *Proceedings of the 2026 International Conference on Multimedia Retrieval*, pp. 758–767, 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. *Advances in Neural Information Processing Systems*, 36:55681–55693, 2023.

S Tao, F Xiang, A Shukla, Y Qin, X Hinrichsen, X Yuan, C Bao, X Lin, Y Liu, TK Chan, et al. Demonstrating gpu parallelized robot simulation and rendering for generalizable embodied ai with maniskill3. *Proceedings of Robotics: Science and Systems,(LosAngeles, CA, USA)*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

Feng Xu, Guangyao Zhai, Xin Kong, Tingzhong Fu, Daniel FN Gordon, Xueli An, and Benjamin Busam. Progressive stage-aware reinforcement for fine-tuning vision-language-action models. *arXiv preprint arXiv:2512.05107*, 2025. URL https://arxiv.org/abs/2512.05107.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, pp. 24725–24742. PMLR, 2022.

Jiahao Xu, Peiyuan Wang, Hanzhuo Zhang, Zihao Yu, Tianyu Fu, Hao Chen, Xuanhao Xiang, Jianbo Yu, Chenchen Fu, and Wanyuan Wang. Grasp-then-plan with failure attribution: A closed two-stage framework for precise and generalizable robotic manipulation. *arXiv preprint arXiv:2606.03385*, 2026.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robo-fac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*, 2025.

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. *arXiv preprint arXiv:2505.10911*, 2025a.

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

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

Meng Zheng, Samhita Marri, Anwesa Choudhuri, Benjamin Planche, Zhongpai Gao, Van Nguyen Nguyen, Terrence Chen, Girish Chowdhary, and Ziyan Wu. Failing forward: Adaptive failure-informed learning for vision-language-action models. *arXiv preprint arXiv:2605.08434*, 2026.

APPENDIX

A EXTENDED EXPERIMENTAL SETUP AND REPRODUCIBILITY

This appendix specifies experimental protocols and implementation details. Baseline–RAISE pairs retain the same architecture, interfaces, demonstrations, initialization, optimization budget, and evaluation distribution.

A.1 SIMULATION TASKS AND EVALUATION PROTOCOL

Tasks and demonstrations. The ten tasks in Table 1 retain their native scenes, robots, action interfaces, and success conditions. Each main experiment uses a fixed *Expert300* set of 300 successful trajectories, with identical paired membership and no imposed clean-to-retry ratio. Successful trajectories containing retries remain included.

Success criteria. Let $s_t \in \{0, 1\}$ be the task-success flag and $c_t \in \{0, 1\}$ the completion predicate for stability, with $c_t \leq s_t$. They coincide when no additional completion conditions apply. For a rollout of length T,

$$S_{\text{once}} = \max_{1 \leq t \leq T} s_t, \qquad S_{\text{end}} = s_T, \qquad S_{\text{stable@3}} = \max_{3 \leq t \leq T} (c_{t-2} c_{t-1} c_t),$$

where $S_{\text{stable@3}} = 0$ for $T < 3$. The stable window can occur anywhere, not only at the endpoint, and denotes consecutive steps rather than trials. Stable Success@3 implies success-once but neither implies nor is implied by success-at-end: a stable window may be followed by failure, and final success may lack a preceding stable window. It rejects isolated successes, not all later instability, and is not a calibrated physical duration. For StackCube, c_t requires stacking and release, linear speed $\leq 0.01\,\text{m/s}$, and angular speed $\leq 0.5\,\text{rad/s}$; other tasks retain their own predicates.

Evaluation and aggregation. Final models in Tables 1 and 2 use 200 fixed environment seeds and at most 250 steps, without stopping at transient success. Training seeds are $\{0, 1, 2\}$ for BC/DP and $\{42, 43, 44\}$ for the main VLA results. Within each task, method, and seed, baseline and RAISE start identically and train independently. Paired variants and training seeds share task-specific evaluation lists; BC/DP also share theirs. Main VLA evaluations use `400005010--400005209`, inclusive. For success counts k_s, $\bar{p} = \frac{1}{3}\sum_s 100k_s/200 = 100\sum_s k_s/600$. Thus, 600 rollouts are three models on 200 common initializations, not 600 distinct environment seeds. Means and gains precede rounding. Equal numeric seeds across tasks do not imply equal scenes; BC/DP and VLA generally use different lists. The earlier StackCube policy studies in Appendices E.3 and E.4 retain their seed-42 protocol.

Simulation backend and control. Simulations use the PhysX CUDA backend. VLA runs on A100/RTX 5090 use ManiSkill3 3.0.0b18 and SAPIEN 3.0.0b1; BC/DP runs use ManiSkill3 3.0.0b21 and SAPIEN 3.0.2. Native control and physics frequencies are 20 and 100 Hz, respectively. Policies retain task-specific joint-position or end-effector-pose increment interfaces, fixed within each baseline–RAISE pair. BC updates its action every native control step, DP replans after eight actions, and $\pi_{0.5}$ after ten. The PegInsertionSide VLA wrapper repeats each action twice. Genuine task failures remain included; technical failures are logged separately rather than relabeled as task failures.

A.2 DATA ROLES AND ISOLATION

Table 4 separates data roles. Training, development, reference, calibration, and final evaluation are trajectory-separated; development data have zero policy-training mass. The reference bank and calibration set are mutually disjoint and exclude evaluation trajectories. Natural failures are technically valid rollouts with no Stable Success@3, not infrastructure exceptions. Thresholds, weights, and checkpoints remain fixed during each comparison; V3-selection chronology is distinguished in Appendix E.3.

Auxiliary experience. The 300-demonstration budget counts policy action supervision only. Within each task, policy families and training seeds share frozen auxiliary trajectory pools; model-dependent quantities are recomputed when required by an ablation. For StackCube, progress adaptation uses 1000 clean-success trajectories and 1000 retry events from 729 successful trajectories. The success reference bank and calibration set contain 1000 trajectories each. Of 489 natural failures, 352 yield confirmed risk anchors (Appendix E.1). These auxiliary sources do not add policy action targets.

Table 4: Data roles and direct gradient use. Auxiliary failure evidence determines weights but supplies no policy action targets.

Data role	Source	Gradient use	Purpose
Expert300	Stable successful demonstrations	Policy	Original action supervision
Ordinary success	Clean stable-success trajectories	Progress model	Stage and absolute-progress supervision
Retry success	Stable successes containing local retries	Progress model	Local degradation–recovery supervision
Retry-risk anchors	Pre-error states from successful retries	None	Retry-derived risk estimation
Natural failures	Failed rollouts from a frozen base policy	None	Failure localization and risk estimation
Success Bank	Independent successful trajectories	None	Local successful-state reference
Calibration set	Independent successful trajectories	None	Evidence-threshold calibration
Development sets	Disjoint clean/retry/controlled rollouts	None	Method development and ablation
Final evaluation	Frozen seeds or physical trials	None	Final performance reporting

A.3 Retry-Sensitive Progress Model

Architecture and inputs. Table 5 specifies simulation models. Language-backbone LoRA and two heads are trained; ROBOMETER-4B's pretrained parameters, visual encoder, and original heads remain frozen. Stage-slot semantics and active labels are task-specific; the within-stage head is shared. Main inputs are native render-camera RGB at 512×512/H8. Initial padding repeats the first real observation, never future frames. Physical-state labels and labeling potentials provide offline supervision, not model or policy inputs.

Eligibility and supervision boundaries. Only stable-success trajectories adapt the model. Physical predicates supply simulation labels, checked by human review (Appendix D). Equation (4) normalizes each valid contiguous nonterminal segment independently; re-entry starts a new segment. Singletons and retry neighborhoods have no absolute-progress targets. The first stable window anchors unit progress; later frames are excluded from model training, not from fixed policy-demonstration rows. Stage, progress, and terminal masks remain distinct. StackCube terminal success without a stable window gives no training endpoint, although observations and diagnostics are retained.

Retry windows and neighboring events. Annotations determine W_r^{pre}, W_r^{near}, and W_r^{post}; predicted minima do not select them. Windows are clipped at trajectory endpoints, the effective supervision boundary, and neighboring attempts. Under the StackCube labeler, a one-step predicate interruption is transient; an interruption of at least two policy steps can initiate a reliable retry. A connected transient can supply context but cannot independently establish an event. Empty windows or invalid ordering exclude an attempt from $\mathcal{L}_{\mathrm{retry}}$ without relocating its keypoint; the reliable degradation interval remains masked from absolute-progress targets. Clipping resolves conflicts involving near-error windows, but W_r^{post} may overlap W_{r+1}^{pre} because these are positive contexts for different setbacks. Means use actual members without supervision padding. A one-member retry window is valid, unlike a singleton segment whose temporal normalization needs a nonzero interval. Figure 3 illustrates this distinction. These event-boundary rules are specific to StackCube; other

Table 5: Simulation progress-model configuration. Loss coefficients follow the stage/progress/retry/terminal order in Eq. (7).

Component	Configuration
Backbone	ROBOMETER-4B
Visual input	render RGB, 512×512, H8
Trainable backbone	language LoRA
LoRA rank / alpha / dropout	64 / 128 / 0.05
LoRA projections	q/k/v/o/gate/up/down
Stage head	$2560 \rightarrow 4$
Within-stage head	$2560 \rightarrow 10$
Main-stream batch	16
Retry auxiliary batch	8 events from 8 trajectories
Optimizer	AdamW
Learning rate	2×10^{-5}
Betas / ϵ	$(0.9, 0.999)$ / 10^{-8}
Weight decay	0.01 for matrix parameters
Gradient clipping	global norm 10
Warmup / scheduler	100 updates / cosine
Training budget	1000 optimizer updates
Loss coefficients	$1/1/2/1$
Precision	BF16 mixed precision
Model selection	fixed final update 1000

tasks use their own annotations. The labeler targets degradation–recovery and has no dedicated stall-only detector.

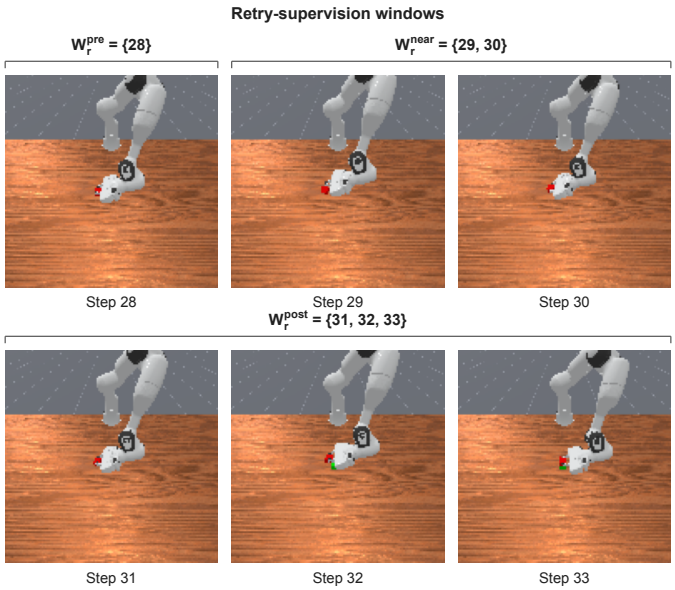

Figure 3: A successful StackCube retry: $W_r^{\text{pre}} = \{28\}$, $W_r^{\text{near}} = \{29, 30\}$, and $W_r^{\text{post}} = \{31, 32, 33\}$. A preceding retry clips the pre-error window from $\{26, 27, 28\}$ to $\{28\}$. Original policy indices mark supervision membership, not input-history length or predicted progress; window sizes are event-specific.

Training streams. The clean-success stream supplies stage, absolute-progress, and terminal supervision. An independent auxiliary stream samples eight retry events from eight distinct successful trajectories and weights events equally. Retry trajectories contribute through this stream, not the clean calibration losses. Both streams optimize Eq. (7). After update 1000, all parameters are frozen

for localization and reweighting; final-failure trajectories never update the model. Appendix B evaluates the resulting model's global progress accuracy and local retry sensitivity. Supervision boundaries and retry-window handling are specified in the preceding paragraphs.

A.4 Base Policies and Learner-Specific Training

Table 6 specifies native objectives and training settings. Paired runs share initialization, data, normalization, seed, and budget. The $\pi_{0.5}$ pair starts independently from `pi05_base`, predicts ten-action chunks, and uses fixed final checkpoints. GTP-FA's grasp front-end remains fixed; its preceding optimization is not rerun or attributed to RAISE.

Table 6: Learner-specific configurations, fixed within each baseline–RAISE pair. OpenVLA is used in Table 2, not Table 1.

Policy	Training data	Optimization	Policy-specific configuration
BC	300/task	Seeds $\{0, 1, 2\}$; batch 1024; LR 3×10^{-4}; 10k iterations	State normalization enabled
Diffusion Policy	300/task	Seeds $\{0, 1, 2\}$; batch 1024; LR 1×10^{-4}; 100k iterations	$T_{\text{obs}} = 2$, $T_{\text{act}} = 8$, $T_{\text{pred}} = 16$; diffusion embedding 64; U-Net $[64, 128, 256]$; 8 groups
$\pi_{0.5}$	300/task	Seeds $\{42, 43, 44\}$; global batch 32; 30k optimizer updates	10-action chunks; fixed final checkpoint
OpenVLA-7B	300/task	Seeds $\{42, 43, 44\}$; AdamW; constant LR 5×10^{-4}; maximum 200k updates/run; batch 16/GPU; accumulation 1	LoRA rank 32, $\alpha = 16$, dropout 0; 224×224 images; deterministic decoding; used only in Table 2

OpenVLA-specific preparation. OpenVLA-7B (Kim et al., 2024) is initialized from pretrained weights and fine-tuned on 300 MPLib trajectories per task. Common preprocessing removes idle actions with negligible end-effector pose change before fixing the training set; RAISE removes no additional samples. We follow the official LoRA fine-tuning configuration in `vla-scripts/finetune.py`, using AdamW with a constant learning rate of 5×10^{-4} and up to 200k optimizer updates per run. Batch and adapter settings are listed in Table 6. Training uses `all-linear` LoRA, BF16 without four-bit quantization, and augmented 224×224 images. Evaluation uses each run's final saved checkpoint and deterministic decoding under Appendix A.1, rather than StARe's protocol. The $\pi_{0.5}$ systems retain their separate 30k-update budget.

A.5 RAISE Integration

Fixed weights from Eq. (16) enter the learner's per-sample objective through Eq. (2). They receive no gradients, and no risk-proportional resampling is introduced. Unit weights recover the original loss and gradients. Unmatched or non-advancing samples remain in training, including samples from successful retries. In the $\pi_{0.5}$ implementation, the 10×32 flow-matching residual is first averaged within each row; the weighted numerator and denominator are then reduced over the full logical batch. Ratios computed separately on devices or microbatches are not averaged. A unit-weight sample remains supervised, although its relative contribution changes when other samples receive larger weights.

Progress inference, reference construction, failure localization, and risk matching are offline preparation steps. Risk annotations and later observations affect only weights, not policy inputs. Deployment adds no progress-model or risk-matching call and preserves the existing grasp front-end where applicable. Auxiliary rollout collection and offline scoring nevertheless incur data and computation costs.

A.6 Real-Robot Platform and Evaluation

A Franka Research 3 uses a Robotiq parallel-jaw gripper and base/wrist Intel RealSense D435i cameras; computation is listed in Table 7. The five tasks in Table 3 use 300 demonstrations and 50

physical trials per method/task. PourWater transfers small balls between cups. End-of-execution completion follows Xu et al. (2026), not simulation Stable Success@3. Baseline measurements are reused; RAISE follows the same task-specific demonstration and evaluation protocol.

Recorded RGB-D, robot/gripper states, grasp poses, and execution histories support perception and retrospective review. Policies retain native inputs, without simulator privileged state, tactile/contact ground truth, future observations, human annotation, or outcome-conditioned action generation. The progress model is separately adapted on successful physical trajectories using base-camera RGB (Sec. 4.4). Reweighting preserves cameras, controller, demonstrations, and evaluation conditions, including the fixed GTP-FA front-end. Annotation semantics appear in Appendix D.

Execution timing. End-effector pose targets are updated at 10 Hz. Policy inference is synchronous, with a new prediction after each block of ten control commands. These command-level rates are distinct from the camera and annotation time bases.

A.7 Reproducibility and Reporting Conventions

Records retain data/checkpoint identities, learner settings, seeds or physical trials, and results. Main three-seed evaluations remain separate from the seed-42 study informing V3 selection. Out-of-memory events, unavailable devices, missing dependencies, simulator crashes, corrupted artifacts, and communication failures are infrastructure exceptions, not genuine task failures.

A.8 Computational Resources

Table 7 lists host configurations, not per-run allocations. Main progress models use LoRA (Table 5); full-parameter ROBOMETER training belongs to a separate development setting.

Table 7: Computational platforms. GPU counts describe hosts rather than per-run allocations.

Primary use	CPU	GPUs
ROBOMETER development and adaptation; VLA training and evaluation	Intel Xeon Gold 6330	8× NVIDIA A100
Task-specific ROBOMETER adaptation; VLA training and evaluation	Intel Xeon Silver 4310	2× RTX 5090
StackCube VLA ablations	AMD Ryzen 9 9950X	2× RTX 4090
Behavior Cloning and Diffusion Policy training and evaluation	Intel Xeon Silver 4310	5× RTX 4090
Real-robot perception and VLA deployment	Intel Xeon Gold 6226R	1× RTX 4090

B Additional Analysis of Retry-Sensitive Progress Learning

We examine global calibration and local retry sensitivity through supervision-strength, data-coverage, directional-loss, and input comparisons. Historical development studies are distinguished from the paired A/B models.

B.1 Evaluation Protocol and Experimental Configurations

Models within each comparison share evaluation trajectories, annotations, and validity masks. Clean successes assess global progress; successful retries assess local degradation–recovery. Targets follow Sec. 3.2 and Appendix A.3; set sizes and experiment-specific settings accompany each table.

Global accuracy. For progress-valid indices $\mathcal{I}_{\text{prog}}$ with $N_{\text{prog}} = |\mathcal{I}_{\text{prog}}|$, define

$$\text{MAE} = \frac{1}{N_{\text{prog}}} \sum_{i \in \mathcal{I}_{\text{prog}}} |V_i - g_i^*|, \qquad \text{Acc@}\delta = \frac{1}{N_{\text{prog}}} \sum_{i \in \mathcal{I}_{\text{prog}}} \mathbf{1}[|V_i - g_i^*| \le \delta], \tag{17}$$

where g_i^* is the fixed target. Stage macro-F1 and terminal-anchor error assess stage classification and deviation from unit progress at completion, using their respective masks.

Retry sensitivity. Using the annotated window means from Sec. 3.2, event satisfaction is

$$s_r = \mathbf{1}[\mu_r^{\text{pre}} > \mu_r^{\text{near}}]\mathbf{1}[\mu_r^{\text{post}} > \mu_r^{\text{near}}]. \tag{18}$$

Both inequalities are strict, with no additional margin. P_{evt} averages events; P_{traj} first averages within trajectories and then weights trajectories equally. We also report $\Delta_{\text{pre}} = \text{mean}_r(\mu_r^{\text{pre}} - \mu_r^{\text{near}})$, $\Delta_{\text{post}} = \text{mean}_r(\mu_r^{\text{post}} - \mu_r^{\text{near}})$, and recovery consistency, the fraction satisfying $\mu_r^{\text{post}} \geq \mu_r^{\text{pre}}$. The latter is diagnostic, not a training constraint. These metrics assess predictions, not policy task success.

B.2 Effect of Retry Supervision Strength

The earlier StackCube base-camera 128×128/H16 study varies $\lambda_{\text{retry}} \in \{0, 1, 2, 3, 4, 5\}$, holding data, sampling, training seed, masks, and the 1000-update budget fixed. Training uses 1000 clean trajectories and 1000 retry events from 750 successful trajectories. Setting $\lambda_{\text{retry}} = 0$ disables only the explicit retry loss. Table 8 averages metrics equally across three separately collected holdouts, not training seeds.

Table 8: Retry-loss strength on three StackCube holdouts with base-camera 128×128/H16 inputs. Fixed update-1000 results are averaged equally across holdouts. Accuracy, satisfaction, and recovery are percentages; $\Delta_{\text{pre/post}}$ use the normalized progress scale.

λ_{retry}	MAE ↓	Acc@0.10 ↑	P_{evt} ↑	P_{traj} ↑	Δ_{pre}	Δ_{post}	Recovery
0	0.05084	87.85	9.20	8.49	-0.0637	0.0108	69.67
1	0.05249	87.68	39.96	38.88	0.0896	0.2012	75.08
2	0.05690	86.62	45.47	43.80	0.1386	0.2664	77.33
3	0.06375	84.64	45.65	44.34	0.1556	0.2968	77.84
4	0.06924	83.11	51.08	50.27	0.1771	0.3206	75.50
5	0.07488	81.49	48.91	47.71	0.1740	0.3094	75.03

Evaluation sets. The three holdouts contain 194/197/199 clean trajectories and 317/336/357 retry trajectories, respectively. Their progress-valid sample counts are 3956/4004/4053, and their retry-event counts are 425/457/488.

Without retry supervision, MAE is lowest but satisfaction is only 9.20%, with negative Δ_{pre}. At $\lambda_{\text{retry}} = 2$, satisfaction improves by 36.27 points while Acc@0.10 falls by 1.23. Larger coefficients either add little satisfaction (3), trade accuracy and recovery for higher satisfaction (4), or worsen both objectives (5). The main render-camera/H8 configuration inherits 2 as a compromise; this sweep evaluates the earlier base-camera/H16 inputs only.

B.3 Data Coverage and Optimization Controls

Coverage. E1–E7 vary training data, not architecture (Table 9). E1–E4 fix 300 clean trajectories and increase retry coverage; E2/E5/E6 fix the 299-event retry pool and increase clean coverage. E7 combines 1000 clean trajectories and 1000 retry events. All runs use base-camera 128×128/H16 inputs, $\lambda_{\text{retry}} = 1$, equal directional coefficients, seed 20261221, and 1000 updates. Final-checkpoint metrics and development-window near-best summaries remain separate.

Retry coverage raises satisfaction from 24.5% to 40.5% (E1–E4), whereas extra clean data with fixed retries change it little (E2/E5/E6). Clean coverage instead improves stage macro-F1 from 0.8520 to 0.8657 (E4/E7). These roles are complementary; covarying retry-event and source-trajectory counts prevent isolating their individual effects.

Directional-loss controls. With E7 membership and $\lambda_{\text{retry}} = 2$ fixed, coefficients c_{pre} and c_{post} multiply the two terms of Eq. (6). Balanced $(1, 1)$ and pre-emphasized $(4/3, 2/3)$ settings preserve the coefficient sum and sampling and use the same two training seeds. Table 10 shows that pre-emphasis increases Δ_{pre} but reduces Δ_{post}, recovery consistency, and joint preference satisfaction for both seeds. A sharper predicted decline does not by itself improve the complete retry pattern.

Table 9: StackCube training-data coverage. Evaluation uses 200 clean trajectories and 200 retry events from 150 trajectories. Final columns use update 1000; near-best is the median of the three highest development-window preference values, not a held-out or training-seed average.

	Training data			Final update 1000		Near-best
ID	Clean traj.	Retry traj.	Retry events	P_{evt} (%)	Stage macro-F1	P_{evt} (%)
E1	300	49	78	24.5	0.8531	29.5
E2	300	219	299	36.5	0.8544	37.0
E3	300	367	500	38.0	0.8541	38.5
E4	300	750	1000	40.5	0.8520	40.5
E5	500	219	299	36.5	0.8511	38.0
E6	1000	219	299	37.5	0.8578	39.5
E7	1000	750	1000	40.5	0.8657	40.5

These development-window results favor balanced weighting but provide limited evidence about training-seed sensitivity.

Table 10: E7 directional-loss controls at $\lambda_{\mathrm{retry}} = 2$ on the development sets of Table 9. Values are historical development-window medians, not fixed-final or holdout results. Satisfaction and recovery are percentages.

$(c_{\mathrm{pre}}, c_{\mathrm{post}})$	Seed	P_{evt}	P_{traj}	Δ_{pre}	Δ_{post}	Recovery
$(1, 1)$	20261222	46.0	47.1	0.1209	0.2469	77.0
$(1, 1)$	20261223	45.5	45.3	0.1350	0.2753	74.5
$(4/3, 2/3)$	20261222	43.5	43.9	0.1394	0.2178	70.5
$(4/3, 2/3)$	20261223	44.5	44.7	0.1480	0.2440	68.5

B.4 Paired Comparison of Visual Input Configurations

Matched conditions. A uses base-camera 128×128/H16 inputs; B uses render-camera 512×512/H8 inputs. The models share training membership, annotations, initialization protocol, balanced directional coefficients, $\lambda_{\mathrm{retry}} = 2$, seed 20261222, and 1000 updates. Both use the fixed final checkpoint. A is trained on the paired corpus, not reused from historical E7. Viewpoint, resolution, and history change jointly, so the comparison does not isolate an individual input factor.

Results and paired outcomes. B improves local satisfaction, valley differences, stage classification, and terminal anchoring, but slightly worsens clean MAE, Kendall τ_b, and monotonicity violations (Table 11). Among 200 events, 76 satisfy the preference under both models, 63 under B only, 15 under A only, and 46 under neither. The net gain of 48 events corresponds to a 24.0-percentage-point improvement, although the A-only cases show that the gain is not universal.

B.5 Task-Wise Progress-Model Evaluation

Configuration B is adapted separately to each task with the shared recipe in Appendix A.3 and task-specific stages, targets, masks, and retries. Table 12 reports ten-task development results at update 1000, assessing task-specific progress learning rather than zero-shot transfer or downstream policy success.

Global accuracy and local sensitivity remain distinct. TransportBox has MAE 0.03278 and Acc@0.10 of 94.84%, but retry satisfaction is 56.00%. Satisfaction ranges from 47.00% on PegInsertionSide to 99.50% on PullCube. Different stage semantics and event distributions preclude interpreting these values as a controlled task-difficulty ranking; both global and local diagnostics are needed.

Table 11: Paired StackCube inputs: A is base-camera 128×128/H16; B is render-camera 512×512/H8. Training uses 1000 clean trajectories and 1000 retry events from 729 trajectories; development uses 200 clean trajectories (3996 progress-valid samples) and 200 retry events from 149 trajectories. Both checkpoints are fixed at update 1000.

Metric	A	B
Clean-success development		
Stage macro-F1 ↑	0.8426	**0.9061**
Sample MAE ↓	**0.05288**	0.05449
Acc@0.05 (%) ↑	69.47	**70.52**
Acc@0.10 (%) ↑	**87.94**	87.81
Acc@0.15 (%) ↑	**93.62**	93.34
Terminal-anchor error ↓	0.07896	**0.04252**
Trajectory-mean Kendall τ_b ↑	**0.9382**	0.9205
Monotonicity violations (%) ↓	**6.88**	9.03
Retry development		
P_{evt} (%) ↑	45.50	**69.50**
P_{traj} (%) ↑	44.52	**68.79**
Δ_{pre}	0.1750	0.3088
Δ_{post}	0.2881	0.4851
Recovery consistency (%) ↑	74.50	**79.50**
Retry loss ↓	1.20688	**1.06744**

B.6 Qualitative Analysis and Learning Dynamics

The following examples use the fixed final checkpoints of the paired A/B models. Learning curves describe their training trajectories separately.

Clean trajectories. Figure 4 aligns two clean-development trajectories with saved predictions and segment-wise targets. In case01, stage-2 segments at steps 6–17 and 19–21 are separated by a stage-1 singleton at 18. The singleton has no absolute-progress target; re-entry restarts $u_t^* = 0$, yielding $g_t^* = b_2 \approx 0.289$. This target reset follows normalization and does not independently establish a physical setback.

Table 12: Task-specific B models on ten tasks at update 1000. Each development set has 200 clean trajectories and 200 retry events; N_{prog} counts valid progress samples and N_{traj} counts retry-source trajectories. Acc@0.10 and preference satisfaction are percentages.

	Clean development				Retry development		
Task	N_{prog}	MAE ↓	Stage macro-F1 ↑	Acc@0.10 ↑	N_{traj}	P_{evt} ↑	P_{traj} ↑
StackCube	3996	0.05449	0.9061	87.81	149	69.50	68.79
PushCube	2567	0.04357	0.8564	90.73	160	89.50	88.63
PullCube	1848	0.03905	0.6707	92.91	200	99.50	99.50
PickCube	1509	0.08370	0.8273	72.63	182	90.50	90.93
LiftPegUpright	1297	0.07392	0.8848	74.79	184	96.00	97.08
PokeCube	4295	0.09163	0.8145	76.44	198	88.50	88.89
PullCubeTool	4509	0.07964	0.8680	76.18	188	94.00	94.33
PegInsertionSide	5381	0.09053	0.8227	67.85	136	47.00	44.22
PlaceApple	9521	0.08667	0.7379	71.69	187	74.00	75.22
TransportBox	14798	0.03278	0.9208	94.84	193	56.00	56.99

Stage macro-F1 retains the original four-slot evaluation convention, including the inactive stage slot for PullCube. N_{prog} is not the denominator for stage classification.

Retry preferences and counterexamples. Figures 5 and 6 show all four paired preference outcomes: both models succeed in case03, B only in case04, A only for the selected case05 event, and neither in case06. The trajectories ultimately succeed; preference failure denotes prediction ordering, not execution failure. Each point is a complete annotated-window mean, not a prediction for

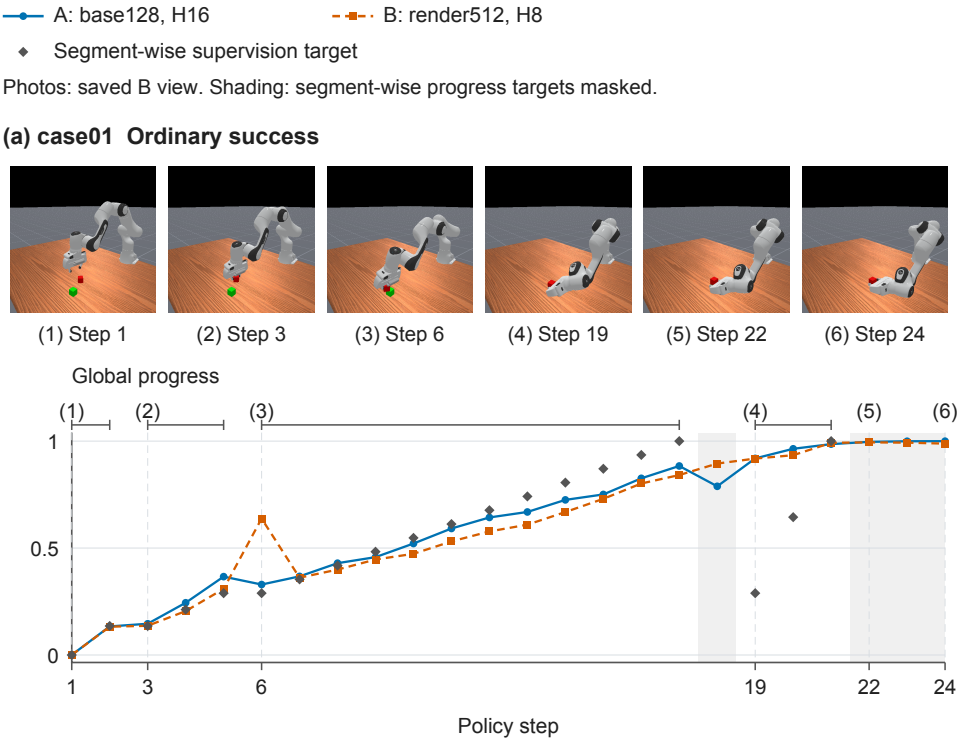

Figure 4: Clean-success predictions and segment-wise targets on StackCube. Numbered B-view images align with policy steps; A/B retain their own inputs. Diamonds mark valid targets and shading excludes absolute-progress supervision only. The case01 reset reflects segment normalization, not an independently measured setback. Saved predictions are unsmoothed and endpoints are not forced.

the displayed frame. The examples were selected post hoc to illustrate outcomes, whose frequencies come from the full 200-event comparison in Appendix B.4.

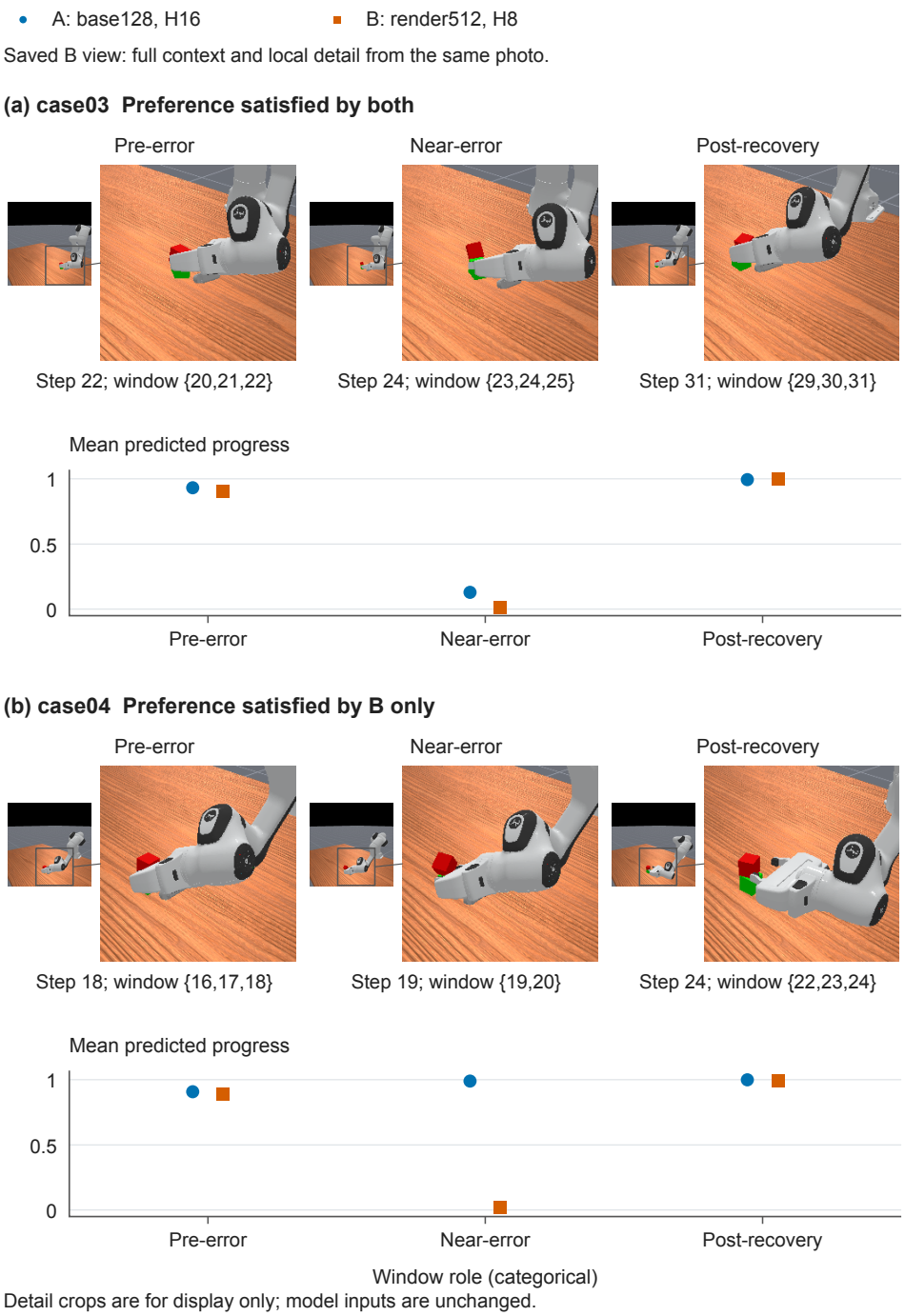

Figure 5: Final A/B retry-window predictions: both satisfy case03; only B satisfies case04. Points average complete annotated windows, and categorical positions do not encode time. Full views and crops share the same B-view image. Post-hoc successful-trajectory examples illustrate prediction ordering, not aggregate performance.

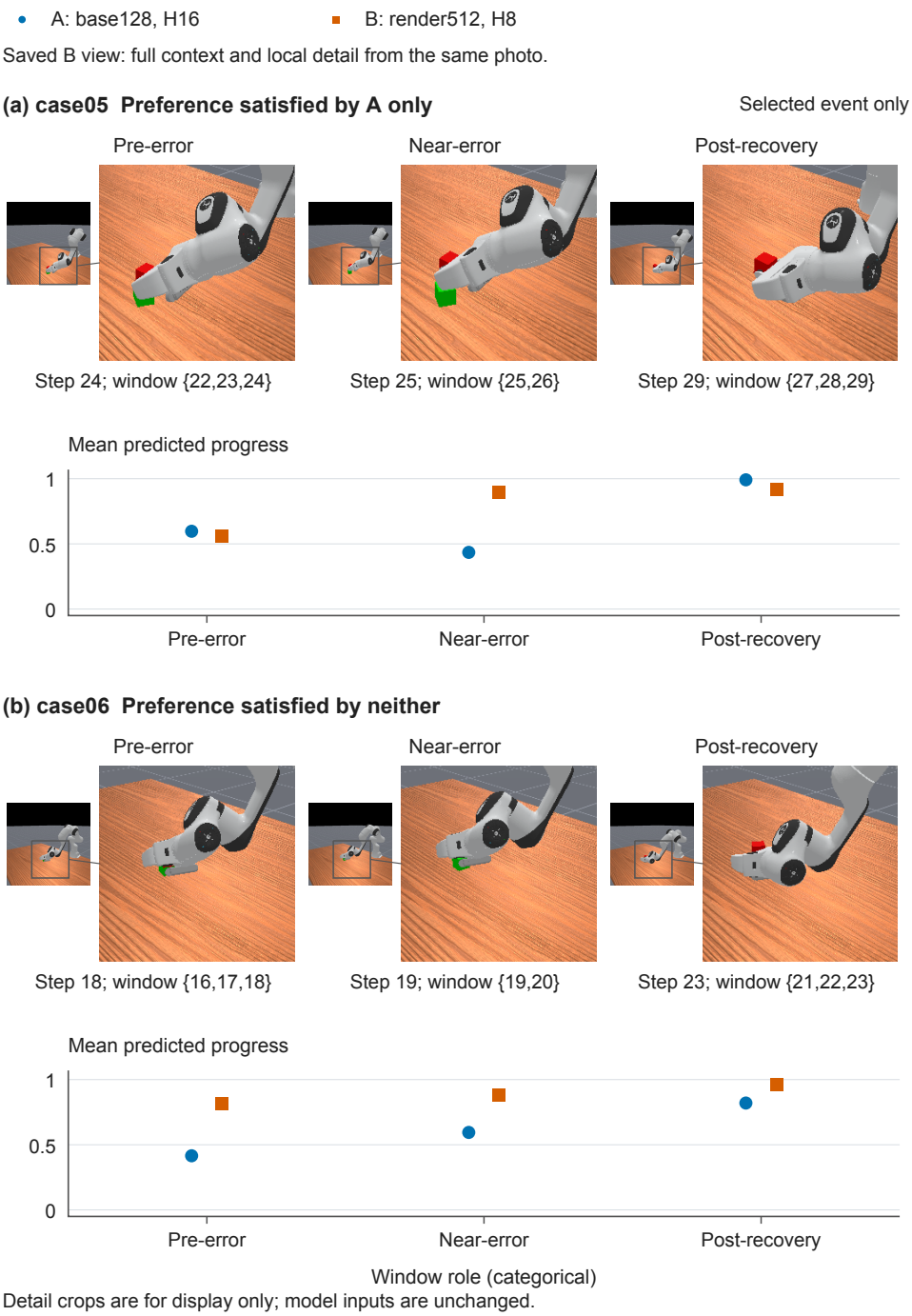

Figure 6: Complementary final-checkpoint cases: only A satisfies the selected case05 event; neither satisfies case06. Both trajectories succeed. Unconnected points are full-window means; paired views are the same saved images. The case05 conclusion concerns its selected event only.

Learning dynamics. Figure 7 separates training losses from development metrics. Both models reduce losses and clean MAE. After early fluctuations, B maintains higher retry satisfaction through update 1000: 69.5% versus 45.5%, with slightly higher clean MAE (0.054492 versus 0.052883). The final-checkpoint comparison therefore preserves the local–global trade-off rather than selecting an intermediate optimum. Remaining prediction errors motivate complementary state-support and action evidence, not treating progress decreases as verified failure onsets.

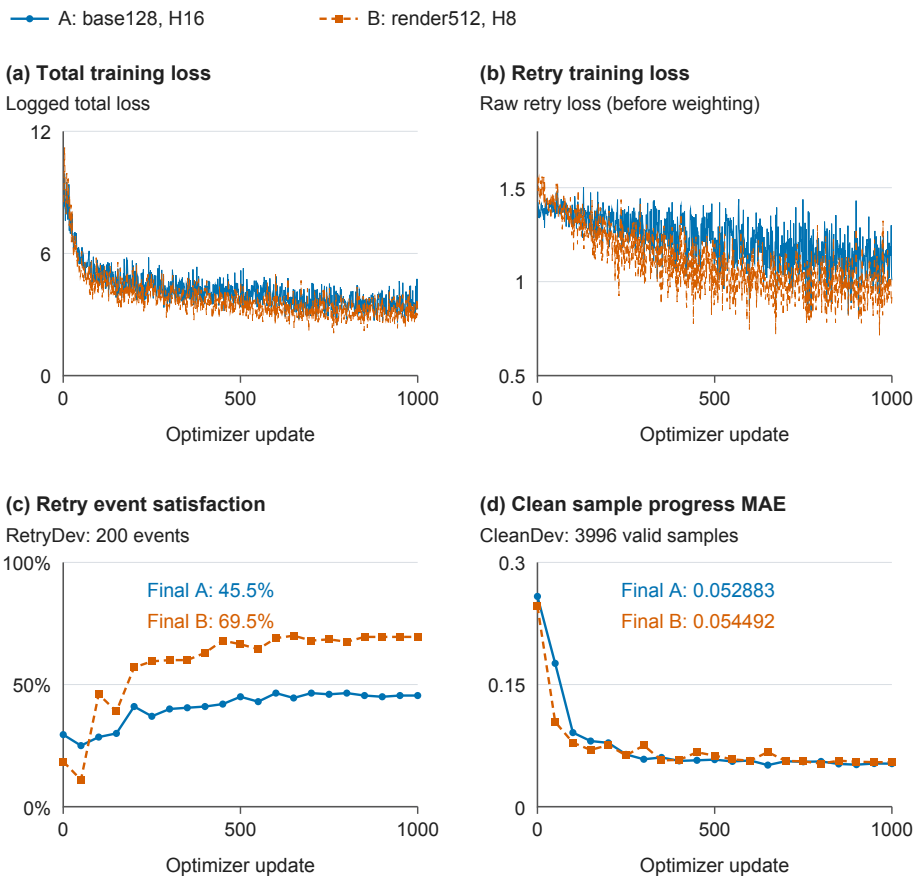

Figure 7: Paired A/B learning dynamics: total loss, raw retry loss before λ_{retry} scaling, retry satisfaction, and clean MAE. Each configuration is one training run. Curves are unsmoothed; no across-seed uncertainty is estimated.

C SUCCESS-REFERENCED FAILURE LOCALIZATION

Appendix B examines progress predictions and their remaining errors. We now evaluate how successful references and temporally corroborated evidence support offline failure detection and onset localization. Logged decision timing is reported separately, and paired A/B comparisons remain distinct from historical detector-development studies.

C.1 EVALUATION PROTOCOL AND METRICS

Data and reference onsets. The paired STACKCUBE study uses four trajectory-disjoint sets: 1000 successful reference trajectories, 1000 successful calibration trajectories, 1000 controlled failures, and 1000 clean evaluation successes. Failures comprise 250 interventions each of wrong direction, action freezing, premature gripper opening, and action perturbation. An example is accepted when the unmodified continuation from the same pre-intervention state reaches stable success but the intervened rollout does not. Its reference $t^*_{a,i}$ is the first actually modified action after bounding and application, not the first altered raw output or later visible consequence. The controlled and clean evaluation sets form Dev2, which informed development and input selection; it is not an independent final test.

Model-specific references and calibration. The paired A/B models from Appendix B.4 use update-1000 checkpoints. Trajectory membership and statistical rules are shared, but representations, reference indices, and thresholds are built separately. Retrieval uses five neighbors, ten progress bins, and same-stage expansion. Conditional calibration requires 20 rows, otherwise falling back to stage-level and global nonterminal statistics. Strong state/action thresholds use successful-calibration 99th percentiles; progress uses the stage-conditioned 1st percentile. Confirmation retains five indices, $0.95 \le r_S < 1$ near-boundary support, and Eq. (12); refinement uses the weak-action 95th percentile and at most five preceding steps.

Detection and localization metrics. Detection counts trajectories, not repeated alarms. Confirmation on a controlled failure is a true positive (TP); a trigger on a clean success is a false positive (FP). False negatives (FN) and true negatives (TN) are the respective nontriggered trajectories. Precision and F1 refer to this balanced failure–success mixture. For confirmation indicator $\gamma_i \in \{0, 1\}$, define

$$e_i = \begin{cases} |\hat{t}_i - t^*_{a,i}|, & \gamma_i = 1, \\ 50, & \gamma_i = 0. \end{cases} \tag{19}$$

For n controlled trajectories,

$$\mathrm{NLE} = \frac{1}{50n} \sum_{i=1}^{n} e_i, \qquad \mathrm{Acc@}k = \frac{1}{n} \sum_{i:\gamma_i=1} \mathbf{1}[e_i \le k], \quad k \in \{3, 5, 10\}. \tag{20}$$

Misses remain in overall denominators. The fixed value 50 is a miss penalty and normalization scale, not the execution horizon. Conditional Acc@k instead divides by the confirmed-failure count. These step-tolerance metrics differ from progress-value tolerances in Appendix B.1.

Uncertainty and timing. Paired 95% confidence intervals use 2000 trajectory-level bootstrap replicates, stratified by four intervention types and clean successes, with seed 20261222. They describe trajectory-sampling variation for fixed models, not training-seed variation. Logged $t^{\log}_{\mathrm{dec}}$ is distinct from $\hat{t}_0$ and $\hat{t}$; dependence on later observations prevents interpreting logged differences as online alarm latency.

C.2 PAIRED DETECTION AND LOCALIZATION RESULTS

B detects 88 additional failures and adds eight false positives (Table 13). Paired intervals exclude zero for the 8.60-point Acc@10 gain and 0.08592 NLE reduction on Dev2. The FPR interval includes zero but does not establish absence of a false-positive cost. B's 69.0% recall differs from its 49.4% overall Acc@10; conditional Acc@10 is 71.59% after excluding 310 misses. Conditional scores use different detected subsets and cannot replace the overall metrics.

Table 13: Paired A/B results on Dev2 (1000 controlled failures, 1000 clean successes). Rates are percentages; $\Delta = B - A$ uses percentage points except for NLE. Paired 95% intervals describe the fixed models. Differences are computed before rounding; conditional metrics use each model's detected subset.

Metric	A	B	Δ	95% CI for Δ
Trajectory-level detection				
TP / FN	602/398	690/310	–	–
FP / TN	122/878	130/870	–	–
Recall ↑	60.20	69.00	+8.80	[+5.00, +12.50]
FPR ↓	12.20	13.00	+0.80	[−1.90, +3.30]
Precision ↑	83.15	84.15	+1.00	[−1.88, +4.02]
F1 ↑	69.84	75.82	+5.99	[+3.24, +8.80]
Balanced accuracy ↑	74.00	78.00	+4.00	[+1.75, +6.15]
Overall localization: 1000 failures per model				
NLE ↓	0.49932	0.41340	−0.08592	[−0.11974, −0.05386]
Acc@3 ↑	26.00	36.40	+10.40	[+6.80, +14.30]
Acc@5 ↑	33.10	41.70	+8.60	[+4.90, +12.60]
Acc@10 ↑	40.80	49.40	+8.60	[+4.80, +12.60]
Conditional localization: 602 A / 690 B triggers				
Conditional Acc@3 ↑	43.19	52.75	–	–
Conditional Acc@5 ↑	54.98	60.43	–	–
Conditional Acc@10 ↑	67.77	71.59	–	–

Table 14: Dev2 results by intervention, with 250 failures per type. Recall and Acc@10 are percentages. NLE and overall localization retain misses according to Eq. (20).

	Recall ↑		NLE ↓		Acc@10 ↑	
Intervention	A	B	A	B	A	B
Wrong direction	72.4	74.0	0.34376	0.34216	60.4	57.2
Action freezing	51.2	57.2	0.59824	0.57776	32.0	25.6
Premature gripper opening	62.0	81.6	0.49032	0.25592	41.2	71.2
Action perturbation	55.2	63.2	0.56496	0.47776	29.6	43.6

Table 15: Earlier E7L2-01 rules on 500 controlled failures and 500 clean successes. G1–G4 vary bank/calibration trajectory counts. FP counts trajectories; Acc@10 and FPR are percentages. A1–A3 retain their earlier definitions.

Group: bank/calibration	Gate	TP	FP	NLE ↓	Acc@10 ↑	FPR ↓
G1: 500/500	A1	283	143	0.54844	38.2	28.6
	A2	243	139	0.62460	29.8	27.8
	A3	14	14	0.97816	1.2	2.8
G2: 500/1000	A1	277	134	0.56016	36.6	26.8
	A2	243	132	0.62284	30.0	26.4
	A3	12	6	0.98264	0.8	1.2
G3: 1000/1000	A1	277	134	0.56016	36.6	26.8
	A2	236	133	0.63516	28.8	26.6
	A3	12	6	0.98264	0.8	1.2
G4: 1000/500	A1	283	143	0.54844	38.2	28.6
	A2	235	138	0.63944	28.2	27.6
	A3	14	14	0.97916	1.0	2.8

C.3 FAILURE-TYPE ANALYSIS

B improves recall for all intervention types (Table 14), but Acc@10 gains concentrate in premature gripper opening (+30.0 points) and action perturbation (+14.0). Acc@10 decreases for wrong direction (-3.2) and action freezing (-6.4), despite higher recall. Lower penalized error or fewer misses need not place more onsets inside a particular tolerance. These controlled STACKCUBE comparisons highlight the importance of evaluating failure detection and onset localization separately across intervention types.

C.4 DEVELOPMENT ANALYSES OF EVIDENCE RULES AND REFERENCE SETS

The reference-set and evidence-rule studies below use the earlier E7L2-01 checkpoint and reuse a cohort of 500 controlled failures and 500 clean successes. Onset-refinement results on Dev2 are reported separately for historical E7L2-01 controls and a matched comparison using the fixed final B checkpoint (Table 17).

Reference scale and earlier gates. A1 uses progress alone; A2 requires current successful-state support; A3 adds a same-index raw-action anomaly (Table 15). These earlier gates differ from the current detector and are unrelated to visual A/B. Restrictive conjunction reduces false alarms but misses almost all failures (G3/A3: 12/500 versus A1: 277/500). Increasing bank/calibration size does not resolve this behavior under these rules.

Temporal and action controls. With applied-action evidence, S0 requires same-index conjunction; R0 permits windowed three-type evidence; R1 uses action/support-initiated two-type confirmation; R2 also permits progress initiation (Table 16). R1 improves detection/localization at a false-positive cost; R2 adds 0.6 Acc@10 points and 2.2 FPR points. State conditioning raises Acc@10 at both $W = 5$ and $W = 7$, also increasing FPR. S95 adds near-boundary support; D2 persistence lowers its FPR from 15.8% to 13.6% while retaining 60.8% Acc@10. Persistence therefore moderates false alarms rather than improving every metric.

Table 16: Historical controls on the reused 500-failure/500-clean cohort. W is the confirmation window, not visual history. TP counts trajectories; FPR and Acc@10 are percentages. Repeated baselines reuse the same records.

Setting	W	TP	FPR ↓	NLE ↓	Acc@10 ↑
(a) Temporal evidence rules					
S0: same-index conjunction	–	12	1.0	0.98048	1.4
R0: windowed three-type	5	45	0.8	0.92748	5.4
R1: action/support anchor	5	300	9.8	0.52880	36.0
R2: any evidence anchor	5	312	12.0	0.51252	36.6
(b) Global versus state-conditioned action					
Global action	5	300	9.8	0.52880	36.0
State-conditioned action	5	392	12.8	0.33540	57.2
Global action	7	308	10.8	0.50764	38.6
State-conditioned action	7	397	12.8	0.31696	60.2
(c) Weak support with state-conditioned action					
S95	5	412	15.8	0.29220	62.2
S95 + D2	5	398	13.6	0.31336	60.8

Onset refinement. Historical E7L2-01 variants C0–C5 retain identical detection outcomes (Table 17(a)). The adopted five-step C2 rule lowers NLE from 0.30250 to 0.28402 and raises Acc@10 from 59.8% to 62.9% relative to C0; longer look-backs attain higher point estimates on several metrics. For the fixed final B checkpoint (Table 17(b)), C2 lowers NLE from 0.44310 to 0.41340 and increases Acc@3, Acc@5, and Acc@10 by 7.2, 3.9, and 6.8 percentage points, respectively, while recall and FPR remain unchanged at 69.0% and 13.0%. These comparisons change only the reported onset, not detection outcomes or the logged decision index; the gains therefore reflect more accurate offline localization rather than earlier online alarms.

Table 17: Onset refinement on StackCube Dev2. (a) Historical E7L2-01 controls: TP/FN/FP/TN remain 821/179/116/884. q_A is the weak-action quantile, L the look-back in steps, and C5 permits a support precursor. (b) Matched comparison using the fixed final B checkpoint, with detection outcomes held unchanged. All Acc@k values include all 1000 controlled failures; recall uses these failures and FPR uses 1000 clean successes. Rates are percentages.

(a) Historical E7L2-01 controls

Variant	q_A	L	Support precursor	NLE ↓	Acc@3 ↑	Acc@5 ↑	Acc@10 ↑
C0	–	0	No	0.30250	41.1	48.7	59.8
C1	0.975	5	No	0.29090	43.9	51.3	61.7
C2	0.950	5	No	0.28402	46.4	52.4	62.9
C3	0.975	10	No	0.27694	45.8	54.9	65.7
C4	0.950	10	No	0.26926	47.9	55.4	67.3
C5	0.950	10	Yes	0.26822	47.1	55.5	67.6

(b) Fixed final B: effect of onset refinement

Reported onset	Recall ↑	FPR ↓	NLE ↓	Acc@3 ↑	Acc@5 ↑	Acc@10 ↑
No refinement ($\hat{t}_0$)	69.0	13.0	0.44310	29.2	37.8	42.6
C2 refinement ($\hat{t}$)	69.0	13.0	**0.41340**	**36.4**	**41.7**	**49.4**

C.5 Temporal Evidence and Qualitative Cases

Evidence coordinates. Figures 8 and 9 show B's saved evidence for three action-perturbation rollouts and one clean success, selected post hoc for accurate attribution, offset attribution, a miss, and a false alarm. Four saved images accompany applied-action deviation, support-distance ratio, and progress regression. Orange curves are evidence, dashed curves are thresholds, the weak support boundary is 0.95, and missing indices remain disconnected. For display, action evidence at index t appears at state $t + 1$, support at t, and g_t at $t + 3$, the last required future state. Ref., Initial, C2, and Log D retain the recorded indices of t_a^*, $\hat{t}_0$, $\hat{t}$, and $t_{\text{dec}}^{\log}$. Display coordinates do not reconstruct online decisions.

D Event Annotation and Human Review

This appendix defines task events and evaluates annotation agreement. Human judgments and automatic–human timing are distinct from the controlled localization evaluation in Appendix C and from policy success.

D.1 Task Stages and Event Definitions

Stages describe execution structure; retries describe local departures. Table 18 lists the real-robot task stages and separate completion criteria. Revisiting a stage does not create a new phase. A local failure is an observable setback, such as a missed or lost grasp or a disrupted tool–object relation. Correction onset marks remedial action; recovery re-establishes the relevant relation, not necessarily final completion. Regrasping a dropped cube can restore the grasp without completing stacking. Routine adjustment alone is not a retry.

D.2 Annotation Procedure and Visual Interface

Simulation predicates applied to recorded physical states generate stages, retries, and completion labels, which human review checks against execution. Failure onset, correction onset, recovery, and the potential-based retry keypoint retain distinct meanings; the labeler stores degradation and recovery boundaries separately. These physical signals construct supervision, not additional model inputs (Appendix A.3).

The real-robot workspace combines base/wrist views, sparse event labels, temporal navigation, and notes (Fig. 10). Labels are confirmed at annotation-grid frames; unmarked frames remain unla-

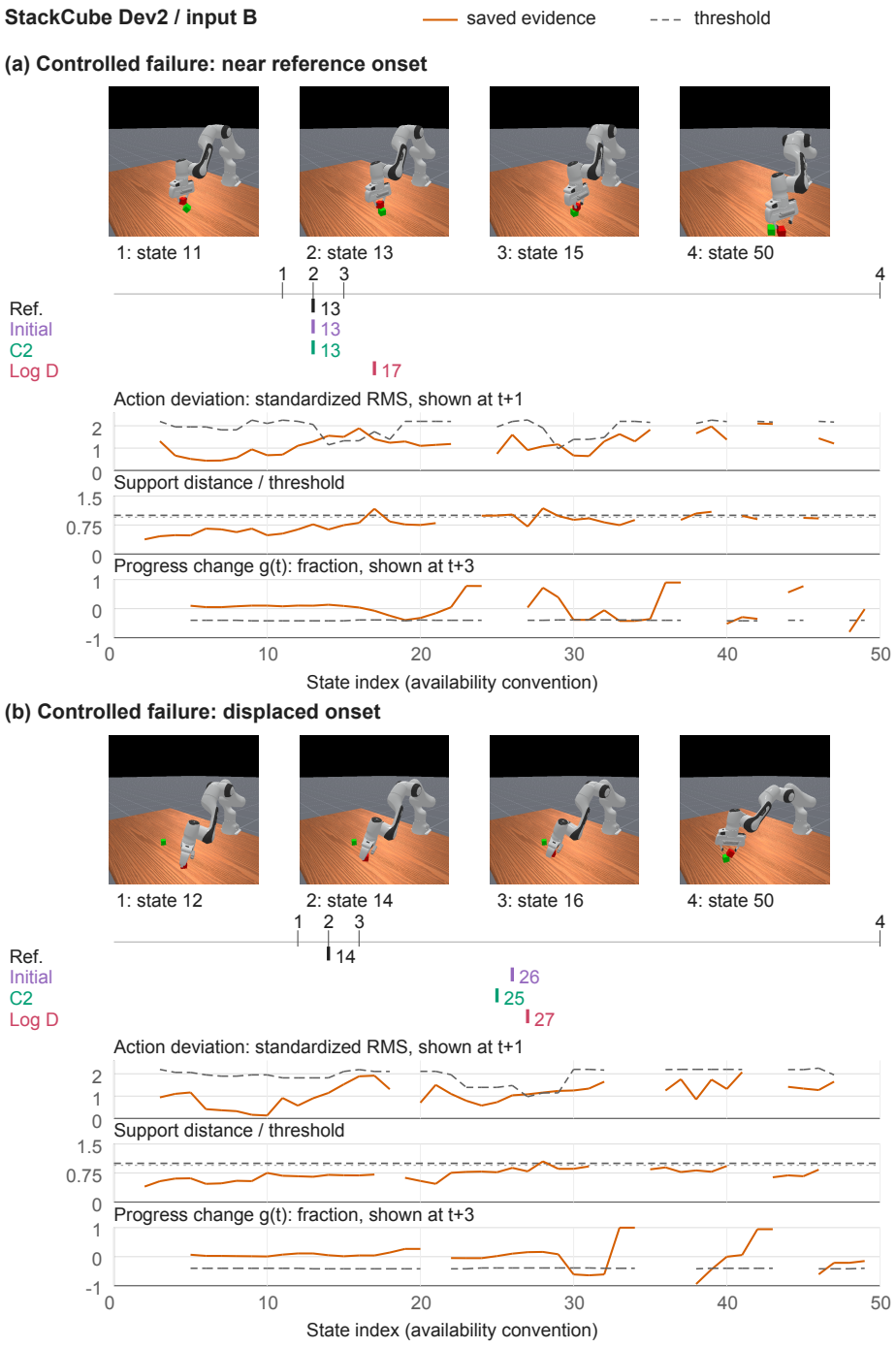

Figure 8: B evidence on action-perturbation rollouts. (a) Initial and refined onsets equal reference 13; Log D is 17. (b) Refinement from 26 to 25 leaves an 11-step error relative to reference 14. Log D is 27, although confirming progress requires state 30. Orange curves are saved evidence, dashed curves are thresholds, and gaps denote unavailable evidence. Log D records attribution rather than measured online alarm time. These post-hoc examples illustrate accurate and offset localization, not their population frequencies.

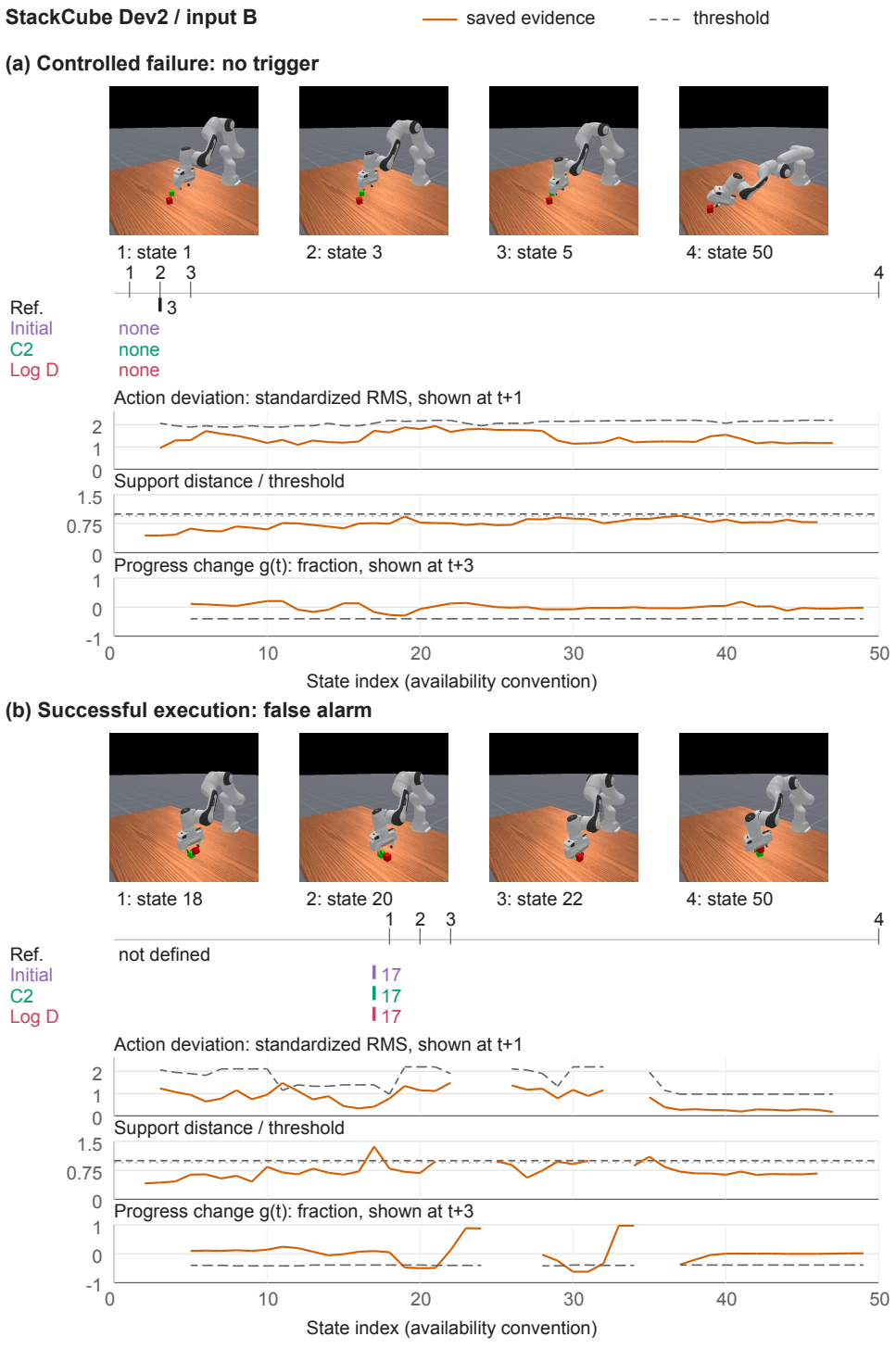

Figure 9: B counterexamples. (a) The reference-3 intervention is missed. (b) A clean success triggers with initial onset, refined onset, and Log D all 17; confirming progress nevertheless requires state 20, and no failure reference exists. Zero logged gap is therefore not zero online latency. Mean logged gaps of 2.397 (A) and 2.230 (B) use different triggered subsets, not matched online timings. Misses and false alarms remain; natural-failure anchors are candidate risk locations, not causal truth.

Table 18: Real-robot annotation stages and separate completion criteria. PourWater transfers small balls between cups.

Task	Process stages	Completion criterion
StackCube	Approach and grasp; transport and align; stack and release.	The upper cube is stacked, released, and stable.
PokeCube	Acquire the rod; align its tip; push the cube toward the target.	The cube reaches the target region and the robot is stationary.
PullCubeTool	Acquire the tool; position the hook; pull the cube.	The cube reaches the designated retrieval region.
Orange-to-Tray	Approach and grasp; transport and align; place and release.	The released orange remains at the target location in the tray.
PourWater	Acquire the container; position it; pour; finish the transfer.	At least 90% of the initial small balls reach the receiving cup.

beled, not implicit negatives. The displayed 10 Hz grid, source-frame indices, timestamps, and policy-action indices require explicit temporal mapping. For StackCube, the current frame is the third of a visual three-frame stacking, release, and stability check; other tasks use Table 18. This observational check is not the simulator's physical-state predicate or the end-of-execution policy evaluation criterion.

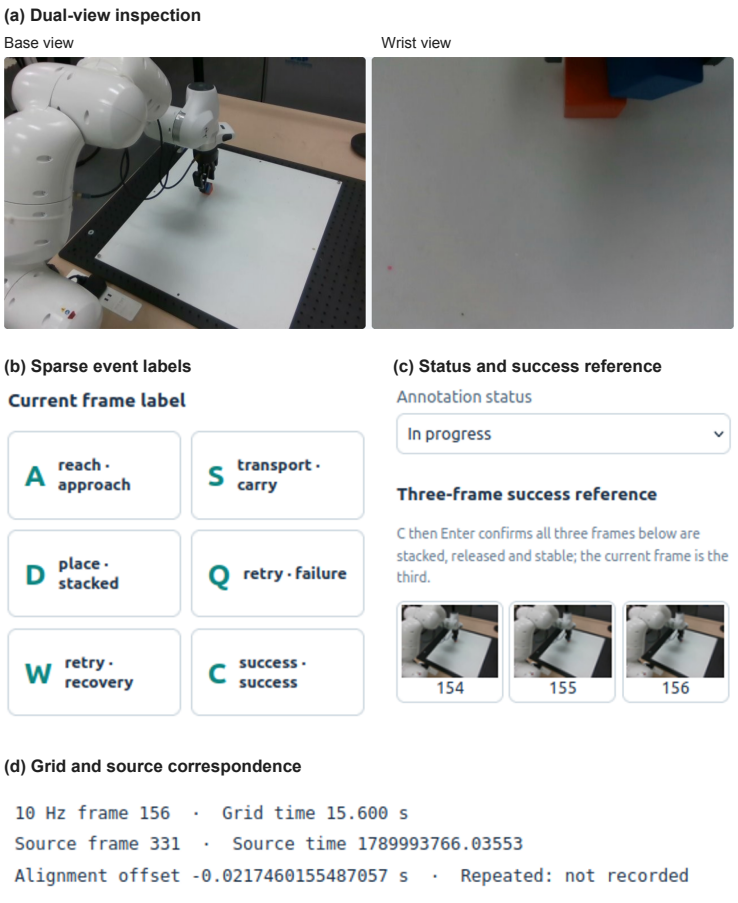

Figure 10: Real-robot annotation workspace: dual-view inspection, sparse event labels, completion reference, and grid/source-time mapping. Panels are display crops from one interface screenshot. Unmarked frames are unlabeled, and the 10 Hz annotation grid is distinct from policy-action indices.

D.3 Agreement on Retry Presence

Two annotators review retry presence on 300 simulated StackCube trajectories and 300 physical trajectories across five tasks. The unit is a binary trajectory judgment, not an event count. Agreement uses the two initial labels only when both responses are valid: 260 simulation pairs and 298 physical pairs. The remaining 40 and two records stay in review totals but not binary agreement; valid disagreements remain included. Let n_{uv} count responses $u, v \in \{0, 1\}$, where 1 denotes retry presence, and $N_{\text{pair}} = \sum_{u,v} n_{uv}$ among N_{review} reviewed trajectories. We report

$$\begin{aligned} \text{Coverage} = \frac{N_{\text{pair}}}{N_{\text{review}}}, \quad P_o = \frac{n_{11} + n_{00}}{N_{\text{pair}}}, \quad \kappa = \frac{P_o - P_e}{1 - P_e}, \\ P_e = \frac{(n_{11} + n_{10})(n_{11} + n_{01}) + (n_{01} + n_{00})(n_{10} + n_{00})}{N_{\text{pair}}^2}. \end{aligned} \tag{21}$$

Coverage and agreement have different denominators; neither annotator is ground truth. Table 19 summarizes valid-pair coverage and agreement between the two annotators.

Table 19: Two-annotator retry-presence judgments. Coverage uses all reviewed trajectories; contingency counts and agreement use valid binary pairs only (1: present, 0: absent). Missing responses are excluded, not converted to negatives. Rates are percentages.

Domain	Reviewed	Valid pairs	Excluded	Coverage (%)
Simulation	300	260	40	86.67
Real robot	300	298	2	99.33

Domain	n_{11}	n_{10}	n_{01}	n_{00}	P_o (%)	κ
Simulation	150	35	42	33	70.38	0.2580
Real robot	294	1	1	2	99.33	0.6633

Simulation labels agree on 183/260 trajectories, with 77 disagreements, $P_o = 70.38\%$, and $\kappa = 0.2580$, indicating substantial ambiguity in visual retry judgments. Physical labels agree on 296/298, with $P_o = 99.33\%$ and $\kappa = 0.6633$, but each annotator labels 295/298 as retry-present and only two receive two negative labels. High physical agreement therefore provides little evidence about retry-absent judgments. Different tasks, observations, and class proportions make the domain contrast descriptive rather than a controlled modality comparison.

D.4 Temporal Correspondence with Human References

For StackCube, we compare automatic and human stage anchors, local failure onsets, recovery points, and completion confirmations. Each category c uses M_c corresponding annotation pairs on the same simulation-step axis:

$$\text{TA}_c(k) = \frac{1}{M_c} \sum_{j=1}^{M_c} \mathbf{1}[|t_{c,j}^{\text{auto}} - t_{c,j}^{\text{human}}| \le k], \qquad k \in \{2, 10\}. \tag{22}$$

Counts refer to paired events, not independent trajectories or autonomous event-detection recall.

Table 20: StackCube automatic–human temporal correspondence. M counts paired annotations and Q_k counts those within k simulation steps; $\text{TA}(k) = 100Q_k/M$. Human, automatic, and paired counts coincide within each reported category.

Annotation type	M	Q_2	TA(2) (%)	Q_{10}	TA(10) (%)
Stage anchor	400	310	77.50	395	98.75
Local failure onset	126	94	74.60	123	97.62
Recovery point	122	90	73.77	120	98.36
Completion confirmation	146	127	86.99	145	99.32

Table 20 shows agreement rates of 73.77%–86.99% within two steps and 97.62%–99.32% within ten steps. Completion has the highest two-step agreement; failure and recovery boundaries vary more. Coarse agreement does not establish precise alignment, and aggregate counts do not identify discrepancy causes. Human–human presence agreement, automatic–human timing, and controlled failure localization answer different questions. None establishes a unique causal onset for natural failures.

E Risk-Guided Reweighting and Policy Learning

Following the localization and annotation analyses in Appendices C and D, we examine how risk evidence shapes learning from successful demonstrations. Appendices E.1–E.4 analyze StackCube weight distributions, risk sources, and demonstration composition under the original seed-42 policy-training protocol. Appendix E.5 extends the analysis to three-seed downstream ablations across seven policy–task settings.

E.1 Data Sources and Reweighting Protocol

Data sources. The motion-planning Expert300 contains 300 successful trajectories and 31,707 valid action-supervision rows. *Natural mix* denotes this unstratified demonstration collection, not the natural-failure pool. Auxiliary retries provide 1000 pre-error anchors from 729 trajectories; 352 of 489 natural failures yield confirmed anchors, and 137 yield none (Table 21). Only Expert300 supplies policy action targets.

Table 21: StackCube Expert300 and auxiliary risk sources. Trajectories, policy-training rows, and risk anchors are different units; only Expert300 supplies action targets.

Source	Trajectories	Contribution
Original Expert300	300	31,707 valid action-supervision rows.
Successful retries	729	1000 event-specific pre-error risk anchors.
Natural failures	489	352 confirmed anchors; 137 untriggered rollouts.

Weights and loss. The frozen update-1000 B model uses render-camera 512×512/H8. Each risk source independently returns one nearest same-task, same-stage anchor by cosine distance, then applies $\delta_z = \delta_u = 0.15$; a rejected anchor is not replaced. Advancement uses $t_i^+ = \min(t_i + 10, T_i)$, $\delta_V = 0.12$, and $\beta = 1$, yielding weights in $[1, 2]$ through Eqs. (14)–(16). V0 uses unit weights, V1 retry risk, V2 natural-failure risk, and V3 their maximum before the shared advancement gate. All rows remain, including 2700 short terminal chunks; padding adds no physical transitions. Loss reduction follows Appendix A.5: the 10×32 residual is averaged per row and the weighted ratio is computed over the full logical batch, preserving targets, normalization, and sampling probabilities.

E.2 Distribution of Training Emphasis

Candidate availability, match acceptance, and enhancement ($w_i > 1$) are distinct. A boundary match can have zero risk, and nonpositive advancement suppresses enhancement. Table 22 uses all 31,707 rows. V1/V2/V3 enhance 8634/15,485/16,177 rows; their unit-weight fractions are 72.77%/51.16%/48.98%.

Figure 11 retains the masses at $w = 1$. Because the gate is shared, $w_i^{\mathrm{V3}} = \max(w_i^{\mathrm{V1}}, w_i^{\mathrm{V2}})$; the cumulative-distribution ordering follows the fusion rule, not policy superiority. The ten largest anchors contribute 66.06%, 37.50%, and 31.69% of extra weight for V1/V2/V3. V3 attributes extra weight to the larger-risk source and splits positive exact ties equally. These statistics describe emphasis, not individual anchor effectiveness.

Matching and advancement examples. Three post-hoc rows from one expert trajectory illustrate distinct cases. At row 3, accepted natural risk 0.1550 and $q_i = 0.2985$ yield $w_i^{\mathrm{V3}} = 1.0463$. At row 19, an accepted natural match but $\Delta V_i = -0.06757$ gives unit weight. At row 0, advancement

Table 22: Offline statistics over all 31,707 Expert300 rows. V3 availability/acceptance requires either source to qualify. Top-10 share is the fraction of extra weight $\sum_i (w_i - 1)$ attributed to the ten largest anchors, not normalized loss share.

Statistic	V1: Retry	V2: Natural	V3: Combined
Candidate coverage (%)	100.00	99.00	100.00
Accepted matches (%)	46.90	76.95	80.69
Enhanced rows, $w > 1$ (%)	27.23	48.84	51.02
Mean raw weight	1.0477	1.1008	1.1116
95th-percentile raw weight	1.3232	1.5136	1.5383
Maximum raw weight	1.8444	1.8669	1.8669
Top-10 anchor extra-weight share (%)	66.06	37.50	31.69

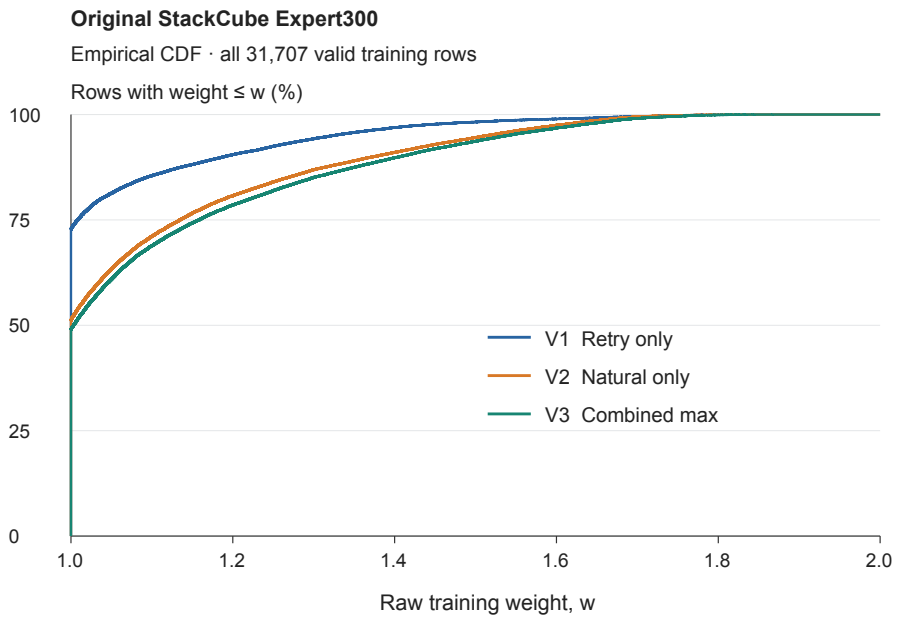

Figure 11: Empirical raw-weight distributions over all 31,707 Expert300 rows, including unit weights. V1 uses retries, V2 natural failures, and V3 their maximum before the shared advancement gate. Saved weights are shown without smoothing or distribution fitting.

is positive but both candidates are rejected, also giving unit weight. These are the first examples of each condition in training-row order, not policy evaluations.

E.3 Risk-Source Ablation

V0–V3 fix Expert300 and start independently from `pi05_base`, using seed 42, global batch 32, and 30,000 updates. Final checkpoints share 200 evaluation seeds, a 250-step horizon, and ten-action chunks. Table 23(a) varies only risk sources; (b) compares V0/V3 on each composition. All differences are relative to the V0 trained on the same dataset.

V3 and V0 tie at 96.0% under both criteria. Retry-only is four points lower and natural-only 0.5 points lower. Combined risk avoids these aggregate reductions but does not exceed uniform training on Expert300; broader enhancement coverage is not itself a policy gain.

Variant selection. An earlier, separate development study favored V2. The Expert300 comparison in Table 23(a) subsequently informed the choice of V3 for later experiments and is reported here as a variant-selection analysis.

Table 23: StackCube risk sources and demonstration composition. (a) Original Expert300; (b) controlled compositions with Natural mix as a separate reference. Entries are successes/200 with percentages in parentheses; gains are relative to each dataset's V0. Natural mix is unstratified and differs in collection source, not a fourth controlled retry ratio.

(a) Original Expert300: risk-source ablation

Variant	Risk source	Once	Stable@3	Δ_{once} (pp)	Δ_{stable} (pp)
V0	None	192 (96.0)	192 (96.0)	0.0	0.0
V1	Retry	184 (92.0)	184 (92.0)	-4.0	-4.0
V2	Natural	191 (95.5)	191 (95.5)	-0.5	-0.5
V3	Combined	192 (96.0)	192 (96.0)	0.0	0.0

(b) Demonstration composition and original-data reference

Dataset	Variant	Clean/retry	Once	Stable@3	Δ_{once} (pp)	Δ_{stable} (pp)
G00	V0	300/0	194 (97.0)	192 (96.0)	0.0	0.0
G00	V3	300/0	193 (96.5)	192 (96.0)	-0.5	0.0
G25	V0	225/75	190 (95.0)	187 (93.5)	0.0	0.0
G25	V3	225/75	191 (95.5)	190 (95.0)	$+0.5$	$+1.5$
G50	V0	150/150	180 (90.0)	179 (89.5)	0.0	0.0
G50	V3	150/150	190 (95.0)	185 (92.5)	$+5.0$	$+3.0$
Natural mix	V0	Unstratified	192 (96.0)	192 (96.0)	0.0	0.0
Natural mix	V3	Unstratified	192 (96.0)	192 (96.0)	0.0	0.0

E.4 Demonstration-Composition Analysis

Composition datasets. G00/G25/G50 contain 0%/25%/50% retry-success trajectories. Each has 300 successes collected with the same frozen V0 policy and retains all 250 actions per trajectory, giving 75,000 rows. Across datasets there are 450 unique trajectories: adjacent mixtures share 225 each, and G00/G50 share 150. Within each dataset, V0/V3 share membership, normalization, initialization protocol, and budget, but train independently. The progress model and external risk pools remain fixed; weights are recomputed for each dataset's rows. G00 excludes retries from policy demonstrations, not from the auxiliary risk pool. Natural mix has a different collection source and remains outside the ordered composition comparison.

Results and interpretation. Stable Success@3 gains over matched V0 are 0.0, 1.5, and 3.0 points for G00/G25/G50 (Table 23(b)); the latter two correspond to three and six additional stable episodes. Success-once gains are -0.5, $+0.5$, and $+5.0$ points. The pattern suggests greater relative benefit when successful demonstrations contain more retries, but absolute stable success falls for both methods as the retry fraction rises. Reweighting partially offsets that decline; it does not show that adding retries improves demonstration quality. G50-V3 reaches success-once in 190 episodes but stable success in 185, preserving the metrics' distinction.

Relation to the main evaluation. Table 1 retains each task's original 300 demonstrations, without selecting a retry ratio from this analysis. The StackCube $\pi_{0.5}$ pair uses the Natural mix corpus and averages training seeds $\{42, 43, 44\}$; the composition comparisons in this subsection retain their original seed-42 protocol.

E.5 Downstream Risk-Source and Component Ablations

Protocol. We evaluate seven policy–task settings spanning five tasks and three policy families (Table 24). Within each setting, variants use the same 300 policy demonstrations, initialization protocol, learner-specific budget, and evaluation seeds as Table 1. BC/DP use training seeds $\{0, 1, 2\}$; $\pi_{0.5}$ uses $\{42, 43, 44\}$. Each final model is evaluated for 200 episodes with a 250-step horizon. The baseline and full-RAISE results are reused from Table 1.

Variants. V1 and V2 retain the full progress model and advancement gate but use retry or natural-failure risk alone. The no-retry-supervision variant uses the matched B input configuration with $\lambda_{\text{retry}} = 0$ and recomputes the dependent references, calibration statistics, and weights. The progress-only variant retains the full progress model but replaces multi-evidence failure localization with a progress-only rule. The no-advancement-gate variant retains the same anchors and risk matches and sets $q_i = 1$. All three component ablations use both risk sources; policies are trained independently using the resulting fixed weights.

Table 24: Downstream ablations: Stable Success@3 (%), averaged over three training seeds with 200 episodes per seed and setting. Avg. equally weights the seven policy–task settings, not the five distinct tasks or the ten tasks in Table 1. Δ is the mean change from V0 in percentage points. Means and differences are computed before rounding.

Configuration	BC Stack	DP Stack	BC Lift	DP Lift	$\pi_{0.5}$ Poke	$\pi_{0.5}$ Tool	$\pi_{0.5}$ Apple	Avg.	Δ (pp)
Uniform (V0)	0.0	39.2	61.2	24.7	44.0	38.0	53.5	37.2	0.0
Retry risk only (V1)	0.0	34.8	55.7	18.2	40.2	31.5	46.7	32.4	−4.8
Natural-failure risk only (V2)	8.8	43.8	65.3	32.2	49.3	43.2	60.8	43.4	+6.1
No retry supervision	5.7	42.7	58.5	29.2	47.2	41.5	57.2	40.3	+3.0
Progress-only localization	0.0	36.7	58.2	21.5	42.3	35.7	49.2	34.8	−2.4
No advancement gate	9.8	47.5	68.2	36.2	52.8	44.7	62.2	45.9	+8.7
Full RAISE (V3)	**13.7**	**49.5**	**71.2**	**41.7**	**56.2**	**49.2**	**64.5**	**49.4**	**+12.2**

Stack, Lift, Poke, Tool, and Apple denote StackCube, LiftPegUpright, PokeCube, PullCubeTool, and PlaceApple.

Results. Full RAISE achieves the highest mean success in each of the seven settings, increasing the aggregate from 37.2% to 49.4% (+12.2 pp). Retry-only weighting lowers the aggregate, whereas natural-failure risk alone improves it but remains 6.0 points below combined risk. Removing retry supervision, replacing multi-evidence localization, and removing the advancement gate reduce the aggregate by 9.1, 14.6, and 3.5 points, respectively. The latter still improves over V0, indicating that advancement gating adds to, rather than fully accounts for, the reweighting benefit. These comparisons support the downstream contribution of the tested designs; they do not establish per-setting statistical significance or universal necessity.

F Additional ManiSkill Benchmark Comparisons

Building on the risk-source and demonstration-composition analyses in Appendix E, we place the fixed GTP-FA-$\pi_{0.5}$ baseline and its RAISE variant alongside published results on shared ManiSkill tasks. The within-system pair evaluates the incremental effect of reweighting; cross-study comparisons provide context under different resources and evaluation protocols.

F.1 Comparison Scope and Reporting

Our rows reuse Table 1: three seeds $\{42, 43, 44\}$, 200 episodes per task/model, and environment seeds `400005010--400005209`. The paired grasp front-end, supervision, and adaptation settings are fixed. Our metric is Stable Success@3; published rows retain source-defined rates without conversion. Task means weight displayed tasks equally and gains are computed before rounding. No uncertainty is inferred from rounded published percentages. The three-task table reuses the four-task subset rather than adding an experiment. Differences in pretraining, demonstrations, controllers, and evaluation preclude equal-resource rankings or attributing cross-study gaps to RAISE.

F.2 Four-Task Comparison with the RDT Release

The RDT release uses 250 trials per method in ten groups of 25 (Liu et al., 2024). Table 25 excludes PlugCharger from its five tasks and recomputes the shared-task means. Our paired gain is +8.42

points, mainly from picking and insertion; their absolute stable-success rates remain 45.0% and 29.2%. RDT's published picking rate is higher, under a different protocol.

Table 25: Four shared ManiSkill tasks from the RDT release. Peg/Pick/Stack/Push denote PegInsertionSide/PickCube/StackCube/PushCube. Published rows retain source metrics; ours use three-seed Stable Success@3 (200 episodes/task/seed). Avg. covers these four tasks; Δ is within our fixed GTP-FA pair. Means and gains are rounded after aggregation.

Method	Peg	Pick	Stack	Push	Avg.
Published reference results: official RDT release					
RDT	13.2	77.2	74.0	100.0	66.10
OpenVLA	0.0	8.0	8.0	8.0	6.00
Octo	0.0	0.0	0.0	0.0	0.00
Diffusion Policy	0.0	40.0	80.0	88.0	52.00
Our evaluation: Stable Success@3, 3 seeds $\times$ 200 episodes/task					
GTP-FA-$\pi_{0.5}$	16.2	26.2	97.2	94.5	58.50
RAISE (GTP-FA-$\pi_{0.5}$)	29.2	45.0	98.2	95.3	66.92
Δ (pp)	+13.0	+18.8	+1.0	+0.8	+8.42

F.3 Three-Task Comparison with PhysGen

Table 26 retains the rates and rounded averages from Song et al. (2026). PhysGen uses 1000 demonstrations and 125 rollouts per task; its comparison includes published and reproduced baselines. Our three-task mean rises from 72.61% to 79.50% (+6.89 pp), mostly through PickCube. The four-task view additionally includes insertion, so the two means summarize different task subsets, not inconsistent model measurements. Aggregate gains do not establish per-task significance or isolate individual risk components.

Table 26: PhysGen's three-task reference rates and rounded means. Our rows reuse the corresponding Table 25 results, not new PhysGen-protocol evaluations. Published metrics are unchanged; ours use three-seed Stable Success@3. Δ is the within-pair gain (pp).

Method	PushCube	PickCube	StackCube	Avg.
Published reference results: PhysGen Table 2				
ACT	76	20	30	42
BC-T	98	4	14	39
Diffusion Policy	88	40	80	69
ICRT	77	78	30	62
RDT	100	77	74	84
π_0	100	60	48	69
PhysGen	100	73	48	74
Our evaluation: Stable Success@3, 3 seeds $\times$ 200 episodes/task				
GTP-FA-$\pi_{0.5}$	94.5	26.2	97.2	72.61
RAISE (GTP-FA-$\pi_{0.5}$)	95.3	45.0	98.2	79.50
Δ (pp)	+0.8	+18.8	+1.0	+6.89

G Task Illustrations and Execution Examples

Complementing the policy analyses in Appendices E and F, this appendix provides visual context for task settings and recorded executions. Simulation scenes, real-robot GTP-FA+RAISE sequences, and recorded baseline outcomes retain their distinct sources. These examples illustrate execution structure rather than provide additional performance estimates.

G.1 Simulation Task Overview

Figure 12 shows the ten task scenes and goals. Seven panels use task-animation frames; PegInsertionSide, PlaceApple, and TransportBox use expert-rollout base-camera observations. The gallery is not a common policy-evaluation cohort and does not specify training resolution or demonstrate learned-policy success.

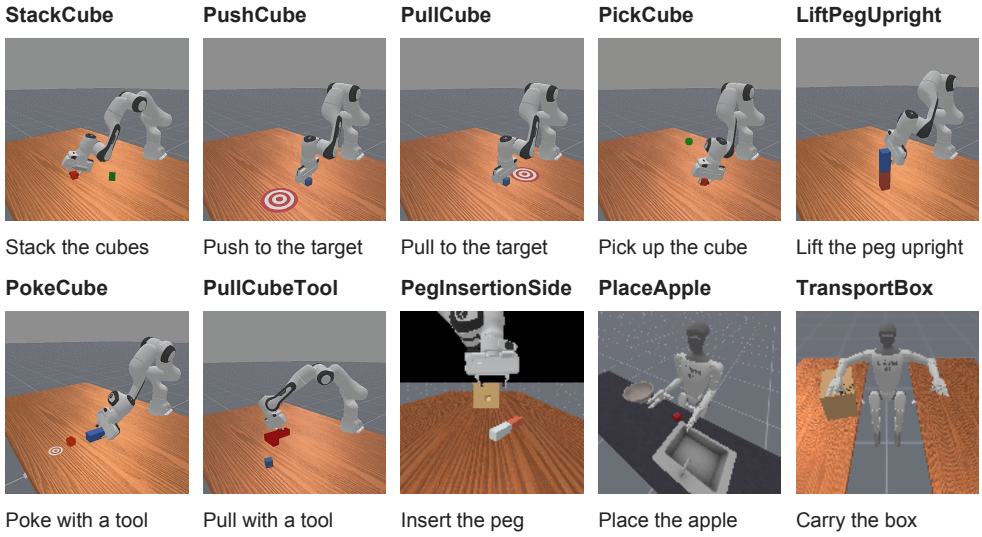

Figure 12: Ten simulation tasks and goals. Task animations supply seven panels; PegInsertionSide, PlaceApple, and TransportBox use expert base-camera observations. Scenes illustrate settings, not a common policy-evaluation cohort.

G.2 Real-Robot Execution Sequences

Figures 13 and 14 illustrate the fixed grasp-conditioned $\pi_{0.5}$ system with RAISE applied to downstream training. Each task shows four chronological moments from one video; full views retain geometry and fixed same-frame crops expose object and gripper details. Crops are display aids, not extra views or policy inputs. Timestamps refer to video presentation, not synchronized action indices or control latency. Column headings describe presentation, not predicted stages or annotated event boundaries. PourWater transfers small balls; its 90% completion criterion is not inferred from sparse frames. These are illustrative sequences, not paired baseline–RAISE trials, success-rate estimates, or independent Stable Success@3 checks.

G.3 Recorded Baseline Success and Failure

Figure 15 retains two G00 V0 executions: seed-42 training on 300 clean successes, final update 30,000, and a 250-step horizon. The upper example fails both metrics; the lower satisfies both. Labels come from complete evaluation records, not selected frames; the intermediate states do not identify failure onset or first success. Different initialization seeds make these independent examples, not a V0–RAISE pair or new evaluations. They illustrate baseline variability without attributing its cause to reweighting.

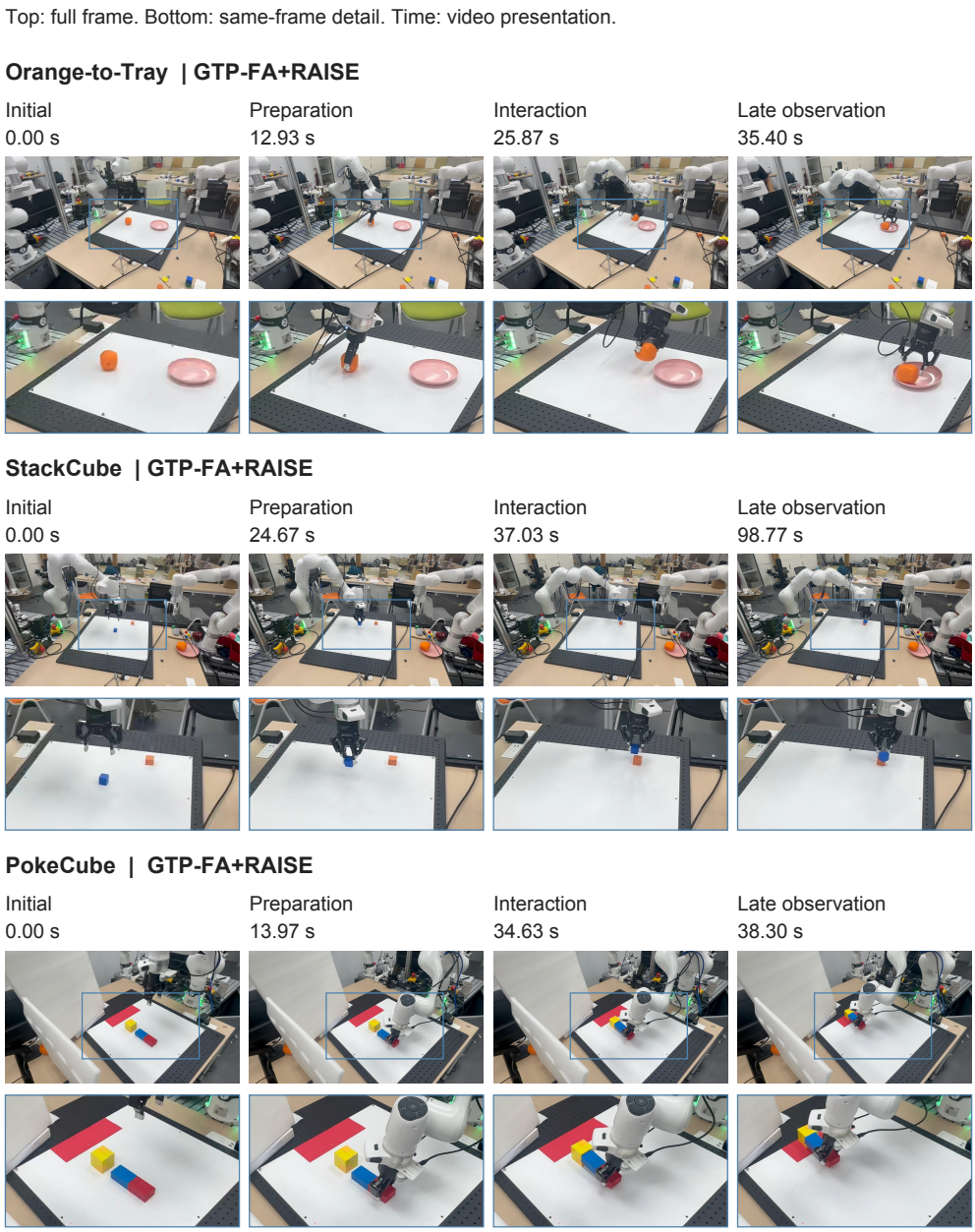

Figure 13: GTP-FA+RAISE on Orange-to-Tray, StackCube, and PokeCube. Each task shows four chronological video moments: full view above, fixed same-frame detail below. Times are video-presentation timestamps; headings are descriptive, not predicted stages.

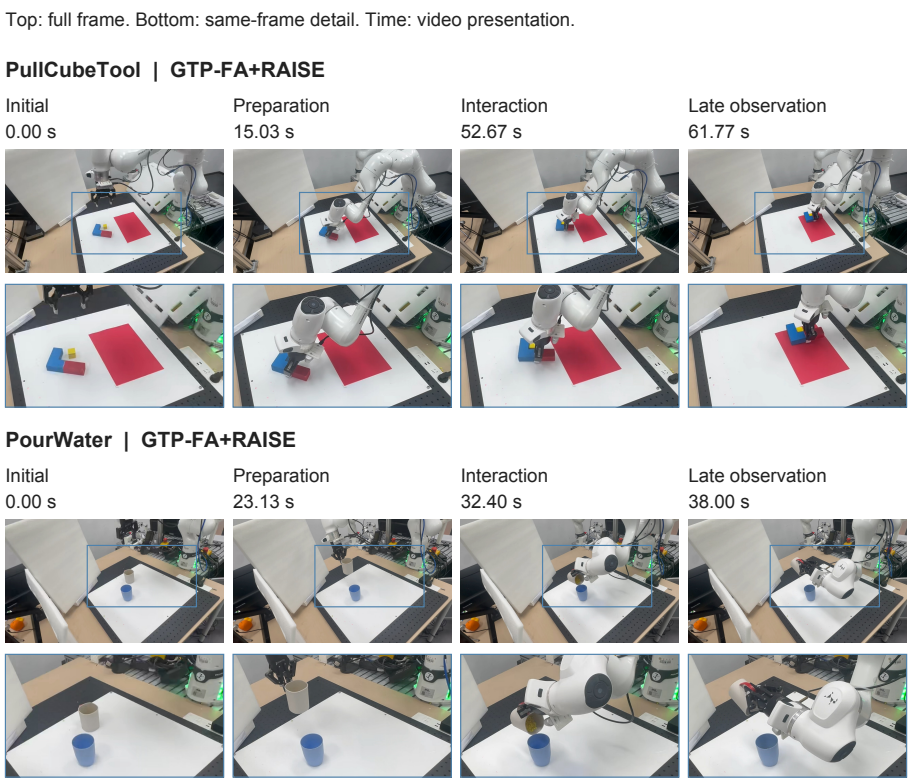

Figure 14: GTP-FA+RAISE on PullCubeTool and PourWater (small-ball transfer). Views and timestamps follow Fig. 13.

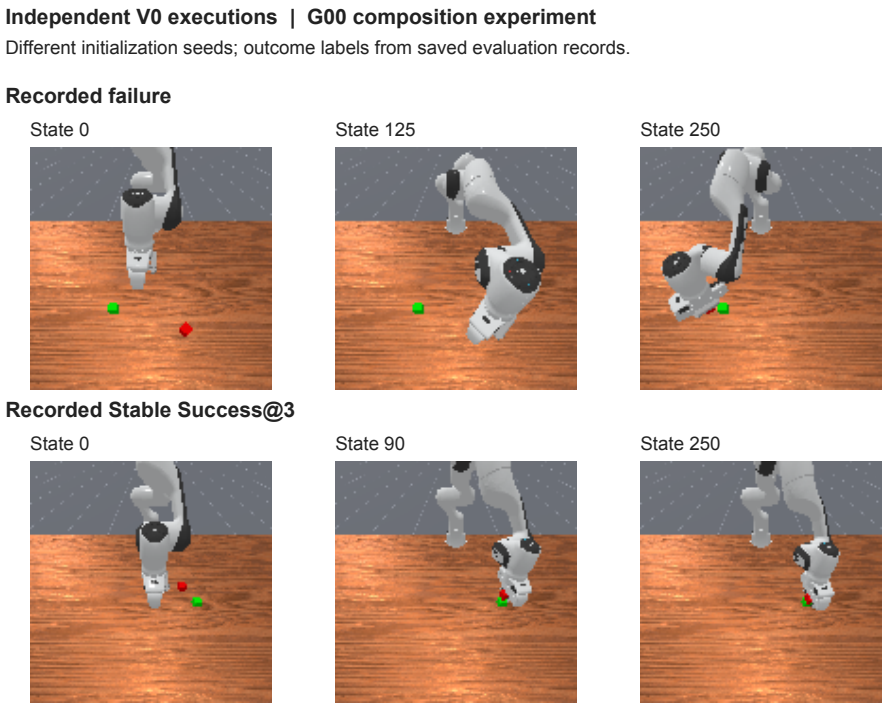

Figure 15: G00 V0 examples: failure (top) and stable success (bottom). Complete records determine outcomes; initializations differ.