R^2-WAM: REPAIR-AND-REJECT POST-TRAINING FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) emerge as a promising foundation for policy refinement by predicting the consequences of sampled actions. However, visually plausible predictions can mislead policy refinement if they fail to reflect the input actions. To address this mismatch, we introduce R^2-WAM, a two-stage repair-and-reject post-training framework that first improves the consistency of predicted futures with input actions, then uses these futures to select inferior action samples for negative fine-tuning. The repair stage grounds imagination in observed robot behavior through a kinematic alignment score that measures agreement between predicted and demonstrated motion, enabling the predicted video to faithfully reflect its input actions. Using the repaired video model, the rejection stage compares imagined outcomes of sampled and demonstrated actions, selectively applying negative fine-tuning to samples whose predicted task progress falls below the demonstrated reference by a prescribed margin. Together, the two stages extend video prediction from representation learning to consequence-based policy refinement without additional environment interaction or changes to the inference procedure. R^2-WAM achieves 93.8% average success on RoboTwin 2.0 across clean and randomized settings. On the long-horizon real-world Fold Shirt task, it achieves 87.5% average success, compared with 0% for Fast-WAM.

1 INTRODUCTION

World Action Models (WAMs) have recently emerged as a promising approach to robotic control by guiding action generation with predicted visual outcomes. Existing WAMs typically learn to predict video and actions jointly in pixel or latent space (Ye et al., 2026b; Zhu et al., 2025; Bi et al., 2026; Li et al., 2026; Yuan et al., 2026), thereby grounding action generation in predicted scene dynamics. Beyond direct robot control, recent studies have explored using action-conditioned world models as simulators to predict the consequences of candidate actions and provide feedback for policy improvement (Guo et al., 2026b; Yang et al., 2026a). This is particularly useful when demonstrations provide limited coverage of execution errors and collecting physical trials is costly (Chandra et al., 2025; Jiang et al., 2025; Xiao et al., 2025; Jiang et al., 2026).

A key prerequisite for effective policy post-training with a world model is that it faithfully predicts how the environment changes in response to actions. Although modern video generation models can produce visually plausible predictions (Wan et al., 2025), they do not necessarily preserve physical properties such as kinematics and dynamics. Consequently, generated videos may appear realistic while failing to reflect the supplied actions (Yang et al., 2026b). Evaluating actions with such predictions can mistake video model errors for policy failures, causing the policy to learn from outcomes its actions would not produce(Jiang et al., 2026; Chen et al., 2026).

To reduce mismatch between imagined and actual consequences, prior methods ground generated videos in robot motion using inverse dynamics (Wang et al., 2026) or extracted 3D flow (Dharmarajan et al., 2025). These methods rely on auxiliary components to obtain motion feedback, but do not ensure that the video faithfully follows a supplied action. Recent methods improve imagined feedback by fine-tuning world models on real policy rollouts (Guo et al., 2026a), updating the model as the policy evolves from task-critical keyframes (Jiang et al., 2026), or training on diverse actions with supervision (Chen et al., 2026). These methods improve action grounding, but leave

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.

open how observed demonstration outcomes can identify unreliable predictions before they are used to evaluate sampled policy actions.

To fill this gap, we propose repairing the video expert using observed consequences of demonstrated actions, then using the repaired expert to reject sampled actions with inferior predicted task progress (Figure 1). To this end, we introduce R^2-WAM, a two-stage repair-and-reject post-training framework comprising predictive repair and action rejection (Figure 2), comprising the following modules:

1) *Predictive repair* turns each demonstration into a test of action fidelity: given the same observation and action, the video expert generates a future that can be compared with what the robot actually did. A kinematic alignment score measures whether the generated robot motion follows the observed motion and selects predictions with substantial discrepancies for negative fine-tuning. This stage improves the video expert before its imagined consequences are used to judge sampled policy actions, reducing the risk that prediction errors become misleading training signals.

2) *Action rejection* uses the repaired video expert to judge what sampled actions are likely to achieve. With the expert frozen, we predict futures for sampled and demonstrated actions from the same initial observation, then compare their task progress using STEAM (Liu et al., 2026). Sampled actions whose predicted progress falls below the demonstrated reference by a prescribed margin are selected for negative fine-tuning, while demonstrations remain the positive targets. This directs policy updates toward actions with evidence of inferior consequences.

Together, the two stages turn video prediction into feedback for policy post-training without additional environment interaction or changes to the inference procedure. We conduct an extensive empirical study across 50 RoboTwin 2.0 tasks and two real-world manipulation tasks, evaluating action fidelity, negative-selection accuracy, and downstream policy success. R^2-WAM improves average success over action-conditioned Fast-WAM by 3.3 percentage points in simulation and raises average success on the long-horizon real-world Fold Shirt task from 0% with Fast-WAM to 87.5%.

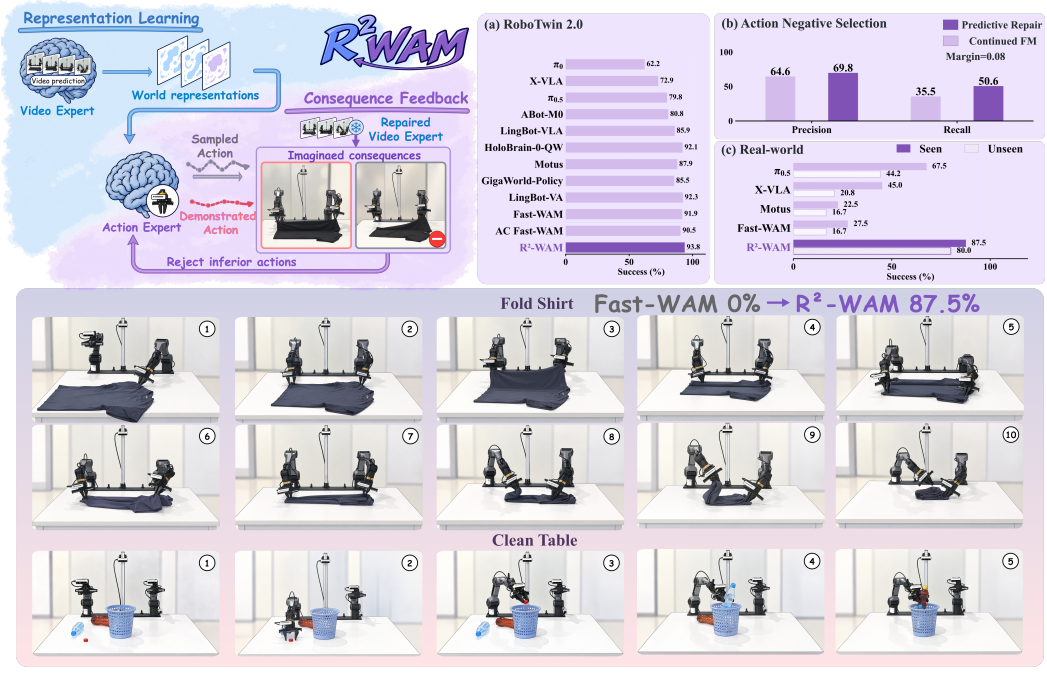

Figure 1: R^2-WAM extends video prediction from representation learning to consequence feedback for policy post-training. (a) Average success on RoboTwin 2.0. (b) Action negative-selection precision and recall against execution-based STEAM labels. (c) Real-world success under seen and unseen conditions. Bottom: execution sequences on Fold Shirt and Clean Table.

2 Related Work

World action models. World action models couple future prediction with action generation through joint diffusion modeling of video and actions (Ye et al., 2026b; Zhu et al., 2025), causal modeling (Li et al., 2026), and latent action representations (Bi et al., 2026). WorldScape Policy 2.0 (Su et al., 2026) extends this framework with memory and multimodal conditioning. Predictions can also guide control through visual representations (Hu et al., 2024) or explicit motion plans: RoboFlow4D (Lin et al., 2026a) predicts multi-frame 3D motion to condition action generation. Fast-WAM (Yuan et al., 2026) couples video and action experts through shared attention, using video prediction to learn world representations without generating future videos at deployment. We build on Fast-WAM to use predicted consequences for policy post-training.

Policy post-training. RECAP (Intelligence et al., 2025a) refines policies through advantage-conditioned learning from experience. World-model-based approaches use synthetic rollouts and predicted consequences for policy fine-tuning (Guo et al., 2026b), advantage estimation (Yang et al., 2026a), and preference optimization (Hung et al., 2025). To improve the reliability of imagined feedback, VLAW (Guo et al., 2026a) adapts its world model using real rollouts, while WoVR (Jiang et al., 2026) regulates imagined rollouts and co-evolves the world model and policy. Our approach uses observed demonstration futures to repair action-inconsistent predictions without additional environment interaction. The repaired video expert then guides selective negative fine-tuning of actions whose predicted progress falls below the demonstrated reference.

3 Preliminaries

World action model. To leverage world representations for action generation without explicit future video generation at inference, R^2-WAM adopts the shared attention world action model architecture of Fast-WAM (Yuan et al., 2026). Each expert demonstration comprises $x_i = (o_i, s_i, \ell_i)$, consisting of a current observation, robot state, and language instruction, together with an expert action chunk $a_i^\star$ and its observed future video $y_i^\star$.

The architecture comprises a video expert and an action expert, both implemented as diffusion transformers. The video stream contains the latent of the current frame and noised future video latents, whereas the action stream contains noised action chunks. Within each MoT block, the two streams interact through shared masked self-attention, followed by cross-attention to state and language representations. In our configuration, the video expert also receives the action chunk. The video expert defines the action-conditioned video model $W_\phi(y \mid x, a)$, and the action expert defines $\pi_\theta(a \mid x)$. The video expert therefore supplies world representations for action generation and maps candidate actions to imagined consequences during post-training.

Flow matching initialization. We jointly train the video and action experts on demonstrations using flow matching. Let z_0 denote an observed future video latent or a demonstrated action chunk. For $t \in [0, 1]$ sampled from the training schedule and $\epsilon \sim \mathcal{N}(0, I)$, we construct $z_t = (1-t)z_0 + t\epsilon$ with target velocity $v^\star = \epsilon - z_0$. The objective for expert $b \in \{\mathrm{v}, \mathrm{a}\}$ is

$$\ell^b(z_0) = \mathbb{E}_{t,\epsilon}\left[\|v_{\psi_b}(z_t, t) - v^\star\|_2^2\right], \tag{1}$$

where $\psi_\mathrm{v} = \phi$ and $\psi_\mathrm{a} = \theta$. Conditioning inputs are omitted for brevity. This objective fits demonstration endpoints but does not compare them with endpoints sampled from the current model. A generated future may therefore fail to realize its conditioning action, and a sampled action may produce an inferior imagined consequence, without receiving a targeted corrective signal.

4 Consequence-Grounded Post-Training

R^2-WAM refines a world action model through predictive repair followed by action rejection (Figure 2). Predictive repair uses observed futures to identify and suppress action-inconsistent video predictions (Section 4.1). With the repaired video expert frozen, action rejection compares the imagined progress of sampled and demonstrated actions to select inferior action samples for negative fine-tuning (Section 4.2).

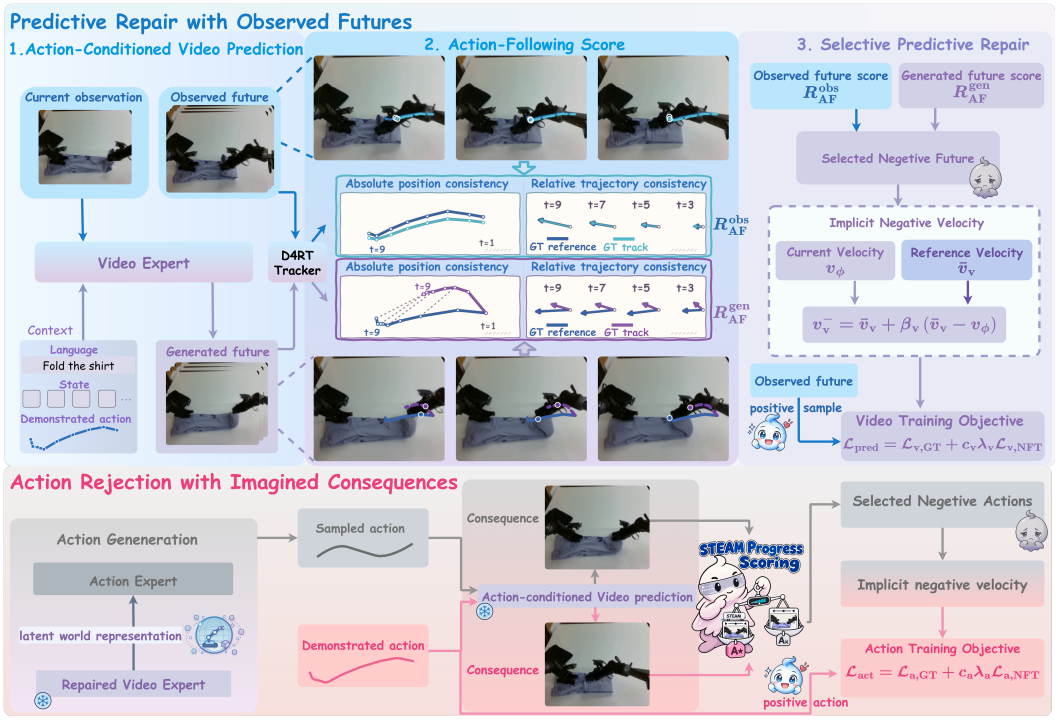

Figure 2: Overview of R^2-WAM. Predictive repair compares generated robot motion with observed motion under the same demonstrated action. The repaired video expert is then frozen and predicts consequences for selecting inferior action samples.

4.1 Predictive Repair with Observed Futures

Predictive repair uses the observed consequences of demonstrated actions to identify predictions that fail to follow their conditioning actions. For each demonstration $(x_i, a_i^\star, y_i^\star)$, the observed future $y_i^\star$ provides a reference for K_v futures sampled under the same action and initial observation:

$$\hat{y}_{i,k} \sim W_\phi(\cdot \mid x_i, a_i^\star), \qquad k = 1, \ldots, K_\mathrm{v}. \tag{2}$$

We assess action consistency in both observed and generated futures through their agreement with the recorded robot motion.

Action-following score. Robot kinematics maps the demonstrated proprioceptive trajectory to reference landmark positions $p_{i,j,f}$ in the camera frame, where j indexes landmarks and f indexes video frames. For a video y, a frozen point tracker estimates landmark image locations and visibility probabilities $q_{i,j,f}(y) \in [0, 1]$. We obtain the tracked positions $\hat{p}_{i,j,f}(y)$ in the same camera frame by lifting the pixels using the kinematic reference depth and known camera intrinsics. To restrict motion comparisons to reliable tracks, we evaluate visibility over $\mathcal{F}_{i,j}$, the frames with an in-image reference projection after the first such frame $f_{i,j}^{\mathrm{init}}$. For nonempty $\mathcal{F}_{i,j}$, trajectory reliability is

$$\rho_{i,j}(y) = \frac{1}{|\mathcal{F}_{i,j}|} \sum_{f \in \mathcal{F}_{i,j}} q_{i,j,f}(y). \tag{3}$$

We introduce a video-level trackability indicator $C(y)$ to indicate whether a video provides reliable landmark trajectories for motion comparison. It equals 1 when at least one landmark trajectory has a nonempty comparison frame set $\mathcal{F}_{i,j}$ and mean visibility $\rho_{i,j}(y) \geq 0.5$, and 0 otherwise.

We introduce two measures of action consistency: absolute-position consistency evaluates agreement with the reference landmark positions, while relative-trajectory consistency evaluates agreement with the reference displacements. Displacements from $f_{i,j}^{\mathrm{init}}$ are

$$\Delta p_{i,j,f} = p_{i,j,f} - p_{i,j,f_{i,j}^{\mathrm{init}}}, \qquad \Delta \hat{p}_{i,j,f}(y) = \hat{p}_{i,j,f}(y) - \hat{p}_{i,j,f_{i,j}^{\mathrm{init}}}(y).$$

The corresponding position and displacement errors are

$$e^{\mathrm{abs}}_{i,j,f}(y) = \|\hat{p}_{i,j,f}(y) - p_{i,j,f}\|_2, \qquad e^{\mathrm{rel}}_{i,j,f}(y) = \|\Delta\hat{p}_{i,j,f}(y) - \Delta p_{i,j,f}\|_2.$$

For each component $g \in \{\mathrm{abs}, \mathrm{rel}\}$, the trajectory score $S^g_{i,j}(y)$ averages Gaussian alignment scores over frames using visibility weights, with σ_g setting the spatial tolerance. For generated futures, $\bar{S}^g_i(y)$ denotes the reliability-weighted mean of these scores over landmarks:

$$S^g_{i,j}(y) = \frac{\sum_{f\in\mathcal{F}_{i,j}} q_{i,j,f}(y) \exp\left[-\frac{(e^g_{i,j,f}(y))^2}{2\sigma_g^2}\right]}{\sum_{f\in\mathcal{F}_{i,j}} q_{i,j,f}(y)}, \quad \bar{S}^g_i(y) = \frac{\sum_{j:\rho_{i,j}(y)\geq 0.5} \rho_{i,j}(y) S^g_{i,j}(y)}{\sum_{j:\rho_{i,j}(y)\geq 0.5} \rho_{i,j}(y)}. \tag{4}$$

For generated futures, the action-following score $R^{\mathrm{gen}}_{\mathrm{AF}}(y, a^\star_i)$ averages the two aggregate scores. The observed-reference score $R^{\mathrm{obs}}_{\mathrm{AF}}(y^\star_i, a^\star_i)$ instead uses selected reliable landmarks to reduce the influence of poor tracks. Let $j^{\max}_{\mathrm{abs}}$ be the reliable landmark with the highest $\rho_{i,j}(y^\star_i)$, and $j^{\max}_{\mathrm{rel}}$ the reliable landmark with the highest $S^{\mathrm{rel}}_{i,j}(y^\star_i)$. Then

$$R^{\mathrm{gen}}_{\mathrm{AF}}(y, a^\star_i) = \big(\bar{S}^{\mathrm{abs}}_i(y) + \bar{S}^{\mathrm{rel}}_i(y)\big)/2, \quad R^{\mathrm{obs}}_{\mathrm{AF}}(y^\star_i, a^\star_i) = \big(S^{\mathrm{abs}}_{i,j^{\max}_{\mathrm{abs}}}(y^\star_i) + S^{\mathrm{rel}}_{i,j^{\max}_{\mathrm{rel}}}(y^\star_i)\big)/2. \tag{5}$$

Aggregating generated trajectories prevents one consistent landmark from masking inconsistent motion in other reliable tracks. These scores are evaluated only when $C(y) = 1$. Appendix A.1 specifies the landmarks, tracking model, and coordinate lifting.

Observed-reference selection. When both futures are trackable, their action-following score gap is

$$\Delta R_{i,k} = R^{\mathrm{obs}}_{\mathrm{AF}}(y^\star_i, a^\star_i) - R^{\mathrm{gen}}_{\mathrm{AF}}(\hat{y}_{i,k}, a^\star_i). \tag{6}$$

We assign each generated future a negative fine-tuning weight $w^{\mathrm{v}}_{i,k} \in [0, 1]$ based on the observed reference:

$$w^{\mathrm{v}}_{i,k} = \begin{cases} 0, & C(y^\star_i) = 0, \\ 1, & C(y^\star_i) = 1,\ C(\hat{y}_{i,k}) = 0, \\ \mathrm{clip}\big((\Delta R_{i,k} - m_{\mathrm{v}})/\tau_{\mathrm{v}}, 0, 1\big), & C(y^\star_i) = C(\hat{y}_{i,k}) = 1. \end{cases} \tag{7}$$

Here m_{v} is the minimum score gap for selecting a trackable prediction, and $\tau_{\mathrm{v}} > 0$ controls how its weight increases beyond the margin. A reliable observed reference is required for negative fine-tuning of generated futures.

Negative fine-tuning. Both stages use the same objective for expert $b \in \{\mathrm{v}, \mathrm{a}\}$. Let $z^\star_i$ and $\hat{z}_{i,k}$ denote its demonstration and sampled endpoints: video latents for $b = \mathrm{v}$ and action chunks for $b = \mathrm{a}$. Demonstration supervision is retained through the flow-matching objective in Equation 1, $\mathcal{L}_{b,\mathrm{GT}} = \frac{1}{N}\sum_{i=1}^{N} \ell^b(z^\star_i)$, where N is the number of demonstrations in the batch. For generated samples selected as negatives, we adopt the implicit negative parameterization of DiffusionNFT (Zheng et al., 2026b):

$$v^-_b(z_t, t) = (1 + \beta_b)\bar{v}_b(z_t, t) - \beta_b v_{\psi_b}(z_t, t), \tag{8}$$

where $\bar{v}_b$ is a frozen copy of the corresponding velocity predictor at the start of its post-training stage, $\beta_b > 0$ controls the negative parameterization, and $\psi_{\mathrm{v}} = \phi$, $\psi_{\mathrm{a}} = \theta$. Negative fine-tuning regresses the implicit velocity v^-_b toward the flow targets of selected generated samples, weighted by $w^b_{i,k}$:

$$\mathcal{L}_{b,\mathrm{NFT}} = \frac{\sum_{i,k} w^b_{i,k} \mathbb{E}_{t,\epsilon}\left[\left\| v^-_b(\hat{z}_{i,k,t}, t) - v^\star_{i,k} \right\|_2^2\right]}{\sum_{i,k} w^b_{i,k} + \delta}. \tag{9}$$

Here $\hat{z}_{i,k,t} = (1-t)\hat{z}_{i,k} + t\epsilon$ and $v^\star_{i,k} = \epsilon - \hat{z}_{i,k}$ follow the flow-matching construction in Equation 1, and $\delta > 0$ ensures a well-defined loss when no negatives are selected. The resulting post-training objective is

$$\mathcal{L}_b = \mathcal{L}_{b,\mathrm{GT}} + c_b \lambda_b \mathcal{L}_{b,\mathrm{NFT}}. \tag{10}$$

Here $\lambda_b > 0$ weights the negative loss, and $c_b \in [0, 1]$ limits its contribution to a prescribed fraction of the demonstration loss. For predictive repair, the endpoints are observed and generated video latents, with weights $w^{\mathrm{v}}_{i,k}$ from Equation 7. We minimize $\mathcal{L}_{\mathrm{v}}$ with respect to ϕ, keeping the action expert fixed. Appendix A.3 specifies the cap and optimization details.

4.2 Action Rejection with Imagined Consequences

Video representations and imagined consequences. Unlike demonstrated actions, newly sampled actions have no recorded consequences. We therefore use the frozen, repaired video expert $W_{\phi^\star}$ to evaluate their imagined outcomes while retaining its role in providing world representations for action generation. We sample K_{a} candidate actions from the current policy:

$$\hat{a}_{i,k} \sim \pi_\theta(\cdot \mid x_i), \qquad k = 1, \ldots, K_{\mathrm{a}}. \tag{11}$$

Let $\tilde{y}_i^\star$ and $\tilde{y}_{i,k}$ denote the futures predicted by the repaired video expert under the demonstrated action $a_i^\star$ and sampled action $\hat{a}_{i,k}$, respectively, from the same initial observation:

$$\tilde{y}_i^\star \sim W_{\phi^\star}(\cdot \mid x_i, a_i^\star), \qquad \tilde{y}_{i,k} \sim W_{\phi^\star}(\cdot \mid x_i, \hat{a}_{i,k}). \tag{12}$$

Consequence-based action selection. We evaluate task progress in predicted futures using a frozen temporal progress model following STEAM (Liu et al., 2026). Its score $A_{\mathrm{STEAM}}(x_i, y)$ measures progress from the current observation to the final frame of y, conditioned on the language instruction. We define the progress gap $\Delta A_{i,k}$ as the difference between the predicted progress of the demonstrated and sampled actions:

$$\Delta A_{i,k} = A_{\mathrm{STEAM}}(x_i, \tilde{y}_i^\star) - A_{\mathrm{STEAM}}(x_i, \tilde{y}_{i,k}). \tag{13}$$

A positive $\Delta A_{i,k}$ indicates lower predicted progress for the sampled action. Actions whose predicted progress falls sufficiently below the demonstrated reference receive negative fine-tuning weights $w_{i,k}^{\mathrm{a}} \in [0, 1]$:

$$w_{i,k}^{\mathrm{a}} = \mathrm{clip}\big((\Delta A_{i,k} - m_{\mathrm{a}})/\tau_{\mathrm{a}}, 0, 1\big). \tag{14}$$

Here m_{a} is the rejection margin, and $\tau_{\mathrm{a}} > 0$ controls the increase in weight beyond it. Appendix B.2 details the scorer training.

Rejection-only action fine-tuning. We apply Equation 10 with $b = \mathrm{a}$, using demonstrated actions $a_i^\star$ as positive endpoints and sampled actions $\hat{a}_{i,k}$ as negative candidates weighted by $w_{i,k}^{\mathrm{a}}$ from Equation 14. The frozen reference $\bar{v}_{\mathrm{a}}$ is initialized from the action expert at the start of this stage. We optimize θ while keeping the repaired video expert and STEAM scorer fixed.

5 Experiments

Our experiments address five questions. **(Q1) Simulation performance:** how does R^2-WAM compare with recent vision language action policies and world action models on RoboTwin 2.0 (Chen et al., 2025)? **(Q2) Negative selection:** does predictive repair improve the agreement between negative selections based on imagined consequences and STEAM judgments of actual executions? **(Q3) Ablation studies:** which post-training components and action training objectives account for the gains in task success? **(Q4) Real-world effectiveness:** does R^2-WAM improve task completion in long-horizon real-world manipulation? **(Q5) Behavioral analysis:** do imagined consequences reveal the lack of progress observed during shirt folding?

5.1 Simulation on RoboTwin 2.0

Experimental protocol. We follow the RoboTwin 2.0 multitask protocol used by Fast-WAM. The training set combines 2,500 demonstrations collected in clean scenes and 25,000 demonstrations collected with scene randomization, spanning 50 bimanual manipulation tasks. We evaluate every task in both clean and randomized environments using 100 closed-loop trials per setting, and report success rates averaged over the 50 tasks.

Our base model is Fast-WAM (Yuan et al., 2026) with an action-conditioned video expert. We first train both experts jointly with flow matching for 5 epochs. Action-conditioned Fast-WAM continues joint flow matching for 2 additional epochs, whereas R^2-WAM applies predictive repair followed by action post-training, training each expert for 2 epochs. Appendix E.1 details the ablation protocol, and Appendix C reports the optimization and sampling settings.

Table 1: Closed-loop success rates (%) on RoboTwin 2.0.

Method	Clean	Randomized	Average
Vision language action policies			
π_0 (Black et al., 2024)	65.9	58.4	62.2
X-VLA (Zheng et al., 2026a)	72.9	72.8	72.9
$\pi_{0.5}$ (Intelligence et al., 2025b)	82.7	76.8	79.8
ABot-M0 (Yang et al., 2026c)	81.2	80.4	80.8
LingBot-VLA (Wu et al., 2026)	86.5	85.3	85.9
HoloBrain-0-QW (Lin et al., 2026b)	91.9	92.3	92.1
World action models			
Motus (Bi et al., 2026)	88.7	87.0	87.9
GigaWorld-Policy (Ye et al., 2026a)	85.6	85.3	85.5
LingBot-VA (Li et al., 2026)	92.9	91.6	92.3
Fast-WAM (Yuan et al., 2026)	91.9	91.8	91.9
Our evaluations			
Action-conditioned Fast-WAM	90.6	90.4	90.5
R^2-WAM	**93.8**	**93.7**	**93.8**

Table 2: Action negative-selection precision and recall (%) at three margins. Reference labels are obtained by applying each margin to STEAM score gaps from actual executions; predicted selections use the same margin on imagined consequences.

	Precision		Recall	
Margin	Continued FM	Repair	Continued FM	Repair
0.02	76.9	**79.6**	44.4	**63.7**
0.05	63.2	**76.8**	38.7	**56.7**
0.08	64.6	**69.8**	35.5	**50.6**

Main comparison. To answer Q1, we compare R^2-WAM with recent vision-language action policies and world action models (Table 1). R^2-WAM achieves the highest success rates in both evaluation settings, reaching 93.8% in clean environments and 93.7% under scene randomization. The gains over LingBot-VA in clean environments and HoloBrain-0-QW under randomization are 0.9 and 1.4 percentage points, respectively. Relative to action-conditioned Fast-WAM, R^2-WAM improves success by 3.2 and 3.3 percentage points in clean and randomized environments, respectively.

5.2 Action Negative Selection

To answer Q2, we test whether predictive repair improves the selection of action negatives from imagined consequences on evaluation episodes. Across 500 contexts from all 50 RoboTwin tasks, we predict consequences for 2,000 shared candidate actions using video experts trained with continued flow matching or predictive repair. Each candidate and its expert reference are also executed from the same simulator state. The same frozen STEAM evaluator scores actual and imagined consequences, providing reference labels from actual executions and predicted selections from imagined futures. These simulator executions are used only for evaluation, not for post-training. We evaluate margins of 0.02, 0.05, and 0.08, applying the same margin to actual and imagined consequence gaps in each comparison. Precision and recall are computed from pooled counts across the 2,000 candidates. Predictive repair improves negative-selection precision and recall at all three margins (Table 2). At the training margin of 0.08, precision increases from 64.6% to 69.8%, while recall rises from 35.5% to 50.6%. With candidate actions and the STEAM evaluator held fixed, the improved agreement with execution-based judgments suggests that predictive repair provides more reliable predictions for selecting action negatives. Appendix E.3 provides the protocol and complete counts.

5.3 Ablation Studies

Contributions of the two stages. To answer Q3, we first examine the contributions of predictive repair and action rejection (Table 3). All four variants train the video expert first, followed by

the action expert. GT FM denotes flow matching on demonstrations alone. Repair adds negative fine-tuning of generated videos selected against observed futures, whereas Rejection adds negative fine-tuning of sampled actions selected by their imagined progress.

Table 3: Component ablation. Success rates (%).

Video	Action	Clean	Rand.
GT FM	GT FM	90.8	90.5
GT FM	Rejection	91.1	90.7
Repair	GT FM	91.9	91.4
Repair	Rejection	**93.8**	**93.7**

Table 4: Action objectives. Success rates (%).

Variant	Clean	Rand.
All negatives	86.8	85.9
Random negatives	87.1	87.2
RECAP + STEAM	92.6	92.3
R^2-WAM (ours)	**93.8**	**93.7**

With the action objective fixed to GT FM, predictive repair increases success from 90.8% to 91.9% in clean environments and from 90.5% to 91.4% under randomization, supporting its role in learning more useful world representations for action generation. Beyond representation learning, repair also strengthens the gains from action rejection: improvements of 0.3 and 0.2 percentage points without repair increase to 1.9 and 2.3 points after repair in clean and randomized environments, respectively. Together with the improved negative-selection accuracy in Table 2, these results support predictive repair as a source of more reliable consequence feedback for negative action fine-tuning.

Action training objective. We next compare action training objectives with the repaired video expert fixed (Table 4). All negatives treats every policy sample as a negative, while Random negatives selects each sampled action independently with probability 0.5. Both retain demonstration flow matching. RECAP + STEAM (Intelligence et al., 2025a) conditions the policy on a binary advantage indicator derived from STEAM estimates.

Rejecting all samples or a random subset underperforms demonstration flow matching alone in both settings (Repair / GT FM in Table 3), showing that poorly selected negatives can degrade the policy despite retaining demonstration supervision. Consequence-based rejection instead outperforms RECAP + STEAM by 1.2 and 1.4 percentage points in clean and randomized environments, respectively. These results support selective rejection as a more effective use of predicted progress than advantage conditioning in this setting.

5.4 Real-World Evaluation

Robot setup and tasks. We evaluate on a PIPER platform with two arms, equipped with one front camera and two RGB cameras mounted on the wrists. The policy receives the three camera views and the current robot state, and predicts bimanual end-effector action chunks. We evaluate two long-horizon tasks covering deformable manipulation and tabletop cleanup. In Fold Shirt, the robot transforms a shirt into a rectangular configuration through an extended sequence of coordinated bimanual actions. In Clean Table, it repeatedly selects, grasps, and places bottles and bottle caps from the tabletop into a trash bin. Hardware, data, and deployment details appear in Appendix F.1.

Baselines. We compare against 4 baselines: $\pi_{0.5}$ (Intelligence et al., 2025b), X-VLA (Zheng et al., 2026a), Motus (Bi et al., 2026), and Fast-WAM (Yuan et al., 2026). Each baseline is trained for 7 epochs on the same demonstrations for each task, using its original architecture, observation processing, and action representation. We initialize R^2-WAM from Fast-WAM after 5 epochs and train each expert for 2 additional epochs using its corresponding post-training objective.

Evaluation protocol. We organize evaluation as a 2×2 design over initial position and object instance. Seen uses the training position region and familiar objects. Unseen Position and Unseen Object vary one factor at a time, while Unseen Position + Object varies both. Each condition contains 20 autonomous trials per method and task. Each trained policy is evaluated across all 4 conditions. Figure 3(a–b) reports success for each task; complete numerical results, Wilson confidence intervals, and the complete protocol are provided in Appendices F.3 and F.4.

Real-world performance. To answer Q4, we compare task success on Fold Shirt and Clean Table (Figure 3(a–b)). On Fold Shirt, a long-horizon bimanual task, R^2-WAM achieves 87.5% average

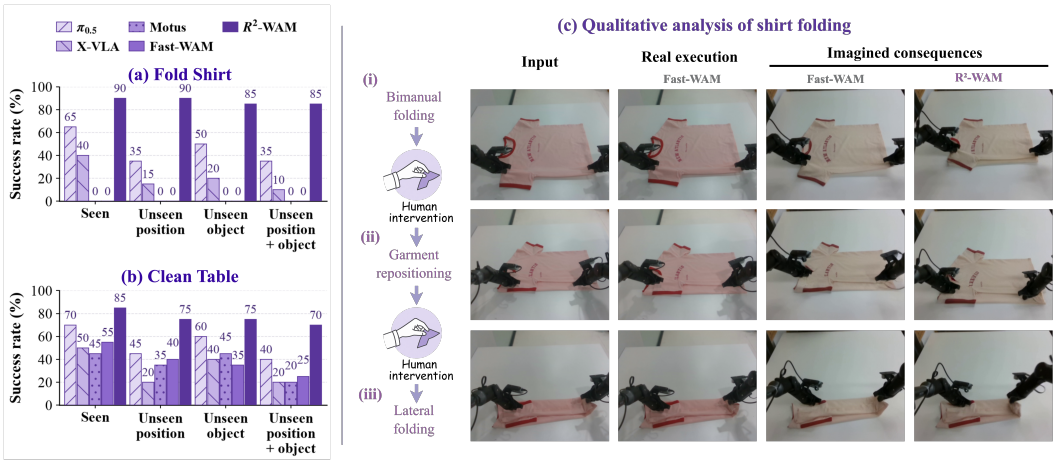

Figure 3: Real-world evaluation and qualitative analysis. (a–b) Success rates on Fold Shirt and Clean Table. (c) Fast-WAM executions and imagined consequences of actions from Fast-WAM and R^2-WAM during (i) bimanual folding, (ii) garment repositioning, and (iii) lateral folding. Human intervention in (c) enables later-stage analysis; these rollouts are excluded from success rates.

success, whereas neither Fast-WAM nor Motus completes the task in any evaluation trial. The VLA baselines, $\pi_{0.5}$ and X-VLA, achieve higher success than these WAM baselines but remain below R^2-WAM across all four evaluation conditions. On Clean Table, R^2-WAM also outperforms all evaluated baselines across the four conditions. Under joint variation, it achieves 70.0% success, exceeding the strongest baseline, $\pi_{0.5}$, by 30 percentage points.

Qualitative analysis of shirt folding. To answer Q5, we compare Fast-WAM executions with imagined consequences at three stages of shirt folding (Figure 3(c)). At each stage, Fast-WAM establishes a grasp or contact but does not complete the subsequent garment motion. During bimanual folding, it lifts the garment without folding it forward. During repositioning, it contacts the garment without pulling it toward the robot. During lateral folding, it holds and lifts a corner without completing the fold.

For each initial observation, we use the same repaired video expert to predict the consequences of actions sampled from Fast-WAM and R^2-WAM. The predictions for Fast-WAM actions show a similar lack of progress to the displayed executions, whereas those for R^2-WAM depict forward folding, garment pulling, and lateral folding, respectively. These examples illustrate the stalled behaviors that consequence feedback is intended to correct: grasping or lifting the garment without advancing the folding sequence. Their appearance in the predicted futures suggests that the repaired video expert can help identify such actions for rejection during post-training, with the aim of improving completion of the full task.

6 Conclusion

We presented R^2-WAM, a repair-and-reject framework for post-training world action models. Predictive repair uses observed demonstration futures to improve the action fidelity of video predictions. The repaired video expert then guides negative fine-tuning of sampled actions whose predicted progress falls below the demonstrated reference. Experiments on RoboTwin 2.0 and two real-world manipulation tasks show improved task completion, including long-horizon shirt folding that Fast-WAM fails to complete. Further analysis shows that predictive repair improves negative-selection accuracy, while action rejection yields larger gains in task success with the repaired video expert.

Reproducibility Statement

Details of predictive repair and action rejection are provided in Appendices A and B. Appendix C specifies model initialization, optimization, sampling, and post-training hyperparameters. Simulation protocols, ablation settings, and negative-selection evaluation are described in Appendix E. Appendix F documents the robot platform, demonstration data, deployment settings, evaluation conditions, and statistical reporting for the real-world experiments.

AI Use Statement

We used generative AI tools to improve the clarity, grammar, and readability of the manuscript. The authors reviewed and revised the AI-assisted text and take full responsibility for the final manuscript.

References

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35101–35113, June 2026.

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.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. DiWA: Diffusion policy adaptation with world models. In *Conference on Robot Learning, CoRL*, volume 305, pp. 3378–3400, 2025.

Sixiang Chen, Jiaming Liu, Jixian Wu, Yichen Guo, Tinghao Wang, Siyuan Qian, Hao Chen, Jiajun Cao, Jian Tang, and Shanghang Zhang. Do robotic world models really follow actions? diagnosing and aligning action-conditioned generation for policy learning. *arXiv preprint arXiv:2608.24885*, 2026.

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

Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. Dream2flow: Bridging video generation and open-world manipulation with 3d object flow. *arXiv preprint arXiv:2512.24766*, 2025.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026a.

Yanjiang Guo, Lucy Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 6121–6138, 2026b.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

Chia-Yu Hung, Navonil Majumder, Haoyuan Deng, Liu Renhang, Yankang Ang, Amir Zadeh, Chuan Li, Dorien Herremans, Ziwei Wang, and Soujanya Poria. Nora-1.5: A vision-language-action model trained using world model-and action-based preference rewards. *arXiv preprint arXiv:2511.14659*, 2025.

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

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Zhennan Jiang, Kai Liu, Yuxin Qin, Shuai Tian, Yupeng Zheng, Mingcai Zhou, Chao Yu, Haoran Li, and Dongbin Zhao. World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation, 2025.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, et al. Wovr: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*, 2026.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Sixu Lin, Junliang Chen, Huaiyuan Xu, Zhuohao Li, Guangming Wang, Yixiong Jing, Sheng Xu, Runyi Zhao, Brian Sheil, Lap-Pui Chau, et al. Roboflow4d: A lightweight flow world model toward real-time flow-guided robotic manipulation. *arXiv preprint arXiv:2605.17522*, 2026a.

Xuewu Lin, Tianwei Lin, Yun Du, Hongyu Xie, Yiwei Jin, Jiawei Li, Shijie Wu, Qingze Wang, Mengdi Li, Mengao Zhao, et al. Holobrain-0 technical report. *arXiv preprint arXiv:2602.12062*, 2026b.

Zhihao Liu, Qiuyi Gu, Yitao Wang, Dongming Qiao, Yixian Zhang, Shuaihang Chen, Liangzhi Shi, Tianxing Zhou, Zefang Huang, Kang Chen, et al. Steam: Self-supervised temporal ensemble advantage modeling for real-world robot learning. *arXiv preprint arXiv:2606.29834*, 2026.

Haisheng Su, Zongdai Liu, Xin Jin, Haoxuan Dou, Chengming Hu, Baorun Li, Zhanwang Liu, Ruiyan Xu, Jianjie Fang, Xin Zhang, et al. Worldscape policy 2.0: Empowering steerable world action modeling with reasoning-augmented memory. *arXiv preprint arXiv:2607.18840*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Ruixiang Wang, Qingming Liu, Yueci Deng, Guiliang Liu, Zhen Liu, and Kui Jia. Eva: Aligning video world models with executable robot actions via inverse dynamics rewards. *arXiv preprint arXiv:2603.17808*, 2026.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Shuailei Ma, He Sun, Yong Wang, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Shuai Zhou, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Qian Zhu, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692v1*, 2026.

Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-env: Leveraging world model as a virtual environment for vla post-training, 2025.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026a.

Tianzhuo Yang, Zihan Shen, Zirui Mi, Zhaoyi Zhang, Jiayi Zhou, Jiaming Ji, Juntao Dai, Jiawei Chen, Boyuan Chen, and Yaodong Yang. Mirabench: Evaluating action-conditioned reliability in robotic world models. *arXiv preprint arXiv:2605.29360*, 2026b.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026c.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K Barral, Raia Hadsell, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7382–7392, 2026.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026a.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026b.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

A Details of Predictive Repair

A.1 Action-Following Score

Section 4.1 defines the geometric scores, tracking reliability, and aggregation rules. Here we specify how the tracked image locations are expressed in the reference camera coordinates. Let $\hat{\mathbf{u}}_{i,j,f}(y)$ be the tracked pixel location of landmark j in frame f, $d_{i,j,f}$ its depth from the demonstrated robot kinematics, and $\boldsymbol{K}$ the fixed camera intrinsic matrix. We use

$$\hat{p}_{i,j,f}(y) = d_{i,j,f}\boldsymbol{K}^{-1}\begin{bmatrix}\hat{\mathbf{u}}_{i,j,f}(y)\\ 1\end{bmatrix}. \tag{15}$$

Observed and generated videos use the same reference depths and camera coordinates. The Gaussian bandwidths in Equation 4 are $\sigma_{\mathrm{abs}} = \sigma_{\mathrm{rel}} = 0.05\,\mathrm{m}$.

Tracking model and landmarks. We use OpenD4RT,[1] an unofficial implementation of D4RT (Zhang et al., 2026), as a frozen point tracker. It processes the head-camera view of each 9-frame clip, comprising the current frame and 8 future frames, at 256×256 resolution; the video expert receives all three views.

For RoboTwin, we place three virtual landmarks in the local coordinate frame of link 6 on each arm:

$$(0,\,0,\,0), \qquad (0.065,\,0.052,\,0), \qquad (0.065,\,-0.052,\,0).$$

All coordinates are in meters, giving six landmarks across the two arms. OpenD4RT returns image trajectories and visibility logits; their sigmoid probabilities provide $q_{i,j,f}$ in Equations 3 and 4.

A.2 Observed-Reference Negative Selection

Equation 7 specifies the selection and weighting rule, including the cases where the observed or generated future cannot be reliably tracked. We use $m_{\mathrm{v}} = 0.02$ and $\tau_{\mathrm{v}} = 0.01$ in all experiments. Thus, for two trackable futures, a generated sample contributes to the negative loss only when $\Delta R_{i,k} > 0.02$, and its weight reaches one when $\Delta R_{i,k} \geq 0.03$. The observed future contributes to the demonstration loss regardless of its tracking reliability.

[1] https://github.com/Lijiaxin0111/Open-d4rt

A.3 Forward-Process Negative Fine-Tuning

The predictive objective combines the demonstration loss defined in Section 4.1 with the weighted negative loss in Equation 9 with $b = \mathrm{v}$. For each generated latent endpoint, the noised latent and flow target are

$$\hat{z}_{i,k,t} = (1-t)\hat{z}_{i,k} + t\epsilon, \qquad v^{\star}_{i,k} = \epsilon - \hat{z}_{i,k}, \qquad \epsilon \sim \mathcal{N}(0, I).$$

The negative loss fits the implicit velocity v^{-}_{v} rather than the trainable velocity directly. The coefficient of the latter in Equation 8 is $-\beta_{\mathrm{v}}$. Sampled endpoints and selection weights are held fixed during each update.

For both experts $b \in \{\mathrm{v}, \mathrm{a}\}$, the coefficient in Equation 10 is

$$c_b = \min\left(1, \frac{r^{b}_{\max}\mathcal{L}_{b,\mathrm{GT}}}{\lambda_b \mathcal{L}_{b,\mathrm{NFT}} + \delta}\right), \tag{16}$$

where $r^{b}_{\max}$ is the maximum ratio of the weighted negative loss contribution to the demonstration loss. We hold c_b constant during differentiation. The constant $\delta > 0$ stabilizes both the cap and the negative-loss normalization; the negative loss is zero when all sample weights are zero. We use $\beta_{\mathrm{v}} = 0.1$, $\lambda_{\mathrm{v}} = 0.1$, $r^{\mathrm{v}}_{\max} = 0.25$, and $\delta = 10^{-8}$. Both stages use the shifted flow-time schedule inherited from Fast-WAM: we draw $u \sim \mathcal{U}(0, 1)$ and set $t = 5u/(1 + 4u)$. The frozen reference is initialized once at the start of predictive repair. Appendix D analyzes the negative objective under a fixed reference.

B Details of Action Rejection

B.1 Forward-Process Negative Fine-Tuning

Action post-training uses Equation 10 with $b = \mathrm{a}$, demonstration endpoints $z^{\star}_i = a^{\star}_i$, and sampled endpoints $\hat{z}_{i,k} = \hat{a}_{i,k}$. The negative loss follows Equation 9, with weights from Equation 14. We use $m_{\mathrm{a}} = 0.08$ and $\tau_{\mathrm{a}} = 0.10$ in simulation, and $m_{\mathrm{a}} = 0.05$ and $\tau_{\mathrm{a}} = 0.04$ on the real robot. The remaining coefficients are $\beta_{\mathrm{a}} = 0.1$, $\lambda_{\mathrm{a}} = 0.2$, and $r^{\mathrm{a}}_{\max} = 0.15$ in both domains. The numerical stability constant, flow-time schedule, and treatment of sampled endpoints and weights are the same as in predictive repair. The loss cap follows Equation 16. The action reference is initialized once from the action expert at the start of this stage and remains frozen.

B.2 Temporal Progress Model

Following STEAM (Liu et al., 2026), we learn task-conditioned temporal progress from pairs of multi-view observations and their language instruction. SigLIP2 so400m-patch14-384 and Gemma-3 270M encode the visual and language inputs, respectively. A 32-way classifier predicts signed temporal displacement from forward and reversed frame pairs, with the expected displacement normalized to $[-1, 1]$. Both encoders are trained jointly with the projection layers, multimodal fusion layers, and temporal classification head.

In simulation, training uses demonstrations from all 50 RoboTwin tasks, comprising 2,500 clean and 25,000 randomized episodes. For real-world evaluation, we train separate scorers for Fold Shirt and Clean Table using the same task demonstrations as policy training (Table 9). Frame pairs are sampled with a maximum temporal separation of k frames: $k = 32$ for RoboTwin and Fold Shirt, and $k = 64$ for Clean Table. The classifier retains 32 temporal bins in all settings.

All scorers share the same optimization settings. Images are resized to 384×384, and each predictor is trained for 16,000 steps using AdamW with learning rate 5×10^{-5}, weight decay 10^{-5}, betas (0.9, 0.95), and a global batch size of 512 on 8 NVIDIA A800 GPUs. The learning rate is shared by the encoders and prediction head. Following STEAM, we train an ensemble of three predictors and take the minimum of their progress estimates as A_{STEAM}. Positive scores indicate progress and negative scores indicate regression. The ensemble remains frozen during WAM post-training and evaluates progress between the current observation and the final frame of each imagined future.

C Implementation Details

C.1 Optimization and Sampling

Post-training starts from a Fast-WAM model trained for 5 epochs by joint video and action flow matching. For real-world experiments, we follow the Fast-WAM initialization: the video DiT, text encoder, and video VAE are initialized from pretrained Wan2.2-5B, and the action expert is initialized according to the Fast-WAM configuration. The initial 5 epochs use real-world demonstrations, without transferring a policy checkpoint trained on RoboTwin. The video expert is then refined for 2 epochs. We then freeze the repaired video expert and train the action expert for 2 epochs. Both stages use AdamW with learning rate 10^{-4}, betas (0.9, 0.95), and zero weight decay. The learning rate follows a linear warmup over the first 5% of training and cosine decay to 10^{-6}. Each stage is trained on 16 NVIDIA A800 GPUs. The predictive and action objectives update approximately 5.00B and 1.02B parameters, respectively. These optimization settings apply to both simulation and real-world post-training.

Predictive repair samples K_{v} futures for each demonstration context. Action post-training samples K_{a} action chunks and rolls out each candidate with the frozen video expert. We use 5-step Euler integration with the shifted flow schedule for both video and action sampling. At inference, following Fast-WAM, the action expert uses 10 Euler steps to generate a 32-step action chunk and executes its first 24 steps before generating the next chunk.

C.2 Post-Training Hyperparameters

Table 5 summarizes the coefficients used in simulation and on the real robot. Predictive repair uses the same settings in both domains. For action post-training, the margin and weighting scale differ between domains; all remaining coefficients are shared.

Table 5: Post-training hyperparameters in simulation and on the real robot.

Parameter	Simulation	Real robot
K_{v}	4	4
m_{v}	0.02	0.02
τ_{v}	0.01	0.01
β_{v}	0.1	0.1
λ_{v}	0.1	0.1
$r^{\mathrm{v}}_{\max}$	0.25	0.25
K_{a}	4	4
m_{a}	0.08	0.05
τ_{a}	0.10	0.04
β_{a}	0.1	0.1
λ_{a}	0.20	0.20
$r^{\mathrm{a}}_{\max}$	0.15	0.15

D Analysis of the Rejection Objective under a Fixed Reference

We provide a pointwise population analysis of the uncapped objective under a fixed reference predictor and a fixed distribution of selected negatives. The analysis concerns the predictor output at a single input; it does not establish convexity in neural network parameters or characterize training dynamics under repeated negative sampling.

Fix a branch $b \in \{\mathrm{v}, \mathrm{a}\}$ and a complete predictor input ξ, including the noised endpoint, flow time, and the conditioning of that branch. We write

$$v = v_{\psi_b}(\xi), \qquad \bar{v} = \bar{v}_b(\xi), \qquad \lambda = \lambda_b, \qquad \beta = \beta_b.$$

Let $p_{\mathrm{GT}}(\xi)$ denote the density of noised inputs induced by demonstration endpoints, and let $p_{\mathrm{neg}}(\xi)$ denote the weighted density induced by selected negative endpoints. The latter incorporates the nonnegative selection weights used by the corresponding branch. Their conditional mean flow targets

are

$$m_{\mathrm{GT}}(\xi) = \mathbb{E}[v^\star \mid \xi, \mathrm{GT}], \qquad m_{\mathrm{neg}}(\xi) = \mathbb{E}[v^\star \mid \xi, \mathrm{neg}]. \tag{17}$$

Suppressing the dependence on ξ, the bias–variance decomposition of the squared flow-matching loss gives the following objective, up to terms independent of v:

$$F_\xi(v) = p_{\mathrm{GT}} \|v - m_{\mathrm{GT}}\|_2^2 + \lambda p_{\mathrm{neg}} \left\|(1+\beta)\bar{v} - \beta v - m_{\mathrm{neg}}\right\|_2^2 . \tag{18}$$

Define the effective regression target

$$\widetilde{m}_{\mathrm{neg}} = \bar{v} + \frac{1}{\beta}\left(\bar{v} - m_{\mathrm{neg}}\right). \tag{19}$$

Since

$$(1+\beta)\bar{v} - \beta v - m_{\mathrm{neg}} = -\beta(v - \widetilde{m}_{\mathrm{neg}}),$$

Equation 18 is equivalent to

$$F_\xi(v) = p_{\mathrm{GT}} \|v - m_{\mathrm{GT}}\|_2^2 + \lambda \beta^2 p_{\mathrm{neg}} \|v - \widetilde{m}_{\mathrm{neg}}\|_2^2. \tag{20}$$

Whenever $p_{\mathrm{GT}} + \lambda\beta^2 p_{\mathrm{neg}} > 0$, the objective is strictly convex in the pointwise function value, with

$$\nabla_v^2 F_\xi = 2\left(p_{\mathrm{GT}} + \lambda\beta^2 p_{\mathrm{neg}}\right) I \succ 0. \tag{21}$$

Its unique minimizer is

$$v_{\mathrm{opt}}(\xi) = \frac{p_{\mathrm{GT}} m_{\mathrm{GT}} + \lambda\beta^2 p_{\mathrm{neg}} \widetilde{m}_{\mathrm{neg}}}{p_{\mathrm{GT}} + \lambda\beta^2 p_{\mathrm{neg}}}. \tag{22}$$

Because

$$\widetilde{m}_{\mathrm{neg}} - \bar{v} = \frac{\bar{v} - m_{\mathrm{neg}}}{\beta},$$

$\widetilde{m}_{\mathrm{neg}}$ extrapolates from the fixed reference in the direction opposite to the conditional mean target of the selected negatives. The pointwise optimum therefore balances the demonstration flow target with this effective regression target. When $p_{\mathrm{neg}} = 0$ and $p_{\mathrm{GT}} > 0$, it reduces to the demonstration optimum m_{GT}.

This analysis differs from the reinforcement guidance derivation of DiffusionNFT (Zheng et al., 2026b). DiffusionNFT obtains its positive and negative distributions by reward reweighting samples from a single reference policy, such that their mixture recovers that policy. In our setting, positive endpoints come from demonstrations, whereas negative endpoints are selected from samples generated by the current model; the two distributions do not form an analogous decomposition. Equation 22 therefore characterizes the minimizer of our rejection objective rather than a direction of reward improvement. It neither guarantees higher environment reward nor implies closed-loop policy improvement.

E Simulation Experiment Details

E.1 Ablation Protocol

The Fast-WAM implementation supports an action-conditioned video expert, whereas its published results use action-unconditioned video prediction. All ablations start from the same action-conditioned Fast-WAM checkpoint trained for 5 epochs. Optimization and sampling settings follow Appendix C.

Predictive and action objectives. In Table 3, each predictive objective is applied for 2 epochs, followed by 2 epochs with the corresponding action objective. When predictive repair or action rejection is omitted, the respective expert instead receives 2 additional epochs of flow matching on the demonstrations. Every setting therefore uses the same data and number of optimizer updates.

Action objective ablation. Table 4 isolates the effect of the action objective using a common frozen video expert after repair. All variants share the action initialization, data order, and training duration. Variants that use sampled actions also share the candidate sampling budget. For all-sample rejection, every sampled action receives weight 1. For random selection, each sampled action independently receives weight 1 with probability 0.5 and weight 0 otherwise. Both variants use the normalization in Equation 9 with $b = \mathrm{a}$, so the negative loss is averaged over the selected samples in the minibatch. It is zero when no sample is selected.

Action and video representations. In RoboTwin, each action is a 32-step sequence of 14 absolute joint and gripper targets, comprising six arm joints and one gripper command per arm. The demonstrations are recorded at 50 Hz. Each video clip contains the current frame and 8 predicted future frames, with the three camera views arranged at a combined resolution of 384×320. The current frame provides visual conditioning and is excluded from the flow-matching loss. Consecutive frames are separated by 4 action steps, so the clip spans the complete action chunk.

E.2 Evaluation of Action-Conditioned Prediction

Experimental setup. We compare five objectives for post-training the video expert. Continued flow matching serves as a control with the same training data and number of optimizer updates, but uses only the observed futures. Repair with PSNR compares generated and observed head-camera images, averaging PSNR over the 8 future frames. Generated futures with an average PSNR below 26 dB are selected as negatives and assigned weight one in the negative fine-tuning objective. We then evaluate the complete action-following score and two variants that remove either relative-trajectory consistency or absolute-position consistency. These variants isolate the contribution of each geometric component to predictive repair.

All repair variants share the initialization, training data, sampling budget, and negative fine-tuning formulation; they differ in how generated futures are selected and weighted.

The evaluation uses 500 randomized validation episodes from RoboTwin 2.0. For each episode, every video model generates one future conditioned on the same observation and demonstrated action. PSNR and SSIM compare the generated future with its observed counterpart and measure visual fidelity. The action-following metrics below instead evaluate whether the generated robot motion realizes the conditioning action.

Action-following metrics. We use the trajectory reliability indicator $C(y)$ defined in Section 4.1. Because each episode provides one generated future, we omit the sample index k from $\hat{y}_{i,k}$ and $\Delta R_{i,k}$. Let

$$\mathcal{I}_{\mathrm{ref}} = \{i : C(y_i^\star) = 1\}, \qquad N_{\mathrm{ref}} = |\mathcal{I}_{\mathrm{ref}}|, \tag{23}$$

denote the episodes whose observed future provides a reliable reference trajectory. The tracking success rate is the proportion of generated futures that satisfy this reliability criterion:

$$\text{Tracking success rate} = \frac{1}{N_{\mathrm{ref}}} \sum_{i \in \mathcal{I}_{\mathrm{ref}}} C(\hat{y}_i). \tag{24}$$

For a trackable generated future, the observed-reference gap is

$$\Delta R_i = R_{\mathrm{AF}}^{\mathrm{obs}}(y_i^\star, a_i^\star) - R_{\mathrm{AF}}^{\mathrm{gen}}(\hat{y}_i, a_i^\star). \tag{25}$$

We use $m_{\mathrm{eval}} = 0.02$ for all reported diagnostic results. The action-following failure rate is

$$\text{AF failure rate} = \frac{1}{N_{\mathrm{ref}}} \sum_{i \in \mathcal{I}_{\mathrm{ref}}} \mathbf{1}\big[C(\hat{y}_i) = 0 \ \vee \ \big(C(\hat{y}_i) = 1 \wedge \Delta R_i > m_{\mathrm{eval}}\big)\big]. \tag{26}$$

We report both quantities as percentages. Tracking success measures whether the generated future contains sufficient robot motion for trajectory comparison, whereas the failure rate additionally includes trackable futures whose action-following score is substantially below the observed reference.

Results. Repair with PSNR reduces the action-following failure rate from 32.0% to 18.6%, indicating that generic visual feedback corrects some action-following errors. The complete action-following score further reduces the failure rate to 6.8% and increases tracking success from 93.0% to 99.2%. It also achieves the highest PSNR and SSIM. These results show that action-following feedback corrects prediction errors that are not captured by visual similarity, without trading off standard video fidelity.

Among the individual geometric components, relative-trajectory consistency provides the stronger action-following signal, yielding a failure rate of 10.6%, compared with 17.0% for absolute-position consistency. Combining both components further reduces the failure rate to 6.8% and achieves the highest trackability, PSNR, and SSIM. The two terms are therefore complementary: relative trajectories characterize the execution of the conditioning action, while absolute positions retain information about kinematic alignment.

Table 6: Evaluation and component analysis of predictive repair on 500 randomized validation episodes from RoboTwin 2.0, with one generated future per episode. Action-following metrics are evaluated on episodes whose observed future provides a reliable reference trajectory.

Predictive post-training	PSNR	SSIM	Tracking success (%)	AF failure (%)
Continued flow matching	30.69	0.9500	93.0	32.0
Repair with PSNR	31.94	0.9576	96.8	18.6
Ours w/o relative-trajectory consistency	32.25	0.9584	97.2	17.0
Ours w/o absolute-position consistency	32.52	0.9607	98.0	10.6
Ours	**33.17**	**0.9642**	**99.2**	**6.8**

Table 7: Policy success rates (%) with different predictive repair scores and the same action rejection procedure on RoboTwin 2.0.

Predictive repair score	Clean	Randomized
PSNR	92.1	91.8
Kinematic alignment (ours)	**93.8**	**93.7**

Effect of the repair score on policy success. We use the PSNR-repaired video expert from Table 6 in the same action rejection procedure and compare the resulting policy with R^2-WAM. As shown in Table 7, repair with the kinematic alignment score improves success over PSNR-based repair by 1.7 and 1.9 percentage points in clean and randomized environments, respectively. This comparison shows that the benefit of motion-based repair extends beyond video metrics to policy success.

E.3 Action Negative Selection from Imagined Consequences

Evaluation protocol. We compare continued flow matching and predictive repair in the clean RoboTwin setting across all 50 tasks. For each task, 10 independently generated successful scripted-expert trajectories provide one context each, giving 500 contexts. The decision point is placed at 60% of the valid 32-step action-chunk start range. The next 32 expert actions form the reference continuation; completion of the entire task within this horizon is not required.

For each context, four candidate action chunks are sampled once from a fixed policy and reused for both video models. Each candidate and the expert continuation are executed from the same restored simulator state, producing 2,000 candidate executions and 500 reference executions. Each video model also predicts futures under the expert continuation and the identical candidate actions. This diagnostic uses simulator executions to evaluate the consequence comparisons, rather than as supervision for the post-training objective.

Reference labels and selection metrics. The same frozen STEAM evaluator scores actual and imagined outcomes. Let $y^{\text{exec}}_{i,k}$ denote the observed consequence of executing candidate $\hat{a}_{i,k}$, and let $y^{\star}_i$ denote the observed expert continuation. The execution-based consequence gap is

$$\Delta A^{\text{exec}}_{i,k} = A_{\text{STEAM}}(x_i, y^{\star}_i) - A_{\text{STEAM}}(x_i, y^{\text{exec}}_{i,k}). \tag{27}$$

A candidate receives a reference negative label when $\Delta A^{\text{exec}}_{i,k} > m$. For each video model, the imagined gap $\Delta A_{i,k}$ is computed as in Equation 13, using that model for both the expert and candidate futures. The candidate is selected when $\Delta A_{i,k} > m$. We evaluate $m \in \{0.02, 0.05, 0.08\}$, using the same threshold for reference labels and predicted selections within each comparison; 0.08 is the margin used for action post-training in simulation.

We treat the reference negative label as the positive class for this diagnostic. Thus, a true positive (TP) is a selected action that also receives a negative label from its actual execution; a false positive (FP) is selected only from its imagined consequence. False negatives (FN) and true negatives (TN) are the corresponding unselected cases. We compute

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{28}$$

Table 8: Action negative selection against reference labels from actual executions. Counts are pooled over 2,000 shared candidate actions; precision, recall, and F1 are percentages. The positive class is an action whose execution-based consequence gap exceeds the listed margin.

Margin	Video model	TP	FP	FN	TN	Precision	Recall	F1
0.02	Continued FM	652	196	816	336	76.9	44.4	56.3
	Predictive repair	935	239	533	293	**79.6**	**63.7**	**70.8**
0.05	Continued FM	437	254	692	617	63.2	38.7	48.0
	Predictive repair	640	193	489	678	**76.8**	**56.7**	**65.2**
0.08	Continued FM	268	147	487	1,098	64.6	35.5	45.8
	Predictive repair	382	165	373	1,080	**69.8**	**50.6**	**58.7**

All metrics use counts pooled over the 2,000 candidates, rather than averages of task-level metrics. The reference labels contain 1,468, 1,129, and 755 negatives at margins of 0.02, 0.05, and 0.08, respectively. They measure relative progress according to STEAM on actual executions, not binary task success.

F Real-World Evaluation Details

F.1 Robot Platform and Policy Deployment

We use a PIPER platform with two arms (Figure 4). Each arm has six actuated joints and a gripper with one degree of freedom. Visual observations are provided by three RGB cameras: one front camera and one camera mounted on each wrist. The raw camera streams are recorded at 480×640 resolution and 30 Hz. At inference, the front view is resized to 320×256, the two wrist views are resized to 160×128, and the views are assembled into a 384×320 combined camera input (height $\times$ width).

The policy predicts a 20D absolute bimanual end-effector action, comprising position, a 6D rotation representation, and gripper state for each arm. The policy uses 10 Euler steps to generate a 32-step action chunk at 30 Hz, with an execution horizon of 24 control steps.

F.2 Offline Real-World Demonstrations

All task adaptation and post-training use offline teleoperated demonstrations; the method does not collect additional environment interactions. Demonstrations for the two evaluated tasks comprise 3,031 trajectories and approximately 42.9 hours of robot experience. Table 9 gives the composition for each task. Training uses one instruction describing the overall task for each trajectory and does not require subtask labels for the long horizon tasks.

Table 9: Offline demonstrations for the two evaluated real-world tasks.

Task	Episodes	Frames	Duration (h)
Fold Shirt	605	3,110,127	28.80
Clean Table	2,426	1,526,405	14.13
Total	3,031	4,636,532	42.93

F.3 Controlled Real-World Variations

We independently vary whether the initial position and object instance are represented in the demonstrations, yielding four evaluation conditions. Unseen conditions introduce position or object-instance variations for one or more objects; the remaining objects may retain their training configurations. Figure 5 shows representative configurations for both tasks. For every method and task, each condition contains 20 autonomous episodes, for a total of 80. Configuration lists are fixed before policy evaluation and shared across all methods.

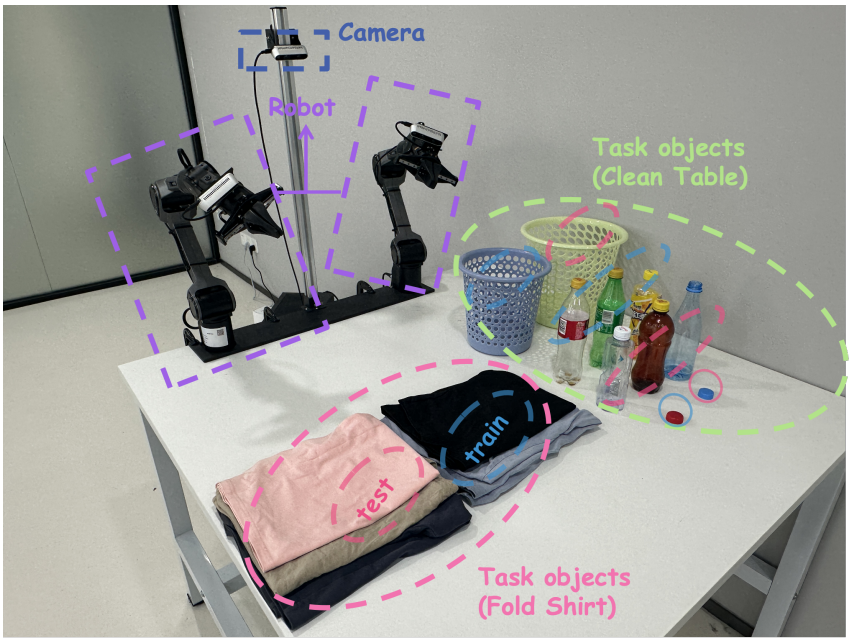

Figure 4: Real-world experimental setup, showing the dual-arm PIPER platform, cameras, and task objects for Fold Shirt and Clean Table.

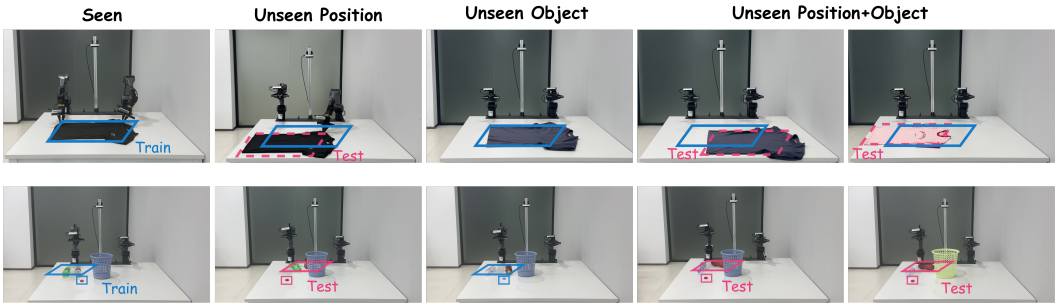

Figure 5: Representative evaluation configurations for Fold Shirt (top) and Clean Table (bottom). Blue and pink outlines indicate training and test position regions, respectively.

Seen. The objects and initial position region are represented in the demonstrations. These trials quantify performance under familiar deployment conditions without replaying demonstration trajectories.

Unseen Position. We retain familiar object instances while varying the initial placement of one or more objects beyond their training regions. For Fold Shirt, evaluation uses five test regions displaced from the central training region. For Clean Table, bottles and bottle caps may occupy either their training or test regions within the same scene.

Unseen Object. We retain the training position regions while introducing one or more object instances not used in training. Fold Shirt uses two training shirts and three unseen shirts. For Clean Table, the training objects comprise three bottles, one bottle cap, and one trash bin; the unseen set contains three additional bottles, one bottle cap, and one trash bin.

Unseen Position + Object. This condition combines both variations within the same scene, including unseen object instances and initial placements outside the training regions.

F.4 Success Criteria and Statistical Reporting

Each quantitative episode runs without human intervention and terminates upon task completion, a time limit defined for each task, or a predefined failure or safety condition. Completion criteria are fixed before model evaluation and applied identically to all methods.

Appendix Table 10 reports the success rate and 95% Wilson score interval for every task and condition, each computed from 20 binary outcomes, corresponding to Figure 3(a–b). The average success under unseen conditions is the unweighted mean over the two tasks and the three conditions with an unseen factor.

Table 10: Task success rates (%) for each real-world condition. Every cell contains 20 autonomous trials; brackets denote 95% Wilson score confidence intervals.

Task	Method	Seen	Unseen position	Unseen object	Unseen position + object
Fold Shirt	$\pi_{0.5}$	65.0 [43.3, 81.9]	35.0 [18.1, 56.7]	50.0 [29.9, 70.1]	35.0 [18.1, 56.7]
	X-VLA	40.0 [21.9, 61.3]	15.0 [5.2, 36.0]	20.0 [8.1, 41.6]	10.0 [2.8, 30.1]
	Motus	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]
	Fast-WAM	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]	0.0 [0.0, 16.1]
	R^2-WAM	**90.0 [69.9, 97.2]**	**90.0 [69.9, 97.2]**	**85.0 [64.0, 94.8]**	**85.0 [64.0, 94.8]**
Clean Table	$\pi_{0.5}$	70.0 [48.1, 85.5]	45.0 [25.8, 65.8]	60.0 [38.7, 78.1]	40.0 [21.9, 61.3]
	X-VLA	50.0 [29.9, 70.1]	20.0 [8.1, 41.6]	40.0 [21.9, 61.3]	20.0 [8.1, 41.6]
	Motus	45.0 [25.8, 65.8]	35.0 [18.1, 56.7]	45.0 [25.8, 65.8]	20.0 [8.1, 41.6]
	Fast-WAM	55.0 [34.2, 74.2]	40.0 [21.9, 61.3]	35.0 [18.1, 56.7]	25.0 [11.2, 46.9]
	R^2-WAM	**85.0 [64.0, 94.8]**	**75.0 [53.1, 88.8]**	**75.0 [53.1, 88.8]**	**70.0 [48.1, 85.5]**

F.5 Component Ablations on Fold Shirt

We evaluate predictive repair and action rejection separately on Fold Shirt (Table 11). Each variant is evaluated over 20 autonomous trials in each of the four conditions described in Appendix F.3, totaling 80 trials. Repair / GT FM applies predictive repair to the video expert while training the action expert with demonstration flow matching. GT FM / Rejection trains the video expert with demonstration flow matching and uses its imagined consequences for action rejection. Neither single-stage variant completes the task in these trials. Predictive repair alone does not yield successful task completion when the action expert is trained solely with demonstration flow matching. Conversely, action rejection without predictive repair also remains insufficient. Combining the two stages achieves 87.5% average success, supporting their complementary roles in long-horizon shirt folding: improving the action fidelity of predicted consequences and using that feedback to refine sampled actions.

Table 11: Component ablation on Fold Shirt. Success rates are averaged over the four evaluation conditions. GT FM denotes flow matching on demonstrations.

Video	Action	Average success (%)
Repair	GT FM	0.0
GT FM	Rejection	0.0
Repair	Rejection	**87.5**

F.6 Real-World Execution Examples

Figures 6 and 7 show representative execution sequences of R^2-WAM on Fold Shirt and Clean Table.

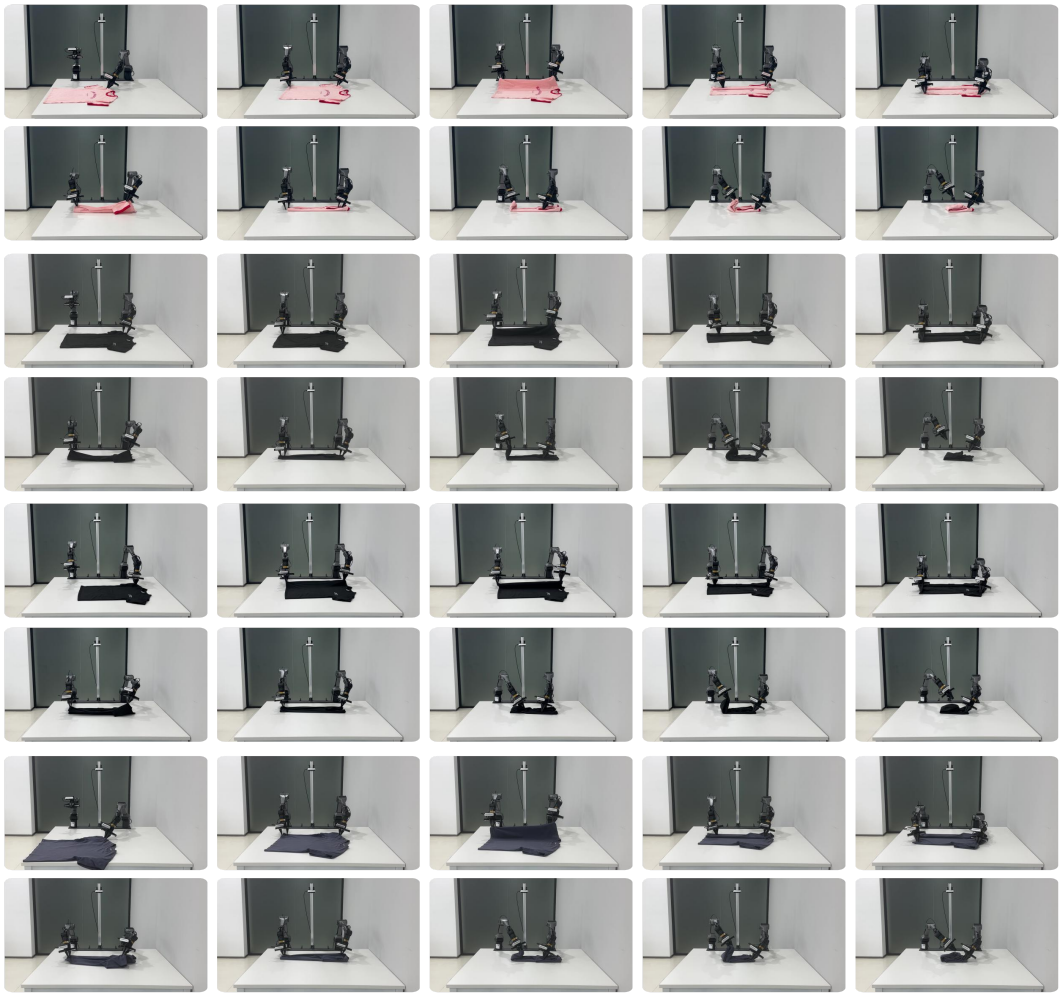

Figure 6: Execution sequences of R^2-WAM on Fold Shirt. Each sequence spans two consecutive rows and proceeds from left to right.



Figure 7: Execution sequences of R^2-WAM on Fold Shirt (top four rows) and Clean Table (bottom two rows). Each Fold Shirt sequence spans two rows, while each Clean Table sequence occupies one row. Frames proceed from left to right.