LOOPWAM: CLOSED-LOOP GENERALIST POLICY IMPROVEMENT FROM DEPLOYMENT ROLLOUTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling expert demonstrations alone does not enable generalist robot policies to improve from deployment. Real-world rollouts contain successes, suboptimal behavior, and failures: imitating every recorded action can reinforce poor behavior, while filtering out failures discards evidence of physical interactions and their consequences. We introduce **LoopWAM**, a closed-loop world-action model that learns directly from this full quality spectrum, using rollouts from both itself and other deployed policies. Source-dependent noise sampling unifies behavior learning and action-conditioned world modeling, while quality steering and quality-weighted supervision retain full video supervision across qualities and regulate action imitation. Video and action tokens interact throughout parallel denoising, allowing learned consequences to inform action generation directly. On RoboChallenge Table30-V2, three deployment-and-training rounds raise success from 23.67% to 43.33%, a 19.7-percentage-point gain, with a final score of 56.80. LoopWAM also achieves 94.5% average success on RoboTwin 2.0 and 49.5% overall success on RoboCasa365 without additional embodied pretraining, obtaining the best overall results among the compared methods across all three benchmarks. On challenging shirt folding task, two rounds of deployment-driven improvement after single-task post-training raise success from 13.20% to 55.56%. These results demonstrate closed-loop improvement from mixed-quality experience under both multi-task learning and single-task post-training. Project Page

1 INTRODUCTION

Scaling human-collected robot data has expanded the capabilities of generalist manipulation policies (O'Neill et al., 2024), but demonstration scaling alone does not close the loop between deployment and continual improvement. Deployment exposes states, interactions, and failure modes beyond curated demonstrations, providing new experience for an embodied data flywheel. Real-world reinforcement learning (RL) (Xu et al., 2026) can improve policy performance during deployment, but may narrow the model's predictive distribution and often struggles to learn multi-task generalist policies. Although distilling a generalist policy from multiple RL specialists (Xiao et al., 2025) can address this issue, the generalist's supervision comes from curated successful rollouts rather than the full interaction history. Therefore, we aim to study direct generalist improvement from the full mixture of successful, suboptimal, and failed deployment rollouts. The challenge is to use this experience productively: imitating every recorded action risks absorbing undesirable behavior, while retaining only high-quality trajectories discards informative interactions and their outcomes.

Recent VLA and world-action models address important pieces of this problem. On the VLA side, $\pi_{0.6}$ (Physical Intelligence, 2025) uses advantage-conditioned learning to incorporate autonomous experience and interventions, while $\pi_{0.7}$ (Physical Intelligence, 2026) uses quality, speed, and mistake metadata to absorb mixed-quality data and distill specialist capabilities into a generalist. These VLA methods use steering during training to model behavior distributions across data qualities and specify a desirable mode at inference for smooth execution, but rely on VLM-based language–vision–action representations without an explicit objective for learning the physical dynamics reflected in action–video transitions. Consequently, suboptimal and failed rollouts provide behavior targets but do not directly supervise learning of their physical consequences. World modeling offers a more elegant solution: WAMs (Pai et al., 2025; Ye et al., 2026b; Li et al., 2026) couple predictive visual representations with action generation, while action-conditioned world models

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.

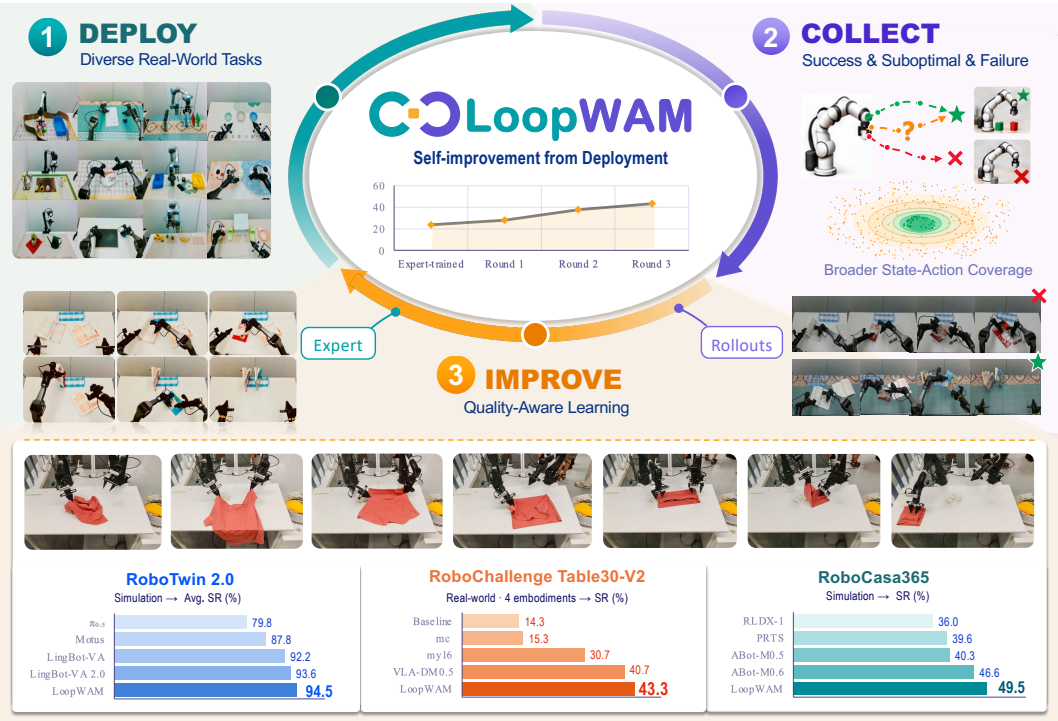

Figure 1: **LoopWAM learns from deployment.** Mixed-quality rollouts return to training through quality steering, source-dependent noise sampling, and quality-weighted supervision. Center: closed-loop self-improvement on RoboChallenge Table30-V2, where three rounds of deployment and retraining increase overall success from 23.67% to 43.33%. Bottom: benchmark comparisons.

(ACWMs) (Guo et al., 2025; Gao et al., 2026) predict future observations conditioned on actions. Combining the two allows ACWM to learn action consequences from data of different qualities, while WAM learns appropriate actions from expert demonstrations. τ_0-WM (Zhou et al., 2026) trains separate WAM and ACWM, routing failure and recovery data to ACWM training and use ACWM as selector for policy improvement. FACT (Peng et al., 2026) implements the two roles within a shared model, but retains sequential action–consequence generation. It only supports success/failure supervision split, with failure consequences influencing the policy implicitly through shared representations. These approaches expose a remaining design challenge: VLA steering lacks natural action–video dependency modeling, while consequence-aware WAMs still exploit rollout data indirectly through an ACWM interface—a simulator for action selection or a subsequent consequence-prediction stage that influences the policy through shared representations. Can a single WAM directly learn from mixed-quality deployment rollouts, with all qualities supervising the model and learned consequences participating directly in action generation?

We introduce **LoopWAM**, a closed-loop World-Action Model that updates a generalist directly from mixed-quality deployment rollouts. Our core idea is to let one WAM directly absorb rollouts across the full quality spectrum and learn from each. To achieve this, we first use *source-dependent noise sampling* to unify WAM behavior learning and ACWM consequence learning within a single model and generation stage. Video and action noise levels are sampled independently, allowing low-noise rollout actions to guide prediction of their observed consequences. Within this unified model, *quality steering* represents behaviors of different qualities and selects the highest-quality condition at deployment. To control how strongly each behavior is imitated, we further apply *quality-weighted supervision*, reducing imitation of suboptimal actions and masking failed-action targets while retaining video supervision across all qualities. Together, these mechanisms let every trajectory supervise the joint model according to its behavioral quality and observed transitions. Video and action tokens interact bidirectionally throughout one parallel denoising process, allowing learned future representations to participate in action formation without a separate consequence-evaluation stage. We close the loop by repeatedly collecting rollouts and retraining the generalist on demonstrations and

accumulated experience before redeployment. On **RoboChallenge Table30-V2** (RoboChallenge Dexmal Team, 2026), three rounds raise overall success from **23.67%** to **28.00%**, **37.67%**, and **43.33%**, a **19.7-percentage-point improvement**, with a final score of **56.80**, ranking first in both success and score among the compared entries (Table 1). LoopWAM achieves **94.5%** average success on **RoboTwin 2.0** (Chen et al., 2025), exceeding the reported 87.5% of FACT and 93.6% of LingBot-VA 2.0, and **49.5%** on the 50-task **RoboCasa365** evaluation (Nasiriany et al., 2026), above ABot-M0.6's 46.6% (Tables 2–3). We further evaluate deployment-driven self-improvement after single-task post-training on challenging shirt folding, which involves deformable objects and multiple coordinated steps. Two deployment-and-training rounds raise success from **13.20%** to **55.56%** (Section 4.4). These results establish that directly learning from mixed-quality rollouts with parallel world–action denoising can deliver leading benchmark performance and repeated deployment improvement in both multi-task and single-task settings.

2 Related Work

Robot Manipulation Policies. Robot manipulation policies increasingly combine scalable imitation learning with pretrained representations. ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2025) learn action sequences from demonstrations, while RT-2 (Zitkovich et al., 2023) and OpenVLA (Kim et al., 2024) transfer vision-language priors to language-conditioned control. Flow-based policies such as π_0 (Black et al., 2024) and $\pi_{0.5}$ (Physical Intelligence et al., 2025) further scale continuous action generation and generalization. Open X-Embodiment (O'Neill et al., 2024) and DROID (Khazatsky et al., 2024) expand training coverage across embodiments and environments. Beyond offline demonstrations, DAgger (Ross et al., 2011) addresses policy-induced distribution shift by collecting expert labels on visited states. LoopWAM studies learning from the recorded observations and actions of deployment rollouts, using trajectory quality to guide policy updates while retaining world supervision from unsuccessful interactions.

World Action Models. Video-based robot policies use predictive visual representations to support control. mimic-video (Pai et al., 2025) conditions an action decoder on representations from a pretrained video model. DreamZero (Ye et al., 2026b) jointly generates video and actions, while LingBot-VA (Li et al., 2026) organizes visual prediction and action decoding within a causal autoregressive framework. LingBot-VA 2.0 (Zhang et al., 2026a) further develops native video-action pretraining with a control-oriented tokenizer. Efficiency-oriented designs include Fast-WAM (Yuan et al., 2026), which retains video co-training but removes future-video generation at inference, and GigaWorld-Policy (Ye et al., 2026a), which learns action-conditioned visual dynamics while allowing action-only inference. Being-H0.7 (Luo et al., 2026) instead uses future observations to supervise latent representations. These studies establish several ways for predictive supervision to support action learning. LoopWAM focuses on closing the deployment–training loop: mixed-quality rollouts broaden world-model training, while joint video–action modeling, quality steering, and weighted action supervision connect consequence awareness with high-quality behavior.

3 Method

LoopWAM learns a world–action model that can be updated with experience collected during deployment. The framework couples video and action generation within a shared denoising process, then uses trajectory quality to distinguish how deployment data supervise world modeling and control. Figure 2 summarizes the architecture and training procedure. We first introduce joint world–action modeling, followed by quality-aware learning from mixed trajectories and the deployment–learning loop.

3.1 Joint World–Action Modeling

We consider language-conditioned manipulation with visual observation o_t, robot state s_t, and instruction ℓ. The prediction targets are an action chunk $A_t = (a_t, \dots, a_{t+H-1})$ and the corresponding future observations V_t, sampled over the same horizon. A pretrained video VAE maps (o_t, V_t) to an observation latent z_o and future latents Z_t. LoopWAM models their joint distribution

$$p_\theta(A_t, Z_t \mid o_t, s_t, \ell, q), \tag{1}$$

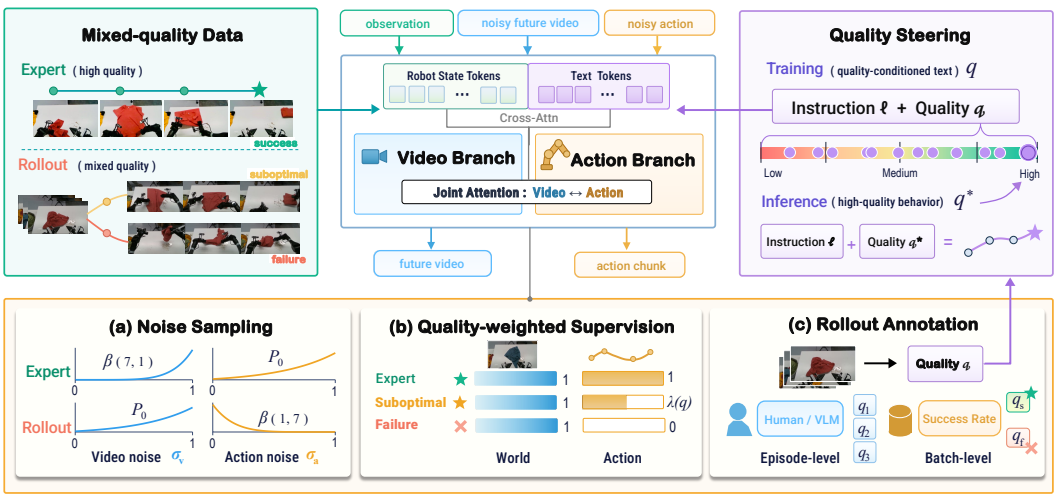

Figure 2: **Overview of LoopWAM.** The video and action branches exchange information through bidirectional attention between their denoising tokens. Training combines quality-conditioned instructions, source-dependent noise sampling, and quality-weighted action supervision. Expert trajectories supervise both modalities, while lower-quality rollouts retain full world supervision and contribute to action imitation according to their quality. The trained model is deployed using the expert score $q^\star$, and recorded experience returns to the next training round.

where q is a numerical score of trajectory quality, as detailed in Section 3.2. This unified formulation enables a single model to learn action generation and visual dynamics from paired action–observation sequences (Zhu et al., 2025).

We instantiate this formulation with a dual-stream diffusion transformer whose video backbone is initialized from Wan2.2-TI2V-5B (Wan et al., 2025). The video and action streams have separate token projections, transformer parameters, and prediction heads, with joint attention connecting the streams at every layer. To train them with flow matching (Lipman et al., 2023), we independently perturb each target X_m, where $X_v = Z_t$ and $X_a = A_t$:

$$X_m^{\sigma_m} = (1-\sigma_m)X_m + \sigma_m \epsilon_m, \qquad \epsilon_m \sim \mathcal{N}(0, I), \quad m \in \{v, a\}. \tag{2}$$

The joint network predicts velocity fields (v_θ^v, v_θ^a) with targets $u_m = \epsilon_m - X_m$. Each stream receives noise level σ_m, with 0 denoting clean data and 1 pure noise; the observation latent z_o remains clean.

The attention structure couples the two predictions throughout denoising. Denote observation, future-video, and action tokens by O, V, and A, respectively. Their block mask, with queries indexing rows and keys indexing columns, is

$$M_{(O,V,A)} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \tag{3}$$

where 1 permits attention. Observation tokens attend only within the initial frame. Both streams cross-attend to the quality-conditioned instruction; robot-state tokens additionally condition the action stream. Together with the training scheme in Section 3.2, this architecture supports learning both embodied world representations and awareness of action consequences. Dense video supervision on expert and mixed-quality rollouts broadens the interactions from which these representations are learned. Through bidirectional attention, the world-prediction loss also trains features used for action generation. This training couples learning how tasks can be completed with learning how particular actions affect their outcomes, encouraging consistency between video and action generation.

3.2 Quality-Aware Learning from Mixed Trajectories

Our training corpus combines expert demonstrations $\mathcal{D}_E$ with rollout data $\mathcal{D}_R$ of arbitrary quality, including successful, suboptimal, and failed executions. These rollouts substantially broaden the

state–action–outcome distribution beyond expert paths, providing world-model supervision even when their actions are undesirable imitation targets. Each trajectory is assigned a source indicator $d \in \{E, R\}$; every training segment extracted from that trajectory uses the same indicator. We combine source-dependent noise sampling, quality conditioning, and selective action supervision to learn from this broader experience while favoring high-quality behavior.

Source-dependent noise sampling. Jointly predicting video and actions in a WAM does not by itself ensure that the predicted futures reflect the consequences of the generated actions. To improve consistency between video and action generation and make world predictions more useful for control, we aim to strengthen action-conditioned world modeling (ACWM) within the same unified model. We therefore use source-dependent noise sampling to make recorded rollout actions accessible when predicting their observed consequences, while retaining joint video–action generation in a single training stage. Specifically, we select different noise distributions for expert and rollout segments, while drawing video and action noise levels independently within each segment:

$$\begin{aligned} d = E: &\quad \sigma_v \sim \mathrm{Beta}(7,1), \quad \sigma_a \sim \mathcal{P}_0, \\ d = R: &\quad \sigma_v \sim \mathcal{P}_0, \qquad\quad \sigma_a \sim \mathrm{Beta}(1,7). \end{aligned} \tag{4}$$

Here $\mathcal{P}_0$ is the base noise distribution obtained by sampling $u \sim \mathcal{U}(0,1)$ and setting $\sigma = \phi_\kappa(u) = \kappa u/[1+(\kappa-1)u]$, with $\kappa = 5$. The Beta distributions (Johnson et al., 1995) are applied directly to σ. For expert data, high video noise emphasizes recovering expert futures from strongly corrupted inputs, alongside action generation over the base noise distribution. For rollouts, low action noise preserves much of the recorded motion while the video stream predicts its visual consequences. Training across these executions broadens the action–outcome distribution seen by the world model, including transitions in which the task remains incomplete or an interaction fails. Joint attention then connects this consequence-learning signal to the representations used for action generation.

Quality steering. Within this unified training scheme, each trajectory is assigned a numerical quality score q, which is shared by all training segments extracted from it. We encode quality as a suffix to the instruction, $\ell_q = [\ell; \texttt{Quality: } q]$, and use its text embeddings to condition both video and action streams. We support two trajectory annotation schemes: (1) *Episode-level annotation*, where human annotators or a vision-language model (VLM) score each trajectory individually for fine-grained quality estimation; and (2) *coarse-level annotation*, where a single human-provided score is assigned to all rollouts from a policy checkpoint based on its overall success rate and motion smoothness. Under coarse annotation, successful trajectories receive the checkpoint score while failed trajectories receive the minimum score, requiring only one annotation per checkpoint. Both schemes represent rollout quality using numerical scores. For expert demonstrations, we assign an expert score $q^\star$ that is higher than the maximum rollout quality score. The quality-conditioned instruction allows different behaviors under the same task instruction to be represented within one model. During deployment, we condition on $q = q^\star$ to steer sampling toward expert-quality behavior.

Quality-weighted supervision. Given these noise levels and quality scores, we retain video prediction on all trajectory qualities and modulate action supervision with a nondecreasing weight $\lambda(q) \in [0,1]$. Expert trajectories receive full action supervision with $\lambda(q^\star) = 1$, while failed trajectories use $\lambda(q) = 0$ and receive no action supervision. For sample i and modality m, define the denoising loss

$$\ell_{m,i} = \omega(\sigma_{m,i}) \left\| v^m_{\theta,i} - (\epsilon_{m,i} - X_{m,i}) \right\|^2_{\text{valid}}, \tag{5}$$

where the norm denotes mean squared error over valid target elements and ω is the common timestep weighting function. The clean observation latent is excluded from video supervision, and padded targets are masked in both modalities. For a minibatch of B segments, the complete objective is

$$\mathcal{L}_{\text{LoopWAM}} = \underbrace{\frac{1}{B}\sum_{i=1}^{B} \ell_{v,i}}_{\text{world supervision}} + \underbrace{\frac{\sum_{i=1}^{B} \lambda(q_i)\ell_{a,i}}{\max\left(1, \sum_{i=1}^{B} \lambda(q_i)\right)}}_{\text{quality-weighted action supervision}}. \tag{6}$$

The source indicator determines corruption, whereas quality controls the behavior condition and action weight. In particular, $\lambda(q) = 0$ removes direct imitation of a trajectory's actions while preserving world-model gradients through the jointly attended video and action representations. Quality steering and weighted supervision thus allow broader transition learning while concentrating action imitation on more desirable behavior. The data preprocessing are specified in Appendix A.

3.3 Closed-Loop Self-Improvement

The deployed policy uses the high-quality action marginal of the joint model:

$$\pi_\theta(A_t \mid o_t, s_t, \ell; q^\star) = \int p_\theta(A_t, Z_t \mid o_t, s_t, \ell, q^\star)\, \mathrm{d}Z_t. \tag{7}$$

In practice, we sample this marginal by jointly denoising video and actions from Gaussian noise, holding the current observation latent fixed. Starting from an expert-trained model, we repeatedly collect and score deployment rollouts, then update the model on expert demonstrations and all accumulated rollouts using Equation 6. Each updated model is redeployed with the expert score $q^\star$ to continue the deployment–learning loop.

More broadly, LoopWAM's self-improvement updates can use rollouts from arbitrary deployed policies, even poorly performing ones. The joint architecture and quality-aware objective accommodate their varying trajectory qualities, so each trajectory contributes according to its observed transitions and behavioral quality. This extends self-improvement to learning from deployment experience collected by other policies as well as LoopWAM itself. We validate this broader setting on RoboTwin 2.0 and RoboCasa365 (Section 4.3).

4 Experiments

4.1 Implementation Details

We initialize the video backbone, text encoder, and video VAE from Wan2.2-TI2V-5B (Wan et al., 2025), and use an action DiT with a hidden dimension of 1024. Each training segment pairs a 64-step action chunk with nine RGB frames sampled at intervals of eight control steps, including the initial observation. Multi-camera views are stitched into a single image, and actions and robot states are normalized using expert-data statistics. We train for **five epochs on 96 Alibaba PPU-810E accelerators** using AdamW with a learning rate of 10^{-4}, weight decay of 0.01, cosine learning-rate decay, bfloat16 mixed precision, and gradient clipping at 1.0. Training follows the quality-conditioned joint objective in Section 3.2, and deployment uses the expert score $q^\star$. Additional preprocessing and quality-weight settings are provided in Appendix A.

4.2 Benchmark Experimental Setup

We evaluate LoopWAM on the real-robot benchmark RoboChallenge Table30-V2 and two simulation benchmarks, RoboTwin 2.0 and RoboCasa365. RoboChallenge evaluates closed-loop self-improvement using LoopWAM's own real-robot deployments. The two simulation benchmarks additionally test learning from different deployed policies by augmenting expert demonstrations with rollouts from both expert-trained LoopWAM and others. Across these settings, successful, suboptimal, and failed rollouts expand the state–action–outcome distribution beyond expert demonstrations.

RoboChallenge Table30-V2. RoboChallenge Table30-V2 (RoboChallenge Dexmal Team, 2026) is a standardized real-robot benchmark comprising 30 tabletop manipulation tasks across four embodiments: ALOHA, DOS-W1, ARX5, and UR5. Its tasks test capabilities including tool use, precise manipulation, and coordinated bimanual control. We follow the official multi-task evaluation protocol and report overall performance across embodiments. Starting from one expert-trained multi-task LoopWAM per embodiment, we conduct three successive rounds of closed-loop self-improvement. In each deployment round, we run each model on its assigned tasks and collect about 100 rollout trajectories per task. These trajectories are quality-annotated and combined with expert demonstrations and rollout trajectories from all previous deployment rounds for further training.

RoboTwin 2.0. RoboTwin 2.0 (Chen et al., 2025) comprises 50 bimanual manipulation tasks evaluated in clean and domain-randomized scenes. Following the multi-task setup of LingBot-VA (Li et al., 2026), we use 2,500 expert demonstrations from clean scenes and 25,000 from heavily randomized scenes. We augment these demonstrations with rollouts generated by expert-trained LoopWAM, X-VLA (Zheng et al., 2025), and $\pi_{0.5}$ (Physical Intelligence et al., 2025). For each task, each policy contributes 20 clean and 130 randomized rollouts, yielding 22,500 trajectories across

Table 1: Performance on RoboChallenge Table30-V2. We report per-embodiment success rates (SR, %) and overall performance across four robot embodiments. The overall metric is computed with task weights of 10/10/7/3. Baselines are sorted by overall SR. Bold denotes the best performance, while blue and light blue highlight LoopWAM and the runner-up, respectively.

Method	ALOHA		DOS-W1		ARX5		UR5		Overall	
	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR
debuggers_vla	5.40	0.00	5.15	2.00	53.21	34.29	27.83	23.33	18.72	11.00
W0	12.25	2.00	4.40	0.00	48.64	30.00	49.50	43.33	21.85	12.00
test_01	6.25	1.00	8.95	3.00	49.50	28.57	61.17	43.33	22.73	12.33
holobrain	11.75	3.00	13.30	6.00	53.64	30.00	70.67	43.33	27.93	14.33
Baseline Model	21.55	6.00	17.35	6.00	60.14	32.86	44.83	26.67	31.48	14.33
MagicBot	4.70	0.00	6.85	5.00	47.57	32.86	58.67	53.33	20.82	14.67
mc	10.60	7.00	2.00	0.00	73.64	47.14	22.50	20.00	23.63	15.33
JV0	12.70	5.00	17.80	16.00	49.64	20.00	58.33	43.33	27.58	16.00
my16	18.95	13.00	36.60	27.00	73.29	54.29	56.50	46.67	41.27	30.67
VLA-DM0.5	26.50	18.00	55.45	42.00	79.07	58.57	86.50	70.00	54.42	40.67
LoopWAM (expert-trained)	23.05	8.00	46.35	38.00	37.00	24.29	43.33	26.67	36.10	23.67
LoopWAM	38.65	26.00	54.90	42.00	76.21	55.71	78.33	76.67	**56.80**	**43.33**

Table 2: Performance on RoboTwin 2.0. We report success rates (SR, %) under clean and randomized evaluation settings. Embodied PT. denotes additional embodied-data pretraining before benchmark training. LoopWAM is trained with the mixed expert–rollout corpus described in Section 4.2. Blue and light blue highlight LoopWAM and the runner-up, respectively.

Method	Embodied PT.	Clean	Rand.	Average
π_0 (Black et al., 2024)	✓	65.9	58.4	62.2
X-VLA (Zheng et al., 2025)	✓	72.9	72.8	72.9
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	✓	82.7	76.8	79.8
ABot-M0 (Yang et al., 2026)	✓	81.2	80.4	80.8
Motus (from Wan2.2) (Bi et al., 2025)	✗	77.6	77.0	77.3
Motus (Bi et al., 2025)	✓	88.7	87.0	87.8
LingBot-VA (Li et al., 2026)	✓	92.9	91.5	92.2
Fast-WAM (Yuan et al., 2026)	✗	91.9	91.8	91.8
AHA-WAM (Cai et al., 2026)	✗	93.4	92.2	92.8
LingBot-VA 2.0 (Zhang et al., 2026a)	✓	93.8	93.4	93.6
LoopWAM	✗	**94.6**	**94.4**	**94.5**

all 50 tasks and the three policies. The combined rollout corpus has approximately the same total duration as the expert corpus. Rollout collection and evaluation use different random seeds to avoid data leakage. Rollout quality is annotated using the coarse-level numerical scores in Section 3.2.

RoboCasa365. RoboCasa365 (Nasiriany et al., 2026) is a large-scale simulation framework for household manipulation in diverse kitchen environments. We evaluate on its 50-task benchmark, which includes atomic and composite manipulation tasks. Training combines expert demonstrations with rollouts generated by expert-trained LoopWAM and RLDX-1 (Kim et al., 2026), with the combined rollout duration approximately matching that of the expert data. Both expert and rollout data follow the Human300 training split: the 16 Composite-Unseen tasks are excluded from training and evaluated zero-shot. As in RoboTwin 2.0, training and evaluation data use different random seeds.

4.3 Benchmark Results

RoboChallenge Table30-V2. Table 1 compares expert-trained and final LoopWAM with leading entries from the official leaderboard (RoboChallenge, 2026). After three deployment-and-training rounds, LoopWAM attains an overall score of 56.80 and 43.33% success, exceeding VLA-DM0.5

Table 3: Performance on RoboCasa365. We report success rates (SR, %) across three evaluation splits. The overall metric is computed with task weights of 18/16/16. Baseline results are obtained from the official leaderboard. Embodied PT. follows the definition in Table 2. Blue and light blue highlight LoopWAM and the runner-up, respectively.

Method	Embodied PT.	Atomic Seen	Composite Seen	Composite Unseen	Overall
Diffusion Policy (Chi et al., 2025)	✗	15.7	0.2	1.3	6.1
π_0 (Black et al., 2024)	✓	34.6	6.1	1.1	14.8
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	✓	39.6	7.1	1.2	16.9
GigaWorld-Policy 0.1 (Ye et al., 2026a)	✓	44.4	11.8	2.9	20.7
GR00T N1.6 (NVIDIA et al., 2025)	✓	51.1	9.4	1.7	21.9
GR00T N1.5 (NVIDIA et al., 2025)	✓	50.7	14.8	2.7	23.9
RLDX-1 (Kim et al., 2026)	✓	67.6	27.9	8.5	36.0
PRTS (Zhang et al., 2026b)	✓	66.3	30.3	**18.8**	39.6
ABot-M0.5 (Chen et al., 2026)	✓	75.6	37.7	3.3	40.3
ABot-M0.6 (Yang et al., 2026)	✓	79.4	48.3	7.9	46.6
LoopWAM	✗	**79.7**	**50.3**	14.9	**49.5**



Figure 3: Closed-loop improvement from real-robot deployment. Top: three deployment-and-training iterations on RoboChallenge Table30-V2, covering 30 tasks across four robot embodiments. Bottom: two deployment-and-training iterations on shirt folding after task-specific expert post-training of the DOS-W1 model. Each plot reports success rate and score over successive improvement rounds.

by 2.38 score points and 2.67 percentage points in success, and ranking first on both metrics among the compared methods. Section 4.4 analyzes the successive deployment updates that produce this improvement over expert-only training.

RoboTwin 2.0. LoopWAM achieves 94.58% success in clean scenes and 94.42% under domain randomization, averaging 94.50% (Table 2). Without additional embodied pretraining, it improves over AHA-WAM by 1.70 percentage points and surpasses the pretrained LingBot-VA 2.0 by 0.90 points on average. Together, these results show that joint learning from expert demonstrations and diverse deployment rollouts yields a competitive multi-task policy under both evaluation settings.

RoboCasa365. LoopWAM achieves 79.7%, 50.3%, and 14.9% success on Atomic-Seen, Composite-Seen, and Composite-Unseen, respectively, yielding 49.5% overall success across the 50 tasks (Table 3). It exceeds ABot-M0.6 by 2.9 percentage points overall without additional embodied pretraining. The improvement extends to held-out composite tasks, demonstrating transfer to task compositions absent from both expert and rollout training data.

4.4 Closed-Loop Improvement from Deployment

We examine deployment-driven improvement in two real-robot settings: multi-task learning and single-task post-training (Figure 3). Both follow the update procedure in Section 3.3: the current policy collects rollouts, which are quality-annotated and combined with expert demonstrations and accumulated deployment trajectories for the next update.

Multi-task deployment. As in Section 4.2, we train multi-task models for ALOHA, DOS-W1, ARX5, and UR5, jointly covering the 30 Table30-V2 tasks. Starting from expert-trained checkpoints, each model undergoes three rounds of deployment and training using experience from its assigned tasks. Overall success increases from 23.67% to 28.00%, 37.67%, and 43.33%, while the scores rise from 36.10 to 42.63, 51.65, and 56.80. Both aggregate metrics improve at every iteration, yielding a 19.7-percentage-point success gain over the initial deployment. This demonstrates that LoopWAM can use its own deployment experience to improve policies trained across multiple tasks.

Single-task post-training. We further evaluate shirt folding, a challenging deformable-object manipulation task that requires multiple coordinated steps. We initialize from the DOS-W1 multi-task model trained on Table30-V2 and post-train it on newly collected expert demonstrations of shirt folding. The resulting task-specific expert-trained checkpoint serves as the starting point for two rounds of deployment and self-improvement. Success rises from 13.20% to 32.54% and 55.56%, while the scores increase from 14.44 to 38.89 and 67.78. These gains show that deployment feedback remains useful after adapting a multi-task model to a challenging individual task, supporting closed-loop improvement under both training regimes.

4.5 Ablation Study

We use the RoboTwin 2.0 setup in Section 4.2, varying only quality steering and rollout action supervision; expert action supervision and video supervision remain unchanged.

Table 4 compares no steering, extra-coarse annotation, and our coarse-level annotation (Section 3.2). Extra-coarse uses `Quality: low`, `medium`, or `high` for failed rollouts, successful rollouts, and experts, respectively; our coarse-level scheme assigns numerical scores by policy checkpoint. **(1)** Quality steering is key to learning from mixed-quality deployment rollouts, enabling the model to distinguish trajectories of different quality. **(2)** Even extra-coarse labels remain effective, demonstrating that fine-grained annotation is not required to benefit from steering. **(3)** Quality-aware rollout action supervision further improves overall performance under both annotation schemes.

Table 4: Ablation study on RoboTwin 2.0. Act. denotes quality-weighted rollout action supervision.

Steering	Act.	Clean	Rand.	Avg.
None	✗	73.5	71.1	72.3
Extra-coarse	✗	92.2	93.3	92.8
Extra-coarse	✓	93.9	93.3	93.6
Coarse	✗	93.2	92.2	92.7
Coarse (ours)	✓	**94.6**	**94.4**	**94.5**

5 Conclusion

We presented LoopWAM for improving generalist world-action policies directly from deployment experience. The key is to learn physical consequences from trajectories across the quality spectrum while controlling how their actions supervise behavior. Source-dependent noise sampling, quality steering, and quality-weighted supervision implement this distinction within joint video–action denoising, allowing consequence representations to participate directly in action generation. The same formulation accepts rollouts from both LoopWAM and other deployed policies. Experiments show the best overall results: a score of 56.80 and 43.33% success on RoboChallenge Table30-V2, 94.5% average success on RoboTwin 2.0, and 49.5% overall success on RoboCasa365. Repeated real-robot deployment raises Table30-V2 success by 19.7 percentage points over three rounds and improves shirt-folding success from 13.20% to 55.56% over two rounds after single-task post-training. These findings support using mixed-quality interaction history as a training resource for continued improvement of both multi-task policies and policies adapted to individual tasks.

6 Reproducibility Statement

We provide training details, data processing procedures, optimization configurations, and qualitative evaluation examples throughout this appendix.

The implementation details in Section A describe the trajectory processing, quality weighting strategy, temporal alignment procedure, and inference noise schedule used by LoopWAM.

The reported results are generated using the final LoopWAM configuration described in the main paper and this appendix.

Reproducing the complete training and evaluation pipeline requires the corresponding datasets, robot environments, and implementation dependencies.

7 AI Use Statement

Generative AI tools assisted with language editing, figure organization, and consistency checking.

The authors are responsible for the manuscript content, experimental design, implementation, and reported results.

References

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A Unified Latent Action World Model. *arXiv preprint arXiv:2512.13030*, 2025.

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

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. AHA-WAM: Asynchronous Horizon-Adaptive World-Action Modeling with Observation-Guided Context Routing. *arXiv preprint arXiv:2606.09811*, 2026.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, et al. ABot-M0.5: Unified Mobility-and-Manipulation World Action Model. *arXiv preprint arXiv:2607.00678*, 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.

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. *IJRR*, 2025.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025.

Norman L. Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous Univariate Distributions*, volume 2. John Wiley & Sons, 2nd edition, 1995. ISBN 9780471584940.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A Large-Scale In-the-Wild Robot Manipulation Dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. RLDX-1 Technical Report. *arXiv preprint arXiv:2605.03269*, 2026. URL https://arxiv.org/abs/2605.03269.

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.

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.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A Latent World-Action Model from Egocentric Videos. *arXiv preprint arXiv:2605.00078*, 2026. URL https://arxiv.org/abs/2605.00078.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. *arXiv preprint arXiv:2603.04356*, 2026.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025.

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-Action Models for Generalizable Robot Control Beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025.

Quanquan Peng, Yutong Liang, Rui Yan, Nicklas Hansen, and Xiaolong Wang. FACT: Failure-Aware Causal Training for World-Action Models. *arXiv preprint arXiv:2608.10232*, 2026.

Physical Intelligence. $\pi_{0.6}$: A VLA That Learns From Experience. *arXiv preprint arXiv:2511.14759*, 2025.

Physical Intelligence. $\pi_{0.7}$: A Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

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*, 2025.

RoboChallenge. RoboChallenge Table30-V2 Leaderboard, 2026. URL https://robochallenge.cn/leaderboard?selectBenchmark=table30_v2&currentPage=1&pageSize=30. Accessed September 22, 2026.

RoboChallenge Dexmal Team. Table30 V2: Evaluating Generalized Models by Real Robots at Scale. Technical report, Dexmal, 2026. URL https://robochallenge.ai/robochallenge_techreport_V2.pdf.

Stephane 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*, volume 15, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

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.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-Improving Vision-Language-Action Models with Data Generation via Residual RL. *arXiv preprint arXiv:2511.00091*, 2025.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL Token: Bootstrapping Online RL with Vision-Language-Action Models. *arXiv preprint arXiv:2604.23073*, 2026. URL https://arxiv.org/abs/2604.23073.

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*, 2026.

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.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native Video-Action Pretraining for Generalizable Robot Control. *arXiv preprint arXiv:2607.08639*, 2026a. URL https://arxiv.org/abs/2607.08639.

Yang Zhang, Jiangyuan Zhao, Chenyou Fan, Fangzheng Yan, Tian Li, Haitong Tang, Sen Fu, Xuan'er Wu, Qizhen Weng, Weinan Zhang, Xiu Li, Chi Zhang, Chenjia Bai, and Xuelong Li. PRTS: A Primitive Reasoning and Tasking System via Contrastive Representations. *arXiv preprint arXiv:2604.27472*, 2026b.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. *arXiv preprint arXiv:2304.13705*, 2023.

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. *arXiv preprint arXiv:2510.10274*, 2025.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-WM: A Unified Video-Action World Model for Robotic Manipulation. *arXiv preprint arXiv:2606.01027*, 2026.

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, 2025. URL https://arxiv.org/abs/2504.02792.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Training and Data Processing Details

A.1 Trajectory-Quality Annotations

We use two numerical quality scales according to annotation granularity. *Episode-level annotation* uses a 1–12 scale, with rollout scores in $[1, 10]$ and expert demonstrations assigned $q^\star = 12$. *Coarse-level annotation* uses a 1–5 scale, with rollout scores in $\{1, 2, 3, 4\}$ and expert demonstrations assigned $q^\star = 5$. In all settings, every failed trajectory receives $q = 1$. Expert demonstrations are treated as optimal reference trajectories and assigned scores above the rollout range; the same expert score is used at deployment. Each score is assigned to a complete trajectory and inherited by all training segments extracted from it.

RoboChallenge Table30-V2. We use human episode-level annotations from the official RoboChallenge platform. Platform operators already rate each rollout on a 1–10 scale according to task completion, with successful trajectories typically receiving scores of at least 8.5. We directly use these platform ratings as rollout quality scores, except that all failed trajectories are assigned $q = 1$. Expert demonstrations receive $q^\star = 12$.

Single-task shirt-folding post-training. We likewise use human episode-level annotation. After each rollout, the data collector assesses task completion before resetting the scene and assigns a score on a 1–10 scale, following the same completion-based scoring approach as RoboChallenge. Failed trajectories receive $q = 1$, and task-specific expert demonstrations receive $q^\star = 12$.

RoboTwin 2.0. We apply coarse-level annotation separately to rollouts from expert-trained LoopWAM, X-VLA, and $\pi_{0.5}$. For each policy, we compute its overall rollout success rate across all tasks and manually inspect approximately 0.7% of its trajectories for motion smoothness and action accuracy. These two assessments determine a single shared score for all successful trajectories from that policy across tasks; every failed trajectory receives $q = 1$. Expert-trained LoopWAM achieves approximately 90% success during rollout collection and produces relatively smooth trajectories, so its successful rollouts receive $q = 4$. X-VLA and $\pi_{0.5}$ each achieve less than 50% success and produce less smooth trajectories, so their successful rollouts receive $q = 3$. Expert demonstrations receive $q^\star = 5$.

RoboCasa365. We use the same coarse-level protocol, assessing each source policy through its overall rollout success rate across tasks and manual inspection of approximately 0.7% of its trajectories for motion smoothness and action accuracy. We obtain two expert-trained LoopWAM policies using 64-step and 32-step action chunks, denoted Policy 1 and Policy 2, respectively, and also collect rollouts from RLDX-1. Following the same scoring criteria as RoboTwin, all successful rollouts from Policy 1 share $q = 3$, those from Policy 2 share $q = 4$, and those from RLDX-1 share $q = 2$. All failed rollouts receive $q = 1$, and expert demonstrations receive $q^\star = 5$.

A.2 Training Details

Temporal alignment and robot representations. The RoboTwin and W1 training configurations use an action horizon of $H = 64$ and a visual stride of eight control steps, yielding nine RGB frames including the initial observation. Camera views are stitched into a single image, and robot states and actions are normalized using expert-data statistics. For W1, absolute joint targets are converted to offsets from the initial state of the chunk before normalization; the inverse transformation restores robot coordinates at execution. To reduce redundant stationary data, the W1 pipeline checks unnormalized state sequences with an absolute tolerance of 10^{-3} and resamples nearly static windows up to three times. Windows starting within the final 20 seconds of expert episodes are exempt from this filtering to retain terminal behavior. Padding is excluded from the denoising losses.

Noise schedules and loss weighting. The base noise sampler uses $\phi_5(u) = 5u/(1 + 4u)$ for $u \sim \mathcal{U}(0, 1)$. Samples drawn from the Beta distributions in Equation 4 are clipped to $[10^{-4}, 1 - 10^{-4}]$ without a further time shift. Both modalities use the same scheduler weighting function in

Equation 5:

$$g(\sigma) = \exp\left[-2(\sigma - \tfrac{1}{2})^2\right] - e^{-1/2},$$
$$\omega(\sigma) = \frac{g(\sigma)}{C + 10^{-10}}, \qquad C = \frac{1}{T}\sum_{j=0}^{T-1} g(\phi_5(1 - j/T)), \quad T = 1000. \tag{8}$$

This normalization is shared by expert and rollout samples. For K inference steps, the schedule is $\sigma_k = \phi_5(1-k/K)$, $k = 0, \ldots, K$. Each modality follows the Euler update $X_m^{k+1} = X_m^k + (\sigma_{k+1} - \sigma_k)v_\theta^m$, with the vector fields evaluated jointly and the initial observation latent fixed throughout sampling.

B Final Recipe and Result Provenance

This section summarizes the final LoopWAM configuration used for training and evaluation, and clarifies the provenance of reported experiments.

All quantitative and qualitative results are generated using the final LoopWAM model configuration described in the main paper and Appendix A. The reported experiments use the same training objective, data processing pipeline, and inference procedure unless otherwise specified.

The qualitative examples shown in this appendix are selected from robot executions collected on held-out environments. These examples are used only for visualization and are not included in the training set.

The reported results reflect the final model checkpoint and evaluation configuration. This appendix provides additional implementation details to facilitate reproduction of the training and evaluation pipeline.

C Failure Recovery and Iterative Execution

Real-world robot manipulation often involves uncertainty caused by imperfect perception, inaccurate execution, and complex object interaction dynamics.

To analyze the ability of LoopWAM to support iterative execution, Figure 4 presents qualitative examples where an initial attempt does not complete the desired task.

After observing the resulting environment state, the model generates subsequent corrective behaviors that improve the execution outcome. These examples highlight the importance of modeling future states and environment transitions beyond predicting isolated actions.

D Qualitative Future Observation Prediction

As a learned robot world model, LoopWAM predicts future visual observations conditioned on the current robot state and interaction history.

To further examine the learned visual dynamics, Figure 5 compares predicted future frames with ground-truth observations from robot executions.

The predictions capture major aspects of scene evolution, including robot motion, object movement, and manipulation progress. Although generated observations may differ from the ground-truth images in fine visual details, they preserve the temporal structure of the interaction process.

E Representative Robot Platforms and Tasks

LoopWAM is evaluated across multiple robot platforms with different embodiments, sensing configurations, and manipulation spaces.

To demonstrate the diversity of learned robot behaviors, we provide additional qualitative examples from ARX, ALOHA, UR5, and W1 platforms. The examples cover object manipulation, coordinated control, fine-grained interaction, and long-horizon tasks.



Figure 4: Qualitative examples of iterative execution and failure recovery. The first attempt may fail due to execution uncertainty, incomplete interaction, or object displacement. Subsequent attempts generate corrective behaviors and achieve successful task completion.

Figure 6 shows representative tasks on the ARX platform. Figure 7 presents examples from the ALOHA platform, which includes coordinated manipulation behaviors. Figure 8 demonstrates single-arm manipulation tasks on the UR5 platform. Finally, Figure 9 presents representative long-horizon tasks from the W1 platform.



Figure 5: Qualitative comparison between predicted future observations from LoopWAM and ground-truth robot executions. The model captures the temporal evolution of robot-object interactions and predicts plausible future visual states.

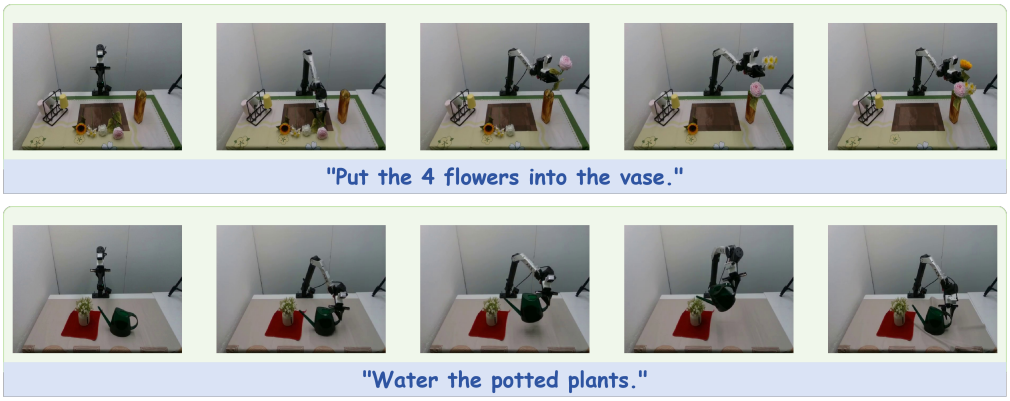

Figure 6: Representative tasks on the ARX robot platform. The examples illustrate diverse manipulation scenarios used for evaluating LoopWAM across different robot embodiments.

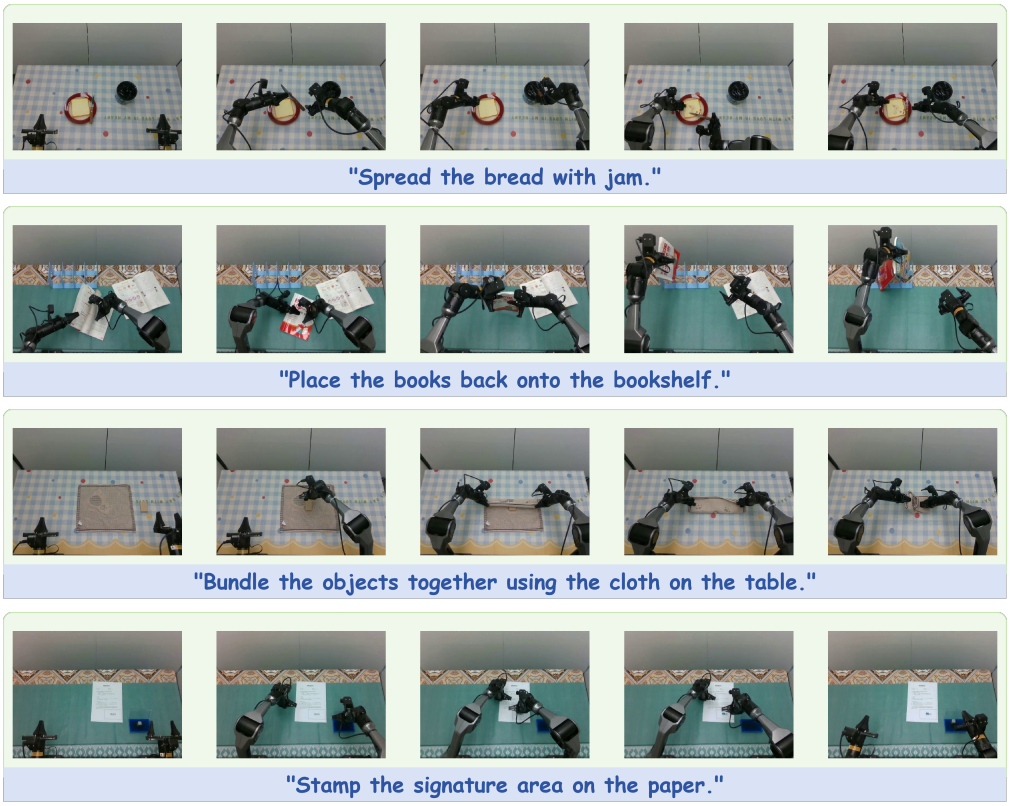

Figure 7: Representative tasks on the ALOHA robot platform. The examples include coordinated manipulation and object interaction requiring temporally consistent control.

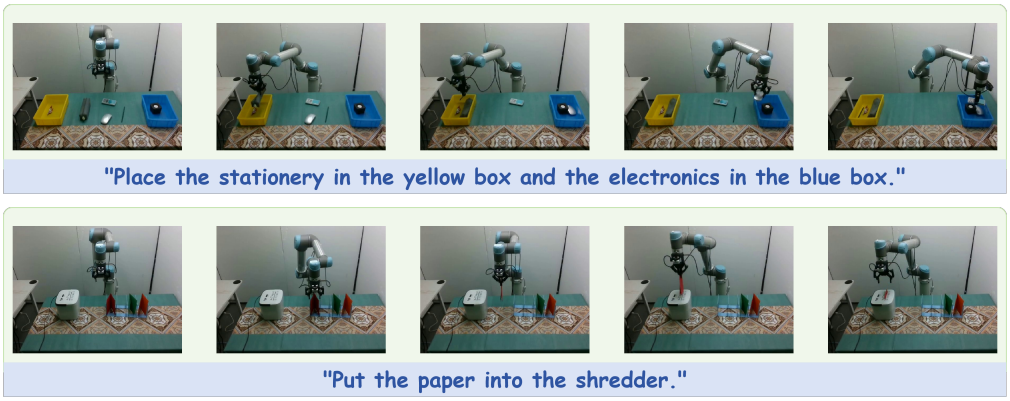

Figure 8: Representative tasks on the UR5 robot platform. The examples demonstrate single-arm manipulation in structured environments.

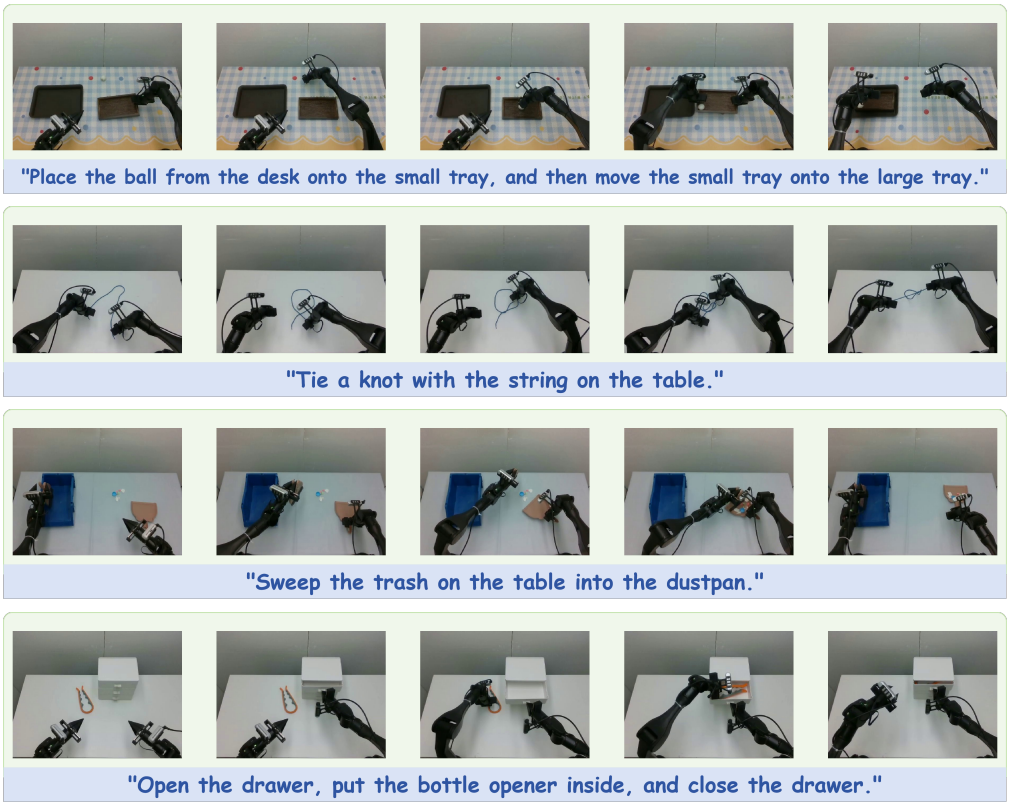

Figure 9: Representative long-horizon tasks on the W1 robot platform. The examples demonstrate sequential manipulation behaviors in complex environments.