DREAMCHASER: A SELF-IMPROVING FRAMEWORK FOR WORLD ACTION MODELS VIA VIDEO-GUIDED ACTION REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) built on pre-trained video generators learn video prediction and action generation under asymmetric supervision. The video branch inherits dense spatiotemporal supervision from large-scale video pre-training, whereas the action branch relies on comparatively limited paired action annotations. We hypothesize that this asymmetry in data scale contributes to a capability gap: a WAM can predict plausible task outcomes that its generated actions fail to realize. We introduce DreamChaser, a self-improving framework that uses the model's video predictions to guide action refinement and acquire additional action supervision through interaction. A lightweight refiner, trained solely on the original demonstrations, uses predicted visual outcomes as targets for correcting local execution deviations. DreamChaser selectively aggregates the resulting rollout data and combines them with the original demonstrations to fine-tune the WAM. This process converts video-guided corrections into additional action supervision, forming an iterative loop of interaction and policy updates without further human demonstrations or an external expert during rollout. On LIBERO, DreamChaser fine-tuning increases FastWAM-joint's average success rate from 97.3% to 97.9% when evaluated without the refiner, suggesting that the collected experience improves the WAM's action policy. On LIBERO-plus, three rounds of self-improvement increase the mean task success from 72.1% to 80.1% across ten tasks. On a real AgileX PiperX, DreamChaser improves the FastWAM-joint average success score on cup sleeving, plate placement, and bottle straightening from 27.0% to 41.7% after two online rounds.

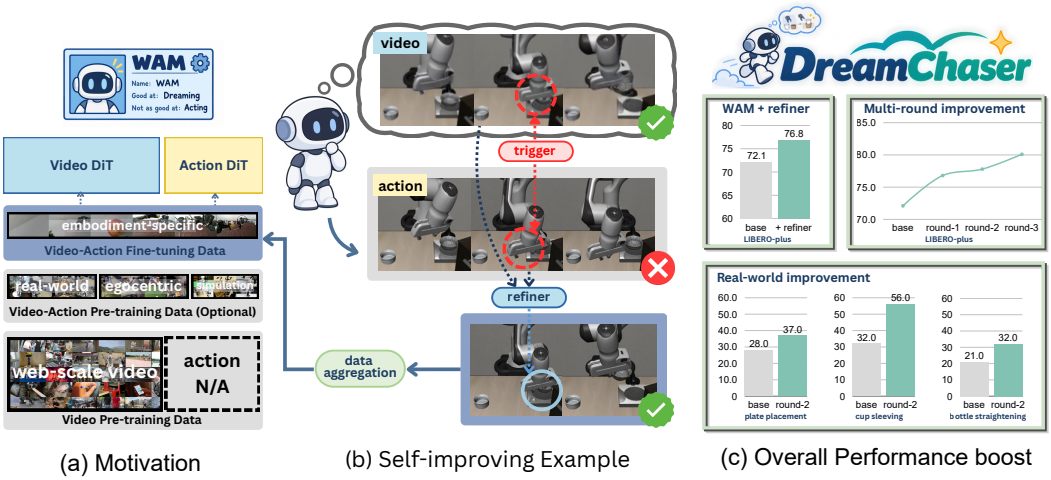
1 INTRODUCTION

World Action Models (WAMs) have demonstrated strong capabilities in robotic manipulation, grounded in jointly modeling future dynamics with actions (Du et al., 2023; Hu et al., 2024; Pai et al., 2025; Ye et al., 2026b; Bi et al., 2026; Yuan et al., 2026; Shen et al., 2026). This line of work builds on the broader world-modeling and video-prediction view of control (Ha & Schmidhuber, 2018; Finn et al., 2016). Video-based WAMs typically combine a pre-trained video model with an action-generation component learned from paired observations and controls (Bi et al., 2026; Yuan et al., 2026). Their appeal is that visual experience can inform control even when much of the pre-training data lacks robot action annotations (Chen et al., 2025; Zhu et al., 2025).

However, the ability to predict a plausible visual outcome does not ensure that a policy can execute the actions needed to realize it. Prior work has identified a video–action generalization gap (Mishra et al., 2026; Liu et al., 2026a), and our motivating observation is that predicted and executed outcomes can diverge during manipulation and error recovery. As shown in 1(b), WAM predicts a successful video of grasping the bowl, however, the trajectory drifts and ultimately leads to task failure. Such discrepancies create an opportunity to use a predicted outcome as a target for local correction.

We view this gap through the lens of asymmetric supervision. Video pre-training provides supervision over diverse scenes and temporal changes (Wan et al., 2025), whereas learning embodiment-specific controls requires paired action annotations, which are less widely available (Wang et al.,

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.



(a) Motivation (b) Self-improving Example (c) Overall Performance boost

Figure 1: **Overview of DreamChaser. (a) Motivation.** WAM is better at generating plausible videos than executable actions. We view this gap through asymmetric supervision. **(b) Visualization of the self-improving pipeline.** The trigger is activated when detecting a mismatch in predicted video and actual observation after executing actions. The refiner publishes action locally, and uses successful rollouts for subsequent self-improve short chunk, saving the task from failure to success. **(c) The performance boost across simulation and real-world experiments.**

2026b). We hypothesize that this asymmetry can contribute to the gap between visual prediction and execution. The practical question is whether the visual knowledge already available in a WAM can provide useful guidance for acquiring additional action-learning experience. Expert-driven data aggregation, such as DAgger and HG-DAgger (Ross et al., 2011; Kelly et al., 2019), addresses states visited by the learner using external corrections. We instead ask: *Can a WAM use its own visual predictions to guide local corrections and collect useful training data without a corrective expert during rollout?*

We introduce **DreamChaser**, a loop of rollout, refinement, selective data aggregation, and WAM fine-tuning (Fig. 2). Starting from the available expert demonstrations, we train the base WAM and a lightweight, visual goal-conditioned action refiner. After executing a WAM action chunk, a task-calibrated trigger compares its predicted visual outcome with the observed result. When the discrepancy is unusually large, the refiner generates a short correction toward the predicted target, after which control returns to the WAM. DreamChaser retains ordinary chunks when the trigger is inactive and accepts refined chunks only when correction reduces the visual discrepancy. The retained interaction data are mixed with the original expert demonstrations to update the WAM. Thus, video predictions guide data acquisition, while actual interaction provides the observations and executed actions used for subsequent learning (Liu et al., 2026a). This transfers the immediate benefit of a refiner from the improvement retained by the updated policy.

Experiments examine both effects on LIBERO, an adapted LIBERO-plus subset, and a physical robot. Without inference-time refinement, the mean LIBERO-plus success rate increases from 72.1% to 80.1% over three rounds, and the mean real-world task-progress score increases from 27.0% to 41.7% over two rounds. Our contributions are:

- We introduce a formulation of WAM improvement that uses predicted visual outcomes as local correction targets and converts the resulting interaction into additional action supervision.
- We propose a modular framework combining a video-action discrepancy trigger, a refiner trained on existing demonstrations, and selective chunk aggregation with expert-data replay.
- We evaluate DreamChaser in simulation and on a physical robot. Across ten adapted LIBERO-plus tasks, three online rounds raise mean success from 72.1% to 81.3% without inference-time refinement; across three real-world tasks, two rounds raise the mean normalized success score from 27.0% to 41.7%.

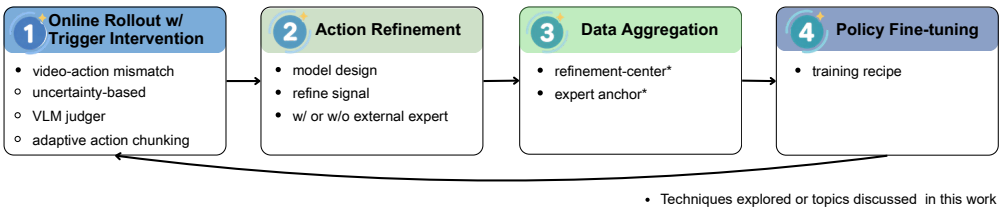


Figure 2: A modular framework for WAM's self-improvement. At each round, the WAM rolls out actions, the trigger detects unreliable execution, the refiner generates corrective actions, and successful segments are aggregated for the next fine-tuning round.

2 Related Work

2.1 World Action Model (WAM)

World Action Models (WAMs) (Wu et al., 2024; Ye et al., 2026b; Team et al., 2024; Ye et al., 2026a; Wang et al., 2026a; Chen et al., 2026; Gu et al., 2026; Liu et al., 2026b) couple future world dynamics with action generation. UniPi (Du et al., 2023) and Large Video Planner (Chen et al., 2025) adopt a two-stage pipeline that first generates future videos and then decodes them into actions. Other methods, such as Mimic-Video (Pai et al., 2025) and Genie Envisioner (Liao et al., 2026), use intermediate representations of video generation to condition action prediction without fully rendering future videos. More recent works, including Unified World Model (Zhu et al., 2025), Motus (Bi et al., 2026), and Fast-WAM (Yuan et al., 2026), more tightly couple video and action generation through shared or joint modeling, enabling a more precise world modeling and action learning. This direction is related to the broader use of learned visual dynamics for planning and control (Ha & Schmidhuber, 2018; Finn et al., 2016).

2.2 Policy Improvement through Interaction

Reinforcement learning improves policies using rewards or value estimates. SimpleVLA-RL (Li et al., 2026) applies reinforcement learning to VLAs, and flow policy gradients (Yi et al., 2026) enable reward-based optimization of flow policies without explicit action likelihoods. RECAP, used for $\pi^*_{0.6}$ (Intelligence et al., 2025), combines demonstrations, autonomous experience, and teleoperated interventions through advantage-conditioned learning. RISE (Yang et al., 2026) instead uses a compositional world model and a progress-value model to improve policies from imagined rollouts. DreamChaser uses supervised fine-tuning on selected environment interactions rather than a reward-based policy-gradient update.

Data aggregation offers another route. DAgger (Ross et al., 2011) queries an expert on learner-visited states; HG-DAgger (Kelly et al., 2019) and Diff-DAgger (Lee et al., 2025) use selective intervention to reduce expert effort. WM-DAgger (Yu et al., 2026) synthesizes recovery data with a world model and filters it using consistency with real demonstration frames. DreamChaser also targets the cost of corrective supervision, but its refiner acts in the environment toward the current WAM's predicted target, and acceptance depends on the observed change in visual discrepancy. This distinction concerns the source and validation of correction data, rather than claiming that expert-free data aggregation is itself new.

3 Method

3.1 Problem Formulation

Let o_t, s_t, and l denote the visual observation, robot proprioceptive state, and language instruction. A WAM produces an H-step action chunk $A_t = a_{t:t+H-1}$ together with a predicted observation sequence $\hat{O}_t = \hat{o}_{t+1:t+H}$, represented abstractly as

$$(\hat{O}_t, A_t) \sim p_\theta(\hat{O}_t, A_t \mid o_t, s_t, l). \tag{1}$$

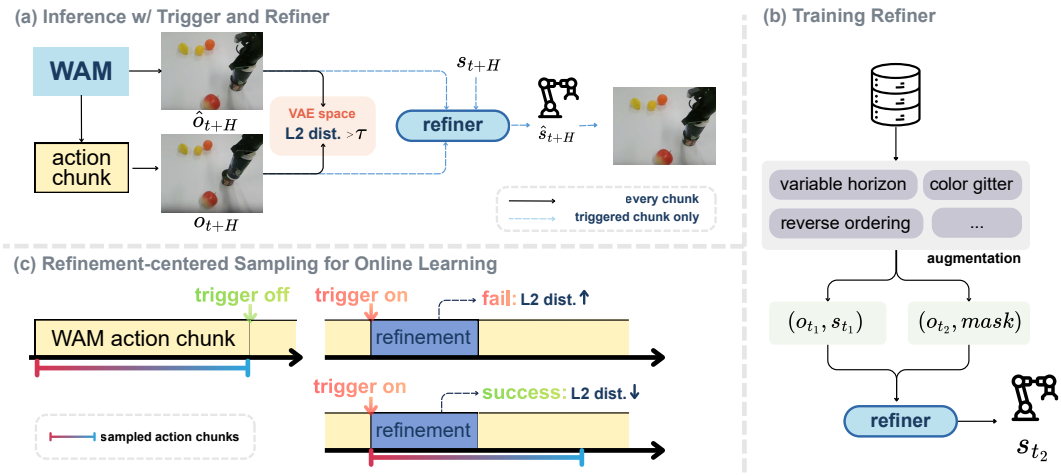


Figure 3: Method details. (a) The video–action mismatch trigger compares the predicted and observed future in the latent space, and call for refiner if trigger is on. (b) Training refiner with augmentation. (c) The refined segment is selectively aggregated and used for online fine-tuning, with three possible situations.

This notation allows different couplings between video and action generation. It does not imply that every WAM renders video during ordinary action inference. For DreamChaser, however, we requires a predicted visual target during collection.

We use the endpoint $v_t = \hat{o}_{t+H}$ as the local target and write π_{θ_i} for the WAM action policy at online round i. The initial policy is fine-tuned on demonstrations $\mathcal{D}_{\text{exp}}$. A separate refiner R_ϕ is trained on the same data and then frozen. The WAM first executes $a_{t:t+H}$. At time $\tau = t + H$, a trigger g_t decide whether to invoke the refiner.

3.2 DreamChaser Framework

At round i, the current WAM π_{θ_i} collects environment rollouts with mismatch-triggered corrections. The controller executes a WAM chunk, checks its endpoint against the predicted target, and, if triggered, executes a corrective segment before returning control to the WAM. The resulting experience is filtered at the chunk level and combined with the original demonstrations for the next policy update. Appendix A.1 specifies the order of operations. The trigger, refiner, and aggregation rule are separate components, while the reported implementation uses the choices below.

3.3 Video-Action Mismatch Trigger

Let E be the VAE encoder. After executing A_t, we measure the L2 distance between the predicted and actual observations

$$d_t^{\text{pre}} = \|E(o_{t+H}) - E(v_t)\|_2. \tag{2}$$

A task-specific reference set $\mathcal{C}_q = \{d_j^{(q)}\}_{j=1}^{N_q}$ contains discrepancies from nominal rollouts of task q. To account for different error scales, we compute an empirical upper-tail rank, following the intuition of rank-based conformal calibration (Angelopoulos & Bates, 2023), and activate the trigger when it falls below a shared threshold α:

$$p_t = \frac{1 + \sum_{j=1}^{N_q} \mathbf{1}[d_j^{(q)} \geq d_t^{\text{pre}}]}{N_q + 1}, \qquad g_t = \mathbf{1}[p_t \leq \alpha]. \tag{3}$$

A small p_t identifies an unusually large discrepancy relative to the reference rollouts.

3.4 ACTION REFINER

When the trigger is activated, the refiner predicts a local robot-state target u_t from the current proprioception s_τ, observed image o_τ, and WAM-predicted visual target v_t. The target is converted to a short action sequence through the robot's target-state control interface:

$$u_t = R_\phi(s_\tau, o_\tau, v_t), \qquad A_t^R = \mathrm{Interp}(s_\tau, u_t; H_R), \quad 0 < H_R < H. \tag{4}$$

Here $A_t^R = a^R_{\tau:\tau+H_R-1}$ is an executable correction, whereas s_τ is a measured state and u_t is a target. We use a ResNet-18-based regressor (He et al., 2016; Black et al., 2023; Walke et al., 2023) for R_ϕ.

To enable refinement without additional expert demonstrations or recovery trajectories, the refiner is trained entirely from observation–action pairs already available in the original WAM training data. However, these data mainly consist of successful transitions and provide limited coverage of deviations. We therefore augment the training data by sampling pairs with variable horizons, visual perturbations, and temporal reversal, exposing the refiner to a broader range of local deviations during training. We show the training procedure in Fig. 3(b).

3.5 SELECTIVE DATA AGGREGATION FOR ONLINE LEARNING

We accept a refinement only if it improves the visual alignment with the target. Keeping the visual target v_t fixed, we compare the discrepancy before and after the H_R corrective actions:

$$d_t^{\mathrm{post}} = \|E(o_{\tau+H_R}) - E(v_t)\|_2, \qquad c_t = \mathbf{1}[d_t^{\mathrm{post}} < d_t^{\mathrm{pre}}]. \tag{5}$$

If the correction is accepted, we construct an H-step training window by concatenating the corrective actions with the subsequent WAM actions:

$$A_\tau^{\mathrm{ref}} = [a^R_{\tau:\tau+H_R-1};\, a^{\mathrm{WAM}}_{\tau+H_R:\tau+H-1}], \tag{6}$$

where $[\cdot;\cdot]$ denotes temporal concatenation. The window starts at τ, immediately after the chunk that triggered the correction. Thus, the corrective actions are used to guide future learning without modifying the actions executed before the trigger.

Each training window is represented as $Z_u = (o_u, s_u, l, a_{u:u+H-1}, o_{u+1:u+H})$, which includes the initial observation, conditioning state, language instruction, executed actions, and resulting observations. At round i, we selectively retain windows based on whether a correction was triggered and, if so, whether it improved the visual alignment:

$$\mathcal{D}^{(i)}_{\mathrm{online}} = \{Z_t : g_t = 0\} \;\cup\; \{Z_{t+H} : g_t = 1,\; c_t = 1\}, \tag{7}$$

where actions in accepted refined windows are given by Eq. 6. This selection retains ordinary windows when no correction is triggered and adds refined windows when the correction is successful. We then combine the retained online data with the original demonstrations to form the training dataset for the next round:

$$\mathcal{D}^{(i+1)}_{\mathrm{train}} = \mathcal{D}_{\mathrm{exp}} \cup \mathcal{D}^{(i)}_{\mathrm{online}}. \tag{8}$$

For parameter-efficient adaptation, we use LoRA (Hu et al., 2021) and assign a higher rank to the action module than to the video module, $r_a > r_v$, allowing more adaptation capacity for action learning. The resulting policy $\pi_{\theta_{i+1}}$ is then used to collect data in the next round. Details of the sampling mixture, optimization settings, and rollout-window conventions are provided in Appendix A.4.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Our experiments are designed to answer three questions: First, can the trigger and the action refiner improve execution without changing the WAM? Second, can fine-tuning on collected experience improve the WAM when evaluated without the refiner? Third, does the improvement remain stable across multiple online rounds and transfer to physical robots? We therefore report a single-round comparison on LIBERO, a three-round study on LIBERO-plus, and a real-world study.

Table 1: LIBERO success rates (%). Fine-tuning gains compare models without the refiner to FastWAM; refiner gains compare each model with and without the refiner. Both are absolute changes in percentage points.

Method	Spatial	Object	Goal	Long	Avg.
FastWAM-joint	97.8	99.8	96.4	95.2	97.3
+ Refiner	98.0	99.8	97.0	96.2	97.8
Refiner gain (pp)	↑ 0.2	0.0	↑ 0.6	↑ 1.0	↑ 0.5
Vanilla-FT	95.6	99.8	97.6	92.8	96.5
Fine-tuning gain (pp)	↓ 2.2	0.0	↑ 1.2	↓ 2.4	↓ 0.8
+ Refiner	95.8	100.0	97.2	96.0	97.2
Refiner gain (pp)	↑ 0.2	↑ 0.2	↓ 0.4	↑ 3.2	↑ 0.7
DreamChaser-FT	98.0	100.0	97.0	96.6	97.9
Fine-tuning gain (pp)	↑ 0.2	↑ 0.2	↑ 0.6	↑ 1.4	↑ 0.6
+ Refiner	97.8	100.0	97.8	96.6	**98.0**
Refiner gain (pp)	↓ 0.2	0.0	↑ 0.8	0.0	↑ 0.1

We follow a fixed training protocol for both the base FastWAM-joint and the action refiner across all experiments. For simulation experiments, the base WAM is trained on the standard dataset for 30K steps. For real-world experiments, we collect 300 task-specific expert trajectories and fine-tune the base WAM for 30K steps.

A ResNet-18-based action refiner (He et al., 2016) is trained separately on the same demonstrations, using randomly sampled temporal horizons shorter than H and the augmentations in Sec. 3.4. It remains frozen during all online rounds.

Multi-round protocol. For LIBERO, we perform one online improvement round. For LIBERO-plus, we perform three rounds. Each task in LIBERO-plus evaluation contains between 40 and 60 evaluation trajectories; the same task-specific evaluation set is used for rollout and evaluation in every round. Initial states and scene layouts are held fixed across rounds, while each update starts from the previous checkpoint and trains on the new data for five epochs, during which expert data is sampled with a fixed rate. The changed initial states and scene layouts in LIBERO-plus are part of the benchmark construction and are not treated as held-out data during this iterative adaptation protocol.

Metrics. Simulation results use task success rate. We report the average success over the four LIBERO suites and task-wise success rates for LIBERO-plus. For real-world experiments, we use a normalized task success score that is more sensitive to variations in the model's capabilities. Definition and task-specific completion criteria are provided in the appendix.

4.2 Simulation Experiments

Benchmarks. We evaluate our method on two simulation benchmarks, LIBERO (Liu et al., 2023) and LIBERO-plus (Fei et al., 2025). LIBERO follows the standard evaluation setting. We fine-tune a LoRA adapter for each task suite. For LIBERO-plus, we construct a subset for each LIBERO-10 task by combining the Initial Robot States suite, which initializes the task from diverse robot states, and the Scene Layouts suite, which shuffles the original object arrangements, and we group the resulting tasks according to their instructions. We fine-tune a LoRA adapter for each instruction, and evaluate the resulting policy under the corresponding setting.

Baselines. For simulation experiments, we compare DreamChaser with three baselines: (a) the base FastWAM-joint, (b) the base WAM equipped with the action refiner but without online fine-tuning, and (c) Vanilla-FT, which fine-tunes the WAM using its own online rollouts without the proposed trigger, refiner, selective aggregation, but with expert-data replay. LIBERO reports one improvement round, whereas LIBERO-plus reports three successive rounds.

Table 2: Task-wise success rates (%) on adapted LIBERO-plus. Initial is the starting WAM; R1–R3 are successive online improvement rounds. Each round is evaluated without and with the action refiner (+R). Fine-tune gain is relative to Initial under the same refiner setting. Refiner gain is relative to the current WAM model. Task labels are abbreviated.

	Initial		Round 1		Round 2		Round 3	
Task	WAM	+R	WAM	+R	WAM	+R	WAM	+R
Moka, both	79.7	84.7	81.4	91.5	88.1	88.1	89.8	91.5
Black bowl, drawer	63.0	83.3	75.9	77.8	77.8	85.2	79.6	91.5
Mug, microwave	68.3	71.7	73.3	76.7	70.0	73.3	76.7	81.6
Book, caddy	90.6	94.3	100.0	96.2	96.2	100.0	100.0	96.2
Two mugs, plates	54.9	58.8	56.9	70.6	62.7	64.7	66.7	70.6
Stove, moka	86.0	80.7	91.2	89.5	89.5	82.5	89.5	86.0
Cream cheese, butter	77.6	77.6	79.6	75.5	75.5	77.6	79.6	75.5
Mug, pudding	76.3	78.0	71.2	88.1	81.4	84.7	81.4	84.7
Soup, cream cheese	58.3	58.3	56.2	60.4	58.3	58.3	60.4	62.5
Soup, tomato sauce	66.1	80.4	82.1	80.4	78.6	78.6	76.8	78.6
Average	72.1	76.8	76.8	80.7	77.8	79.3	80.1	81.9
Fine-tune gain (pp)	–	–	↑ 4.7	–	↑ 5.7	–	↑ 8.0	–
Refiner gain (pp)	–	↑ 4.7	–	↑ 3.9	–	↑ 1.5	–	↑ 1.8

Results. Table 1 reports the results on LIBERO. We compare the base FastWAM-joint with vanilla fine-tuning and our DreamChaser-based fine-tuning, evaluated both without and with the action refiner. Results show that the action refiner improves average success across all three models: the original base model, Vanilla-FT, and DreamChaser-FT. Compared with the base model, Vanilla-FT leads to a decrease in success rate, particularly on long-horizon tasks. We attribute this degradation to the accumulation of sub-optimal behaviors during rollout, which may introduce additional bias when the model is fine-tuned on its own trajectories. In contrast, DreamChaser-FT improves the success rate over the base model.

Table 2 reports the results on adapted LIBERO-plus. Across all three rounds of self-improvement, the action refiner consistently improves the average success rate, although its effect varies across tasks. For example, the refiner yields an 11.9% improvement on "put the black bowl in the bottom drawer of the cabinet and close it," while its effect is negative on "put both the cream cheese box and the butter in the basket" (-4.1 %) and "turn on the stove and put the moka pot on it" (-3.5 %). This variation reflects differences in recovery difficulty and the optimal trigger thresholds across tasks. Meanwhile, the overall self-improvement pipeline consistently increases the average success rate across rounds, reaching 80.1% without the refiner and 81.9% with the refiner.

The extent of improvement through multiple rounds generally correlates with the gains brought by the refiner, while a mild improving trend can still be observed in tasks where the refiner provides limited direct gains. This highlights the steering effect of the refiner. Useful video predictions can steer failed rollouts toward successful behaviors and provide additional successful action data. Although refinement may occasionally perturb successful rollouts, our aggregation strategy retains only successfully refined trajectories, preventing disturbing chunks from being propagated to subsequent rounds. Thus, even when the refiner provides limited direct improvement on a task, its successful corrections can still contribute to the overall multi-round improvement process.

4.3 Ablation and Mechanism Analysis

We organize the ablations around the framework of DreamChaser: the trigger determines when refinement is invoked, the refiner provides corrective actions, selective aggregation decides which corrections become training data, and expert replay stabilizes fine-tuning. Table 3 shows the ablation study in LIBERO-plus.

Table 3: Component ablations on adapted LIBERO-plus. Success-rate drops are measured against the complete DreamChaser system.

Configuration	Avg. success (%)	Drop (pp)
Full DreamChaser	**80.1**	–
Without mismatch trigger	76.2	↓ 3.9
Without selective aggregation	71.4	↓ 8.7
Without expert-data replay	45.8	↓ 34.3

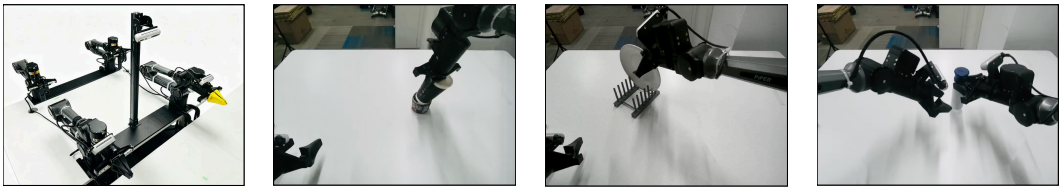


Figure 4: Real-world manipulation tasks on the AgileX PiperX platform. The experiments cover cup sleeving and plate placement with a single arm, and bottle straightening with two arms.

Removing the mismatch trigger, selective aggregation, or expert-data replay lowers the reported mean by 5.1%, 9.9%, and 35.5%, respectively. The largest drop occurs without demonstration replay, consistent with the importance of retaining an expert-data anchor.

4.4 Real-world Experiments

Setup. We conduct real-world experiments on an AgileX PiperX robotic platform equipped with a front-facing RealSense D455 camera and two wrist-mounted RealSense D435 cameras (4). To assess the methods across diverse manipulation settings, we consider three tasks with different arm configurations, including two single-arm tasks and one dual-arm task. Specifically, these tasks involve cup sleeving (pick up the cup and place it in the sleeve), plate placement (pick up the plate and place it into the rack), and bottle straightening (pick up the bottle and set it upright). Cup sleeving and plate placement are single-arm tasks, whereas bottle straightening is a dual-arm task. We report a normalized success score rather than binary completion: the score credits intermediate task progress and is therefore more informative for partially completed real-world trajectories. All methods use the same initial conditions. DreamChaser performs two rounds of online self-improvement; the two entries reported for DreamChaser-FT in Table 4 correspond to the first and second rounds.

Baselines. We compare $\pi_{0.5}$, the initial base model FastWAM-joint, the base model with the frozen refiner, Vanilla-FT, and two DreamChaser-FT checkpoints. the "+ Refiner" row evaluates correction of the unchanged FastWAM-joint. The DreamChaser-FT rows evaluate the updated WAM without the refiner. The $\pi_{0.5}$ result provides a reference policy rather than a matched-pretraining ablation.

Table 4: Normalized real-world task success scores (0–100; higher is better). The mean averages three tasks equally.

Method	Cup sleeve	Plate	Bottle	Mean
$\pi_{0.5}$	84.0	52.0	54.0	63.3
FastWAM-joint	32.0	28.0	21.0	27.0
+ Refiner	59.0	38.0	34.0	43.7
Vanilla-FT	28.0	23.0	20.0	23.7
DreamChaser-FT, round 1	47.0	34.0	28.0	36.3
DreamChaser-FT, round 2	56.0	37.0	32.0	41.7
Gain vs. FastWAM-joint	↑ 24.0	↑ 9.0	↑ 11.0	↑ 14.7

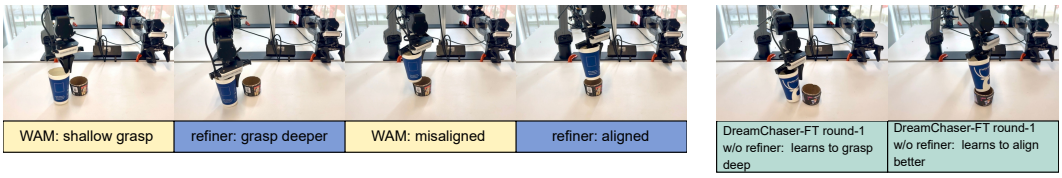


Figure 5: **Qualitative results for DreamChaser.** Equipped with the action refiner, the base WAM grasps the cup more deeply and performs additional alignment before releasing it into the cup sleeve, improving task execution. After fine-tuning on the aggregated data, round-1 DreamChaser-FT learns similar behaviors, demonstrating that the refinements can be transferred to the WAM through fine-tuning.

Results and Analysis. Table 4 reports the real-world results in terms of task success and mean progress score. The updated WAM improves its mean progress score from 27.0% to 36.3% after the first round of refinement and further to 41.7% after fine-tuning, with consistent gains across all three tasks. These results suggest that the action refiner helps the WAM make meaningful progress toward task completion and that the resulting refinement data can be used to further improve the policy. We provide qualitative results in Fig. 5. As illustrated, the refiner enables the base WAM to grasp the cup more deeply and perform additional alignment before placing it into the cup sleeve. After fine-tuning on the aggregated data, round-1 DreamChaser-FT learns similar behaviors, suggesting that the refined actions can be effectively transferred to the WAM through proposed method.

5 Limitations and Future Work

A limitation of our framework is the capability of the goal-conditioned action refiner. We use a lightweight ResNet-18 to predict the target robot state and interpolate the corresponding actions. While sufficient to demonstrate video-guided refinement, this approach may produce actions that are inconsistent with the WAM-generated trajectory, limiting refinement data quality and cumulative gains across self-improvement rounds. This issue is particularly evident in real-world fine-tuning, where oscillatory behaviors can emerge. Future work will explore more capable refiners that reason over predicted futures and generate temporally coherent corrections, potentially by integrating agentic embodied AI systems or LLM-based reasoning for higher-level action refinement.

6 Conclusion

In this work, we introduced DreamChaser, a self-improving framework that closes this video–action gap by using the WAM's own predictions as guidance for action refinement. DreamChaser detects unreliable execution with a calibrated video–action mismatch trigger, applies a lightweight goal-conditioned refiner, and selectively aggregates successful corrections together with the original demonstrations for subsequent fine-tuning.

Our experiments show that the proposed components are complementary. The refiner improves execution without changing the WAM parameters, while the selective rollout, refine, aggregate, fine-tune loop converts these corrections into additional action supervision. On LIBERO-plus, the multi-round experiments demonstrate that the same framework can be repeatedly applied across task-specific initial states and scene layouts. Real-world experiments on the AgileX PiperX platform further show that video-guided refinement and online updates can improve the success rate of manipulation tasks without requiring a new human intervention policy during rollout.

More broadly, DreamChaser suggests a way to turn the visual predictions already present in a WAM into executable-action supervision. Instead of treating video generation and action generation as separate capabilities, the framework uses their disagreement as a signal for online improvement. We hope this perspective motivates future WAMs that learn from their own predictions and interactions, while improving the temporal coherence and generality of the refiner used to guide that process.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development, provide feedback on research methodology and experimental design, and polish and improve the clarity and readability of the manuscript. We have reviewed all AI-assisted work and verified the correctness and appropriateness of the resulting code, methodology, experiments, and manuscript text. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study does not involve human subjects or personal data. All robotic experiments were performed in controlled environments. Since policies deployed on physical robots may behave unpredictably in unfamiliar settings, thorough validation and appropriate safety measures are necessary before deployment.

REPRODUCIBILITY STATEMENT

We document the methodology, model configurations, experimental protocols, and evaluation metrics in the main paper and appendix. To support reproducibility, we plan to release the codebase and deployment instructions upon publication.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

Boyuan Chen, Tianyuan Zhang, Haoran Geng, Caiyi Zhang, Peihao Li, Kiwhan Song, William T Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025.

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.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Zhaopeng Gu, Bingke Zhu, Tianxi Lin, Guibo Zhu, Yingying Chen, Kai Wang, Tingyu Yuan, Chaoyang Zhao, Zhaowen Li, Peng Su, et al. Harnesswam: Bridging prediction and deliberation in world action models. *arXiv preprint arXiv:2608.09516*, 2026.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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.

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*, 2025.

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

Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4845–4852. IEEE, 2025.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 125846–125869, 2026.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, et al. Genie envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 88446–88463, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Yuejiang Liu, Fan Feng, Lingjing Kong, Weifeng Lu, Jinzhou Tang, Kun Zhang, Kevin Murphy, Chelsea Finn, and Yilun Du. World action verifier: Self-improving world models via forward-inverse asymmetry. *arXiv preprint arXiv:2604.01985*, 2026a.

Yushan Liu, Peibo Sun, Shoujie Li, Yifan Xie, Lingfeng Zhang, Xintao Chao, Shiyuan Dong, Fang Chen, Xiao-Ping Zhang, and Wenbo Ding. Oa-wam: Object-addressable world action model for robust robot manipulation. *arXiv preprint arXiv:2605.06481*, 2026b.

Utkarsh A Mishra, Yongxin Chen, Danfei Xu, Yang Liu, Xi Chen, and Jiayuan Mao. Understanding and mitigating the video-action generalization gap via temporal ratio. *arXiv preprint arXiv:2607.08127*, 2026.

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.

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

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey. *arXiv preprint arXiv:2606.20781*, 2026.

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

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023.

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.

Junke Wang, Qihang Zhang, Shuai Yang, Yiming Luo, Yujun Shen, Zuxuan Wu, Yu-Gang Jiang, and Yinghao Xu. Repwam: World action modeling with representation visual-action tokenizers. *arXiv preprint arXiv:2606.13674*, 2026a.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026b.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024.

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*, 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.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, et al. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

Anlan Yu, Zaishu Chen, Peili Song, Zhiqing Hong, Haotian Wang, Desheng Zhang, Tian He, Yi Ding, and Daqing Zhang. Wm-dagger: Enabling efficient data aggregation for imitation learning with world models. *arXiv preprint arXiv:2604.11351*, 2026.

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.

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 APPENDIX

A.1 EXECUTION ORDER AND TRAINING WINDOWS

The sequence below makes the time indexing in Sec. 3 explicit. The WAM first executes an action chunk and the trigger evaluates the resulting observation at the chunk endpoint. If refinement is activated, the refiner generates a correction from the post-chunk state. The WAM is then replanned from this corrected state, and its subsequent actions provide the padding portion of the refined training window. If the episode terminates before the window can be completed, the candidate window is discarded. Overlapping windows are allowed and their retained samples are combined by set union.

Algorithm 1 DreamChaser collection and update

Require: Initial WAM θ_0, frozen refiner ϕ, demonstrations $\mathcal{D}_{\text{exp}}$, task calibration sets $\mathcal{C}_q$, rounds K

for $i = 0, \ldots, K-1$ **do**
  Initialize trajectory buffer $\mathcal{B}_i$ and candidate window starts $\mathcal{U}_i$
  for each collection episode e **do**
    Set $t = 0$; observe (o_t, s_t) and instruction l
    while the episode continues **do**
      Sample a WAM chunk and its predicted endpoint $\hat{o}_{t+H}$
      Execute the chunk and record the resulting trajectory in $\mathcal{B}_i$
      Set $\tau = t + H$; compute d_t^{pre}, p_t, and g_t
      if $g_t = 0$ **then**
        Add (e, t) to $\mathcal{U}_i$; set $t \leftarrow \tau$
      else
        Generate and execute the correction A_t^R from the corrected state
        Replan subsequent WAM actions from the corrected state
        Compute d_t^{post} against the same predicted endpoint
        if $d_t^{\text{post}} < d_t^{\text{pre}}$ **then**
          Add (e, τ) to $\mathcal{U}_i$
        end if
        Set $t \leftarrow \tau + H_R$
      end if
    end while
  end for
  Materialize retained windows within each episode; merge overlaps by union
  Fine-tune θ_i on $\mathcal{D}_{\text{exp}} \cup \mathcal{D}_{\text{online}}^{(i)}$ to obtain θ_{i+1}
end for

A.2 CALIBRATED TRIGGER

The discrepancy reference set is collected once from ten nominal rollout episodes for each task using the initial WAM. It is then kept fixed during online self-improvement. We use $\alpha = 0.1$ on LIBERO, $\alpha = 0.3$ on LIBERO-plus, and $\alpha = 0.8$ in the real-world experiments. The same encoder and preprocessing pipeline is used for calibration and live evaluation. For multi-camera observations, the camera views are concatenated in a T-shaped layout before encoding. The predicted endpoint is aligned with the observation at $t + H$, consistent with the temporal alignment used during training.

These settings make the trigger a task-calibrated tail criterion: a current discrepancy is considered unusual relative to nominal execution for that task. The calibration set is not updated during later rounds, so changes in trigger behavior can be attributed to the updated policy rather than to a moving reference distribution.

A.3 Real-world Task-progress Score

For task q, let $T_q = 50$ be the number of evaluation trials. The normalized score averages fixed completion criteria over trials:

$$S_q = \frac{100}{T_q K_q} \sum_{n=1}^{T_q} \sum_{k=1}^{K_q} b_{qnk}, \qquad \bar{S} = \frac{1}{3} \sum_{q=1}^{3} S_q, \tag{9}$$

where $b_{qnk} \in \{0, 1\}$ indicates whether trial n satisfies criterion k. The task rubrics are:

- **Cup sleeving:** pick up the cup (0.5) and place it in the sleeve (0.5).
- **Plate placement:** pick up the plate (0.5) and insert it into the rack (0.5).
- **Bottle straightening:** the left arm picks up the bottle (1/3), the right arm receives it (1/3), and the right arm straightens it upright (1/3).

The score credits each completed criterion. A failed later stage does not remove credit for an earlier completed stage, while a failed prerequisite receives zero credit for that criterion. The same criteria, trial count, and evaluation conditions are used for every policy and online round.

A.4 Implementation and Evaluation Details

Table 5 summarizes the implementation settings used in the experiments. The WAM variant is FastWAM-joint. The video generator uses ten sampling steps. The refiner is trained once before online interaction and remains frozen during collection and self-improvement.

Table 5: Implementation and evaluation settings.

Component	Setting
WAM	FastWAM-joint; 30,000 initialization steps; ten video sampling steps.
Action chunk	$H = 16$ for LIBERO and LIBERO-plus; $H = 32$ for real-world experiments.
Refiner horizon	$H_R = 4$ in simulation; $H_R = 16$ in real-world experiments.
Refiner inputs	Image and robot state are processed separately. The refiner predicts a residual action.
Refiner objective	The original FastWAM training loss.
Optimizer	AdamW; learning rate 1.0×10^{-4} for base training and 1.0×10^{-5} for fine-tuning; batch size 128.
Fine-tuning	30,000 initialization steps; five epochs on new data per LIBERO-plus round.
LoRA	Applied to the q, k, v, and o projections, with $r_a = 128$ and $r_v = 8$.
Data mixture	Expert replay to online data ratio 4:1 in simulation and 9:1 in real-world experiments.
Online data	Each round uses as many collection trajectories as evaluation trajectories and retains online data from previous rounds.
Checkpoint	The final checkpoint of each round is used for the next round and evaluation.

A.5 Ablation Protocol

All ablations are conducted on LIBERO-plus with the same trajectories, three online rounds, and five fine-tuning epochs as the full method. In the without-trigger variant, the refiner is invoked at every eligible window; all triggered refinements are retained for training. In the without-selective-aggregation variant, every triggered refinement is added to the online pool regardless of discrepancy reduction. In the without expert-data-replay variant, the original demonstrations are removed from the fine-tuning mixture entirely. These settings isolate the contributions of triggering, data selection, and replay while keeping the interaction budget fixed.