Trust-Calibrated VLA Policy Post-Training with Evolving Imagination

Wenzhan Li[1,2] **Yiran Qin**[3,4,5] **Heng Zhou**[4,6] **Huirui Wang**[1]
Yulan Guo[1,2] **Ruimao Zhang**[1✉]
[1]Sun Yat-sen University [2]Shenzhen Loop Area Institute
[3]The Chinese University of Hong Kong, Shenzhen [4]Shanghai AI Laboratory
[5]University of Oxford [6]University of Science and Technology of China

Abstract: Vision-Language-Action (VLA) models are often pretrained via imitation learning, making reinforcement learning (RL) post-training important when deployment moves beyond demonstration coverage. Existing refinement paradigms expose a reliability–efficiency tension: model-free refinement avoids simulator bias but is interaction-intensive, whereas model-based refinement can exploit unreliable imagination even as its simulator adapts. We propose **Model-Evolving Reinforcement Learning (MERL)**, a trust-calibrated framework that couples policy optimization with an online-updated simulator comprising a world model and a reward proxy head. Across stages, simulator reliability controls the imagined-data ratio and rollout horizon; within rollouts, chunk-level trust controls replay selection and optimization weight. For recursive imagination without exact grounded continuations, a residual predictor estimates trust without querying future ground truth. On perturbed LIBERO-PRO tasks, MERL reaches 79.7% average success, improves high-threshold learning speed by 10.9% over the matched model-free control, and outperforms same-protocol external baselines across three training seeds. Fixed-data adaptation from 50 demonstrations per task achieves 80.0% average success on two physical-robot tasks. Project website: https://merl-project.github.io/.

Keywords: World Models, Reinforcement Learning, VLA Policy Post-Training

1 Introduction

Vision-Language-Action (VLA) models have become a scalable paradigm for language-conditioned robot manipulation, mapping visual observations and language instructions directly to executable actions [1, 2, 3, 4, 5]. Yet most VLA policies still rely heavily on imitation learning, leaving their robustness tightly bounded by expert demonstration coverage. When deployment moves beyond this coverage, errors accumulate, recovery remains weak, and adaptation to unseen perturbations is limited. Reinforcement learning (RL) therefore offers a natural post-training route, allowing policies to improve through interaction rather than remain constrained by demonstrations.

However, applying RL to VLA post-training reveals a progression of bottlenecks as methods move from real interaction toward imagined experience. Model-free refinement updates policies directly from real interaction and therefore avoids simulator bias, but it is costly, sample-inefficient, and often unstable in long-horizon, high-dimensional manipulation settings [6, 7, 8, 9, 10]. Static model-based refinement reduces interaction cost by training policies with imagined rollouts from learned simulators [11, 12, 13, 14, 15], but its simulator can become stale as the policy-induced distribution shifts. Online-updated model-based refinement further adapts the simulator during policy improvement to better match newly induced states and environments [16, 17]. Simulator adapta-

✉Corresponding author: zhangrm27@mail.sysu.edu.cn.

10th Conference on Robot Learning (CoRL 2026), Austin, Texas, USA.

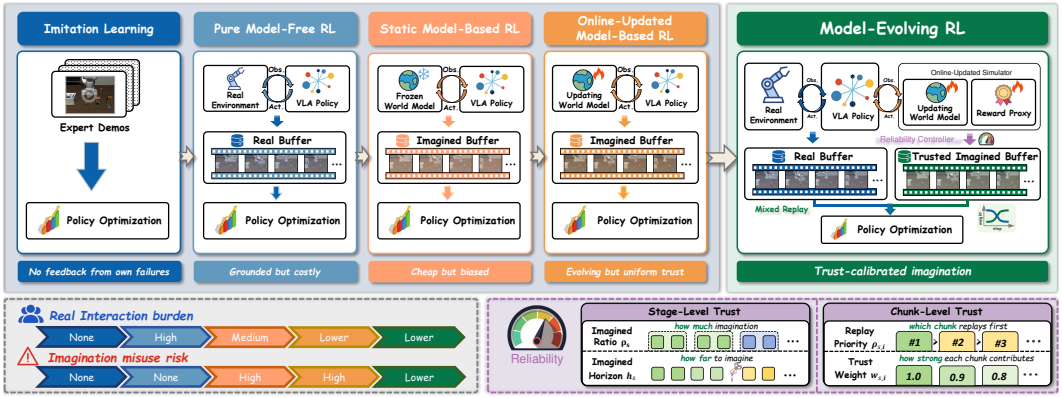


Figure 1: **Paradigm evolution toward trust calibration.** The columns illustrate design patterns represented by our matched refinement controls, from imitation and model-free RL to static and online-updated model-based RL. MERL uses simulator reliability to schedule, select, and weight rollouts before the subsequent mixed real–imagined policy optimization.

tion addresses staleness, while selective model usage and reliability-aware VLA refinement also study how imagined data should influence learning [18, 19, 17]. A remaining question is how one reliability interface can jointly govern the stage-level use and chunk-level influence of visual imagination as the policy and simulator adapt. This auditing gap becomes fundamental because simulator reliability is neither stationary nor homogeneous; it changes over training and varies across imagined rollouts. This raises two central questions: *how should VLA refinement estimate the trustworthiness of simulator-generated imagination as the simulator evolves, and how should this trust govern which imagined experience enters policy optimization?*

Fig. 1 illustrates the trust-calibrated mechanism introduced by MERL. To produce imagination, MERL uses an evolving simulator composed of a world model and a reward proxy head, which provide short-horizon visual futures and local progress estimates for policy optimization. The main contribution is not the simulator architecture itself, but the trust layer placed between simulator-generated imagination and policy learning. As the simulator evolves, MERL estimates trust at the level of each online refinement cycle through stage-level trust. This global reliability estimate is converted into scheduling signals that regulate the strength and temporal extent of simulator-generated imagination. During policy optimization, MERL applies trust at the level of generated rollout segments through chunk-level trust. Each imagined chunk is locally assessed, and its reliability determines replay priority and optimization weight. This hierarchy specifies how trust is estimated from an evolving simulator and how it is operationalized before imagined experience shapes the policy.

The core insight behind MERL is that simulator evolution and simulator trust are distinct problems. Updating the simulator can make imagination more relevant to the current policy, but it does not guarantee that every generated rollout is equally useful for optimization. MERL therefore treats trust as an auditing interface between simulator-generated imagination and policy learning. Stage-level trust prevents premature overuse of unreliable imagination, while chunk-level trust suppresses locally biased samples even when average simulator quality improves. This design keeps the update close to grounded refinement when the simulator is weak and increases the use of imagination as its estimated reliability improves. Fig. 1 summarizes these design axes. Our matched online-updated control retains simulator adaptation while removing trust scheduling and chunk control, enabling a direct test of their contribution.

Our contributions are three-fold: **(1)** We identify the reliability bottleneck of evolving simulators in online VLA post-training, and formulate it as a *trust-calibration* problem where imagined rollouts should influence policy learning according to their estimated reliability. **(2)** We propose **MERL**, a trust-calibrated model-evolving reinforcement learning framework that combines online simulator evolution with two-level trust control, enabling adaptive use of grounded policy updates and

simulator-generated imagination. **(3)** We provide controlled ablations, same-protocol external comparisons, no-oracle trust diagnostics, and fixed-data physical deployment. On perturbed LIBERO-PRO tasks, MERL reduces high-threshold refinement stages over the matched MFRL control by **10.9%** and improves final success over the matched Online-Updated MBRL control by **3.5** points.

2 Related Work

Vision-Language-Action Policies. VLA policies map images and instructions to robot actions. Early systems explore tokenized language-conditioned control [1], while action chunking improves fine-grained manipulation [20]. Later works build on pretrained vision-language backbones and richer action representations to improve semantic grounding, transferability, and generalist control [2, 3, 4, 5, 21, 22, 23]. Most VLA policies are pretrained and adapted through supervised imitation, leaving robustness bounded by demonstration coverage.

Model-Free Reinforcement Learning for VLA Post-Training. To overcome the limitations of pure imitation learning, a growing line of work studies model-free reinforcement learning (MFRL) for post-training VLA policies. Early efforts show that reinforcement learning can improve pretrained VLA policies through interaction [6], while subsequent works scale this paradigm to long-horizon and multi-task settings with trajectory-level optimization, distributed training, and VLA-specific rollout pipelines [7, 8]. More recent studies further explore online fine-tuning for flow-based VLA models, unified RL infrastructures, continual test-time adaptation, and stronger long-horizon policy improvement [9, 10, 24, 25, 26]. These works demonstrate that reinforcement learning can substantially improve VLA policies beyond supervised fine-tuning. However, MFRL still depends heavily on costly real or simulator interaction, and its performance is often constrained by sample inefficiency, reward sensitivity, and unstable long-horizon optimization.

Model-Based Reinforcement Learning for VLA Post-Training. Complementary to MFRL, model-based reinforcement learning (MBRL) reduces interaction cost by learning a simulator and refining policies with imagined experience. In VLA post-training, recent methods instantiate this simulator as a visual world model for policy refinement. Static MBRL refines policies in learned visual simulators [11, 12, 13, 14, 15]. RISE combines visual prediction and progress-value estimation for imagined updates [27]. Online-updated MBRL adapts the simulator to policy-induced distribution shifts [16, 17]. Selective model usage is established in classical MBRL: MBPO uses short branched rollouts [28], Pan et al. mask transitions using ensemble uncertainty [18], and Huang et al. learn transition weights from real-loss meta-gradients [19]. WoVR also studies reliable world-model simulation and policy–model co-evolution for VLA post-training [17]. Our contribution is the coupling of stage-level ratio/horizon scheduling with chunk-level replay/gradient control through visual and proxy residual estimates, including no-oracle scoring of recursive VLA imagination. Online simulator updating supplies the adaptive model; the two-level reliability-to-policy interface is the focus of MERL.

3 Method

Fig. 2 summarizes the refinement cycle: grounded data update the simulator, stage-level trust schedules the imagined ratio and horizon, and chunk-level trust controls replay and optimization. We formalize these components below under sparse terminal success supervision.

3.1 Problem Formulation

We consider online post-training of a pretrained Vision-Language-Action (VLA) policy π_θ in a perturbed manipulation environment. Training proceeds in online stages indexed by s. Each stage denotes one refinement cycle, where the current policy collects grounded rollouts, the simulator is updated and evaluated, imagined chunks are generated with a scheduled ratio and horizon, and the policy is optimized on mixed real–imagined experience.

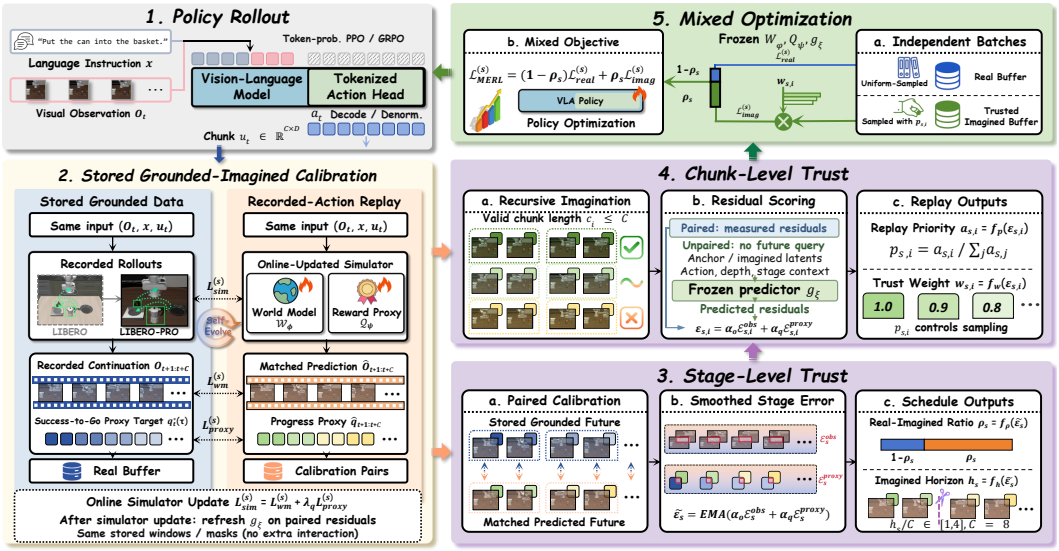


Figure 2: **Overview of the MERL pipeline.** At each online stage s, the policy collects grounded rollouts and updates an evolving simulator composed of a world model $\mathcal{W}_\phi$ and a reward proxy head $\mathcal{Q}_\psi$. Calibration reuses stored grounded continuations with the same anchors and recorded actions; stage-level trust maps their measured errors to ρ_s and h_s. A residual predictor refreshed after simulator update supplies no-oracle scores for unpaired recursive chunks and is frozen during imagination and policy optimization. Chunk-level trust then evaluates generated imagined chunks, assigns replay probabilities $p_{s,i}$ and optimization weights $w_{s,i}$, and combines trusted imagined chunks with grounded rollouts for mixed policy optimization.

Let t denote an anchor time at which the policy issues a new action chunk. Given a visual observation $\mathbf{o}_t \in \mathcal{O}$ and task instruction $\mathbf{x} \in \mathcal{X}$, the policy samples categorical action tokens $\mathbf{a}_t \sim \pi_\theta(\cdot \mid \mathbf{o}_t, \mathbf{x})$ and decodes/denormalizes them to $\mathbf{u}_t \in \mathbb{R}^{C\times D}$. Here C is the action chunk length and D is the dimension of each environment-step action; policy ratios use token probabilities, while the environment and simulator consume continuous commands. For a relative offset j, $\mathbf{o}_{t+j}$ is the post-transition observation; $\mathbf{u}_{t+j}$ denotes the aligned low-level action input, with the chunk context distinguishing it from the full $\mathbf{u}_t$. Executing the policy in the real environment induces a trajectory τ with a sparse terminal success label $z(\tau) \in \{0, 1\}$.

At stage s, the grounded batch records action chunks and observed continuations:

$$\mathcal{B}^{\text{real}}_s = \{(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t, \mathbf{o}_{t+1:t+C}, z(\tau))\},$$

where $\mathbf{o}_{t+1:t+C}$ is produced by real environment interaction under $\mathbf{u}_t$. The imagined batch is

$$\mathcal{B}^{\text{imag}}_s = \{(\tilde{\mathbf{o}}_t, \mathbf{x}, \tilde{\mathbf{u}}_{t:t+h_s-1}, \hat{\mathbf{o}}_{t+1:t+h_s}, \hat{\mathbf{q}}_{t+1:t+h_s})\},$$

where $\tilde{\mathbf{o}}_t$ is an anchor observation sampled for imagination, $\tilde{\mathbf{u}}_{t:t+h_s-1}$ is the action sequence obtained by rolling out the current policy inside the simulator, $\hat{\mathbf{o}}$ denotes simulator-predicted observations, and $\hat{\mathbf{q}}$ denotes proxy progress scores along the imagined continuation. The scheduled horizon satisfies $1 \le h_s \le H_{\max}$, where h_s is measured in environment steps and $H_{\max}$ is the maximum allowed imagined rollout horizon. Thus, C counts actions per policy call, whereas h_s counts imagined environment steps. When $h_s > C$, the policy is queried recursively inside the simulator and the resulting action chunks are unrolled into $\tilde{\mathbf{u}}_{t:t+h_s-1}$.

We optimize π_θ on the two batches while controlling bias from imperfect simulator predictions.

3.2 Online Simulator Evolution

MERL instantiates the online-updated simulator with a world model $\mathcal{W}_\phi$ and a reward proxy head $\mathcal{Q}_\psi$. We reserve *world model* for the video predictor and use *evolving* to denote online adaptation

of the composite simulator. At stage s, both components are updated from grounded trajectories collected by the current policy. The simulator then outputs two signals for imagination: predicted visual continuations from $\mathcal{W}_\phi$ and local proxy progress scores from $\mathcal{Q}_\psi$. This online update makes the simulator better aligned with the current policy-induced distribution than a frozen simulator, but it does not by itself guarantee that all generated rollouts are reliable.

Given a visual observation $\mathbf{o}_t$, instruction $\mathbf{x}$, and action chunk $\mathbf{u}_t$, the world model predicts one chunk of future observations as $\hat{\mathbf{o}}_{t+1:t+C} = \mathcal{W}_\phi(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t)$, where C is the action chunk length. When the scheduled imagined horizon satisfies $h_s > C$, the policy and world model are recursively queried inside the simulator until the rollout reaches h_s environment steps. For each predicted observation-action pair, the reward proxy produces a bounded progress score $\hat{q}_{t+j} = \mathcal{Q}_\psi(\hat{\mathbf{o}}_{t+j}, \mathbf{u}_{t+j}) \in [0, 1]$. Thus, an imagined rollout provides both a visual sequence $\hat{\mathbf{o}}_{t+1:t+h_s}$ and a proxy-score sequence $\hat{\mathbf{q}}_{t+1:t+h_s}$ for policy optimization. Implementation follows App. A.1.

The real environment provides only a sparse terminal success label $z(\tau) \in \{0, 1\}$, which does not directly supervise intermediate imagined progress. We therefore train the proxy head with a truncated success-to-go target. For a trajectory τ of length T, define

$$q_t^\star(\tau) = z(\tau)\, \gamma_q^{\min(T-1-t,\, H_q-1)}, \tag{1}$$

where $\gamma_q \in (0, 1]$ is a discount factor and H_q is a truncation horizon. This target lies in $[0, 1]$: for successful trajectories, states closer to completion receive larger values, while failed trajectories remain zero. The target preserves terminal success and supervises local progress for proxy training.

The simulator is trained with $\mathcal{L}_{\text{sim}}^{(s)} = \mathcal{L}_{\text{wm}}^{(s)} + \lambda_q \mathcal{L}_{\text{proxy}}^{(s)}$, where $\mathcal{L}_{\text{wm}}^{(s)}$ supervises visual prediction, $\mathcal{L}_{\text{proxy}}^{(s)}$ supervises the proxy head with $q_t^\star(\tau)$, and λ_q controls the proxy-loss weight. Detailed objectives are provided in App. A.2. Since visual prediction errors can still induce biased proxy scores on imagined observations, the next section estimates simulator reliability before deciding the imagined-data ratio and rollout horizon.

3.3 Trust-Calibrated Imagination Scheduling

Stage-level trust determines how strongly and how far MERL uses simulator-generated imagination at training stage s. Its input is the simulator mismatch measured after online simulator update, and its outputs are the imagined-data ratio ρ_s and rollout horizon h_s for the subsequent policy update. The need to schedule both quantities comes from a simple fact: proxy supervision on imagined observations inherits visual prediction errors from the world model.

Simulator-induced proxy bias. Consider an anchor time t and an imagined continuation of horizon $h \le h_s$. Let $\hat{q}_t^{(h)}$ denote the discounted aggregate of proxy scores evaluated on simulator-predicted observations $\{\hat{\mathbf{o}}_{t+j}\}_{j=1}^h$, and let $q_t^{(h)}$ denote the corresponding analytical aggregate evaluated on grounded observations $\{\mathbf{o}_{t+j}\}_{j=1}^h$ under the same action continuation. The grounded aggregate is not used as an online training signal; it only measures the bias introduced when the proxy head is evaluated on imagined observations. We define the pointwise visual prediction discrepancy as $\delta_{s,t,j} = \Delta(\hat{\mathbf{o}}_{t+j}, \mathbf{o}_{t+j})$, where $\Delta(\cdot, \cdot)$ measures observation-feature discrepancy.

Proposition 1. If $\mathcal{Q}_\psi$ is L_q-Lipschitz with respect to observation features when the action input is fixed, then for an imagined continuation of horizon h,

$$\left| \hat{q}_t^{(h)} - q_t^{(h)} \right| \le L_q \sum_{j=1}^{h} \gamma_q^{j-1} \delta_{s,t,j}. \tag{2}$$

This bound shows that imagined proxy bias accumulates from visual prediction discrepancies along the rollout. Thus, even a well-supervised proxy head can produce biased progress estimates when evaluated on inaccurate imagined observations, especially as the horizon grows. This motivates scheduling the imagined horizon according to simulator reliability rather than using a fixed rollout length. Formal definitions and proof are provided in App. B.

Stage-level reliability estimate. After updating the simulator, MERL estimates a stage-level reliability error $\bar{\varepsilon}_s = \alpha_o \mathcal{E}_s^{\text{obs}} + \alpha_q \mathcal{E}_s^{\text{proxy}}$ from stored grounded windows. The simulator replays each recorded action continuation from the same anchor and instruction; this calibration adds no environment interaction. Here $\mathcal{E}_s^{\text{obs}}$ measures short-horizon visual prediction mismatch, $\mathcal{E}_s^{\text{proxy}}$ measures proxy-progress mismatch against the success-to-go target, and α_o, α_q are reliability-aggregation weights. To reduce online fluctuations, MERL uses an exponentially smoothed error $\tilde{\varepsilon}_s$ as the global control signal. Concrete definitions of these terms are given in App. A.3.

Scheduling rule. MERL maps $\tilde{\varepsilon}_s$ to two coupled scheduling variables, $\rho_s = f_\rho(\tilde{\varepsilon}_s)$ and $h_s = f_h(\tilde{\varepsilon}_s)$, where both mappings are non-increasing with respect to $\tilde{\varepsilon}_s$. The ratio $\rho_s \in [0, 1]$ controls the contribution of imagined experience in policy optimization, while h_s controls how far the simulator is rolled out from each anchor. Thus, unreliable simulators induce smaller imagined ratios and shorter horizons, whereas more reliable simulators permit stronger and longer imagination. Implementation-specific schedules are provided in App. A.3.

Effect of joint scheduling. The scheduler must control both the amount of imagined data and the rollout horizon because both factors scale the influence of simulator error. Let Δ_s denote the aggregate discrepancy between imagined supervision and its grounded counterpart at stage s. For scheduling analysis, let $\delta_{s,j}$ denote the stage-level visual prediction discrepancy at relative offset j, obtained by aggregating or upper-bounding $\delta_{s,t,j}$ over sampled anchors.

Proposition 2. Suppose $\Delta_s \le K_\Delta \rho_s \sum_{j=1}^{h_s} \delta_{s,j}$ for some task-dependent constant $K_\Delta > 0$. If $\delta_{s,j} \le \tilde{\varepsilon}_s$ at every relative offset $j \le h_s$, the stage-level discrepancy satisfies

$$\Delta_s \le K_\Delta \, \rho_s \, h_s \, \tilde{\varepsilon}_s. \tag{3}$$

The measured average error is a scheduling statistic, not a certified upper bound on every rollout discrepancy; the proposition provides neither a convergence guarantee nor an empirical certificate. Controlling only ρ_s can still allow long biased rollouts, while controlling only h_s can still admit too much unreliable imagined data. We therefore couple the imagined-data ratio and rollout horizon through one shared stage-level trust signal (App. B).

3.4 Trust-Aware Imagined Replay for Policy Optimization

Stage-level trust controls the global use of imagination through the ratio ρ_s and horizon h_s. However, imagined chunks generated within the same stage can still have heterogeneous local reliability. Chunk-level trust therefore determines which imagined chunks are more likely to enter policy optimization and how strongly they affect the update.

For each imagined chunk i generated at stage s, MERL estimates a chunk-level reliability error $\varepsilon_{s,i} = \alpha_o \mathcal{E}_{s,i}^{\text{obs}} + \alpha_q \mathcal{E}_{s,i}^{\text{proxy}}$, where $\mathcal{E}_{s,i}^{\text{obs}}$ and $\mathcal{E}_{s,i}^{\text{proxy}}$ denote estimated local observation mismatch and proxy-progress mismatch. A smaller $\varepsilon_{s,i}$ indicates higher local trust. MERL converts this error into a normalized replay probability $p_{s,i}$ for sampling imagined chunks and an optimization weight $w_{s,i}$ for scaling their policy-gradient contribution. Thus, $p_{s,i}$ controls local admission into the imagined batch, while $w_{s,i}$ controls local influence after sampling. The exact priority normalization and weight functions are provided in App. A.4.

Scoring without future ground truth. Exact grounded–predicted pairs use measured visual and proxy residuals under the same recorded action continuation. For policy-generated chunks without such pairs, a residual predictor g_ξ takes anchor/imagined latents, action, rollout depth, and stage context, and estimates $(\widehat{\mathcal{E}}_{s,i}^{\text{obs}}, \widehat{\mathcal{E}}_{s,i}^{\text{proxy}})$. We refresh g_ξ on exact stored pairs after each simulator update, then freeze it during imagination and the ensuing policy update. Recursive policy calls consume predicted observations; their chunk scores use only the predicted residuals, never a query to the future environment state. These estimates enter the same weighted error $\varepsilon_{s,i}$ and the same $p_{s,i}, w_{s,i}$ mappings as measured residuals. App. A.4 specifies the information flow, and App. D.4 evaluates depths 2–4 with future observations withheld from scoring.

Table 1: Matched-control results on LIBERO-PRO. SR and AUC are on a percentage scale; S2T-H counts refinement stages. See Tab. 2 for three-seed comparisons.

	Suite SR (%) ↑				Overall		
Method	Spatial	Object	Goal	Long	Avg. SR ↑	AUC ↑	S2T-H ↓
SFT	38.6	42.4	23.8	15.9	30.2	–	–
MFRL	80.4	83.1	79.4	66.0	77.2	67.9	103.4
Static-MBRL	76.8	79.4	76.3	60.7	73.3	59.8	168.9
Static-MBRL+Trust	78.2	80.8	78.4	62.8	75.1	63.5	145.8
Online-Updated MBRL	79.5	82.0	77.0	66.1	76.2	65.4	139.6
MERL (Ours)	**82.8**	**85.2**	**81.9**	**68.8**	**79.7**	**70.6**	**92.1**

At stage s, MERL samples grounded chunks into $\mathcal{B}_s^{\text{real}}$ and imagined chunks into $\mathcal{B}_s^{\text{imag}}$. The two batches are sampled independently; grounded–imagined pairing is used only for reliability calibration when paired continuations are available, not for policy optimization. Both branches use the same clipped policy-gradient form, while the imagined branch is additionally weighted by $w_{s,i}$. The resulting real–imagined objective at stage s is

$$\mathcal{L}_{\text{MERL}}^{(s)} = (1-\rho_s)\mathcal{L}_{\text{real}}^{(s)} + \rho_s \mathcal{L}_{\text{imag}}^{(s)}. \tag{4}$$

Here $\mathcal{L}_{\text{real}}^{(s)}$ is computed on grounded rollouts, and $\mathcal{L}_{\text{imag}}^{(s)}$ is computed on imagined rollouts sampled according to $p_{s,i}$ and weighted by $w_{s,i}$. Therefore, ρ_s controls the stage-level mixture between real and imagined branches, whereas $p_{s,i}$ and $w_{s,i}$ control the local admission and influence of individual imagined chunks. The token-level branch objectives, validity masks, and complete online refinement procedure are provided in Apps. A.5 and A.6.

4 Experiments

We evaluate whether MERL improves policy post-training under deployment shifts, whether simulator updates and two-level trust provide complementary gains, and whether predicted trust is informative for recursive imagination and physical deployment.

Setup and metrics. We use LIBERO-PRO [29], with Spatial, Object, Goal, and Long suites. Task semantics remain fixed while deployment conditions shift, isolating adaptation from new-task acquisition. Methods share OpenVLA-OFT initialization [21], action chunks, and grounded budgets. A training stage is one outer refinement cycle, distinct from environment transitions or task horizon. After stage s, each online method has $6s$ grounded trajectories under the same $\leq 3072s$ transition cap; calibration reuses stored windows and adds zero environment interaction. We report final success rate (SR), normalized area under the SR learning curve (AUC), and stages to threshold (S2T). S2T-H averages S2T@80 for Spatial/Object/Goal and S2T@65 for Long. Threshold crossing is linearly interpolated between evaluated checkpoints, allowing fractional values. Imagination and simulator updates incur model computation; S2T-H compares refinement speed under matched grounded budgets. App. C gives the metrics, budget accounting, and implementation.

Matched controls. We isolate components using the same policy optimizer and reward interface. **SFT** evaluates the initialization without RL; **MFRL** uses grounded refinement only; **Static-MBRL** uses a frozen simulator, fixed ratio/horizon, and uniform replay; **Static-MBRL+Trust** adds both trust levels to the frozen simulator; **Online-Updated MBRL** updates the simulator while retaining fixed scheduling and uniform replay. **MERL** combines simulator updates and both trust levels. These controls test components; concrete external algorithms are compared separately below.

Main results. Tab. 1 shows gains in both final task performance and learning efficiency. Relative to MFRL, MERL improves Avg. SR from 77.2 to 79.7 and AUC from 67.9 to 70.6, while reducing S2T-H from 103.4 to 92.1 (10.9%). Static-MBRL underperforms MFRL, consistent with

Table 2: Same-protocol external comparisons on LIBERO-PRO: mean $\pm$ standard deviation over three training seeds. † denotes a port changing benchmark/model assets while retaining the defining algorithm; each WoVR seed has an independent simulator.

Method	Paradigm	Avg. SR ↑	AUC ↑	S2T-H ↓
SimpleVLA-RL	Model-free	77.1 ± 1.1	67.8 ± 1.0	104.1 ± 7.0
WMPO†	Static model-based	75.4 ± 1.2	63.2 ± 1.3	149.6 ± 10.2
WoVR†	Online-updated	77.8 ± 0.9	68.4 ± 0.9	101.3 ± 6.2
MERL	Trust-calibrated	$\mathbf{79.7 \pm 0.7}$	$\mathbf{70.6 \pm 0.8}$	$\mathbf{92.1 \pm 4.8}$

Table 3: AgileX PiPER X closed-loop results: 20 trials/task after offline refinement from 50 demonstrations/task without extra physical training rollouts. Average-SR CIs are 95% intervals.

Method	Block-to-Box	Lid-to-Pot	Avg. SR (%) [95% CI] ↑
SFT	8/20	6/20	$35.0\,[22.1, 50.5]$
MFRL	12/20	10/20	$55.0\,[39.8, 69.3]$
Online-Updated MBRL	13/20	11/20	$60.0\,[44.6, 73.7]$
MERL-Visual	16/20	15/20	$77.5\,[62.5, 87.7]$
MERL (full)	**17/20**	**15/20**	$\mathbf{80.0}\,[\mathbf{65.2}, \mathbf{89.5}]$

biased or stale imagination. Adding trust improves the frozen-simulator control, but it remains below MERL. Online-Updated MBRL improves over Static-MBRL, yet MERL raises its Avg. SR by 3.5 points and reduces S2T-H from 139.6 to 92.1. These comparisons support the complementary roles of simulator adaptation and reliability control.

External positioning and uncertainty. We compare SimpleVLA-RL [8], WMPO [14], and WoVR [17]. Methods in Tab. 2 share SFT initialization, split, grounded budget, action chunk, optimizer/reward interface, checkpoints, and evaluation. MERL has the strongest mean SR/AUC/S2T-H among these methods. Against WoVR, paired 95% confidence intervals (CIs) give $\Delta\text{AUC} = 2.2\,[1.2, 3.3]$ and $\Delta\text{S2T-H} = -9.2\,[-15.4, -3.1]$. Against the matched Online-Updated MBRL control, the three-seed contrast gives $\Delta\text{AUC} = 5.0\,[3.8, 6.2]$. External comparisons establish overall standing; the matched control isolates trust at a shared update cadence.

Where do the gains come from? The component ablation in Tab. 5 retains simulator evolution while enabling either trust level alone. Adding stage-level trust raises AUC from 65.4 to 68.4 and reduces S2T-H from 139.6 to 113.2. Adding chunk-level trust raises AUC to 67.6 and reduces S2T-H to 121.5. Using both gives the strongest Avg. SR, AUC, and S2T-H, consistent with complementary global scheduling and local replay control. Simulator diagnostics and cross-suite results are in App. D. One-at-a-time $\{0.5, 1, 2\}\times$ sweeps of scheduler sharpness κ and proxy-loss weight λ_q change Avg. SR/AUC by at most $1.4/1.6$ points; every setting retains at least $+1.8/+3.1$ points over the matched online-updated control (App. D.5).

Learning dynamics and no-oracle trust. Fig. 3 links policy improvement to the two trust controls. Lower simulator error permits larger ρ_s and h_s; lower chunk error receives larger replay probability and weight. For unpaired recursive chunks, we evaluate the residual predictor with matched future observations withheld from scoring. Across depths 2–4 ($N = 1500$), Pearson $r = 0.68$, Rank-AUC $= 0.80$, ECE $= 0.067$, and top-30% low-error precision $= 0.77$. At depths 3–4 ($N = 900$), the corresponding values are $0.64/0.77/0.074/0.74$, indicating useful but weaker prediction deeper into imagination. Matched-count refinements from Top/Mid/Bottom trust bins achieve policy AUC $69.9 \pm 0.8/68.3 \pm 0.9/66.7 \pm 1.0$ over three seeds, with $\Delta_{\text{Top}-\text{Bottom}} = 3.2\,[2.0, 4.4]$. These results connect predicted trust to policy utility under this protocol. Full depth-wise CIs and residual-predictor calibration diagnostics for these recursive rollouts appear in App. D.4.

Full-method physical deployment. Tab. 3 compares fixed-data adaptation on Block-to-Box and Lid-to-Pot under held-out placements and colors. All methods share data, hardware interface, update

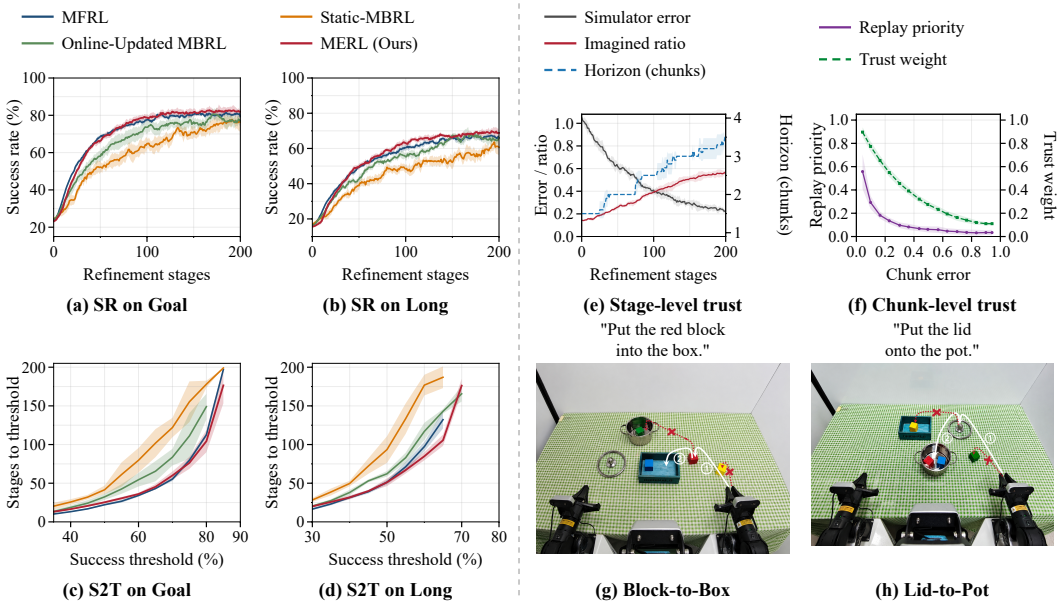


Figure 3: Learning dynamics, two-level trust, and real-robot task setups. Left: SR and S2T on Goal and Long. Right: trust diagnostics and AgileX PiPER X setups for Block-to-Box and Lid-to-Pot. All training axes count outer refinement stages. Panel (e) displays the horizon in action-chunk units $h_s/C \in [1, 4]$, with $C = 8$; panel (f) displays replay priority and trust weight against chunk error.

budget, optimizer, and action horizon. Online-Updated MBRL uses visual imagination; MERL-Visual uses visual-only trust ($\alpha_q = 0$); full MERL enables $\mathcal{Q}_\psi$ and joint visual/proxy trust. Full MERL achieves the largest observed average SR (80.0%), while MERL-Visual obtains 77.5%. Their CIs overlap, so these trials support full-method deployment without resolving a significant gain from enabling the proxy alone. Training uses only the fixed demonstrations; physical evaluation is closed-loop inference. App. E provides the full protocol, qualitative rollouts, and failure modes.

5 Conclusion and Limitations

Conclusion. We presented MERL, a trust-calibrated framework for VLA policy post-training with an online-updated simulator. Stage-level scheduling and chunk-level replay control turn estimated simulator reliability into explicit policy-update decisions. Controlled LIBERO-PRO experiments, same-protocol external comparisons, no-oracle trust diagnostics, and fixed-data physical deployment support the benefits of this coupling under the evaluated deployment shifts.

Limitations. Our validation covers tokenized VLA policies and structured tabletop tasks with fixed RGB sensing; LIBERO-PRO preserves task semantics and does not establish broad new-task learning or contact-rich generalization. Trust is a predictive estimate based on visual and proxy residuals, and its accuracy decreases at deeper rollout depths; it does not certify physical correctness. The trust-bin experiment evaluates predictive policy utility, while individual chunks' causal contributions remain unresolved. Online simulator updates add computation, and calibration depends on grounded coverage even though it reuses stored data. The physical study uses two tasks and offline refinement, with overlapping confidence intervals between full MERL and MERL-Visual; it does not establish online physical RL or a statistically resolved proxy-only gain. Broader sensing, continuous-action generative policies, and larger closed-loop robot studies remain future work.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, 2023. doi:10.15607/RSS.2023.XIX.025.

[2] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025.

[4] D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, et al. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi:10.15607/RSS.2024.XX.090.

[5] K. Black, N. Brown, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025. doi:10.15607/RSS.2025.XXI.010.

[6] J. Liu, F. Gao, B. Wei, X. Chen, Q. Liao, Y. Wu, C. Yu, and Y. Wang. What Can RL Bring to VLA Generalization? An Empirical Study. In *Advances in Neural Information Processing Systems*, volume 38, pages 97121–97151, 2025. doi:10.52202/085713-3248.

[7] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang. VLA-RL: Towards Masterful and General Robotic Manipulation with Scalable Reinforcement Learning. *arXiv preprint arXiv:2505.18719*, 2025.

[8] H. Li, Y. Zuo, J. Yu, Y. Zhang, Z. Yang, K. Zhang, X. Zhu, Y. Zhang, T. Chen, G. Cui, et al. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning. In *International Conference on Learning Representations*, 2026.

[9] K. Chen, Z. Liu, T. Zhang, Z. Guo, S. Xu, H. Lin, H. Zang, X. Li, Q. Zhang, Z. Yu, et al. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models. *arXiv preprint arXiv:2510.25889*, 2025.

[10] H. Zang, M. Wei, S. Xu, Y. Wu, Z. Guo, Y. Wang, H. Lin, P. Wang, H. Yuan, Y. Zhang, et al. RLux-VLA: A Unified and Efficient Framework for Reinforcement Learning of Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.089.

[11] H. Li, P. Ding, R. Suo, Y. Wang, Z. Ge, D. Zang, K. Yu, M. Sun, H. Zhang, D. Wang, et al. VLA-RFT: Vision-Language-Action Reinforcement Fine-tuning with Verified Rewards in World Simulators. *arXiv preprint arXiv:2510.00406*, 2025.

[12] J. Xiao, Y. Yang, X. Chang, R. Chen, F. Xiong, M. Xu, W.-S. Zheng, and Q. Zhang. World-Env: Leveraging World Model as a Virtual Environment for VLA Post-Training. *arXiv preprint arXiv:2509.24948*, 2025.

[13] Z. Jiang, K. Liu, Y. Qin, S. Tian, Y. Zheng, M. Zhou, Z. Zhang, C. Yu, H. Li, and D. Zhao. World4RL: Diffusion World Models for Policy Refinement With Reinforcement Learning for Robotic Manipulation. *IEEE Robotics and Automation Letters*, 11(10):12240–12247, 2026. doi:10.1109/lra.2026.3728345.

[14] F. Zhu, Z. Yan, Z. Hong, Q. Shou, X. Ma, and S. Guo. WMPO: World Model-based Policy Optimization for Vision-Language-Action Models. In *International Conference on Learning Representations*, 2026.

[15] J. Zhang, Z. Huang, C. Gu, Z. Ma, and L. Zhang. Reinforcing Action Policies by Prophesying. *arXiv preprint arXiv:2511.20633*, 2025.

[16] X. Liu, Z. Bai, H. Ci, K. Y. Ma, and M. Z. Shou. World-VLA-Loop: Closed-Loop Learning of Video World Model and VLA Policy. *arXiv preprint arXiv:2602.06508*, 2026.

[17] Z. Jiang, S. Zhou, Y. Jiang, Z. Huang, M. Wei, Y. Chen, T. Zhou, Z. Guo, H. Lin, Q. Zhang, et al. WoVR: World Models as Reliable Simulators for Post-Training VLA Policies with RL. *arXiv preprint arXiv:2602.13977*, 2026.

[18] F. Pan, J. He, D. Tu, and Q. He. Trust the Model When It Is Confident: Masked Model-based Actor-Critic. In *Advances in Neural Information Processing Systems*, volume 33, pages 10537–10546, 2020.

[19] W. Huang, Q. Yin, J. Zhang, and K. Huang. Learning to Reweight Imaginary Transitions for Model-Based Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7848–7856, 2021. doi:10.1609/aaai.v35i9.16958.

[20] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi:10.15607/RSS.2023.XIX.016.

[21] M. J. Kim, C. Finn, and P. Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, 2025. doi:10.15607/RSS.2025.XXI.017.

[22] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[23] Z. Zhang, H. Xu, Z. Yang, C. Yue, Z. Lin, H.-a. Gao, Z. Wang, and H. Zhao. Elucidating the Design Space of Torque-aware Vision-Language-Action Models. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 4019–4037. PMLR, 2025.

[24] Z. Bai, C. Gao, and M. Z. Shou. EVOLVE-VLA: Test-Time Training from Environment Feedback for Vision-Language-Action Models. *arXiv preprint arXiv:2512.14666*, 2025.

[25] A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. B. Connors, J. Darpinian, K. Dhabalia, J. Di Carlo, D. Driess, et al. $\pi^*_{0.6}$: a VLA That Learns From Experience. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.087.

[26] Y. Li, X. Ma, J. Xu, Y. Cui, Z. Cui, Z. Han, L. Huang, T. Kong, Y. Liu, H. Niu, et al. GR-RL: Going Dexterous and Precise for Long-Horizon Robotic Manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[27] J. Yang, K. Lin, W. Zhang, J. Li, T. Lin, L. Wu, Y.-Q. Zhang, H. Zhao, P. Luo, Z. Su, et al. Self-Improving Robot Policy with Compositional World Model. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.012.

[28] M. Janner, J. Fu, M. Zhang, and S. Levine. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[29] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. LIBERO-PRO: Towards Robust and Fair Evaluation of Vision-Language-Action Models Beyond Memorization. *arXiv preprint arXiv:2510.03827*, 2025.

[30] Y. Guo, L. X. Shi, J. Chen, and C. Finn. Ctrl-World: A Controllable Generative World Model for Robot Manipulation. *arXiv preprint arXiv:2510.10125*, 2025.

[31] Y. Qin, J. Ma, L. Kang, W. Li, Y. Jiao, X. Wen, X. Song, H. Zhou, J. Yu, Z. Yin, et al. ComSim: Building Scalable Real-World Robot Data Generation via Compositional Simulation. *arXiv preprint arXiv:2604.11386*, 2026.

[32] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Komeili, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, et al. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*, 2025.

[33] L. Mur-Labadia, M. Muckley, A. Bar, M. Assran, K. Sinha, M. Rabbat, Y. LeCun, and N. Ballas. V-JEPA 2.1: Unlocking Dense Features in Video Self-Supervised Learning. In *Computer Vision – ECCV 2026*, pages 671–689. Springer, 2026. doi:10.1007/978-3-032-37369-4_36.

[34] G. Lu, B. Jia, P. Li, Y. Chen, Z. Wang, Y. Tang, and S. Huang. GWM: Towards Scalable Gaussian World Models for Robotic Manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9263–9274, 2025.

[35] W. Huang, Y.-W. Chao, A. Mousavian, M.-Y. Liu, D. Fox, K. Mo, and F.-F. Li. PointWorld: Scaling 3D World Models for In-The-Wild Robotic Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20765–20779, 2026.

[36] Z. Zheng, X. Peng, Y. Lou, C. Shen, T. Young, X. Guo, B. Wang, H. Xu, H. Liu, M. Jiang, et al. Open-Sora 2.0: Training a Commercial-Level Video Generation Model in $200k. *arXiv preprint arXiv:2503.09642*, 2025.

[37] Team Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, et al. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv preprint arXiv:2503.20314*, 2025.

[38] M. Q. Ali, A. Sridhar, S. Matiana, A. Wong, and M. Al-Sharman. Humanoid World Models: Open World Foundation Models for Humanoid Robotics. *arXiv preprint arXiv:2506.01182*, 2025.

[39] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, et al. DreamGen: Unlocking Generalization in Robot Learning through Video World Models. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 5170–5194. PMLR, 2025.

[40] Z. Ren, Y. Wei, X. Yu, G. Luo, Y. Zhao, B. Kang, J. Feng, and X. Jin. VideoWorld 2: Learning Transferable Knowledge from Real-world Videos. *arXiv preprint arXiv:2602.10102*, 2026.

[41] D. Ha and J. Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[42] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019.

[43] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi:10.1038/s41586-025-08744-2.

[44] D. Hafner, W. Yan, and T. Lillicrap. Training Agents Inside of Scalable World Models. *arXiv preprint arXiv:2509.24527*, 2025.

[45] M. Burchi and R. Timofte. Learning Transformer-based World Models with Contrastive Predictive Coding. *arXiv preprint arXiv:2503.04416*, 2025.

[46] M. Hauri and F. Zenke. Dreamer-CDP: Improving Reconstruction-free World Models Via Continuous Deterministic Representation Prediction. *arXiv preprint arXiv:2603.07083*, 2026.

A Full MERL Formulation

This appendix provides the full formulation of MERL beyond the compact description in the main paper. We first clarify notation, then detail the simulator architecture, simulator training objective, reliability schedules, chunk-level replay and weighting rules, policy objective, and online training algorithm. All symbols are consistent with the main text.

Notation clarification. We distinguish four indices throughout the formulation. The symbol s indexes online training stages, where each stage corresponds to one outer-loop refinement cycle. The symbol t denotes an anchor time at which the policy issues an action chunk. The symbol j denotes a relative environment-step offset from the anchor time. The symbols h and h_s denote imagined rollout horizons measured in environment steps.

The action chunk length is denoted by C, while the maximum allowed imagined rollout horizon is denoted by $H_{\max}$. Thus, C specifies how many low-level actions a single policy call outputs, whereas h_s specifies how far the simulator is rolled out from an anchor at stage s, with $1 \le h_s \le H_{\max}$. When $h_s > C$, the policy and simulator are applied recursively inside the imagined rollout. We use $\hat{\mathbf{o}}$ for simulator-predicted observations and $\tilde{\mathbf{o}}$ for anchor observations sampled for imagination. For a relative offset j, $\delta_{s,t,j}$ denotes the pointwise visual prediction discrepancy between $\hat{\mathbf{o}}_{t+j}$ and $\mathbf{o}_{t+j}$ at stage s and anchor time t. Its stage-level aggregate or upper bound over sampled anchors is denoted by $\delta_{s,j}$. In contrast, $\bar{\varepsilon}_s$ and $\tilde{\varepsilon}_s$ denote raw and smoothed stage-level reliability errors, while $\varepsilon_{s,i}$ denotes the estimated local reliability error of imagined chunk i.

For proxy notation, $q_t^\star(\tau)$ denotes the success-to-go proxy target constructed from the terminal success label of trajectory τ. The scalar $\hat{q}_{t+j}$ denotes a proxy prediction at relative offset j. For a horizon h, $\hat{q}_t^{(h)}$ denotes the discounted proxy aggregate evaluated on simulator-predicted observations, while $q_t^{(h)}$ denotes the corresponding analytical aggregate evaluated on grounded observations under the same action continuation. When discussing replay, $\widetilde{\mathcal{B}}_s^{\mathrm{imag}}$ denotes the candidate imagined chunks generated or retrieved at stage s, while $\mathcal{B}_s^{\mathrm{imag}}$ contains imagined chunks sampled from these candidates for subsequent policy optimization.

A.1 Simulator Architecture

MERL instantiates the evolving simulator with two components: a controllable video-prediction world model $\mathcal{W}_\phi$ and a lightweight reward proxy head $\mathcal{Q}_\psi$. The simulator takes an observation, instruction, and action input as conditions, and returns both predicted visual continuations and scalar proxy progress scores. The world model supports visual imagination, while the proxy head converts sparse task success into local progress supervision that can be used during policy optimization.

World-model backbone. The world model follows controllable generative world models for robot manipulation [30]. Given a current observation $\mathbf{o}_t$, task instruction $\mathbf{x}$, and action chunk $\mathbf{u}_t \in \mathbb{R}^{C\times D}$, it predicts one chunk of future observations:

$$\hat{\mathbf{o}}_{t+1:t+C} = \mathcal{W}_\phi(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t), \tag{5}$$

where C is the VLA action chunk length. In implementation, RGB observations are encoded into visual latent tokens, and action chunks are projected into action-conditioning tokens. The prediction backbone uses action-conditioned spatio-temporal transformers so that future visual tokens can attend to the corresponding action context. The predicted sequence can be decoded into RGB frames for visualization or kept in latent space for reliability estimation and proxy evaluation.

The scheduled imagined horizon h_s in the main text is not restricted to a single model call. When $h_s > C$, MERL recursively queries the policy and world model inside the simulator. Specifically, the policy produces a new action chunk from the latest predicted observation, and $\mathcal{W}_\phi$ predicts the next visual chunk until the rollout reaches h_s environment steps. Thus, C denotes the output length of one policy/world-model chunk, whereas h_s denotes the scheduled simulator rollout horizon.

Reward proxy head. The reward proxy head $\mathcal{Q}_\psi$ maps observation-action latents to a bounded scalar progress score. Given a grounded or predicted observation, a visual encoder produces observation tokens, and an action projection maps the aligned action input into action tokens. The observation tokens attend to the action tokens through lightweight cross-attention blocks, and the fused representation is passed to an MLP head. Its positive-class probability provides the proxy's bounded scalar estimate of local success-to-go progress in the interval $[0, 1]$.

On grounded observations, the proxy evaluates the aligned observation-action pair as

$$\hat{q}_{t+j} = \mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j}) \in [0, 1],$$

On imagined observations, the same proxy instead evaluates the predicted pair as

$$\hat{q}_{t+j} = \mathcal{Q}_\psi(\hat{\mathbf{o}}_{t+j}, \mathbf{u}_{t+j}) \in [0, 1].$$

Here $\mathbf{u}_{t+j}$ denotes the action input aligned with the transition at relative offset j; it can be instantiated as a per-step action or a local action window depending on the implementation. The proxy notation retains observation-action conditioning independently of action tokenization.

Role separation. The world model and reward proxy head serve different purposes. The world model predicts what visual future may follow an action-conditioned rollout, while the proxy head estimates how much local task progress that transition represents. This separation avoids forcing the video predictor to directly model sparse terminal success, and allows the progress estimator to be updated from success-to-go targets without changing the visual prediction objective. It also provides two reliability signals: visual mismatch measures prediction fidelity, while proxy mismatch measures progress-supervision reliability.

A.2 Detailed Simulator Objective

At training stage s, MERL updates the simulator from grounded trajectories collected by the current policy. The world model is supervised by one-chunk visual prediction, and the reward proxy head is supervised by a dense success-to-go target derived from sparse terminal success.

World-model objective. The grounded batch for one-chunk prediction is

$$\mathcal{B}_s^{\text{real}} = \{(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t, \mathbf{o}_{t+1:t+C}, z(\tau))\}, \tag{6}$$

From each anchor and action chunk, the world model predicts C future observations:

$$\hat{\mathbf{o}}_{t+1:t+C} = \mathcal{W}_\phi(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t). \tag{7}$$

The world-model loss aggregates temporally weighted discrepancies over grounded chunks:

$$\mathcal{L}_{\text{wm}}^{(s)} = \frac{1}{|\mathcal{B}_s^{\text{real}}|} \sum_{(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t, \mathbf{o}_{t+1:t+C}, z) \in \mathcal{B}_s^{\text{real}}} \sum_{j=1}^{C} \omega_j d(\hat{\mathbf{o}}_{t+j}, \mathbf{o}_{t+j}), \tag{8}$$

where $d(\cdot, \cdot)$ denotes an observation discrepancy and $\omega_j \geq 0$ is a temporal weighting coefficient. Depending on the backbone, d can be instantiated in pixel space, feature space, latent space, or as the reconstruction-equivalent term of a diffusion denoising objective.

Success-to-go proxy target. The environment provides a sparse terminal success label $z(\tau) \in \{0, 1\}$. To supervise local progress, MERL defines a truncated discounted success-to-go target:

$$q_t^\star(\tau) = z(\tau)\, \gamma_q^{\min(T-1-t,\, H_q-1)}, \tag{9}$$

where T is the trajectory length, $\gamma_q \in (0, 1]$ is the proxy discount factor, and H_q is the truncation horizon. The target satisfies $q_t^\star(\tau) \in [0, 1]$. For successful trajectories, states closer to completion receive larger values; for failed trajectories, the target remains zero. This target preserves the terminal task objective while providing bounded dense supervision for the proxy head.

Reward proxy objective. Grounded success-to-go supervision trains the proxy via

$$\mathcal{L}_{\text{proxy}}^{(s)} = \frac{1}{|\mathcal{B}_s^{\text{real}}|} \sum_{(\mathbf{o}_t, \mathbf{x}, \mathbf{u}_t, \mathbf{o}_{t+1:t+C}, z) \in \mathcal{B}_s^{\text{real}}} \sum_{j=0}^{C-1} \ell_q\big(\mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j}), q_{t+j}^\star(\tau)\big)\,, \tag{10}$$

where $\ell_q(\cdot,\cdot)$ is the proxy loss. In our implementation, $\mathcal{Q}_\psi$ is trained as a binary progress classifier with soft target $q_{t+j}^\star(\tau)$, and its positive-class probability is used as the scalar proxy score. With continuous progress labels, ℓ_q can instead use mean squared error or smooth ℓ_1.

Joint simulator objective. The simulator is updated by

$$\mathcal{L}_{\text{sim}}^{(s)} = \mathcal{L}_{\text{wm}}^{(s)} + \lambda_q \mathcal{L}_{\text{proxy}}^{(s)}, \tag{11}$$

where $\lambda_q \geq 0$ controls the proxy-loss weight in simulator training. This objective updates the visual predictor and the proxy estimator on grounded data from the current policy-induced distribution. The training weight λ_q is distinct from the reliability weights α_o, α_q in App. A.3.

A.3 Concrete Reliability Schedules

The main paper defines stage-level trust through two monotone scheduling functions f_ρ and f_h. Here we give the concrete instantiation used in MERL. The scheduler takes a smoothed stage-level reliability error $\tilde{\varepsilon}_s$ as input and outputs the imagined-data ratio ρ_s and rollout horizon h_s.

Stage-level reliability error. At training stage s, MERL estimates simulator reliability from grounded calibration chunks collected by the current policy. We reuse stored grounded windows and replay their recorded actions from the same anchor and instruction after simulator update. The stored observations supply calibration targets without additional environment interaction. Multi-chunk stored windows also supervise the residual predictor described in App. A.4. The raw stage-level reliability error combines visual prediction mismatch and proxy-progress mismatch:

$$\bar{\varepsilon}_s = \alpha_o \mathcal{E}_s^{\text{obs}} + \alpha_q \mathcal{E}_s^{\text{proxy}}, \tag{12}$$

where $\alpha_o, \alpha_q \geq 0$ are reliability-aggregation weights. The observation mismatch is

$$\mathcal{E}_s^{\text{obs}} = \frac{1}{|\mathcal{B}_s^{\text{real}}|} \sum_{(\cdot) \in \mathcal{B}_s^{\text{real}}} \sum_{j=1}^{C} \omega_j d(\hat{\mathbf{o}}_{t+j}, \mathbf{o}_{t+j}), \tag{13}$$

where C is the action chunk length, $d(\cdot,\cdot)$ is the observation discrepancy, and $\omega_j \geq 0$ is a temporal coefficient. Proxy mismatch compares estimated progress and its target:

$$\mathcal{E}_s^{\text{proxy}} = \frac{1}{|\mathcal{B}_s^{\text{real}}|} \sum_{(\cdot) \in \mathcal{B}_s^{\text{real}}} \sum_{j=0}^{C-1} \nu_j \left|\mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j}) - q_{t+j}^\star(\tau)\right|, \tag{14}$$

where $\nu_j \geq 0$ is a temporal coefficient for proxy mismatch. Thus, $\mathcal{E}_s^{\text{obs}}$ measures how accurately the world model predicts grounded visual futures, while $\mathcal{E}_s^{\text{proxy}}$ measures how accurately the reward proxy estimates local progress on grounded observation-action pairs.

Exponential smoothing. Online simulator updates can make the raw reliability error fluctuate across stages. To avoid unstable schedule changes, MERL uses an exponential moving average:

$$\tilde{\varepsilon}_s = \beta_\varepsilon \tilde{\varepsilon}_{s-1} + (1 - \beta_\varepsilon)\bar{\varepsilon}_s, \qquad \beta_\varepsilon \in [0, 1). \tag{15}$$

The scheduler interprets smaller $\tilde{\varepsilon}_s$ as higher stage-level simulator reliability.

Confidence score. We convert the smoothed error into a normalized confidence score:

$$\chi_s = \left(1 + \left(\frac{\tilde{\varepsilon}_s}{\varepsilon_{\text{ref}}}\right)^\kappa\right)^{-1}, \qquad \chi_s \in (0, 1], \tag{16}$$

where $\varepsilon_{\text{ref}} > 0$ is a reference error scale and $\kappa > 0$ controls the sharpness of the transition. This inverse-power form maps lower reliability error to higher confidence. The reference scale absorbs the units and normalization of the observation and proxy mismatch terms.

Imagination ratio schedule. The imagined-data ratio is

$$\rho_s = \rho_{\min} + (\rho_{\max} - \rho_{\min})\chi_s, \qquad 0 \le \rho_{\min} \le \rho_{\max} \le 1. \tag{17}$$

Here ρ_s controls the contribution of imagined experience in the mixed policy objective. In all experiments, we set $\rho_{\min} = 0.05$ and $\rho_{\max} = 0.95$. The lower bound keeps the update close to real-interaction refinement when the simulator is unreliable, while still retaining a small imagined branch for calibration and replay maintenance. The upper bound makes the reliable-simulator regime close to model-based refinement, while retaining grounded experience for calibration.

Imagined horizon schedule. The imagined rollout horizon is

$$h_s = \lfloor h_{\min} + (h_{\max} - h_{\min})\chi_s \rfloor, \qquad 1 \le h_{\min} \le h_{\max} \le H_{\max}. \tag{18}$$

Here h_s is measured in environment steps, and $H_{\max}$ is the maximum allowed imagined rollout horizon. It should not be confused with the action chunk length C. In our implementation, we set $h_{\min} = C$ and $h_{\max} = H_{\max} = 4C$. Since $C = 8$, the scheduled horizon is 8–32 environment steps; Fig. 3(e) displays h_s/C, ranging from 1 to 4 action chunks. A non-multiple of C uses only the valid portion of the final chunk, so its number of full-chunk equivalents can be fractional. When $h_s > C$, the policy and world model are recursively rolled out inside the simulator until the scheduled horizon is reached. This design keeps imagination local when simulator reliability is low, while allowing longer multi-chunk rollouts only after the simulator becomes more reliable.

A.4 Chunk-Level Replay Priority and Optimization Weight

At training stage s, MERL generates a candidate imagined batch $\widetilde{\mathcal{B}}_s^{\text{imag}}$ by sampling anchor observations from recent grounded trajectories and rolling out the current policy inside the evolving simulator. Stage-level trust controls the overall imagined ratio ρ_s and rollout horizon h_s, but chunks within $\widetilde{\mathcal{B}}_s^{\text{imag}}$ can still differ in local reliability. Chunk-level trust therefore controls two operations before policy optimization: which imagined chunks are sampled into $\mathcal{B}_s^{\text{imag}}$, and how strongly each sampled chunk affects the imagined policy-gradient loss.

Chunk-level reliability error. Each candidate's local error combines visual and proxy residuals:

$$\varepsilon_{s,i} = \alpha_o \mathcal{E}_{s,i}^{\text{obs}} + \alpha_q \mathcal{E}_{s,i}^{\text{proxy}}, \tag{19}$$

where $\mathcal{E}_{s,i}^{\text{obs}}$ and $\mathcal{E}_{s,i}^{\text{proxy}}$ denote estimated local visual mismatch and estimated local progress mismatch, respectively. For exact grounded–predicted pairs, we measure:

$$\mathcal{E}_{s,i}^{\text{obs}} = \sum_{j=1}^{c_i} \omega_j d(\hat{\mathbf{o}}_{i,j}, \mathbf{o}_{i,j}), \qquad \mathcal{E}_{s,i}^{\text{proxy}} = \sum_{j=0}^{c_i-1} \nu_j \left| \hat{q}_{i,j} - q_{i,j}^{\star} \right|. \tag{20}$$

Here $\hat{\mathbf{o}}_{i,j}$ and $\hat{q}_{i,j}$ are the imagined observation and proxy score at offset j of chunk i, while $\mathbf{o}_{i,j}$ and $q_{i,j}^{\star}$ denote the corresponding grounded observation and success-to-go target when available. Here $c_i \le C$ is the number of valid transitions in chunk i, so local residuals do not sum over the full multi-chunk horizon h_s. Exact pairing requires the same anchor, instruction, and recorded action continuation; proximity of two windows alone does not constitute a pair.

Residual predictor for unpaired recursive chunks. Let $\mathbf{v}_i^{\text{anchor}}$ and $\hat{\mathbf{v}}_i$ denote anchor and imagined visual latents, d_i the rollout depth in action chunks, and $\mathbf{c}_s$ the stage context. For a chunk without an exact grounded continuation, the deployed predictor computes

$$(\widehat{\mathcal{E}}_{s,i}^{\text{obs}}, \widehat{\mathcal{E}}_{s,i}^{\text{proxy}}) = g_\xi(\mathbf{v}_i^{\text{anchor}}, \hat{\mathbf{v}}_i, \mathbf{u}_i, d_i, \mathbf{c}_s). \tag{21}$$

After each update of $(\mathcal{W}_\phi, \mathcal{Q}_\psi)$, we replay exact stored grounded–predicted windows to obtain residual targets from Eq. (20), refresh g_ξ, and freeze it for imagination and policy optimization. For recursive calibration windows, recorded actions can be unrolled from the anchor through predicted intermediate observations, while stored futures supply the residual labels. At deployment,

the current policy instead samples later chunks from predicted observations; those unpaired chunks use predicted residuals. Both residual outputs enter $\widehat{\varepsilon}_{s,i} = \alpha_o \widehat{\mathcal{E}}^{\text{obs}}_{s,i} + \alpha_q \widehat{\mathcal{E}}^{\text{proxy}}_{s,i}$ and the same replay/weighting rules below. Future grounded observations are never predictor inputs or queried to score these recursive chunks. The depth validation in App. D.4 holds matched futures out of scoring and uses them only afterward to measure prediction quality. The Residual predictor in Tab. 7 is this no-oracle component; Combined MERL uses visual/proxy fusion with measured residuals where exact pairs exist and predicted residuals otherwise. It is not a separate source of oracle information. A smaller estimated error indicates higher predicted reliability, subject to calibration coverage.

Replay probability. Each chunk receives the inverse-error priority

$$a_{s,i} = (\varepsilon_{s,i} + \zeta_p)^{-\alpha_p}, \tag{22}$$

where $\zeta_p > 0$ prevents numerical singularity and $\alpha_p \geq 0$ controls the strength of reliability prioritization. Normalizing these candidate priorities gives the replay probability used for chunk sampling:

$$p_{s,i} = \frac{a_{s,i}}{\sum_{m \in \widetilde{\mathcal{B}}^{\text{imag}}_s} a_{s,m}}. \tag{23}$$

Thus, $p_{s,i}$ is a sampling probability over candidate imagined chunks, not a rank. When $\alpha_p = 0$, imagined chunks are sampled uniformly; larger α_p concentrates replay on lower-error chunks.

Optimization weight. The per-chunk policy weight depends on reliability:

$$w_{s,i} = \text{clip}\left(\exp(-\eta_w \varepsilon_{s,i}), w_{\min}, 1\right), \tag{24}$$

where $\eta_w \geq 0$ controls the decay rate and $w_{\min} \in (0, 1]$ prevents the imagined branch from vanishing entirely. The replay probability $p_{s,i}$ controls whether an imagined chunk is sampled into $\mathcal{B}^{\text{imag}}_s$, whereas the optimization weight $w_{s,i}$ controls how strongly the sampled chunk contributes to the imagined loss. This separation lets MERL bias replay toward reliable imagination while still reducing gradient influence from uncertain chunks.

A.5 Full Policy Objective

At training stage s, MERL constructs a grounded batch $\mathcal{B}^{\text{real}}_s$ and an imagined batch $\mathcal{B}^{\text{imag}}_s$ for policy optimization. The grounded batch is sampled from grounded interaction data, while the imagined batch is sampled from the candidate imagined batch $\widetilde{\mathcal{B}}^{\text{imag}}_s$ according to the replay probabilities $p_{s,i}$. The two batches are sampled independently. Grounded–imagined pairing is used only for reliability calibration when paired continuations are available; it is not required for policy optimization.

Mixed objective. We combine grounded and imagined policy updates as

$$\mathcal{L}^{(s)}_{\text{MERL}} = (1 - \rho_s)\mathcal{L}^{(s)}_{\text{real}} + \rho_s \mathcal{L}^{(s)}_{\text{imag}}. \tag{25}$$

Here ρ_s sets the stage-level mixture of grounded and imagined experience; $p_{s,i}$ and $w_{s,i}$ respectively control local replay admission and gradient influence.

Policy ratio. For chunk i and valid action token ℓ, the current-to-old policy probability ratio is

$$r_{i,\ell}(\theta) = \frac{\pi_\theta(a_{i,\ell} \mid \mathbf{o}_i, \mathbf{x}_i, \mathbf{a}_{i,<\ell})}{\pi_{\theta_{\text{old}}}(a_{i,\ell} \mid \mathbf{o}_i, \mathbf{x}_i, \mathbf{a}_{i,<\ell})}. \tag{26}$$

The index ℓ denotes a valid categorical action token; $\mathbf{o}_i$ is the grounded or predicted conditioning observation, and $\mathbf{a}_{i,<\ell}$ is prior-token context when used by the action head. Padding and invalid action elements are excluded from the token sum. Decoded/denormalized continuous commands are used for execution, and their densities are not substituted for categorical token probabilities. We write the per-chunk clipped token objective in the following compact form:

$$\ell_{\text{clip}}(i; \theta, \hat{A}_i) = -\sum_{\ell} \min\left(r_{i,\ell}(\theta)\hat{A}_i, \text{clip}\left(r_{i,\ell}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\right)\hat{A}_i\right), \tag{27}$$

where $\hat{A}_i$ is the group-relative advantage associated with chunk i, and ϵ_{clip} is the clipped policy-gradient threshold. The clipping symbol ϵ_{clip} is distinct from the visual discrepancy $\delta_{s,t,j}$.

Algorithm 1 Full MERL Online Training Procedure

Require: Pretrained policy π_θ, evolving simulator $(\mathcal{W}_\phi, \mathcal{Q}_\psi)$, perturbed environment $\mathcal{E}$
Require: Scheduler hyperparameters $\varepsilon_{\text{ref}}, \kappa, \rho_{\min}, \rho_{\max}, h_{\min}, h_{\max}, H_{\max}, \beta_\varepsilon$
Require: Replay hyperparameters $\alpha_p, \eta_w, \zeta_p, w_{\min}$, temporal weights $\{\omega_j\}, \{\nu_j\}$, clipping threshold ϵ_{clip}, total training stages S

Initialize smoothed stage-level error $\tilde{\varepsilon}_0$
for $s = 1, 2, \ldots, S$ **do**
  Collect grounded trajectories using current policy π_θ in environment $\mathcal{E}$
  Construct grounded batch $\mathcal{B}_s^{\text{real}}$
  Update world model $\mathcal{W}_\phi$ with $\mathcal{L}_{\text{wm}}^{(s)}$ in Eq. (8)
  Update reward proxy head $\mathcal{Q}_\psi$ with $\mathcal{L}_{\text{proxy}}^{(s)}$ in Eq. (10)
  Replay stored exact grounded–predicted windows; compute $\bar{\varepsilon}_s$ with Eq. (12)
  Refresh g_ξ on measured residuals; freeze simulator and predictor for imagination/policy update
  Smooth it to obtain $\tilde{\varepsilon}_s$ using Eq. (15)
  Compute confidence score χ_s using Eq. (16)
  Set imagined ratio ρ_s and rollout horizon h_s using Eqs. (17)–(18)
  Generate candidate imagined batch $\widetilde{\mathcal{B}}_s^{\text{imag}}$ by rolling out π_θ inside $(\mathcal{W}_\phi, \mathcal{Q}_\psi)$ for horizon h_s
  for each imagined chunk $i \in \widetilde{\mathcal{B}}_s^{\text{imag}}$ **do**
    Estimate chunk-level reliability error $\varepsilon_{s,i}$ using Eq. (19)
    Compute unnormalized priority score $a_{s,i}$ using Eq. (22)
    Compute optimization weight $w_{s,i}$ using Eq. (24)
  end for
  Normalize all candidate scores to $p_{s,i}$ using Eq. (23)
  Sample imagined batch $\mathcal{B}_s^{\text{imag}}$ from $\widetilde{\mathcal{B}}_s^{\text{imag}}$ according to $p_{s,i}$
  Update policy π_θ using $\mathcal{B}_s^{\text{real}}$, $\mathcal{B}_s^{\text{imag}}$, and the mixed objective in Eq. (25)
end for
return Refined policy π_θ

Grounded branch. The grounded branch averages the clipped objective over grounded chunks:

$$\mathcal{L}_{\text{real}}^{(s)} = \frac{1}{|\mathcal{B}_s^{\text{real}}|} \sum_{i \in \mathcal{B}_s^{\text{real}}} \ell_{\text{clip}}(i; \theta, \hat{A}_i^{\text{real}}). \tag{28}$$

Grounded advantages $\hat{A}_i^{\text{real}}$ use the same environment feedback or reward interface as MFRL.

Imagined branch. We weight the shared clipped objective by local trust:

$$\mathcal{L}_{\text{imag}}^{(s)} = \frac{1}{|\mathcal{B}_s^{\text{imag}}|} \sum_{i \in \mathcal{B}_s^{\text{imag}}} w_{s,i}\, \ell_{\text{clip}}(i; \theta, \hat{A}_i^{\text{imag}}). \tag{29}$$

The imagined advantage $\hat{A}_i^{\text{imag}}$ is computed from simulator-generated progress signals. Because $\mathcal{B}_s^{\text{imag}}$ is already sampled from $\widetilde{\mathcal{B}}_s^{\text{imag}}$ according to $p_{s,i}$, Eq. (29) should be read as a trust-weighted imagined update rather than an unbiased estimator of uniform imagined replay. This is intentional: MERL uses chunk-level trust to bias both replay admission and gradient contribution toward more reliable imagined experience. The GRPO-style instantiation shares its optimizer with the controls; trust acts through replay admission and loss weighting.

A.6 Full Training Algorithm

Algorithm 1 summarizes the complete online training procedure of MERL. To keep notation consistent with the main text, the algorithm uses $\mathcal{B}_s^{\text{real}}$ for grounded batches, $\widetilde{\mathcal{B}}_s^{\text{imag}}$ for candidate imagined chunks, and $\mathcal{B}_s^{\text{imag}}$ for imagined chunks sampled into policy optimization.

B Theoretical Proofs

This appendix provides the proofs and auxiliary bias arguments used to justify the trust mechanisms in MERL. The propositions establish conditional bias bounds rather than convergence guarantees. They show how visual prediction discrepancies propagate into proxy supervision and how this bias depends on imagined-data usage and rollout horizon.

B.1 Proof of Proposition 1

Proposition 1 explains why the imagined rollout horizon should depend on simulator reliability. The reward proxy head is supervised on grounded observation-action pairs, but during imagination it is evaluated on simulator-predicted observations. The proposition bounds the resulting proxy-supervision bias by visual prediction discrepancies.

At anchor t and horizon h, the discounted aggregate on imagined observations is

$$\hat{q}_t^{(h)} := \sum_{j=1}^{h} \gamma_q^{j-1} \mathcal{Q}_\psi(\hat{\mathbf{o}}_{t+j}, \mathbf{u}_{t+j}), \tag{30}$$

and define the corresponding analytical aggregate on grounded observations as

$$q_t^{(h)} := \sum_{j=1}^{h} \gamma_q^{j-1} \mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j}). \tag{31}$$

Both aggregates use the same anchor time, horizon, discount factor, and action continuation. They differ only in whether the proxy head is evaluated on simulator-predicted observations or grounded observations. The grounded aggregate serves analysis, not online training.

Assume that $\mathcal{Q}_\psi$ is L_q-Lipschitz with respect to observation features when the action input is fixed. For any two observations $\mathbf{a}, \mathbf{b}$ and the same action input $\mathbf{u}$,

$$|\mathcal{Q}_\psi(\mathbf{a}, \mathbf{u}) - \mathcal{Q}_\psi(\mathbf{b}, \mathbf{u})| \le L_q \Delta(\mathbf{a}, \mathbf{b}), \tag{32}$$

Here $\Delta(\mathbf{a}, \mathbf{b})$ measures discrepancy in observation-feature space. At stage s and anchor t, define the pointwise visual prediction error at relative offset j as

$$\delta_{s,t,j} := \Delta(\hat{\mathbf{o}}_{t+j}, \mathbf{o}_{t+j}) \tag{33}$$

$$\left|\hat{q}_t^{(h)} - q_t^{(h)}\right| = \left|\sum_{j=1}^{h} \gamma_q^{j-1} \Big(\mathcal{Q}_\psi(\hat{\mathbf{o}}_{t+j}, \mathbf{u}_{t+j}) - \mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j}) \Big)\right| \tag{34}$$

$$\le \sum_{j=1}^{h} \gamma_q^{j-1} \left|\mathcal{Q}_\psi(\hat{\mathbf{o}}_{t+j}, \mathbf{u}_{t+j}) - \mathcal{Q}_\psi(\mathbf{o}_{t+j}, \mathbf{u}_{t+j})\right| \tag{35}$$

$$\le L_q \sum_{j=1}^{h} \gamma_q^{j-1} \Delta(\hat{\mathbf{o}}_{t+j}, \mathbf{o}_{t+j}) \tag{36}$$

$$= L_q \sum_{j=1}^{h} \gamma_q^{j-1} \delta_{s,t,j}. \tag{37}$$

The first inequality uses the triangle inequality; the second uses the Lipschitz condition in Eq. (32). Together, these inequalities give the discounted proxy-supervision bias bound:

$$\left|\hat{q}_t^{(h)} - q_t^{(h)}\right| \le L_q \sum_{j=1}^{h} \gamma_q^{j-1} \delta_{s,t,j}. \tag{38}$$

□

Interpretation. The bound shows that imagined proxy bias is controlled by a discounted accumulation of visual prediction discrepancies. Even when the proxy head is well supervised on grounded data, evaluating it on inaccurate imagined observations can distort the progress signal. This accumulation motivates reliability-dependent scheduling of h_s.

B.2 Proof of Proposition 2

Proposition 2 explains why stage-level trust controls both the imagined-data ratio ρ_s and the imagined rollout horizon h_s. The statement should be interpreted as a scaling argument rather than a convergence theorem: if the discrepancy of imagined supervision grows with both how much imagination is used and how far it is rolled out, then scheduling only one of the two is insufficient.

Assume the stage-level imagined-supervision discrepancy admits the following bound:

$$\Delta_s \le K_\Delta \, \rho_s \sum_{j=1}^{h_s} \delta_{s,j}, \tag{39}$$

where $K_\Delta > 0$ is a task-dependent constant, ρ_s is the imagined-data ratio, h_s is the imagined rollout horizon, and $\delta_{s,j}$ is the stage-level visual prediction discrepancy at relative offset j. The quantity $\delta_{s,j}$ can be understood as an aggregate or upper bound of $\delta_{s,t,j}$ over sampled anchors.

Assume the smoothed reliability error bounds every discrepancy within the horizon:

$$\delta_{s,j} \le \tilde{\varepsilon}_s, \qquad \forall j \le h_s. \tag{40}$$

Here $\tilde{\varepsilon}_s$ is the normalized and smoothed stage-level reliability error used by the scheduler. Substitution of Eq. (40) into Eq. (39) bounds the aggregate discrepancy by

$$\Delta_s \le K_\Delta \, \rho_s \sum_{j=1}^{h_s} \tilde{\varepsilon}_s \tag{41}$$

$$= K_\Delta \, \rho_s \, h_s \, \tilde{\varepsilon}_s. \tag{42}$$

Consequently, the aggregate discrepancy satisfies the stated joint scheduling bound:

$$\Delta_s \le K_\Delta \, \rho_s \, h_s \, \tilde{\varepsilon}_s, \tag{43}$$

This establishes the joint ratio/horizon bound stated in Proposition 2. □

Interpretation. The bound makes explicit that imagined-supervision discrepancy scales with three factors: simulator reliability error $\tilde{\varepsilon}_s$, imagined-data ratio ρ_s, and rollout horizon h_s. Reducing only ρ_s can still allow long biased rollouts, while reducing only h_s can still admit too much unreliable imagined data. MERL therefore couples the ratio and horizon through a shared reliability signal.

B.3 Reliability-Aware Replay Bias Bound

The following proposition complements the stage-level analysis by justifying the chunk-level replay and weighting rule used in MERL. It concerns local heterogeneity among generated imagined chunks and is omitted from the main paper for space.

Proposition 3. Assume the bias of chunk i at stage s admits the following local reliability bound:

$$b_{s,i} \le K_b \varepsilon_{s,i}, \tag{44}$$

for some constant $K_b > 0$, where $\varepsilon_{s,i}$ is the estimated chunk-level reliability error. Let

$$\bar{w}_{s,i} = \frac{w_{s,i}}{\sum_m w_{s,m}} \tag{45}$$

denote normalized weights for a per-unit-weight bias comparison. Here B_s^{imag} refers to the bias of this normalized weighted mixture; the policy loss retains its chunk-count normalization in Eq. (29). Under the stated chunk-bias assumption, the aggregate bias satisfies

$$B_s^{\mathrm{imag}} \le K_b \sum_i \bar{w}_{s,i} \varepsilon_{s,i}. \tag{46}$$

For the comparison with uniform replay, suppose the chunk errors are ordered as

$$\varepsilon_{s,(1)} \le \varepsilon_{s,(2)} \le \cdots \le \varepsilon_{s,(N)} \tag{47}$$

and the corresponding normalized weights are sorted in the opposite order,

$$\bar{w}_{s,(1)} \ge \bar{w}_{s,(2)} \ge \cdots \ge \bar{w}_{s,(N)}, \qquad \sum_{i=1}^{N} \bar{w}_{s,(i)} = 1, \tag{48}$$

The reliability-weighted error then cannot exceed its uniform-weight counterpart:

$$\sum_{i=1}^{N} \bar{w}_{s,(i)} \varepsilon_{s,(i)} \le \frac{1}{N} \sum_{i=1}^{N} \varepsilon_{s,(i)}. \tag{49}$$

For monotone, well-calibrated weights, non-uniform chunk errors yield an aggregate bias bound at least as tight as that under uniform weighting, and typically tighter.

Proof. We define the imagined-branch bias as the normalized weighted sum of chunk biases:

$$B_s^{\text{imag}} := \sum_i \bar{w}_{s,i} b_{s,i}. \tag{50}$$

Using the chunk-wise bias assumption in Eq. (44), we obtain

$$B_s^{\text{imag}} = \sum_i \bar{w}_{s,i} b_{s,i} \tag{51}$$

$$\le \sum_i \bar{w}_{s,i} K_b \varepsilon_{s,i} \tag{52}$$

$$= K_b \sum_i \bar{w}_{s,i} \varepsilon_{s,i}, \tag{53}$$

This establishes the reliability-weighted imagined-branch bias bound in Eq. (46).

To compare with uniform weighting, let $u_i := 1/N$. Consider the sorted errors $\varepsilon_{s,(1)} \le \cdots \le \varepsilon_{s,(N)}$ and oppositely sorted normalized weights $\bar{w}_{s,(1)} \ge \cdots \ge \bar{w}_{s,(N)}$. Define the cumulative deviation

$$S_k := \sum_{i=1}^{k} \left(\bar{w}_{s,(i)} - \frac{1}{N} \right), \qquad k = 1, \ldots, N-1. \tag{54}$$

Because $\{\bar{w}_{s,(i)}\}$ is sorted in decreasing order and sums to one, it majorizes the uniform weights, which implies $S_k \ge 0$ for all k. By Abel summation,

$$\sum_{i=1}^{N} \left(\bar{w}_{s,(i)} - \frac{1}{N} \right) \varepsilon_{s,(i)} = \sum_{k=1}^{N-1} S_k \left(\varepsilon_{s,(k)} - \varepsilon_{s,(k+1)} \right). \tag{55}$$

Since $S_k \ge 0$ and $\varepsilon_{s,(k)} - \varepsilon_{s,(k+1)} \le 0$, each term on the right-hand side is non-positive. Therefore,

$$\sum_{i=1}^{N} \bar{w}_{s,(i)} \varepsilon_{s,(i)} \le \frac{1}{N} \sum_{i=1}^{N} \varepsilon_{s,(i)}. \tag{56}$$

This proves the comparison of reliability-aware and uniform weighting in Eq. (49). □

Interpretation. Proposition 3 motivates chunk-level trust: generated chunks can have heterogeneous local errors even when average simulator reliability is acceptable. If the trust weights are reasonably calibrated, assigning larger optimization weight to lower-error chunks yields a tighter aggregate bias upper bound than uniform weighting. This supports the joint use of stage-level scheduling and chunk-level replay and weighting for reliable policy refinement in MERL.

C Experimental Details

This appendix provides implementation details for the experiments in the main paper. We describe the benchmark scope, policy backbone, evaluation metrics, training infrastructure, and baseline implementations. The main text reports compact summaries, while this appendix specifies how the metrics are computed and how the online training budget is accounted for.

C.1 Benchmark and Perturbation Scope

We evaluate MERL on LIBERO-PRO [29], which extends LIBERO-style language-conditioned manipulation tasks with controlled deployment-time perturbations. The main experiments cover four perturbed suites: Spatial, Object, Goal, and Long. These suites keep task semantics comparable while shifting spatial layouts, object states, goal conditions, or long-horizon execution contexts. This protocol evaluates online VLA post-training under deployment distribution shifts rather than open-ended acquisition of entirely new task categories. It directly tests the failure mode targeted by MERL: frozen simulators become stale under policy-induced shifts, while online-updated simulators still require reliability auditing before policy optimization.

C.2 Policy Backbone and Action Representation

All methods use the same OpenVLA-OFT [21] policy checkpoint as initialization. In our refinement implementation, the policy uses a tokenized action head compatible with clipped policy-gradient optimization. In particular, MERL optimizes a GRPO-style objective that requires old-policy and new-policy action probabilities to compute policy ratios. Token-based VLA policies naturally expose such action distributions and support stochastic rollout sampling. By contrast, flow-matching or diffusion-action VLAs require additional likelihood estimation, trajectory-level reparameterization, or a different RL objective to compute comparable policy ratios. Extending MERL to those continuous-action generative policies is conceptually possible, but it is orthogonal to the trust-calibrated imagination mechanism and left for future work.

Following the OpenVLA-OFT setting, the policy predicts an action chunk of length $C = 8$ with a 7-dimensional action interface. We use a single global RGB observation and the language instruction as policy input in simulation. Unless otherwise specified, proprioceptive inputs are disabled to keep the setup aligned with the global-camera deployment setting used in our real-robot validation.

C.3 Evaluation Metrics

Let $\mathcal{G} = \{\mathrm{Spatial}, \mathrm{Object}, \mathrm{Goal}, \mathrm{Long}\}$ denote the set of perturbed evaluation suites. For suite $g \in \mathcal{G}$, let $\mathcal{T}_g$ denote its task set. We evaluate a policy at checkpoints indexed by $k = 0, \ldots, K$. Let n_k be the cumulative outer-loop refinement stage when checkpoint k is evaluated; it does not count environment transitions, optimizer minibatches, or task-horizon steps. All online methods are evaluated at the same checkpoint set $\{n_k\}_{k=0}^{K}$ under the same online training budget. For model-based methods, simulator-generated rollouts are part of the training procedure; S2T therefore measures learning speed along the common online training axis rather than claiming a pure real-interaction count.

Success rate. For task $m \in \mathcal{T}_g$ and evaluation rollout r, let $y_{g,m,r}^{(k)} \in \{0, 1\}$ indicate whether checkpoint k succeeds. The success rate of suite g at checkpoint k is

$$\mathrm{SR}_g(k) = 100 \cdot \frac{1}{|\mathcal{T}_g| R} \sum_{m \in \mathcal{T}_g} \sum_{r=1}^{R} y_{g,m,r}^{(k)}, \tag{57}$$

where R is the number of evaluation rollouts per task. Final average SR across the four suites is

$$\text{Avg. SR} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \mathrm{SR}_g(K). \tag{58}$$

Area under the training curve. AUC measures online learning efficiency over the full refinement trajectory. For suite g, we integrate the SR curve by the normalized trapezoidal rule:

$$\mathrm{AUC}_g = \frac{1}{n_K - n_0} \sum_{k=1}^{K} \frac{\mathrm{SR}_g(k-1) + \mathrm{SR}_g(k)}{2} \left(n_k - n_{k-1}\right). \tag{59}$$

We then average the suite-wise AUC values to obtain the overall learning-curve metric:

$$\mathrm{AUC} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \mathrm{AUC}_g. \tag{60}$$

Since SR is expressed in percentage, AUC is also reported on a percentage scale. This metric rewards methods that improve early and remain strong throughout online refinement.

Steps to threshold. S2T measures how many online training steps are required to reach a target success level. For suite g and threshold $\eta \in [0, 100]$, define

$$\mathrm{S2T}_g(\eta) = \inf \left\{ n \mid \mathrm{SR}_g(n) \geq \eta \right\}, \tag{61}$$

where $\mathrm{SR}_g(n)$ denotes the success-rate curve as a function of cumulative online training steps. In practice, policies are evaluated only at discrete checkpoints $\{n_k\}_{k=0}^{K}$. If $\mathrm{SR}_g(k) < \eta \leq \mathrm{SR}_g(k+1)$, we estimate the threshold-crossing point by linear interpolation:

$$\mathrm{S2T}_g(\eta) = n_k + \frac{\eta - \mathrm{SR}_g(k)}{\mathrm{SR}_g(k+1) - \mathrm{SR}_g(k)} \left(n_{k+1} - n_k\right). \tag{62}$$

If a method does not reach the threshold within the training budget, we mark it as not reached in threshold-sweep plots and use the maximum budget for aggregate summaries. Although online training steps are evaluated at discrete checkpoints, S2T can be fractional because the threshold-crossing point is interpolated between checkpoints and later averaged across suites. Smaller S2T therefore denotes faster threshold attainment under the shared protocol.

High-threshold summary. The main table reports a compact high-threshold summary, denoted as S2T-H. We use $\eta_g = 80$ for Spatial, Object, and Goal, and $\eta_g = 65$ for Long:

$$\text{S2T-H} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \mathrm{S2T}_g(\eta_g), \qquad \eta_g = \begin{cases} 80, & g \in \{\mathrm{Spatial}, \mathrm{Object}, \mathrm{Goal}\}, \\ 65, & g = \mathrm{Long}. \end{cases} \tag{63}$$

The lower threshold for Long reflects its harder long-horizon structure under the same training budget. All thresholds are fixed before method comparison and are shared across methods. Because each suite-level S2T is interpolated and S2T-H averages across suites, the reported S2T-H values are shown with one decimal place. We report full threshold-sweep curves to show that the comparison is not determined by a single threshold choice.

Metrics for SFT. SFT is evaluated as a fixed pretrained policy without online refinement. Therefore, we report its final SR as a reference but do not report AUC or S2T in the main table. Learning-curve plots show SFT as a horizontal reference.

C.4 Grounded Budgets and Statistical Comparisons

The common training axis counts outer-loop refinement stages. After stage s, every online method has collected $6s$ grounded trajectories subject to the same $\leq 3072s$ transition cap. Trajectory lengths can differ through termination; the cap is a shared interaction allowance, not a claim that every method realizes an identical transition count. Grounded–predicted calibration and residual-predictor refresh reuse stored windows and add zero environment interaction. Imagined rollouts are model computation and do not consume this grounded interaction budget. Consequently, the 10.9% S2T-H reduction against matched MFRL describes faster threshold attainment under the common interaction protocol, rather than a measured wall-clock or environment-transition saving.

Table 4: Paired 95% CIs over three seeds. AUC: points; S2T-H: refinement stages.

Comparison	Metric	Difference [95% CI]
MERL – WoVR	ΔAUC $\uparrow$	$2.2\,[1.2, 3.3]$
MERL – WoVR	ΔS2T-H $\downarrow$	$-9.2\,[-15.4, -3.1]$
MERL – matched Online-Updated MBRL	ΔAUC $\uparrow$	$5.0\,[3.8, 6.2]$

Tab. 2 reports means and standard deviations from three training seeds. SimpleVLA-RL, WMPO, WoVR, and MERL share SFT initialization, split, grounded budget, action chunk, optimizer/reward interface, checkpoints, and evaluation. WMPO and WoVR are same-protocol ports that change benchmark/model assets while retaining their defining algorithms; each WoVR seed uses an independent simulator. The component controls in Tab. 1 retain their original point estimates and serve a different comparison from these external algorithms. Paired three-seed contrasts appear in Tab. 4.

C.5 Training Infrastructure

All main experiments are conducted on a single node with $4\times$ NVIDIA H100 GPUs with 80GB memory. We allocate three GPUs for actor rollout, policy inference, log-probability computation, and policy optimization, and reserve one GPU for simulator update and simulator inference. This separation avoids resource contention between the VLA policy and the evolving simulator, and allows simulator updates to proceed without blocking all actor workers.

A standard 100-stage online run takes approximately 35 hours under our current implementation. Each online training stage consists of grounded rollout collection, simulator update, reliability estimation, imagined rollout generation, and mixed policy optimization. We use Ray-based distributed execution inherited from the VLA-RL training stack, with FSDP-style sharding and gradient checkpointing for the actor model. Headless rendering uses an EGL/GLX-compatible service.

C.6 Baseline Implementation

The component controls share the same policy initialization, task suite, grounded budget, rollout temperature, action chunk length, and evaluation protocol. The concrete external methods in Tab. 2 are evaluated separately as same-protocol comparisons; they are not renamed component controls.

SFT. Pretrained OpenVLA-OFT is evaluated directly on perturbed LIBERO-PRO.

MFRL. MFRL performs model-free policy refinement. It uses the same GRPO-style policy optimizer and environment reward interface as MERL, but disables simulator-generated imagined rollouts. At a matched budget, this control isolates gains from imagination.

Static-MBRL. Static-MBRL enables simulator-generated imagined rollouts but keeps the simulator frozen during online policy refinement. It uses a fixed imagined-data ratio, a fixed imagined rollout horizon, and uniform imagined replay. This baseline tests whether merely adding imagined data is sufficient when the simulator is not updated online.

Static-MBRL+Trust. Static-MBRL+Trust also keeps the simulator frozen, but applies the same stage-level scheduling and chunk-level replay weighting as MERL. This baseline isolates the effect of trust calibration without online simulator evolution.

Online-Updated MBRL. Online-Updated MBRL updates the simulator during policy refinement, but uses generated imagination with fixed admission and uniform replay. This baseline corresponds to the online-updated model-based refinement paradigm discussed in the main paper and tests whether simulator update alone is sufficient.

Table 5: Component ablation on simulator evolution and two-level trust. Values preserve the original matched-control evaluation; three-seed external comparisons are reported separately in Tab. 2.

Variant	Avg. SR ↑	AUC ↑	S2T-H ↓
MFRL	77.2	67.9	103.4
Static-MBRL	73.3	59.8	168.9
Online-Updated MBRL	76.2	65.4	139.6
Online-Updated+StageTrust	77.9	68.4	113.2
Online-Updated+ChunkTrust	77.5	67.6	121.5
MERL (Ours)	**79.7**	**70.6**	**92.1**

Table 6: Simulator diagnostics on held-out policy-induced trajectories. The frozen simulator corresponds to Static-MBRL, while the evolving simulator is updated online during MERL training.

Simulator	PSNR ↑	SSIM ↑	LPIPS ↓	FVD ↓	Proxy MAE ↓	Proxy MSE ↓
Frozen simulator	22.4	0.684	0.238	286.0	0.183	0.057
Evolving simulator	**25.9**	**0.759**	**0.171**	**198.0**	**0.126**	**0.031**

MERL. MERL updates the simulator online and applies both stage-level and chunk-level trust. Stage-level trust controls the imagined-data ratio and rollout horizon, while chunk-level trust controls replay probability and optimization weight for individual imagined chunks.

D Additional Ablations and Diagnostics

This section provides supplementary analyses omitted from the main paper due to space limits. We focus on three aspects: simulator evolution, trust-estimator validity, and cross-suite robustness. We do not report a separate perturbation-severity sweep because the main benchmark already evaluates heterogeneous perturbation suites under a fixed real-interaction budget. Instead, robustness is assessed by performance consistency across Spatial, Object, Goal, and Long suites, together with the real-robot deployment validation in App. E.

D.1 Component Ablation

The matched ablation retains the policy initialization, grounded budget, and refinement protocol. StageTrust enables ratio/horizon scheduling; ChunkTrust enables reliability-based replay and loss weighting. Their joint use gives the best results among these variants.

D.2 Simulator Evolution Diagnostics

A central claim of MERL is that online simulator evolution improves the quality of simulator-generated imagination under the current policy-induced distribution. We therefore evaluate the simulator separately from downstream policy performance. The comparison is conducted on held-out grounded trajectories collected during online refinement, using the same anchor observations, instructions, and action chunks for all simulators. This held-out comparison isolates simulator prediction quality from the effects of downstream policy optimization.

Quantitative metrics. For visual prediction, we report PSNR and SSIM as fidelity metrics, and LPIPS and FVD as perceptual and temporal consistency metrics. For the reward proxy head, we report MAE and MSE against the truncated success-to-go target. Higher PSNR/SSIM and lower LPIPS/FVD/MAE/MSE indicate better simulator reliability.

Results. Tab. 6 shows that online simulator evolution improves both visual prediction and proxy-progress estimation. Compared with the frozen simulator, the evolving simulator improves PSNR from 22.4 to 25.9, reduces LPIPS from 0.238 to 0.171, and reduces FVD from 286.0 to 198.0. The proxy error also decreases, with MAE reduced from 0.183 to 0.126. These results support the

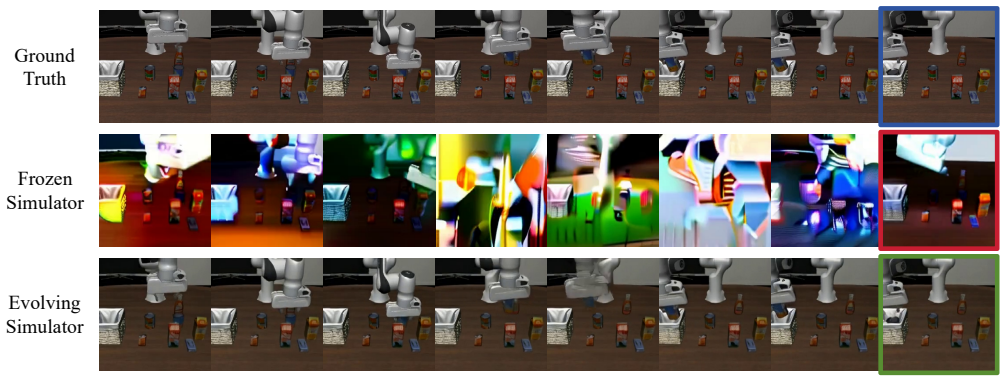


Figure 4: Qualitative comparison of simulator predictions. Rows show ground-truth future frames, predictions from the frozen simulator, and predictions from the evolving simulator under the same rollout condition. The evolving simulator produces visual continuations that remain closer to the ground truth over the prediction horizon.

Table 7: Trust-estimator validity and downstream policy performance.

Estimator	Pearson $r \uparrow$	Rank-AUC $\uparrow$	ECE $\downarrow$	Top30 Prec. $\uparrow$	AUC $\uparrow$
Random / uniform	0.02	0.50	0.176	0.31	66.1
WM latent error only	0.55	0.72	0.114	0.66	67.0
Proxy residual only	0.49	0.68	0.126	0.61	66.6
Residual predictor	0.71	0.82	0.063	0.79	69.4
Combined MERL estimator	**0.79**	**0.87**	**0.044**	**0.86**	**70.6**

need for simulator evolution: as the policy visits states that are not fully covered by the initial data distribution, updating the simulator keeps imagined rollouts better aligned with newly induced trajectories. However, improved average simulator quality does not imply that every imagined rollout is reliable, which motivates the trust diagnostics in App. D.3.

Qualitative visualization. Fig. 4 compares simulator-predicted future frames against the ground-truth trajectory under the same rollout condition. The frozen simulator quickly departs from the real visual sequence, producing strong color artifacts, distorted robot geometry, and misplaced objects in later frames. By contrast, the evolving simulator preserves the tabletop layout, robot pose, and object configurations more faithfully over the prediction horizon. This visualization complements Tab. 6 by showing that online simulator evolution improves not only aggregate prediction metrics, but also the visual reliability of imagined rollouts used for policy refinement.

D.3 Trust-Estimator and Trust-Chain Diagnostics

We next examine whether the reliability estimates used by MERL are informative. This analysis addresses a potential concern that the trust mechanism may simply act as a heuristic schedule. We evaluate whether estimated reliability correlates with actual rollout error, whether low-error chunks are preferentially replayed, and whether trust calibration stabilizes policy updates.

Estimator validity. We compare several reliability estimators. For each estimator, we measure correlation with observed rollout residuals, ranking quality, calibration error, and downstream performance. Here Rank-AUC measures how well the estimator ranks lower-error chunks ahead of higher-error chunks, ECE measures calibration error, and Top30 Low-Error Precision measures the fraction of truly low-error chunks among the top 30% most trusted samples.

Tab. 7 shows that the combined estimator is more predictive than single-source reliability signals. Using only world-model latent error or proxy residual improves over random scoring, but both miss

Table 8: Trust-chain diagnostics. Negative correlations indicate that lower estimated error leads to more imagination, longer rollout horizons, higher replay probability, or larger optimization weight.

Diagnostic link	Statistic	Value
Simulator error vs. imagined ratio ρ_s	Spearman	-0.967
Simulator error vs. rollout horizon h_s	Spearman	-0.963
Chunk error vs. replay probability $p_{s,i}$	Spearman	-0.750
Chunk error vs. trust weight $w_{s,i}$	Spearman	-0.950
High-error quartile replay share / uniform share	Ratio	0.180
KL standard deviation reduction vs. Online-Updated MBRL	Relative	0.430

Table 9: No-oracle trust estimation with matched future observations withheld from scoring. The first two rows give estimator reference results; subsequent rows report recursive-depth estimates with the corresponding trajectory-bootstrap 95% confidence intervals.

Setting	Pearson $r \uparrow$	Rank-AUC $\uparrow$	ECE $\downarrow$	Top30 Prec. $\uparrow$
Residual predictor	0.71	0.82	0.063	0.79
Combined MERL	0.79	0.87	0.044	0.86
Depth 2 ($N = 600$)	0.72 [0.68,0.76]	0.82 [0.79,0.85]	0.061 [0.052,0.070]	0.80 [0.76,0.84]
Depths 3–4 ($N = 900$)	0.64 [0.60,0.68]	0.77 [0.74,0.80]	0.074 [0.065,0.083]	0.74 [0.70,0.78]
All 2–4 ($N = 1500$)	0.68 [0.65,0.71]	0.80 [0.78,0.82]	0.067 [0.060,0.074]	0.77 [0.74,0.80]

part of the local failure structure. The combined estimator achieves the highest correlation, ranking quality, and downstream AUC, supporting both residual sources for local chunk-level trust.

Trust-chain diagnostics. We further verify whether the estimated trust signals are actually used as intended by the scheduler and replay mechanism. Tab. 8 reports correlations between reliability errors and the resulting schedules, replay probabilities, and policy weights.

These diagnostics confirm the intended trust chain. At the stage level, lower simulator error increases the imagined-data ratio ρ_s and permits longer rollout horizons h_s. At the chunk level, lower chunk-level reliability error $\varepsilon_{s,i}$ leads to higher replay probability $p_{s,i}$ and larger trust weight $w_{s,i}$. High-error chunks are substantially under-sampled compared with uniform replay, and policy-update variability is reduced relative to Online-Updated MBRL. This supports the interpretation that MERL improves performance by auditing imagined data before it shapes policy optimization, rather than by merely adding more imagined samples.

D.4 No-Oracle Trust Across Rollout Depths

This validation isolates the deployed residual predictor from access to matched future observations. The predictor is frozen after calibration. For recursive chunks at depths 2–4, matched future ground truth is withheld from scoring and used only afterward to evaluate residual estimates. Depth counts action chunks from the grounded anchor, not individual environment steps. The held-out set contains 600 depth-2 chunks and 900 depth-3–4 chunks, totaling 1500 chunks. Intervals in Tab. 9 are trajectory-bootstrap 95% confidence intervals.

Pearson r measures association between predicted and observed residuals. Rank-AUC measures ranking of lower-error chunks; ECE compares predicted trust with empirical reliability; Top30 precision is the fraction of truly low-error chunks among the top 30% most trusted samples. Prediction remains informative at depths 3–4, although ranking and calibration degrade relative to depth 2. Predicted trust remains useful within the evaluated horizons, while grounded calibration and conservative stage-level scheduling are retained throughout refinement.

Table 10: Matched-count trust-bin policy AUC: mean $\pm$ SD over three seeds.

Trust bin	Top	Mid	Bottom
Policy AUC $\uparrow$	69.9 ± 0.8	68.3 ± 0.9	66.7 ± 1.0

Table 11: Cross-suite robustness across perturbed LIBERO-PRO suites. Avg. SR measures overall performance, Worst SR measures the weakest suite, and Std. measures sensitivity across suites.

Method	Avg. SR $\uparrow$	Worst SR $\uparrow$	Std. $\downarrow$
MFRL	77.2	66.0	6.61
Static-MBRL	73.3	60.7	7.36
Online-Updated MBRL	76.2	66.1	**6.08**
MERL (Ours)	**79.7**	**68.8**	6.37

Downstream policy utility. We refine policies with matched numbers of chunks from Top, Mid, and Bottom trust bins, under the same update protocol and three training seeds. Tab. 10 reports the resulting policy AUC. The Top–Bottom contrast is $3.2\,[2.0, 4.4]$ (95% CI). Matching chunk counts limits the explanation based on sample quantity alone; the experiment tests the utility of trust-ranked data under this controlled refinement protocol.

D.5 Hyperparameter Sensitivity

We vary scheduler sharpness κ and simulator proxy-loss weight λ_q one at a time, using multipliers $\{0.5, 1, 2\}$ relative to their default settings. The former changes the reliability-to-confidence mapping in Eq. (16); the latter changes proxy supervision in Eq. (11). Across these settings, Avg. SR and AUC change by at most 1.4 and 1.6 points, respectively. Every setting remains at least $+1.8$ Avg. SR and $+3.1$ AUC points above the controlled Online-Updated MBRL baseline. The bounds apply to this one-at-a-time sweep and do not imply robustness to arbitrary joint changes.

D.6 Cross-Suite Robustness

Although we do not report a manually tuned perturbation-severity sweep, we evaluate robustness by measuring consistency across heterogeneous LIBERO-PRO perturbation suites. This avoids introducing an additional perturbation protocol while still testing whether a method remains effective under different spatial, object, goal, and long-horizon shifts.

Results. Tab. 11 shows that MERL obtains the strongest average and worst-suite success rates. Its cross-suite standard deviation is not the lowest; this statistic reflects variation in suite difficulty and differs from across-seed uncertainty. For robustness, the more relevant observation is that MERL raises the weakest-suite performance from 66.1 for Online-Updated MBRL and 66.0 for MFRL to 68.8. This indicates that trust-calibrated imagination improves performance under the most difficult evaluated perturbation setting, rather than only improving average performance on easier suites.

E Real Robot Evaluation

We further evaluate MERL on a physical robot to examine whether trust-calibrated imagination can support real-world deployment. This study is intended as a real-robot deployment validation rather than a full online real-world RL benchmark. Because real-robot interaction is expensive and safety-constrained, we collect a small set of expert demonstrations, perform offline refinement on a server, and deploy the refined checkpoints back to the robot for inference-only evaluation.

Platform. Experiments are conducted on a single-arm AgileX PiPER X tabletop manipulation platform equipped with a two-finger gripper. The robot is observed by a fixed global RGB camera, following a PiPER-family tabletop deployment setup similar to prior compositional simula-

tion platforms [31]. At deployment time, the policy receives the global camera image and the language instruction, predicts a 7-dimensional action command, and sends the command to the robot controller. All evaluated methods use the same global RGB camera pose, workspace boundary, control interface, reset protocol, and control frequency.

Real-robot data collection. We collect expert demonstrations on two tabletop manipulation tasks: *Block-to-Box*, where the robot grasps a wooden block and places it into a box, and *Lid-to-Pot*, where the robot grasps a cooker lid and places it on the cooker. For each task, we collect 50 demonstrations. Each demonstration contains synchronized RGB video, executed actions, robot joint states, and gripper states. During collection, object locations and object colors are varied to introduce moderate pose and appearance generalization while keeping the task semantics fixed.

Fixed-data offline refinement. All methods use the same 50 demonstrations per task, hardware interface, offline update budget, optimizer, and action horizon. The demonstration dataset is fixed throughout refinement, and no additional physical policy rollouts are collected. SFT provides the supervised policy reference; MFRL is the grounded-only refinement control on these fixed data. Its name denotes the control family and does not imply online physical interaction in this study. Online-Updated MBRL adapts the world model on stored robot data and uses visual imagination. MERL-Visual uses the visual-only trust configuration, with $\alpha_o = 1$ and $\alpha_q = 0$. Full MERL enables the reward proxy $\mathcal{Q}_\psi$ and joint visual/proxy trust. The visual-only variant disables proxy supervision for imagined refinement, while the full variant evaluates the complete simulator/trust interface.

At each offline refinement stage, simulator adaptation and calibration reuse the demonstrations. The policy mixes grounded demonstration chunks with imagined chunks; stage-level trust controls the amount and horizon of imagination, and chunk-level trust controls replay and loss weighting. The residual predictor is refreshed from exact stored pairs and frozen for unpaired recursive scoring as in App. A.4. The server-refined checkpoints are then deployed for closed-loop evaluation.

Evaluation protocol. Each method is evaluated on 20 trials per task under the same held-out object placements and color variations. Success requires completing the command without human intervention within the episode horizon. Failures include incorrect approach, unstable grasp, premature release, collision, placement outside the target, or timeout. All methods share camera pose, workspace, control interface, reset protocol, and control frequency.

Real-robot results. Tab. 3 reports all five methods, including the visual-only configuration and proxy-enabled full method. Online-Updated MBRL achieves 13/20 Block-to-Box and 11/20 Lid-to-Pot successes (60.0% average). MERL-Visual achieves 16/20 and 15/20 (77.5%), preserving the visual-only result. Full MERL achieves 17/20 and 15/20 (80.0%), while SFT and MFRL obtain 35.0% and 55.0%. The average-SR 95% CIs are $[22.1, 50.5]$, $[39.8, 69.3]$, $[44.6, 73.7]$, $[62.5, 87.7]$, and $[65.2, 89.5]$ for SFT, MFRL, Online-Updated MBRL, MERL-Visual, and full MERL, respectively. The full and visual-only intervals overlap substantially; these trials support feasibility of full-method deployment without isolating a statistically significant proxy contribution.

Qualitative rollout visualization. Fig. 6 visualizes MERL deployment on two real-robot tasks. For each task, we show the policy's real closed-loop execution frames together with simulator-generated visual continuations conditioned on the corresponding rollout context. The real frames demonstrate that the refined policy can approach the target, execute grasping, and complete placement under held-out object poses and appearance variations. The simulator rows provide an aligned observation-space view of the same manipulation process, showing that the evolving simulator preserves the main scene layout, robot motion, and object-level task progress along the rollout. This four-row visualization complements the success-rate table by showing both physical execution and the simulator imagination used by MERL during real-robot adaptation.

Scope and limitations. This study covers fixed-data offline adaptation and closed-loop physical deployment on two tabletop tasks. It does not evaluate online physical RL or additional policy-

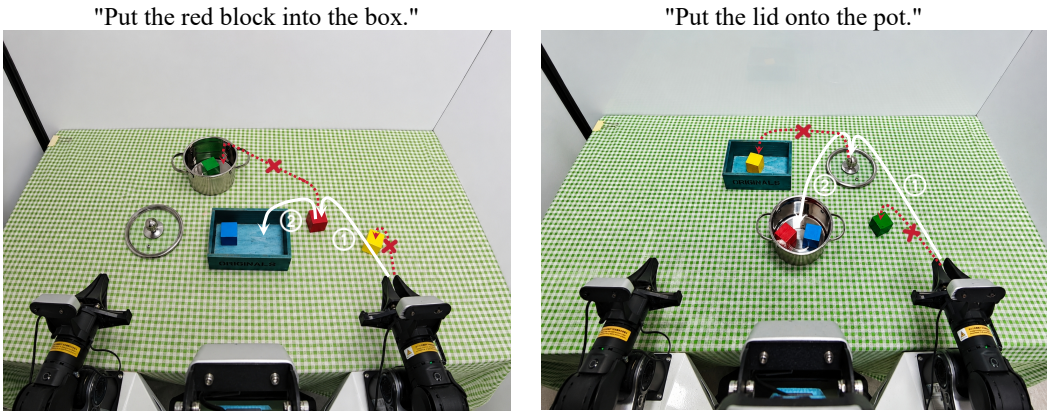


Figure 5: AgileX PiPER X task setups for fixed-data adaptation and closed-loop physical evaluation.

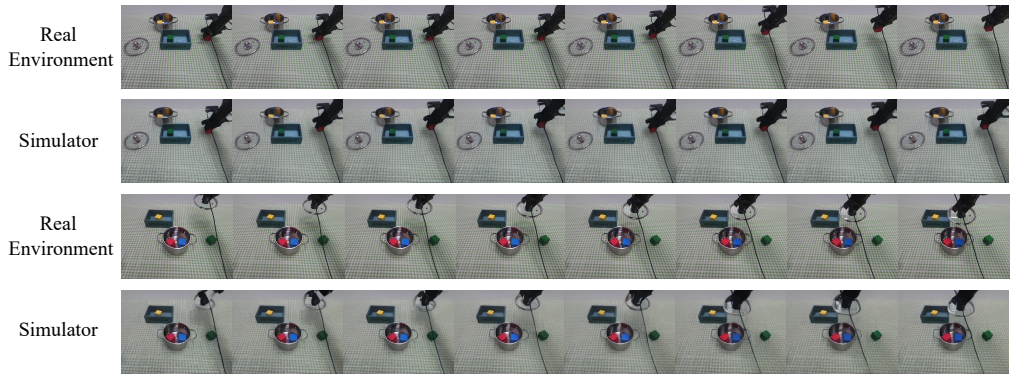


Figure 6: MERL real-robot deployment and simulator visualization on AgileX PiPER X. From top to bottom, rows show real Block-to-Box rollout, simulator-generated Block-to-Box continuation, real Lid-to-Pot rollout, and simulator-generated Lid-to-Pot continuation.

driven data collection. Remaining failures include fixed-camera depth ambiguity, gripper slippage, camera–robot calibration drift, and contact dynamics that short-horizon RGB prediction may not capture. The limited trial count and overlapping intervals constrain conclusions about the proxy-specific gain. Broader physical tasks and interactive refinement require further evaluation.

F Additional Related Work on World Models

This section complements the main related work by positioning the evolving simulator in MERL within recent world-model paradigms. Existing world models differ in their representation space, prediction target, and compatibility with downstream policy refinement. The central distinction for MERL is not only how a future state is predicted, but whether the predicted future can be safely used as training data for VLA policy post-training.

Implicit predictive world models. A first line of work studies implicit or latent predictive world models, where the objective is to learn predictive representations rather than reconstruct future observations in pixel space. JEPA-style models are representative of this direction, predicting abstract latent targets while discarding unpredictable visual details. Recent examples include V-JEPA 2, which combines large-scale self-supervised video pretraining with a small amount of robot interaction data for understanding, prediction, and planning [32], and V-JEPA 2.1, which further improves dense visual grounding and robot grasping [33]. These methods provide strong abstractions and are attractive for planning, but they are less directly aligned with our setting, where the

VLA policy consumes visual observations and the refinement loop benefits from explicit generated rollouts that can be evaluated, filtered, and replayed.

3D spatial world models. A second line explicitly represents the world in 3D, using point-based, Gaussian-based, or geometry-aware structures. This direction improves spatial consistency and physical grounding beyond 2D future prediction. Representative examples include GWM, which uses Gaussian splatting and diffusion-based dynamics for robotic manipulation [34], and PointWorld, which unifies state and action in a shared 3D point-flow space for in-the-wild robotic manipulation [35]. Such models are promising for spatial reasoning and contact-aware control, but they often require specialized reconstruction pipelines and 3D state representations. By contrast, MERL targets online VLA post-training with RGB observations and chunked actions, where a visual rollout interface is easier to integrate with existing image-conditioned policies.

Video-generative world models. MERL is most closely related to video-generative world modeling. Large-scale video generation models such as Open-Sora 2.0 [36] and Wan [37] have improved the realism, controllability, and scalability of video synthesis. Robotics and embodied-AI works further adapt video generation for action-conditioned prediction, synthetic trajectory generation, and visual world knowledge transfer, including humanoid video world models [38], DreamGen-style synthetic trajectory generation [39], and transferable visual world knowledge from large-scale videos [40]. This branch is attractive for VLA refinement because it operates in the same observation space as image-conditioned policies and can generate rollouts that resemble downstream policy inputs. However, video realism alone is not sufficient for policy learning. Most video-generative world models optimize future-frame plausibility or temporal coherence, but do not by themselves decide whether a generated rollout provides reliable supervision for policy optimization. MERL augments visual imagination with a reward proxy and two-level trust that schedules, selects, and weights rollouts using simulator reliability.

RSSM-style world models for reinforcement learning. Another influential line comes from model-based reinforcement learning, starting from early neural world models [41] and recurrent state-space models such as PlaNet [42], and later developed through the Dreamer family. In this paradigm, the world model is part of the reinforcement-learning loop and usually predicts latent transitions, rewards, and continuation signals. Representative examples include DreamerV3 for latent imagination [43], Dreamer 4 for scalable imagination training [44], TWISTER for transformer-based world modeling with contrastive predictive coding [45], and Dreamer-CDP for reconstruction-free deterministic representation prediction [46]. These methods are natively integrated with RL because reward prediction is part of the modeling stack. However, they typically operate in compact latent state spaces and are not designed around modern VLA policies that take high-dimensional visual observations and output chunked actions. MERL borrows the reinforcement-learning motivation of using imagination for policy improvement, while retaining a visual simulator interface compatible with VLA post-training.

Position of MERL. MERL is not proposed as a new world-model architecture. Instead, it addresses a complementary question: once a visual simulator produces imagined rollouts for VLA post-training, how should those rollouts be trusted? The evolving simulator keeps imagination aligned with newly collected grounded trajectories, while stage-level trust controls the imagined-data ratio ρ_s and rollout horizon h_s, and chunk-level trust controls replay probabilities $p_{s,i}$ and optimization weights $w_{s,i}$. This auditing layer explicitly regulates how imagined experience enters policy optimization, complementing advances in simulator fidelity.