WORLDGUIDE: LEARNING SUCCESS–FAILURE BOUNDARIES IN LATENT WORLD MODELS FOR VISION-LANGUAGE-ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models offer a promising way to improve Vision-Language-Action policies by capturing the consequences of actions. However, models trained primarily on expert demonstrations have limited exposure to failure outcomes and may struggle to distinguish visually similar successful and failed interactions. We propose **WorldGuide**, a framework that learns these distinctions in latent space and uses them to guide policy training. WorldGuide combines predictive pretraining on successful and failed trajectories with contrastive learning on matched success–failure pairs. The learned predictor then provides a differentiable reward to guide joint optimization of the policy and visual encoder. The predictor is discarded after training, so deployment requires no additional world-model inference. Extensive experiments show that WorldGuide substantially improves VLA reliability and achieves state of the art performance on LIBERO 100 and SimplerEnv, reaching **96.8%** and **72.0%**, respectively. Code will be publicly available.

1 INTRODUCTION

Vision-Language-Action (VLA) models Intelligence et al. (2025); Zhang et al. (2026); Bjorck et al. (2025) translate visual observations and language instructions into robot actions, providing a unified framework for learning manipulation skills from demonstrations. While imitation learning directly supervises policies to reproduce expert actions, it offers limited explicit supervision about how those actions affect subsequent states. World models complement this action supervision by predicting how interactions evolve. Recent latent world model approaches Bu et al. (2025); Cen et al. (2025); Sun et al. (2026); Zheng et al. (2025) introduce future-representation prediction as an auxiliary objective, encouraging policies to encode action-relevant state transitions and thereby improving manipulation performance.

Using latent predictions to guide policy improvement, however, requires the world model to capture the consequences of actions proposed by the policy, including those that depart from expert behavior. Consider a policy that closes its gripper slightly away from an object and then attempts to lift it. Although the action sequence resembles a successful grasp, the object remains on the table. A model trained predominantly on successful demonstrations receives limited supervision for this outcome and may still predict a future resembling successful execution. As illustrated in Fig. 1 (a), comparing such predictions with a successful goal can yield similar feedback for actions with different outcomes, providing little guidance for avoiding the failed interaction.

Training on failure trajectories expands this coverage by exposing the world model to unsuccessful transitions. Yet observing these transitions does not ensure that their outcome-relevant differences are prominent in the learned representation. A future-feature prediction objective does not explicitly prioritize differences according to their importance for task completion. Small discrepancies in contact or alignment may therefore contribute little to the latent distance even when they change the outcome. This motivates complementing predictive learning with supervision that makes successful and failed interactions more distinguishable.

Outcome labels provide such supervision, but how the corresponding trajectories are compared matters. Arbitrary successful and failed executions can differ in scene configuration, task progress, and motion patterns, allowing their representations to be separated using incidental cues. We therefore

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

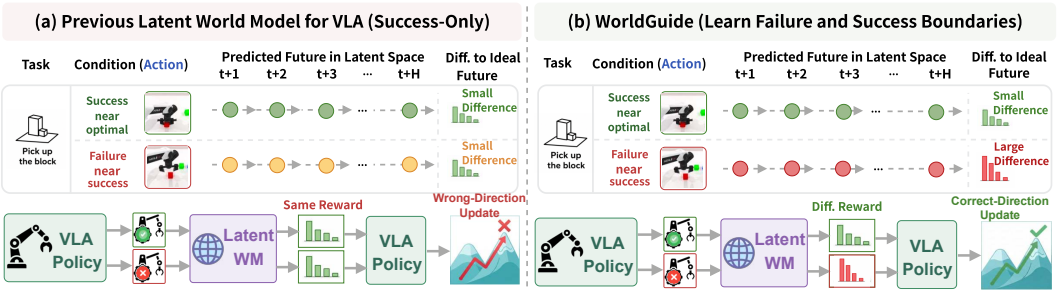


Figure 1: **Schematic motivation for WorldGuide.** (a) Success-only training poorly constrains latent dynamics for failures. (b) WorldGuide combines failure-rich predictive learning with matched comparisons to yield outcome-sensitive representations for policy guidance.

construct within-task comparisons between trajectories that are visually similar segments from the same task, matched around annotated failure onsets. A contrastive objective uses these matched examples to encourage outcome-discriminative representations, while the predictive objective preserves supervision for their state transitions. Together, these objectives address two complementary requirements: learning what happens during unsuccessful interactions and retaining the subtle distinctions that separate them from successful execution.

We introduce **WorldGuide**, which learns success–failure boundaries to guide policy optimization through three training stages. First, Failure-Rich Pretraining (Stage 1) mixes expert demonstrations with diverse failure trajectories, ensuring the world model learns complete physical dynamics rather than just success cases. Second, Boundary-Aware Contrastive Finetuning (Stage 2) pairs success and failure trajectories from the same task that look nearly identical in observations and actions. This forces the model to learn the true boundary between success and failure instead of relying on superficial shortcuts. Third, End-to-End Optimization with Differentiable Reward (Stage 3) freezes the trained world model and uses it as a penalty. When a candidate action is predicted to lead to a failed future, its prediction error is directly backpropagated to update the action head and visual encoder, steering the policy away from mistakes with zero deployment cost. In summary, our main contributions are summarized as follows:

- We study the limitations of learning latent predictive feedback from successful demonstrations and examine how failure data and matched success–failure supervision affect outcome discrimination.
- We propose *WorldGuide*, a three-stage training recipe that turns failure knowledge into policy guidance through failure-rich predictive pretraining, contrastive fine-tuning on matched interactions, and differentiable reward-guided policy training.
- Extensive simulation and real-world experiments show that *WorldGuide* achieves competitive performance with the current SOTA model WAM while setting a new SOTA among VLA baselines (achieving **96.8%** on LIBERO-100). Crucially, *WorldGuide* reduces inference latency by **2.2**$\times$ **to 19.9**$\times$ with zero deployment overhead from the world model.

2 Related Work

Robot Manipulation Policies. Data scale and model capacity have reshaped manipulation policies. Visuomotor imitation evolved into generalist formulations as RT recast control as tokenized sequence modeling, a paradigm OpenVLA consolidated over cross-embodiment datasets. Subsequent works introduced generative action heads via flow matching (π_0, $\pi_{0.5}$) for continuous high-frequency control, pursued efficient inference via linear sequence models (RoboMamba Liu et al. (2024)), or scaled foundation policies to humanoids (GR00T Bjorck et al. (2025)). Parallelly, world-action models (e.g., UniVLA Bu et al. (2025), Video Policy Hu et al. (2024), UWM Zhu et al. (2025), FLARE Zheng et al. (2025), Cosmos Kim et al. (2026)) couple action generation with explicit future state prediction for long-horizon planning. Despite their generality, these systems rely almost exclusively on successful demonstrations, optimizing to succeed more often rather than recognizing failure boundaries.

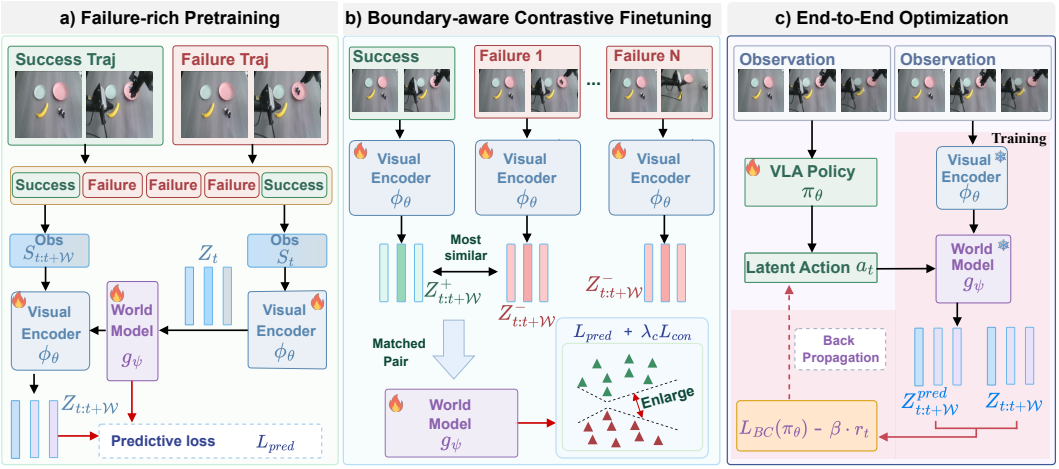


Figure 2: **Overview of WorldGuide. (a) Stage 1:** Failure-Rich Pretraining learns latent dynamics from successful and failed trajectories. **(b) Stage 2:** Boundary-Aware Contrastive Fine-Tuning distinguishes visually matched success–failure pairs while retaining the predictive objective. **(c) Stage 3:** End-to-End Optimization uses a differentiable reward from the frozen world model to train the VLA policy. Flame and snowflake icons indicate trainable and frozen modules, respectively.

World Models for Vision-Language-Action Learning. World models are integrated into VLA learning via three main paradigms. First, as *simulators* Liu et al. (2026); Xiao et al. (2025); Li et al. (2025); Zhu et al. (2026): generative video models synthesize rollouts for policy post-training, though hallucinations can yield physically inconsistent futures and unreliable rewards. Second, via *joint learning* Cen et al. (2025); Bu et al. (2025): state prediction is coupled with action prediction at the feature level to internalize dynamics, but requiring inference-time rollouts inflates latency. Third, via *JEPA-style objectives* Sun et al. (2026): latent prediction error acts as a training-only reward, avoiding inference overhead. However, across all paradigms, models are trained strictly on successful data without capturing failure boundaries. WorldGuide departs from this by pretraining on failure-rich transitions, employing boundary-aware contrastive learning on success–failure pairs, and back-propagating boundary rewards directly into action and visual modules.

3 WorldGuide

As illustrated in Fig. 2, WorldGuide comprises three sequential stages: (1) Stage 1 pretrains the visual encoder and latent world model on a failure-rich corpus to capture complete physical dynamics; (2) Stage 2 applies a boundary-aware contrastive objective on matched success–failure pairs to emphasize outcome-decisive execution differences; and (3) Stage 3 freezes the world model to optimize the VLA policy end-to-end via gradient backpropagation, completely discarding the world model at deployment for zero inference-time overhead.

3.1 Preliminaries

Vision-language-action policy. A VLA policy π_θ maps a visual observation s_t and a task instruction $lang$ to an action chunk $\hat{a}_{t:t+\mathcal{W}} = \pi_\theta(s_t, lang)$, where $\mathcal{W}$ denotes the observation and action horizons. In standard imitation learning, policies are trained on expert demonstrations $\mathcal{D}^+ = \{\xi_i\}_{i=1}^N$, where each trajectory $\xi = \{(s_t, a_t)\}_{t=1}^{|\xi|}$ is annotated with instruction $lang$ and succeeds in completing the task (pred action: $\hat{a}_{t:t+\mathcal{W}}$, gt action: $a_{t:t+\mathcal{W}}$). To incorporate predictive dynamics, the visual encoder ϕ_θ can be shared with an auxiliary world model, encouraging the policy's latent space to capture action-conditioned state transitions.

Latent World Model and Success Bias. Following the JEPA paradigm Sun et al. (2026), the latent world model g_ψ predicts future semantic features rather than high-dimensional pixels:

$$z_{t:t+\mathcal{W}} = g_\psi\big(\phi_\theta(s_t),\, a_{t:t+\mathcal{W}}\big), \tag{1}$$

where $\mathcal{W}$ denotes the prediction horizon. The predictor is trained via an L_1 prediction objective:

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_{\xi\sim\mathcal{D}}\left[\left\|g_\psi\left(\phi_\theta(s_t), a_{t:t+\mathcal{W}}\right) - \phi_\theta(s_{t:t+\mathcal{W}})\right\|_1\right] + L_{reg}, \tag{2}$$

where $\mathcal{L}_{\text{reg}}$ denotes the regularization loss employing the SIGReg regularizer (Details in Appendix A.2). Ideally, g_ψ measures alignment with the goal state via the latent distance between predicted future states and a target goal feature $z^\star_{t:t+\mathcal{W}} = \phi_\theta(s_{t:t+\mathcal{W}})$. However, when trained purely on expert demonstrations $\mathcal{D}^+$ (failure $\mathcal{D}^-$), the predictor suffers from **success bias**: its learned transition distribution collapses onto the success manifold. Consequently, failure states are erroneously mapped near successful latents, yielding optimistically low distance penalties even after execution errors: a failure mode that motivates our three stages framework. **Note: t denotes the global timestep of the action chunk within the full trajectory, while τ represents the local frame index within the future action chunk**.

3.2 Stage 1: Failure-Rich Pretraining

To capture a complete transition distribution, the world model must observe non-expert executions. We curate a failure dataset $\mathcal{D}^-$ from two complementary sources. First, *simulator perturbations* create failures through noisy control, randomized object placements, and scripted errors. Preserving these failing episodes supplies *breadth*, teaching the model coarse, low-level mistakes, like knocked-over objects or missed grasps. Second, *intermediate checkpoints* saved during VLA policy training supply *depth*. Their failures arise at critical decision points such as missed grasps or premature releases, producing subtle visual discrepancies that directly dictate task outcomes and densely cover the success–failure boundary. We joint pretrain g_ψ and ϕ_θ on the enriched corpus $\mathcal{D}^+\cup\mathcal{D}^-$ using the predictive loss in Equation 2. Modeling this failure-rich dynamics forces g_ψ to capture erroneous state transitions than snapping predictions back to the success manifold. As a result, the Stage 1 world model reliably penalizes coarse execution errors, setting the stage for Stage 2 to sharpen its resolution on near-boundary failures. More details can be found in Appendix.

3.3 Stage 2: Boundary-Aware Contrastive Fine-Tuning

Progress-anchored pairing. As shown in Fig. 3, Stage 2 builds each success failure pair by restricting candidates using a temporal proximity criterion before selecting a counterpart in feature space. Since failure rollouts share initial execution with successful demonstrations, step index s serves as a unified task phase. The annotated failure onset t_f (Appendix C.2) anchors candidates within tolerance $\Delta = \mathcal{W}/2$ ($|t - t_f| \le \Delta$). Among valid candidates, the failure counterpart ξ_i^- is chosen by feature distance over action chunk window $\mathcal{W}_\sqcup()$:

$$\xi_i^- = \underset{\substack{\xi^-\in\mathcal{D}^-_{l_i}\\ \left|t-t_f\right|\le\Delta}}{\arg\min} \frac{1}{|\mathcal{W}|}\sum_{\tau\in\mathcal{W}}\left\|\phi_\theta\big(s_{t+\tau}(\xi_i^+)\big) - \phi_\theta\big(s_{t+\tau}(\xi^-)\big)\right\|_2^2, \tag{3}$$

where $\mathcal{D}^-_{l_i}$ is the failure set of task l_i, ξ_i^+ is the successful trajectory, and $\mathcal{W}_t(\xi)$ denotes the observation sequence window of length $\mathcal{W}$ centered at time t. Let $\mathcal{H}$ denotes the *hit set* comprising all valid success–failure pairs filtered via Equation 3. Segments without valid candidates ($i \notin \mathcal{H}$) are trained solely under $\mathcal{L}_{\text{pred}}$. The selected counterpart is separated using loss weighted distance d_m to satisfy Equation 4. Finally, failure rollouts are truncated at post failure retry loops (Appendix C.3), preventing repeated grasp attempts from dominating training.

Loss-weighted contrastive objective. Having matched pairs in the observation space, Stage 2 separates them within the latent space to focus gradients on task-decisive transitions. For each matched pair $i \in \mathcal{H}$, both trajectories are propagated through the encoder ϕ_θ and the predictor g_ψ over the window $\mathcal{W}$, yielding predicted future spatial-temporal latent tensors $z^+_{t+\mathcal{W}}, z^-_{t+\mathcal{W}} \in \mathbb{R}^{|\mathcal{W}|\times H\times W\times d}$. To evaluate their divergence without letting background noise dominate, each spatial-temporal tensor is summarized by pooling across both temporal and spatial dimensions into a channel descriptor $u = \frac{1}{|\mathcal{W}|HW}\sum_{\tau,h,w} z_{t+\tau,h,w} \in \mathbb{R}^d$. We then compute the loss-weighted distance $d_m\big(z^+_{t+\mathcal{W}}, z^-_{t+\mathcal{W}}\big)$:

$$d_m\big(z^+_{t+\mathcal{W}}, z^-_{t+\mathcal{W}}\big) = \sum_{\tau\in\mathcal{W}}\alpha_\tau\cdot\left\|g\odot\big(z^+_{t+\tau} - z^-_{t+\tau}\big)\right\|_F^2, \tag{4}$$

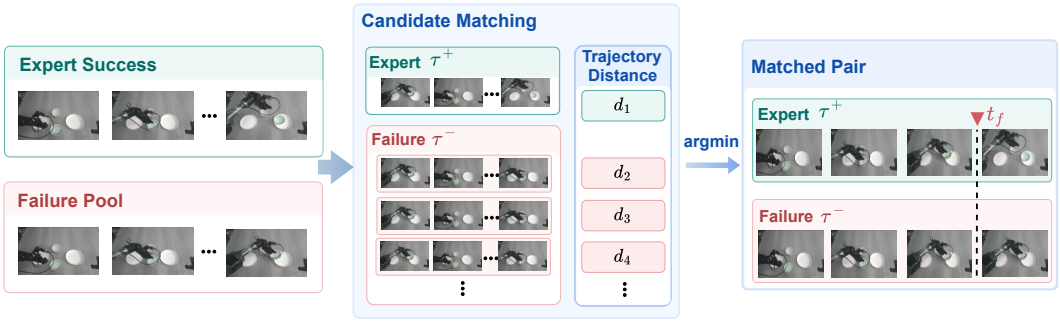


Figure 3: **Success–failure matching for contrastive learning.** Each successful segment is paired with the visually closest failure candidate from the same task within a temporal window to provide contrastive supervision; a dashed line marks the failure onset t_f.

$$\text{where} \quad \alpha = \text{softmax}\left(\frac{1}{LH_aQ}\sum_{l=1}^{L}\sum_{k=1}^{H_a}\sum_{q=1}^{Q} A_{q,\cdot}^{(l,k)}\right) \in \Delta^{|\mathcal{W}|}, \qquad g = \sigma(W_g u) \in (0,1)^d. \tag{5}$$

Here, $\odot$ denotes channel-wise broadcasting over spatial dimensions (H, W), and $\|\cdot\|_F^2$ sums over spatial positions and channels. Dynamically, the temporal salience profile $\alpha \in \Delta^{|\mathcal{W}|}$ aggregates the predictor's cross-attention maps $A^{(l,k)}$ across L layers, H_a heads, and Q query tokens to highlight execution-critical frames, while the channel mask $g = \sigma(W_g u) \in (0,1)^d$ suppresses static context to emphasize outcome-decisive channels.

The distance d_m enters a one-sided repulsive hinge loss over all matched pairs in the hit set $\mathcal{H}$:

$$\mathcal{L}_{\text{con}} = \frac{1}{|\mathcal{H}|}\sum_{i\in\mathcal{H}}\left[m - d_m\left(z^+_{t:t+\mathcal{W}}, z^-_{t:t+\mathcal{W}}\right)\right]_+, \tag{6}$$

where $m = 0.7$ denotes the margin and $[\cdot]_+ = \max(0, \cdot)$. Stage 2 finetunes the pretrained world model via:

$$\mathcal{L}_{\text{ft}} = \mathcal{L}_{\text{pred}} + \lambda_c \mathcal{L}_{\text{con}}, \tag{7}$$

with λ_c balancing the contrastive loss and is set to 0.11. Consequently, minimizing Equation 7 pushes matched pairs beyond the margin and establishes a sharp, explicit boundary in the latent space that responds acutely to critical failure modes, thereby providing the exact structured latent signals required by Stage 3.

3.4 Stage 3: End-to-End Optimization with Differentiable Reward

We freeze g_ψ as a differentiable reward. For each state s_t from expert data and policy rollouts, with a goal frame $s_{t:t+\mathcal{W}}$ from a successful demonstration of the same task, the reward of an action chunk is

$$r_t = -\,d_m\Big(g_\psi\left(\phi_\theta(s_t),\, \hat{a}_{t:t+\mathcal{W}}\right), \phi_\theta(s_{t:t+\mathcal{W}})\Big). \tag{8}$$

The policy is then optimized end-to-end while freezing the world model (ϕ_θ and g_ψ), where ϕ_θ is distinct from the VLA's vision encoder.

$$\mathcal{L} = \mathcal{L}_{\text{BC}}(\pi_\theta) - \beta\,\mathbb{E}\big[r_t\big], \tag{9}$$

where $\mathcal{L}_{\text{BC}}$ is the imitation objective on expert data and β balances exploration toward reward-optimal behavior, which is set to 0.1. Reward gradients backpropagate through the frozen world model into the action head and VLM: the former favors goal-directed action chunks over failure-prone ones, while the latter aligns representations with safety-critical semantics. At deployment, the world model is discarded, leaving the VLA policy to execute independently.

4 Experiments

We comprehensively evaluate WorldGuide across three widely recognized simulation benchmarks: LIBERO, RoboTwin, and SimplerEnv as well as on a suite of real-world robotic manipulation tasks. Details are provided in Appendix.

Table 1: **Quantitative Evaluation on LIBERO and SimplerEnv Benchmarks.** Results are reported as percentages. Best and second best overall results are highlighted in dark green and light green, respectively.

(a) LIBERO Benchmark

Method	Category	Model Size	Latency (ms)	Speedup	100	Goal	Object	Spatial	Average
OpenVLA-OFT Kim et al. (2024)	VLA	7B	277	–	94.5	97.9	98.4	97.6	97.1
$\pi_{0.5}$ Intelligence et al. (2025)	VLA	3.5B	220	–	92.4	98.0	98.2	**98.8**	96.9
GR00T-N1.6 Bjorck et al. (2025)	VLA	3.3B	259	–	94.4	97.5	98.5	97.7	97.0
UniVLA Bu et al. (2025)	Latent-VLA	7B	–	–	92.0	95.6	96.8	96.5	95.2
Mantis Yang et al. (2026)	Latent-VLA	5.8B	–	–	94.2	94.4	99.2	**98.8**	96.7
VLA-JEPA Sun et al. (2026)	Latent-VLA	3B	–	–	95.8	97.2	99.6	96.2	97.2
Cosmos-Policy Alhaija et al. (2025)	WAM	2.1B	1413	1.0×	97.6	**98.2**	**100.0**	98.1	**98.5**
LingBot-VA Zhang et al. (2026)	WAM	5.5B	4482	1.0×	**98.5**	97.2	99.6	98.5	**98.5**
Fast-WAM Yuan et al. (2026)	WAM	6B	486	1.0×	95.2	97.0	**100.0**	98.2	97.6
WorldGuide (Ours)	**VLA (SOTA)**	**4B** *(+0.5B Train)*	**225**	**2.2×–19.9×**	**96.8**	**98.6**	**99.6**	**98.6**	**98.4**

SimplerEnv Benchmark

Method	Google Robot					WidowX Robot				
	Pick	Move	Drawer	Place	Average	Spoon	Carrot	Block	Eggplant	Average
LAPA* Ye et al. (2025)	–	–	–	–	–	70.8	45.8	54.2	58.3	**57.3**
villa-x Chen et al. (2026)	81.7	55.4	38.4	4.2	44.9	48.3	24.2	19.2	71.7	40.8
UniVLA Bu et al. (2025)	–	–	–	–	–	–	–	–	–	42.7
RoboVLMs Li et al. (2026)	77.3	61.7	43.5	24.1	51.7	45.8	20.8	4.2	79.2	37.5
GR00T N1 Bjorck et al. (2025)	0.7	1.9	2.9	0.0	1.4	1.4	0.0	0.0	13.9	3.8
MoTo Chen et al. (2025b)	74.0	60.4	43.1	–	–	–	–	–	–	–
OpenVLA-OFT Kim et al. (2024)	–	–	–	–	–	34.2	30.0	30.0	72.5	41.8
π_0 Black et al. (2024)	72.7	65.3	38.3	–	–	29.1	0	16.6	62.5	27.1
π_0-Fast Pertsch et al. (2025)	75.3	67.5	42.9	–	–	29.1	21.9	10.8	66.7	48.3
VLA-JEPA Sun et al. (2026)	88.3	64.1	59.3	49.1	65.2	75.0	70.8	12.5	70.8	57.3
WorldGuide (Ours)	**91.3**	**75.0**	**57.6**	**64.1**	72.0	**84.0**	**68.0**	**18.8**	**88.5**	**64.8**

4.1 Simulation Setup and Baselines

Experimental Setup. We evaluate on LIBERO Liu et al. (2023) (100 (Long), Spatial, Goal and Object, Franka Panda), RoboTwin Chen et al. (2025a) (50 tasks, 14-DoF Aloha-AgileX), and SimplerEnv Li et al. (2024) (WidowX and Google Robot). Stage 1 pretraining uses 2K/2K, 6.6K/5.61K, and 140K/17K successful/failed trajectories for 60K, 120K, and 100K steps, respectively, with RoboTwin using 12/50 tasks. Stage 2 runs for 10K/20K/17K steps, followed by expert-data fine-tuning for 50K/110K/100K steps (stage 3). Real-robot experiments use 300 expert and 60 failure trajectories, with 30K/10K Stage 1/2 steps and 40K fine-tuning steps. All experiments use 8 NVIDIA H20 GPUs. OpenVLA-OFT and GR00T Bjorck et al. (2025) serve as the primary baselines for LIBERO/RoboTwin and SimplerEnv, respectively. Further details are in the Appendix.

4.2 Simulation Evaluation

LIBERO. As shown in Tab. 1, WorldGuide sets a new SOTA among standard zero-overhead VLAs with a **98.4%** overall average on LIBERO, including **96.8%** on LIBERO-100, outperforming OpenVLA-OFT (97.1%) and GR00T-N1.6 (97.0%) by over 1.3%. Crucially, WorldGuide matches heavy Generative WAMs (e.g., LingBot-VA at 98.5%) while reducing per-step latency to 225 ms ($2.2\times$ –$19.9\times$ speedup). By discarding the world model during deployment, our framework eliminates the prohibitive memory and sampling overhead inherent to generative rollouts. Compared to expert-only latent models like VLA-JEPA (97.2%), WorldGuide's margin expands to $+2.4\%$ on LIBERO-Spatial (98.6%) and $+1.0\%$ on LIBERO-100 (96.8%), proving the necessity of boundary-aware contrastive supervision over failure rollouts.

SimplerEnv. Relative to its closest method VLA-JEPA (Tab. 1), which shares a similar predictive paradigm but trains purely on expert demonstrations, WorldGuide exhibits clear superiority, outperforming VLA-JEPA by **+6.8%** on Google Robot and **+7.5%** on WidowX Robot. This gap highlights a key limitation of expert-only training: policies trained solely on successful executions cannot recognize the early signs of failure when perturbed by domain shifts. In contrast, our contrastive supervision explicitly penalizes trajectories near failure onsets (t_f), teaching the world model to recognize dangerous states.

RoboTwin. As shown in Tab. 2, WorldGuide achieves the best average on RoboTwin 2.0 in both the clean setting at 85.7% and the domain-randomized setting at 86.1%, outperforming GR00T-N1.5 by 4.1 and 2.2 points and ranking among the top two on 8 of 12 tasks. The largest gain appears on

Table 2: Comparison on RoboTwin 2.0 across 12 tasks. Each entry reports the success rate under the clean and the domain-randomized setups as x / y.

Method	Task					
	Press Stapler	**Move Playingcard Away**	**Place Object Stand**	**Place Container Plate**	**Turn Switch**	**Lift Pot**
$\pi_{0.5}$	0.97 / 0.95	0.94 / 0.98	0.87 / 0.87	0.95 / 0.93	0.62 / 0.69	0.99 / 0.99
π_0-Fast	0.96 / 0.97	0.95 / 0.98	0.86 / 0.92	0.92 / 0.98	0.63 / 0.67	1.00 / 0.99
GR00T-N1.5	0.98 / 0.98	0.99 / 0.99	0.97 / 0.96	0.85 / 0.89	0.58 / 0.64	0.99 / 1.00
StarVLA-OFT	0.97 / 0.96	1.00 / 0.98	0.84 / 0.85	0.90 / 0.97	0.67 / 0.60	1.00 / 1.00
WorldGuide	1.00 / 0.97	0.99 / 1.00	0.94 / 0.93	0.99 / 0.97	0.66 / 0.69	0.99 / 1.00

Method	**Adjust Bottle**	**Place Phone Stand**	**Place Mouse Pad**	**Pick Diverse Bottles**	**Rotate Qrcode**	**Move Stapler Pad**
$\pi_{0.5}$	0.95 / 0.91	0.51 / 0.66	0.66 / 0.57	0.55 / 0.51	0.59 / 0.59	0.39 / 0.41
π_0-Fast	0.90 / 0.95	0.49 / 0.69	0.53 / 0.48	0.65 / 0.63	0.50 / 0.61	0.33 / 0.39
GR00T-N1.5	1.00 / 0.96	0.92 / 0.98	0.75 / 0.68	0.53 / 0.62	0.87 / 0.87	0.37 / 0.51
StarVLA-OFT	1.00 / 0.99	0.90 / 0.95	0.55 / 0.49	0.52 / 0.62	0.83 / 0.81	0.44 / 0.41
WorldGuide	1.00 / 1.00	0.92 / 0.98	0.77 / 0.73	0.69 / 0.66	0.89 / 0.85	0.45 / 0.56

Move-Stapler-Pad, where WorldGuide reaches 0.45/0.56 versus 0.44/0.41 for the baseline, a 15-point improvement under randomization. Turn-Switch is the exception, where fine-grained contact dominates and $\pi_{0.5}$ performs best. The learned boundary thus transfers to 14-DoF bimanual control and remains robust under strong domain randomization.

4.3 Real-World Evaluation

We conduct real-world experiments on the ARX LIFT2 platform, collecting 300 human demonstrations covering three pick-and-place tasks of deliberately tight tolerances, so that small pose errors cascade into failure. π_0 and $\pi_{0.5}$ are fine-tuned on the same demonstrations and evaluated under identical settings, 20 trials per task per method. Further configurations are in the Appendix.

Table 3: Real-world results on the ARX LIFT2. Success rate (%) over **20 trials** per task per method

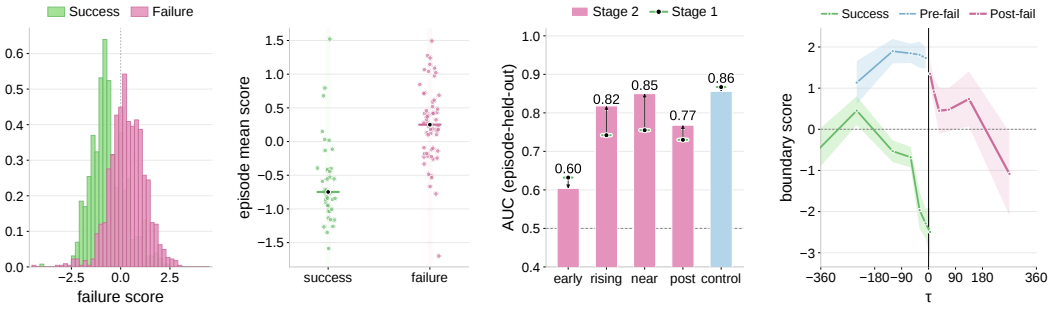
Method \ Task	Purple square	Green circle	Yellow triangle	Avg.
π_0	10%	**20%**	5%	11.7%
$\pi_{0.5}$	10%	15%	10%	11.7%
WorldGuide	**20%**	**20%**	**15%**	**18.3%**

As shown in Tab. 3, all three policies operate in the same low-success regime, with π_0 and $\pi_{0.5}$ both averaging 11.7%, confirming that the difficulty is intrinsic to the tasks rather than any training budget. WorldGuide outperforms $\pi_{0.5}$ across all three tasks and beats or ties π_0, raising the pooled success rate to 18.3% (+57% relative gain). Gains align directly with geometric sensitivity: vanishing on rotationally symmetric circles, doubling on squares, and tripling on triangles, which are hardest to grasp reliably. This distribution confirms that improvements concentrate at pose critical steps near the failure boundary rather than reflecting generic capacity gains (Fig. 6). Given 20 trials per task, the core signal lies in this consistent directional alignment rather than individual margins. Finally, all policies struggle with failure recovery, frequently repeating unsuccessful motions after initial errors.

4.4 Further Analysis and Ablation Study

Q1: Are Failure and Success Truly Separable in Feature Space? To evaluate feature separability, an episode-held-out linear readout (PCA + Logistic Regression) is fitted on predicted future embeddings across LIBERO (Fig. 4 & Tab. 4).

The readout utilizes linear probes (logistic regression on world-model embeddings) trained and evaluated via episode-held-out cross-validation to ensure zero data leakage across episodes. The primary probe emits a signed *failure score* (positive indicates failure). Single-window scores yield distinct distributions (Fig. 4a), while their per-episode average (*episode mean score*) confirms trajectory-level separation beyond local window noise (Fig. 4b). Quantitatively, Stage 2 elevates episode-level AUC from 0.81 to 0.87 and critical-window AUC from 0.74 to 0.79. To characterize when the failure signature emerges relative to the annotated error, we train a second, *phase-matched* probe to derive a *boundary score*. Intuitively, this score measures whether a window resembles the moments immediately preceding an error (positive) or the winding-down phase of a success (negative). Formally, the probe is trained on late-stage failure windows ($\tau \in [-150, 0)$) and late success win-



(a) Measured: score separation (b) Measured: episode separation (c) Stage 1 → Stage 2: AUC (d) Measured: boundary score on τ

Figure 4: Failure success separability of the world model predictor.

dows ($p \geq 0.70$), thereby decoupling elapsed time from classification. Projecting all windows, including the unseen post-fail segment ($\tau \geq 0$), onto this boundary score reveals that successful trajectories remain negative and deepen toward completion, whereas failing ones surge positive as $\tau \to 0$ (Fig. 4d). Before the error, Stage 2 separates impending failure from late success at 0.87 AUC (vs. 0.78 in Stage 1); beyond the error, this signature persists on held-out post-fail windows (0.74 AUC, with 81% of episodes above the success median). Together, these results confirm that Stage 2 encodes a genuinely predictive failure signature rather than memorizing training artifacts.

Q2: How much does the boundary-aware contrastive objective matter? Stage 2 improves all decision-critical readouts, window AUC from 0.74 to 0.79 and episode AUC from 0.81 to 0.87, yet the gains are uneven across phases: near-failure windows improve the most from 0.75 to 0.85, mid-phase windows follow from 0.74 to 0.82, and early-phase windows slightly drop from 0.63 to 0.60. The boundary-direction readout benefits most and generalizes furthest: fitted only on $\tau \in [-150, 0)$, it rises from $0.78/0.74$ to $0.87/0.85$ on near-failure and never-trained post-failure windows, confirming that the contrastive objective concentrates representational capacity on the failure boundary rather than on generic progress or scene cues. Results from success only training further confirm the importance of failure-mixed training.

Table 4: **Feature separability under Stage 2 boundary-aware fine-tuning.** Episode-held-out linear probing AUCs (Fig. 4) before (Stage 1) and after (Stage 2) fine-tuning. Middle rows detail AUC across task progress p and failure offset τ. Bottom rows evaluate early-warning generalization: training the probe strictly on near-failure frames ($\tau \in [-150, 0)$ frames) and testing transferability to unseen future frames.

Readout / Evaluation Setting	**Stage 1**	**Stage 2**
Window AUC (decision-critical)	0.74	**0.79**
Episode AUC	0.81	**0.87**
Phase: Early ($p < 0.35$)	0.63	0.60
Phase: Mid ($0.35 \leq p < 0.70$)	0.74	**0.82**
Phase: Near-failure ($p \geq 0.70$)	0.75	**0.85**
Phase: Post-failure ($\tau \geq 0$)	0.73	**0.77**
Boundary direction: Near-failure	0.78	**0.87**
Boundary direction: Unseen post-failure	0.74	**0.85**

Q3: How much does each training stage contribute to policy performance? As shown in Tab. 5, incorporating failure rollouts during Stage 1 pretraining outpaces training on success only data across all metrics, confirming that observing erroneous transitions builds a broader dynamics prior. Stage 2 contrastive finetuning yields consistent additional gains, particularly on challenging domains like RoboTwin and SimplerEnv, as the induced margin sharpens the success failure boundary in latent space to guide better policy decisions. Gains on LIBERO are more modest due to its near saturated baseline.

Table 5: **Stage-wise ablation.**

Setting	Stage		RoboTwin		LIBERO	SimplerEnv	
	1	2	Clean	Randomized		Google	WidowX
Base VLA	✗	✗	80.1%	80.2%	96.5%	68.8%	60.2%
Stage 1 (success only)	✓	✗	82.4%	81.9%	97.5%	69.6%	63.0%
Stage 1	✓	✗	83.0%	83.5%	97.8%	70.7%	63.4%
Stage 1 + 2	✓	✓	85.7%	86.1%	98.4%	72.0%	64.8%

Q4: What makes a world-model reward informative? Fig. 5 replaces only the Stage-3 reward under identical policy training. Goal similarity and prediction MSE from a success-only world model yield limited gains, reaching only $81.0/81.2\%$ and $82.5/82.8\%$ on RoboTwin, and their AUCs stay near chance at 0.47–0.50 even in our failure-rich latent space, so the limitation lies in the reward readout itself. The boundary reward $-d_m()$ instead achieves an AUC of 0.85 and improves

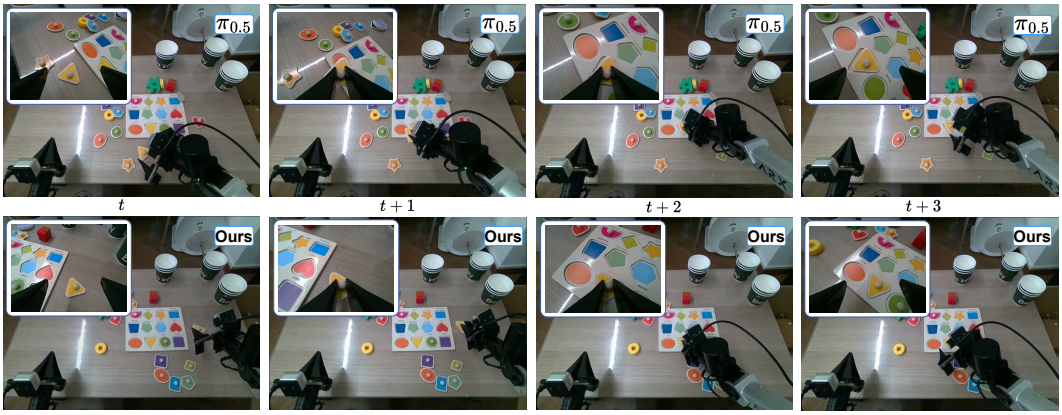


Figure 6: Fine-grained manipulation comparison between $\pi_{0.5}$ and WorldGuide in real world.

RoboTwin to $85.7/86.1\%$ and LIBERO to 98.4%. Stage 3 is thus effective only when the reward can distinguish failure from success, which in turn depends on the failure-aware representation learned in Stages 1 and 2.

Q5: How does world model reshape the representation? Fig. 7 shows the attention weights between action tokens and image patches, overlaid on the raw observations as a thresholded thermal map; only the top 3%–10% of attention mass is colored. The hotspot is not a static saliency prior but consistently lands on the manipulated object across scenes. This demonstrates that the attention mechanism dynamically grounds decision-making on task-relevant contact regions rather than background context.

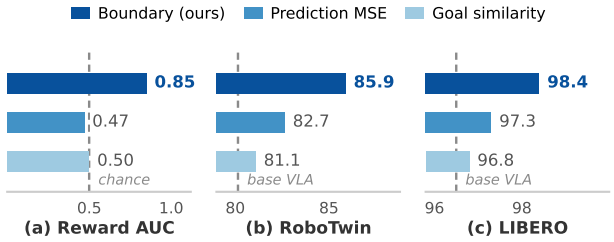


Figure 5: **Reward-source ablation**.

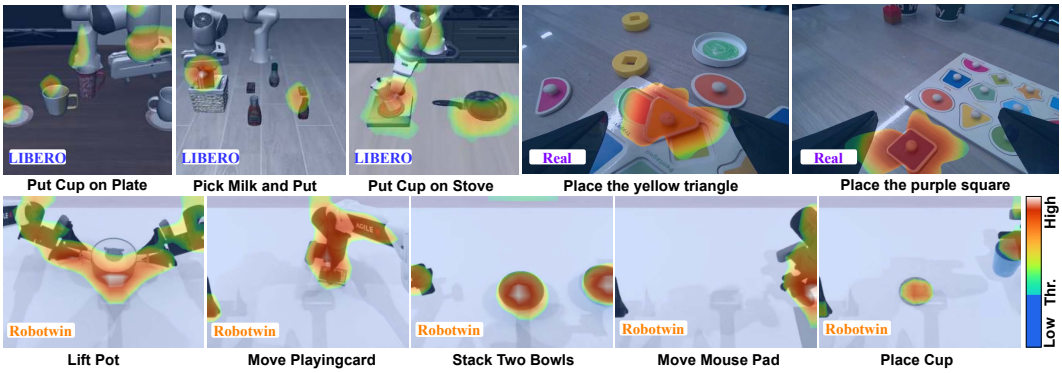


Figure 7: Attention weight matrix of latent action tokens to image tokens.

5 Conclusion

We introduced WorldGuide, a latent world model that enhances VLA robustness by explicitly modeling the boundary between successful and failed behaviors. Learning jointly from success and failure trajectories with boundary-aware contrastive learning yields failure-sensitive representations, and the world model is discarded at inference. Experiments across simulation and real-world tasks show improved robustness and task success with no additional inference overhead, highlighting the promise of failure-aware latent world modeling for embodied intelligence.

AI USE STATEMENT

In this work, generative AI tools were used to assist with method code implementation, language polishing, figure layout and color adjustments, as well as automatic temporal timestamp annotation for t_f. Specifically, the open source Qwen 3.5 model was applied to suggest timestamp annotations for transform frames, and all model generated annotations were manually sampled and cross verified by the authors to guarantee data quality. Generative AI was not used to develop theoretical models or conceptual frameworks, or to formulate or refine hypotheses. It was not used to provide essential elements for mathematical proofs, assist in writing proofs, or support qualitative or thematic data analysis.

All AI assisted outputs were thoroughly reviewed and verified by the authors. Any code or annotations generated with the assistance of large language models were independently tested and verified for correctness. The authors take full responsibility for the final content of this work, including all text, claims, data annotations, and materials produced with the assistance of generative AI. Please refer to the Appendix for further details.

REPRODUCIBILITY STATEMENT

Section 3 specifies the WorldGuide architectures, training objectives, and inference procedures. Section 4 details the training datasets, benchmark splits, evaluation metrics, and comparison baseline settings. The Appendix A provides full model configurations, optimization hyperparameters, evaluator definitions, and metric rules. Upon acceptance, we will publicly release all source code, evaluation protocols, and model checkpoints required to reproduce all reported results.

REFERENCES

Hassan Abu Alhaija, Jose Alvarez, Maciej Bala, Tiffany Cai, Tianshi Cao, Liz Cha, Joshua Chen, Mike Chen, Francesco Ferroni, Sanja Fidler, et al. Cosmos-transfer1: Conditional world generation with adaptive multimodal control. *arXiv preprint arXiv:2503.14492*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

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

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. Villa-x: enhancing latent action modeling in vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 70673–70703, 2026.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19752–19763. IEEE, 2025b.

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

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

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

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, et al. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8(2):158–172, 2026.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

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

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-vla-loop: Closed-loop learning of video world model and vla policy. *arXiv preprint arXiv:2602.06508*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-env: Leveraging world model as a virtual environment for vla post-training. *arXiv preprint arXiv:2509.24948*, 2025.

Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42505–42515, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025.

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

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

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

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 62486–62502, 2026.

A Implementation Details

A.1 Architecture and Base Models

WorldGuide is implemented on the StarVLA training framework. The policy backbone is Qwen3-VL-4B-Instruct; per benchmark we attach the standard action head of the corresponding base VLA (discrete OFT action tokens for LIBERO and RoboTwin, a DiT-B flow-matching head for SimplerEnv), so that "Base VLA" in all ablations refers to the same publicly comparable recipe fine-tuned on the same demonstrations. The world model g_ψ follows the JEPA design: a ViT visual encoder ϕ_θ shared with the policy and a lightweight transformer predictor that maps a window of past latents and actions to the predicted future latent. Tab. 6 summarizes the per-benchmark configurations.

Table 6: Architecture and per-benchmark configurations. The world model is discarded at deployment; only the VLA policy executes.

	LIBERO	RoboTwin 2.0	SimplerEnv
VLM backbone	Qwen3-VL-4B	Qwen3-VL-4B	Qwen3-VL-4B
Action head	OFT tokens	OFT tokens	DiT-B flow matching
Action space	Δqpos, 7-DoF	abs qpos, 14-DoF	ΔEE, 7-DoF
Action chunk H	8	50	16
WM encoder ϕ_θ	ViT-L/14, 224^2	ViT-L/14, 224^2	ViT-T/14, 224^2
WM window (ctx + pred)	$8+1$, frameskip 1	$50+1$, frameskip 1	$16+1$, frameskip 1
WM predictor	6-layer, 16-head	6-layer, 16-head	4-layer, 8-head
Latent dim	768	768	768
WM parameters	0.5B	0.5B	0.5B
Deployment params	4B (no WM)	4B (no WM)	4B (no WM)

The world model predicts next-frame latents in parallel over sliding contexts via causal teacher-forcing, eliminating autoregressive rollouts during training. For clarity, we simplify the frame input at step t to s_t. Within a temporal window, the output at position t predicts the latent of frame s_{t+1} conditioned on all preceding frames $\leq s_t$, allowing a single forward pass to yield all next-frame predictions simultaneously—each supervised under its respective historical context. Regarding window configurations, the world model observes an $(8{+}1)$-frame sequence for LIBERO and a $(50{+}1)$-frame sequence for RoboTwin, matching the policy's observation range. For SimplerEnv, the data registry provides a $(16{+}17$-frame window aligned with the 16-step action chunk, where the first frame conditions the VLM and the full 17-frame sequence feeds into the world model.

Table 7: Stage-wise training configurations across different environments.

	Stage 1 (pretrain)	Stage 2 (contrastive)	Stage 3 (reward)
LIBERO			
Steps	60,000	10,000	50,000
Batch / device	16	16	16
Learning rate	5×10^{-5} (WM)	5×10^{-5} (WM)	1×10^{-5} (VLA)
LR schedule	cosine $\rightarrow 1 \times 10^{-6}$	cosine $\rightarrow 1 \times 10^{-6}$	cosine $\rightarrow 2 \times 10^{-6}$
Warmup ratio	0.1	0.1	0.1
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-8}
Prediction loss $\mathcal{L}_{\text{pred}}$	1.0	1.0	–
SIGReg regularizer	0.09	0.09	–
Contrastive loss $\mathcal{L}_{\text{con}}$	–	0.11	–
Reward weight β	–	–	0.1
Initialization	Rand	Stage 1 WM	Stage 2 WM
RoboTwin			
Steps	120,000	20,000	110,000
Batch / device	16	16	16
Learning rate	6.5×10^{-5} (WM)	6.5×10^{-5} (WM)	2×10^{-5} (VLA)
LR schedule	cosine $\rightarrow 2 \times 10^{-6}$	cosine $\rightarrow 2 \times 10^{-6}$	cosine $\rightarrow 2.5 \times 10^{-6}$
Warmup ratio	0.1	0.1	0.1
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-8}
Prediction loss $\mathcal{L}_{\text{pred}}$	1.0	1.0	–
SIGReg regularizer	0.09	0.09	–
Contrastive loss $\mathcal{L}_{\text{con}}$	–	0.11	–
Reward weight β	–	–	0.1
Initialization	Rand	Stage 1 WM	Stage 2 WM
SimplerEnv			
Steps	100,000	17,000	100,000
Batch / device	16	16	16
Learning rate	5×10^{-5} (WM)	5×10^{-5} (WM)	1×10^{-5} (VLA)
LR schedule	cosine $\rightarrow 1 \times 10^{-6}$	cosine $\rightarrow 1 \times 10^{-6}$	cosine $\rightarrow 1.5 \times 10^{-6}$
Warmup ratio	0.1	0.1	0.1
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-8}
Prediction loss $\mathcal{L}_{\text{pred}}$	1.0	1.0	–
SIGReg regularizer	0.09	0.09	–
Contrastive loss $\mathcal{L}_{\text{con}}$	–	0.11	–
Reward weight β	–	–	0.1
Initialization	Rand	Stage 1 WM	Stage 2 WM
Real Env			
Steps	30,000	10,000	40,000
Batch / device	24	24	24
Learning rate	2.5×10^{-5} (WM)	2.5×10^{-5} (WM)	2×10^{-5} (VLA)
LR schedule	cosine $\rightarrow 2.5 \times 10^{-6}$	cosine $\rightarrow 2.5 \times 10^{-6}$	cosine $\rightarrow 2.5 \times 10^{-6}$
Warmup ratio	0.1	0.1	0.1
Weight decay	1×10^{-2}	1×10^{-2}	1×10^{-2}
Prediction loss $\mathcal{L}_{\text{pred}}$	1.0	1.0	–
SIGReg regularizer	0.09	0.09	–
Contrastive loss $\mathcal{L}_{\text{con}}$	–	0.11	–
Reward weight β	–	–	0.1
Initialization	Rand	Stage 1 WM	Stage 2 WM

A.2 Training Configuration

All stages share the same optimizer setup: AdamW with cosine learning-rate decay to a minimum floor, gradient clipping at a threshold of 1.0, mixed-precision training (bf16), and gradient checkpointing distributed across 8× NVIDIA H20 GPUs. Tab. 7 summarizes the stage-wise training configurations for LIBERO, RoboTwin, SimplerEnv, and real-robot evaluations, with detailed in Tab. 6 and Tab. 7.

Stage-2 Hyperparameters. The temporal match tolerance is set to half of action chunk size, with a hinge margin of $m = 0.7$ and a contrastive loss weight of $\lambda_c = 0.11$. Both the temporal salience profile α and the channel gate g (Equation 5) are enabled by default in all main experiments; their corresponding ablation variants (i.e., uniform α and $g \equiv \mathbf{1}$) are evaluated in Tab. 8. Candidate failure windows are indexed via a 64 entry LRU cache and encoded in a single batched forward pass (with gradients disabled) per step, ensuring that nearest-candidate selection incurs negligible computational overhead.

Regularization. To prevent representation collapse during predictive loss minimization in Stages 1 and 2, we incorporate SIGReg, a VICReg-style variance–covariance regularizer acting directly on the predicted latent representations.Specifically, SIGReg encourages the latent distribution to match an isotropic standard Gaussian prior by matching their empirical characteristic functions over random projections. In our setup, we configure SIGReg with 17 evaluation knots and 1024 random projection directions, applying a weighting coefficient of 0.09 across all environments. By penalizing feature cross-correlations and maintaining spatial variance across latent dimensions without requiring negative pairs or asymmetric architectures, this regularizer ensures that the world model produces informative, high entropy representations that effectively ground downstream reward modeling and policy learning.

B Evaluation Protocols and Real-World Setup

B.1 Simulation Benchmarks

LIBERO covers four suites (Object, Goal, Spatial, and 100) average success rates. **RoboTwin 2.0** is evaluated over 12 tasks under both the clean and the domain-randomized settings (scene clutter, lighting, backgrounds, object configurations, and instruction paraphrases) on the 14-DoF Aloha-AgileX bimanual platform. Each cell in Tab. 2 is reported as clean/randomized. **SimplerEnv** evaluates real-to-sim visual generalization (lighting, textures, camera poses) on the Google Robot and WidowX embodiments. All policies are served through a WebSocket policy server (client/server split between the evaluation environment and the model host) and execute with the different action-chunk frequency (8 chunk sizes (LIBERO), 50 chunk sizes (Robotwin), 16 chunk sizes (SimplerEnv)). The latency in Tab. 1 (225 ms per chunk for WorldGuide) is measured on this deployment path, i.e., the world model contributes nothing at inference time.



Figure 8: Real world experimental setup: ARX Lift 2 with PICO.

B.2 Real-World Setup

Real robot experiments use the ARX LIFT2 dual arms platform teleoperated with a PICO headset. We collect 300 human demonstrations for three pick-and-place tasks with different object geometries (purple square, green circle, yellow triangle) and deliberately tight tolerances, so that small pose errors cascade into failure. π_0, $\pi_{0.5}$, and WorldGuide are fine-tuned on the same demonstrations and evaluated interleaved under identical lighting and object layouts, 20 trials per task per method, to control for environment drift. Since the aforementioned tasks can be completed using a single arm, we collect demonstrations using only one arm. During training and deployment, we mask out the data and actions associated with the irrelevant arm. For real-world model training, we use the open-source LeRobot framework.

C FAILURE CORPUS CONSTRUCTION

C.1 TWO COMPLEMENTARY SOURCES

The failure set $\mathcal{D}^-$ merges two sources, retained by an automatic task-completion check.

Simulator perturbations (broad). We execute tasks in the simulator with perturbed initial object poses, injected action-control noise, and scripted skill violations that disregard predefined conditions (e.g., grasping before alignment). The resulting failed action sequences and scene data are added to $\mathcal{D}^-$, covering coarse and visually salient errors that span the broader regions of the failure space.

Under-converged models (deep). We execute intermediate checkpoints saved during standard VLA fine-tuning in the simulator and retain the resulting failure episodes. During real-world execution, these under-converged models often produce failures that are visually almost indistinguishable from successful executions, yet errors emerge at critical stages, such as missed grasps, incorrect object grasps, or premature gripper release.

C.2 VLM KEYFRAME ANNOTATION OF t_f

For each failure event, we render the collected data into a video and use the prompt template below (the English rendering is shown here, while the native-language prompt is included chinese in the released code) to annotate the failure onset time t_f with Qwen3.5-VL-27B over the entire video. Each event is assigned a failure onset timestamp t_f, which is subsequently used for the progress-anchored pairing in Equation 3.

VLM keyframe-annotation prompt (English rendering)

```
You are an expert in robot-manipulation video analysis.
Watch the entire video and analyze the arm's failed operations.
For each manipulated object, determine exactly one timestamp:
fail:  the earliest moment at which the robot's grasp, placement,
pick, rotate and other manipulation fails.
Requirements:
-- Analyze the entire video.
-- Return the time in seconds (s).
-- Precision:  one decimal place.
-- Exactly one fail time per manipulated block.
-- If a timestamp is mildly ambiguous, give the most reasonable
estimate from the video content; do not return null.
-- Do not output any explanation.
-- Output JSON only.
Output format (strict):
{ "fail":  2.0 }
```

The decoding pipeline is deliberately defensive: (i) generated text is truncated at the `</mthink>` tag, and the trailing answer is parsed as JSON; (ii) if JSON parsing fails, the first `...` block is extracted via regular expressions as a fallback; and (iii) events that remain unparsable are skipped rather than inferred. The validated timestamps are converted to frame indices at the corpus frame rate (20 fps). We then perform manual verification by reviewing a random sample of annotated videos.

C.3 FAILURE-ONSET-BASED TRUNCATION

Rollouts that continue after the first failure often enter prolonged post-failure retry loops, introducing redundant data that can interfere with training. We therefore truncate each unlabeled failure episode based on the annotated failure onset timestamp t_f, retaining two additional action chunks after t_f before truncation.

C.4 Matched Pairing for Stage 2

Stage 2 performs online success-failure pairing using Equation 3. At timestamp t, for each loaded successful segment of task l, we first collect *all* annotated failure events of the same task whose failure onset satisfies $|t - t_f| \leq \Delta$. A failure video is considered a candidate if the current timestamp t falls within an action-chunk-sized window around its annotated t_f. We then extract the corresponding failure window according to its timestamp and feed it together with the successful window into the feature encoder to obtain high-dimensional representations. Among the candidates, we select the failure segment with the smallest feature distance to the successful segment. Algorithm 1 summarizes one Stage 2 training step.

Algorithm 1 Stage-2 Boundary-Aware Contrastive Training Step

Require: Success batch $\mathcal{B}^+$; failure dataset $\mathcal{D}^-$; pretrained encoder ϕ_θ and predictor g_ψ; margin m; loss weight λ_c; match tolerance Δ; window size $|\mathcal{W}|$
Ensure: Updated model parameters θ, ψ

$\mathcal{L}_{\text{pred}} \leftarrow$ Compute prediction loss on $\mathcal{B}^+$ and sampled failure batch from $\mathcal{D}^-$ (masking past truncation)
$\mathcal{H} \leftarrow \varnothing$ ▷ Initialize hit set for matched pairs
for all success sample $i \in \mathcal{B}^+$ **do**
  Candidate set $\mathcal{C}_i \leftarrow \left\{\xi^- \in \mathcal{D}^-_{l_i} \;\middle|\; |t - t_f| \leq \Delta\right\}$ ▷ Filter candidates by progress anchor
  if $\mathcal{C}_i \neq \varnothing$ **then**
    $\xi_i^- \leftarrow \arg\min_{\xi^- \in \mathcal{C}_i} \frac{1}{|\mathcal{W}|} \sum_{\tau \in \mathcal{W}} \left\| \phi_\theta(\mathcal{W}_t(\xi_i^+)) - \phi_\theta(\mathcal{W}_t(\xi^-)) \right\|_2^2$ ▷ Feature matching, Equation 3
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(i, \xi_i^-)\}$
  end if
end for
Compute predicted latent sequences $z^+_{t+\mathcal{W}}$ and $z^-_{t+\mathcal{W}}$ for all pairs $(i, \xi_i^-) \in \mathcal{H}$
$\mathcal{L}_{\text{con}} \leftarrow 0$
for all matched pair $(i, \xi_i^-) \in \mathcal{H}$ **do**
  $\alpha \leftarrow \text{softmax}\left(\frac{1}{L H_a Q} \sum_{l=1}^{L} \sum_{h=1}^{H_a} \sum_{q=1}^{Q} A^{(l,h,q)} \right)$ ▷ Extract temporal salience profile, detached via $\text{sg}(\cdot)$
  $u \leftarrow \frac{1}{|\mathcal{W}|} \sum_{\tau \in \mathcal{W}} z^+_{t+\tau}$ ▷ Compute trajectory summary descriptor
  $g \leftarrow \sigma(W_g u)$ ▷ Compute dynamic channel mask, Equation 5
  $d_m \leftarrow \sum_{\tau \in \mathcal{W}} \alpha_\tau \cdot \left\| g \odot (z^+_{t+\tau} - z^-_{t+\tau}) \right\|_F^2$ ▷ Weighted distance, Equation 4
  $\mathcal{L}_{\text{con}} \leftarrow \mathcal{L}_{\text{con}} + \left[m - d_m \right]_+$ ▷ Hinge loss, Equation 4
end for
$\mathcal{L}_{\text{con}} \leftarrow \frac{1}{|\mathcal{H}|} \mathcal{L}_{\text{con}}$
return Total fine-tuning loss $\mathcal{L}_{\text{ft}} = \mathcal{L}_{\text{pred}} + \lambda_c \mathcal{L}_{\text{con}}$

D Q1 Probing Protocol

This section specifies exactly how the separability analysis of Q1/Q2 (Fig. 4, Tab. 4) is constructed, so that every reported number can be reproduced from the raw checkpoints.

Evaluation set. We evaluate the probe on a held-out split that is excluded from all training mixtures. It contains 109 failure episodes from LIBERO-100, LIBERO-Object, LIBERO-Spatial, and LIBERO-Goal, each with a verified failure timestamp t_f, together with 100 task-matched successful episodes. We use the same preprocessing pipeline as the world model, including video decoding, resizing, and normalization. Each probe input is a 9-frame window consisting of 8 context frames and 1 target frame, together with the 8 executed Δqpos actions.

Window sampling. We sample 16 windows from each failure episode: 8 uniformly over task progress and 8 concentrated in the decision-critical region $p \in [0.7, 0.98]$ relative to the annotated failure point. Successful episodes are sampled using the corresponding progress grid. For failure episodes, we additionally use the relative time $\tau = t - t_f$ to identify the temporal position of each window, which can be mapped to the corresponding task progress p.

Latent readout. Each window is passed through the world model to obtain its predicted future latent z via a forward pass (Equation 1). The resulting features are z-normalized within each task, reduced to 64 dimensions with PCA, and classified using logistic regression with episode-level GroupKFold(5). Thus, no window from an episode is used to train the probe that evaluates that episode. The resulting out-of-sample predictions are z-normalized within each task and used as the *failure score*. We report both window-level scores and episode-level mean scores.

Phase-wise analysis. To examine when failure information becomes separable, we compute AUC separately in four temporal phases: early ($p < 0.35$), mid ($0.35 \leq p < 0.70$), near-failure ($p \geq 0.70$), and post-failure ($\tau \geq 0$). As a control, we also distinguish early and late windows within successful episodes only. This control remains essentially unchanged after Stage 2 ($0.87 \rightarrow 0.86$), indicating that the observed improvement is not explained by generic task-progress encoding.

Boundary-score analysis. We further test whether the learned boundary generalizes beyond the failure region used for training. The classifier is trained only on near-failure windows from the final 7.5,s before failure ($\tau \in [-150, 0)$ at 20 fps), together with near-goal windows from successful episodes. It is then applied to all windows of the held-out episodes, including post-failure windows that are never used for training. We report the AUC on near-failure windows, the AUC on unseen post-failure windows, and the fraction of failure episodes whose mean post-failure score exceeds the median score of successful episodes. The complete evaluation procedure is summarized in Algorithm 2.

Algorithm 2 Episode-Held-Out Probing Protocol

Require: windows $\mathcal{W}_{\text{probe}} = \{(\hat{z}_i, y_i, \Delta t_i, p_i, e_i, l_i)\}_{i=1}^{N}$, where $\hat{z}_i$ is the predicted future latent of window i, $y_i \in \{0, 1\}$ is its outcome label (1: success, 0: failure), $\Delta t_i = t_i - t_f$ is its temporal offset from failure onset, p_i is task progress, e_i is its episode ID, l_i is its task ID; number of folds $K = 5$

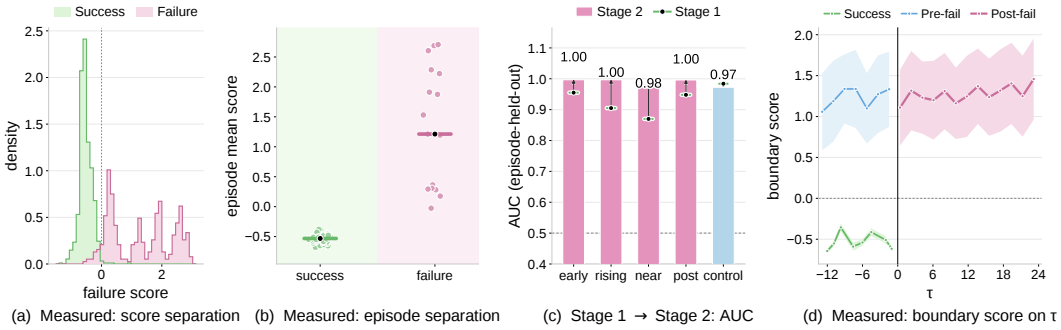
Require: K episode-disjoint folds $\{(\mathcal{I}_{\text{tr}}^{(k)}, \mathcal{I}_{\text{te}}^{(k)})\}_{k=1}^{K}$ obtained by GroupKFold grouped by episode ID e

z-normalize $\hat{z}_i$ independently within each task l_i
for $k = 1, \ldots, K$ **do**
    fit PCA with 64 components on $\{\hat{z}_i : i \in \mathcal{I}_{\text{tr}}^{(k)}\}$
    project training and test latents using the fitted PCA
    fit a logistic regression readout on the projected training windows
    $s_i^{(k)} \leftarrow$ logistic-regression decision value for all $i \in \mathcal{I}_{\text{te}}^{(k)}$
end for
z-normalize s_i independently within each task l_i
return task-normalized s_i as the *failure score*
$\bar{s}_e \leftarrow \frac{1}{|\mathcal{W}_e|} \sum_{i \in \mathcal{W}_e} s_i$ for each episode e, where $\mathcal{W}_e = \{i : e_i = e\}$ ▷ episode-level score
compute window-level ROC-AUC from (s_i, y_i)
compute episode-level ROC-AUC from $(\bar{s}_e, \bar{y}_e)$ using the corresponding episode outcomes
compute phase-wise ROC-AUC within progress-matched bands
compute the success-only early-vs-late control
Boundary probing: restrict probe-training windows to $\{\Delta t_i \in [-150, 0)\} \cup \{y_i = 1,\ p_i \geq 0.7\}$
repeat the same episode-held-out PCA and logistic fitting procedure using only the restricted training windows
apply each fitted readout to *all* windows in its held-out episodes, including windows after failure onset
compute near-failure and unseen post-failure ROC-AUC
return boundary scores and corresponding evaluation metrics

E Additional Ablations

E.1 Separability of Failure and Success in Single In-Distribution Tasks

The readout utilizes linear probes (logistic regression on world-model embeddings) trained and evaluated via episode-held-out cross-validation to ensure zero data leakage across episodes on RoboTwin: 22 successful rollouts with VLM-annotated subtask timestamps versus 20 timeout failures (40 s each, 14-DoF bimanual control). The primary probe emits a signed *failure score* (positive indicates failure). Single-window scores yield near-disjoint distributions (Fig. 9a), while their



(a) Measured: score separation (b) Measured: episode separation (c) Stage 1 → Stage 2: AUC (d) Measured: boundary score on τ

Figure 9: **Success–failure separability on RoboTwin `stack_blocks_three`.** (a) Single-window failure-score distributions. (b) Episode mean scores. (c) Stage-wise AUC. (d) Boundary score swept along the time offset τ from the expected task end.

per-episode average (*episode mean score*) confirms trajectory-level separation beyond local window noise, ranking all 17 failing trajectories above all 92 successful ones (Fig. 9b). Quantitatively, episode-level AUC reaches 1.000 with window-level AUC at 0.993 under an EER of 3.2%. Separability remains uniform across execution progress: 0.997, 0.997, and 0.982 over early, rising, and near-completion stage windows, and 0.996 on post-completion windows, compared to 0.972 for the success-only progress control.

To characterize when the failure signature emerges relative to the expected task end, we train a second, *phase-matched* probe to derive a *boundary score*. Intuitively, this score measures whether a window resembles the moments immediately preceding an error (positive) or the winding-down phase of a success (negative). Formally, the probe is trained on late-stage failure windows ($\tau \in [-150, 0)$ frames) and late success windows ($p \geq 0.70$), thereby decoupling elapsed time from classification. Projecting all windows, including the unseen post-boundary segment ($\tau \geq 0$), onto this boundary score reveals that successful trajectories remain negative and deepen toward completion, whereas failing ones are already strongly positive well before the boundary and remain flat through the timeout horizon without decaying back toward the success side (Fig. 9d). Before the boundary, the probe separates impending failure from late success at 0.98 AUC; beyond it, the signature persists on held-out post-boundary windows (0.996 AUC, with every failing episode lying above the success median). Together, these results indicate that the failure-rich world model encodes a persistent, outcome-decisive failure signature for this task family, present from early execution rather than emerging only at a discrete error onset.

E.2 Contrastive-Negative and Weighting Variants

Tab. 8 isolates the two core design choices of Stage 2, demonstrating that both are critical. Counterpart selection plays a dominant role: replacing the progress-anchored nearest-candidate rule with random opposite-outcome failures of the same task costs 7.4 points on RoboTwin and 2.9 points on LIBERO. Because random pairs differ in scene layout and execution progress, the contrastive gradient is wasted separating incidental cues rather than outcome-decisive ones. The progress anchor alone recovers most of this gap ($84.7/84.2$), confirming that task-phase alignment is the primary filter, while feature-nearest selection contributes the final point by concentrating supervision on visually subtle discrepancies. The separation metric contributes at a comparable magnitude:

Table 8: **Stage-2 design choices.** Policy success rates (%) on RoboTwin (clean/randomized placements) and LIBERO with one Stage-2 component replaced at a time; metric variants replace d_m in both the contrastive loss and the Stage-3 reward.

Variant	RoboTwin	LIBERO
Counterpart selection (separated with d_m)		
Random opposite-outcome	78.3/77.1	95.5
Progress-anchored, random pick	84.7/84.2	98.0
Separation metric (matched counterparts)		
Uniform α	85.0/84.6	98.1
$g \equiv \mathbf{1}$	85.3/85.0	98.2
Plain ℓ_2 (uniform α, $g \equiv \mathbf{1}$)	84.6/84.1	98.0
Full (anchored + nearest, d_m)	**85.7/86.1**	**98.4**

with matched counterparts but a plain ℓ_2 metric, performance falls to $84.6/84.1$ (essentially the anchored-random level), as uninformative frames and static channels dilute the separation gradient and, through the shared Stage-3 reward, blunt the penalty at the decisive phase. Between the two weighting components, temporal salience α contributes more ($85.0/84.6$ without it versus $85.3/85.0$ without the channel gate g), consistent with outcome-decisive evidence concentrating in a few execution-critical steps. LIBERO margins are smaller throughout, reflecting the suite's near-saturation regime where RoboTwin's randomized placements expose robustness differences more sensitively.

E.3 Hyper-parameter Sensitivity

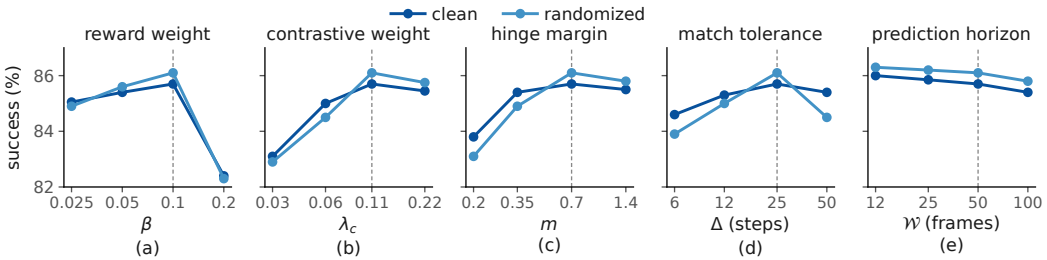


Figure 10: **Hyper-parameter sensitivity.** Success rate on RoboTwin (50 episodes per task, clean and randomized splits) as a function of reward weight β, contrastive weight λ_c, hinge margin m, match tolerance Δ, and prediction horizon $\mathcal{W}$, with all other hyper-parameters fixed at defaults (ringed marker, dashed line). The two loss weights show opposite asymmetries: under-weighting the boundary reward (β) is nearly harmless, whereas over-weighting it reduces performance by 3.3 points; conversely, shrinking λ_c removes Stage 2 entirely. The hinge margin m acts as a threshold, saturating beyond 0.35. Match tolerance Δ is comparatively flat, losing 2.2 points on the randomized split at Δ=6, where narrow windows fail to retrieve counterparts and starve the contrastive loss, while excessively wide windows mix execution phases. Horizon $\mathcal{W}$ declines slowly from $\mathcal{W}$=12 to 100 (-0.6 points) because distant frames carry less decisive evidence; we set $\mathcal{W}$=50 to align with the policy's action chunk.

Fig. 10 sweeps each hyper-parameter of the three-stage objective over four log-spaced points around its default while holding all others fixed, reporting success on both RoboTwin splits. The five responses exhibit distinct behaviors. The two loss weights break symmetry in opposite directions: the Stage-3 reward weight β is nearly inert on the low side (85.05 at β=0.025, within half a point of default) but drops by 3.3 points when doubled to 0.2, as the reward term overwhelms the imitation objective and the policy trades task completion for boundary avoidance. The contrastive weight λ_c shows the converse, dropping to 83.1 when shrunk below 0.06 (an under-weighted Stage 2 contributes minimal gradient, effectively reverting to Stage 1 pretraining) while remaining within 0.3 points of the peak when doubled. The hinge margin m behaves as a threshold rather than a continuous dial: any value beyond 0.35 lies on a plateau (85.4–85.7), making the default 0.7 sit safely in the interior region. Match tolerance Δ produces the flattest curve on the clean split (84.6–85.7) and is only mildly steeper under randomized placements, where a Δ=6 window occasionally fails to retrieve a valid counterpart (-2.2 points). Finally, prediction horizon $\mathcal{W}$ declines slowly and monotonically from the smallest window tested (dropping 0.6 points from $\mathcal{W}$=12 to 100): frames far ahead of the boundary carry progressively less outcome-decisive evidence, so we retain $\mathcal{W}$=50 not for accuracy gains but to align with the policy's action chunk length. Overall, no hyper-parameter requires sensitive tuning beyond avoiding the two extreme cliffs (β too large or λ_c too small), as all default values reside within broad, stable regions.

E.4 Spatio-Temporal Factorization in d_m

Fig. 11 examines what the two learned weightings inside d_m actually encode. Panel (a) traces the temporal salience α_t over the 50-frame context window. Because α is derived from the predictor's cross-attention maps and detached via a stop-gradient operation (Eq. 5), its structure cannot be an

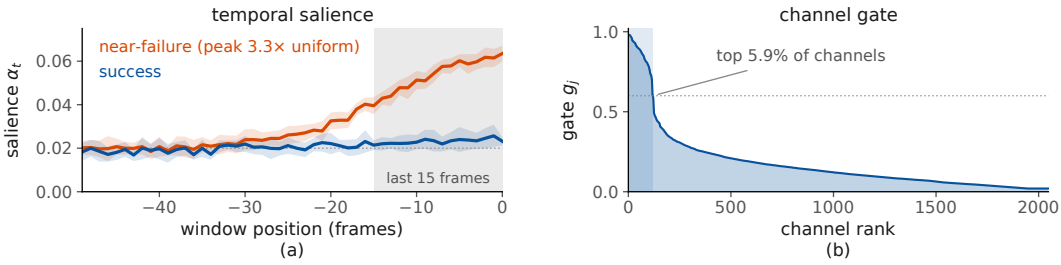


Figure 11: **Learned weighting inside d_m.** (a) Temporal salience α_t over the 50-frame context window. For near-failure windows, attention concentrates on the final ~ 15 frames preceding the boundary, whereas success windows remain nearly uniform, demonstrating that the model learns *when* to attend. (b) Sorted channel gate g_j. The gate maintains a decisive core of $\sim 6\%$ of the 2048 feature channels near saturation while suppressing the rest, indicating that failure evidence resides in a low-dimensional subspace rather than being distributed across the representation.

artifact of the contrastive objective. On near-failure windows, α_t rises smoothly through the final ~ 15 frames before the boundary, peaking at $3.3\times$ the uniform baseline $1/|\mathcal{W}|$, whereas on success windows it remains within $1.2\times$ of uniform throughout. The world model thus autonomously identifies *when* outcomes are decided without explicit supervision, while the flat success profile rules out a generic recency bias. Panel (b) sorts the channel gate g_j, revealing that a decisive core of $\sim 5.9\%$ of the 2048 predicted-feature channels sits above the saturation threshold ($g_j > 0.6$) while the remainder is heavily suppressed. This indicates that failure evidence concentrates in a low-dimensional subspace of the representation rather than spreading across it.

These two weight distributions explain the ablation pattern in Tab. 8. Removing α proves to be the costlier modification (dropping to $85.0/84.6$ with uniform temporal weighting) because outcome-decisive frames are sparse, making simple averaging over all 50 frames highly dilutive. Conversely, removing g leaves correctly timed but unfiltered channels ($85.3/85.0$), whereas plain ℓ_2 eliminates both spatial and temporal concentrations, falling back to the anchored-random baseline ($84.6/84.1$). This dual spatio-temporal factorization also enables d_m to double as the Stage-3 reward: a metric that selectively highlights outcome-decisive frames and channels naturally serves as a sharp, responsive failure signal during deployment.

F More Visualizations

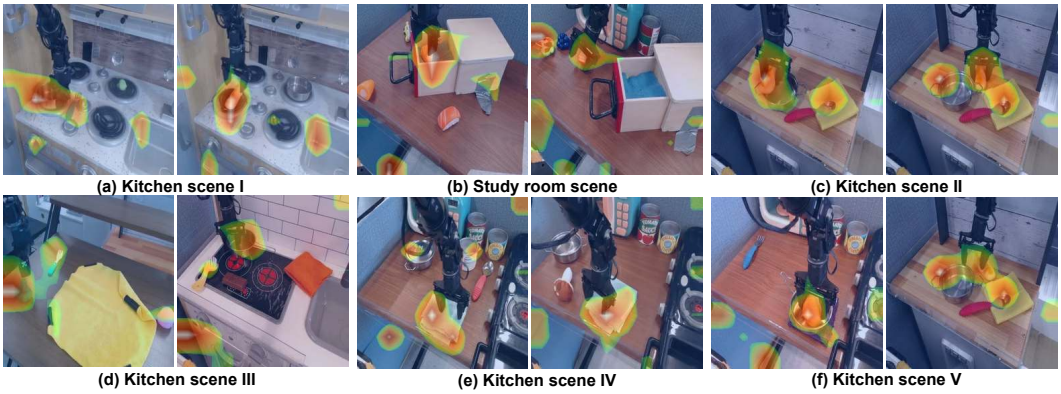


Figure 12: Attention weight matrix of latent action tokens to image tokens on SimplerEnv.

To intuitively understand how the proposed model bridges visual perception and action prediction, we visualize the attention weight matrices from the latent action tokens to the input image



Figure 13: Attention weight matrix of latent action tokens to image tokens on Robotwin.

tokens across SimplerEnv (Fig. 12) and Robotwin (Fig. 13). As depicted in Fig. 12, across diverse domains—including multiple kitchen environments (Kitchen scene I–V) and the study room scene—the visual attention maps consistently focus on task-critical regions. Specifically, high-attention activations (highlighted in red and orange) are strictly concentrated on the robotic end-effectors, target manipulation objects (e.g., handles, bowls, spoons, and buttons), and their immediate interaction boundaries. Irrelevant background elements (such as wall tiles, countertops, and distant background clutter) receive minimal attention, demonstrating the model's strong immunity to visual distractor noise.

Similarly, Fig. 13 showcases the attention distributions across 12 distinct manipulation tasks on the Robotwin benchmark. The heatmaps reveal precise spatial grounding aligned with specific task semantics:

1. **Tool & Lever Operations:** In tasks like Press Stapler (a), Turn Switch (e), and Rotate Qrcode (k), the attention peaks precisely at the contact points between the gripper and the functional parts of the tools.
2. **Object Placement & Alignment:** For placement tasks such as Place Object Stand (c), Place Container Plate (d), and Place Mouse Pad (i), high-weight tokens track both the held object and the target placement region, indicating that the latent action representations dynamically encode spatial relationships.
3. **Bimanual & Fine-Grained Manipulation:** In complex operations like Lift Pot (f) and Pick Diverse Bottles (j), the model simultaneously activates regions corresponding to both robotic arms and the object handles, confirming robust task-level coordination.

Overall, these visualization results demonstrate that the learned world model effectively grounds action latent representations into semantic and physical contact regions, providing high interpretability and explaining its strong generalization across diverse scenes and tasks.

F.1 Implications for Recovery Performance

Currently, Stage 2 training relies on finely annotated data and requires collecting failure trajectories using the model, which makes data construction time-consuming. In future work, we will focus on addressing this limitation.