ACWM-RM: A REFERENCE-FREE REWARD MODEL FROM COMPLEMENTARY MISMATCH PREFERENCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning has gained attention for improving action-conditioned world models (ACWMs), as it enables reward-driven optimization of visual quality (VQ) and action-following fidelity (AF) beyond flow-matching optimization. Realizing this potential requires reward signals that reliably rank videos by both VQ and AF. However, existing rewards capture only one dimension: metric-based rewards assess VQ but require ground-truth videos and do not directly assess AF, while inverse dynamics model (IDM) rewards assess AF without references but do not guarantee VQ. Moreover, IDMs, typically trained on real videos, may provide unreliable rewards for generated videos containing visual artifacts and physical inconsistencies. In this work, we introduce the Action and Video Mismatch Benchmark (AVM-Eval), comprising 67.2k pairs that evaluate how reliably IDMs assess AF on real and generated videos. We then propose ACWM-RM, a reward model that jointly assesses VQ and AF without ground-truth videos during reward computation. To train it, we first construct the Action and Video Mismatch Preference Dataset (AVM-Pref) with 506.3k pairs providing complementary supervision: action-mismatch pairs supervise AF, while video-mismatch pairs supervise VQ and expose the model to generated videos. We then pretrain an action-conditioned predictor on real videos to learn forward dynamics before jointly training it with a patch-weighted score head on AVM-Pref to transfer these dynamics to AF assessment on generated videos. Evaluation on AVM-Eval reveals the limited reliability of IDMs on generated videos, while downstream RL experiments show that ACWM-RM outperforms reward baselines in both criteria.

1 INTRODUCTION

Action-conditioned world models (ACWMs) (Yang et al., 2023; Wu et al., 2024; Zhu et al., 2025; Liao et al., 2025) predict videos conditioned on robot actions, supporting embodied policy evaluation and optimization. For these applications, generated videos must preserve visual quality while faithfully reflecting the commanded actions. However, standard flow-matching training (Lipman et al., 2022) does not explicitly optimize these criteria. This motivates reinforcement learning (RL) such as Flow-GRPO (Liu et al., 2026a; Wu et al., 2026a), which optimizes these criteria using rewards.

The effectiveness of RL, however, depends on reward signals that reliably rank videos generated under the same conditioning input, assigning higher scores to those with better visual quality (VQ) and action-following fidelity (AF). Existing reward signals fall into two main categories. First, full-reference rewards (Wu et al., 2026b) use metrics such as PSNR, SSIM, or LPIPS to compare generated videos with the corresponding ground-truth videos, providing VQ supervision but requiring reference videos and not directly assessing AF. Second, inverse dynamics model (IDM) rewards (Seo et al., 2026; Chen et al., 2026a) assess AF without requiring reference videos by comparing actions inferred from generated videos with the input actions used to generate those videos, but do not directly assess VQ. Moreover, IDMs are typically trained on real-world videos, where motion and interactions obey physical laws, but are applied during RL to ACWM-generated videos that may contain visual artifacts and physical inconsistencies, such as robot-arm deformation and interpenetration. To quantify this issue, we generate 100 videos with each of four ACWMs under identical inputs. As shown in Figure 1, ACWM-generated videos exhibit more visual artifacts and physical inconsistencies than real-world videos. This gap raises the question of whether strong action-mismatch discrimination on real videos translates to reliable assessment of generated videos.

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.

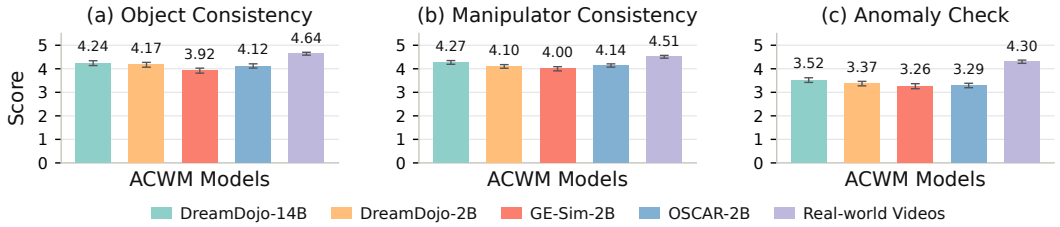


Figure 1: Evaluation scores for generated and real-world videos assigned by GPT-5.6 sol using RBench (Deng et al., 2026) (1: worst; 5: best). Prompts and criteria are detailed in Appendix F.

In this work, we first construct the Action and Video Mismatch Benchmark (AVM-Eval), comprising 67.2k pairs. It evaluates how reliably IDMs assess AF on real videos using action-mismatch pairs and on generated videos using video-mismatch pairs. We then propose a reward model (ACWM-RM) for jointly assessing VQ and AF without ground-truth videos at reward computation time. To train ACWM-RM, we construct the Action and Video Mismatch Preference Dataset (AVM-Pref) with 506.3k pairs of two types that provide complementary supervision: Action-mismatch pairs contrast original and perturbed actions for the same real video, providing AF supervision; Video-mismatch pairs contrast different generated videos under the same actions, providing VQ supervision while exposing the model to the generated-video distribution. For training strategy, we first pretrain an action-conditioned predictor on features extracted from real robot videos by a frozen V-JEPA 2.1 encoder (Mur-Labadia et al., 2026), establishing a dynamics representation for AF evaluation. We then jointly optimize the predictor and patch-weighted score head on AVM-Pref, transferring the dynamics learned from real-world videos to AF assessment of generated videos.

Our evaluation of four architecturally distinct IDMs on AVM-Eval shows that accurate AF assessment on real videos does not ensure reliable AF evaluation of generated videos. The best-performing IDM achieves 86.1% accuracy on action-mismatch pairs from real videos but only 70.0% on video-mismatch pairs from generated videos. To evaluate ACWM-RM's downstream effectiveness as an RL reward, we fine-tune DreamDojo-2B with Flow-GRPO, comparing ACWM-RM against the best-performing IDM on AVM-Eval, a metric-based reward, and a VLM-based reward. On held-out tasks, ACWM-RM yields the best VQ and AF, while maintaining VLM-assessed physical consistency (PC) scores close to the original model before RL fine-tuning.

Our main contributions: (1) We introduce AVM-Eval and show that, across four IDMs, accurate AF assessment on real videos does not ensure reliable AF evaluation of generated videos. (2) We develop AVM-Pref and ACWM-RM, combining dynamics pretraining with complementary mismatch preferences to assess both criteria without references during reward computation. (3) We demonstrate that ACWM-RM outperforms reward baselines in VQ and AF when used for RL fine-tuning.

2 Related Work

2.1 Action-Conditioned World Models

Action-conditioned world models generate videos conditioned on past observations and actions, simulating the resulting robot interactions to support policy evaluation and planning. To learn transferable dynamics, DreamDojo (Gao et al., 2026) pretrains on human videos using continuous latent actions and adapts to target robots with robot data. To condition predictions on actions, iVideoGPT (Wu et al., 2024) models observations and actions as token sequences with an autoregressive transformer, whereas IRASim (Zhu et al., 2025) strengthens action-frame alignment through frame-level conditioning in a diffusion transformer. These predictive capabilities enable downstream applications: Genie Envisioner (Liao et al., 2025) uses GE-Sim for action-conditioned simulation and closed-loop policy evaluation. Such applications require generated videos to preserve both VQ and AF, motivating our study of rewards that jointly assess these properties for ACWM post-training.

2.2 Reward Modeling for World Models

Reliable reward signals are essential for effective RL, as they determine which properties of videos are optimized. For general video generation, VideoReward (Liu et al., 2026b) learns human pref-

erences over visual quality and alignment. For embodied world models based on instructions, RoWorld (Peng et al., 2026) and RoboAlign-R1 (Wu et al., 2026a) assess video quality and instruction alignment without evaluating correspondence with numerical actions. For action-conditioned world models, RLVR-World (Wu et al., 2026b) provides visual rewards through ground-truth video comparisons, while RLIR (Ye et al., 2025) provides AF rewards by using IDM. Other dynamics-based rewards target different properties: EVA (Wang et al., 2026) assesses action executability using IDM. However, reference-based rewards require ground truth, and IDMs that are accurate on real videos does not ensure reliable evaluation of generated videos or joint assessment of VQ and AF.

3 PRELIMINARIES

Given an initial frame I_0 and an action chunk $\mathbf{a} = (a_1, \ldots, a_K)$, an ACWM p_θ predicts video $\hat{\mathbf{I}} = (\hat{I}_1, \ldots, \hat{I}_K)$ by sampling $\hat{\mathbf{I}} \sim p_\theta(\cdot \mid c)$, where $c = (I_0, \mathbf{a})$. To improve VQ and AF for ACWMs, Flow-GRPO is used with reward-based supervision. The reward R is typically computed using metrics or IDMs. Metric-based rewards compare generated videos $\hat{\mathbf{I}}$ with ground-truth videos $\mathbf{I} = (I_1, \ldots, I_K)$ using a metric D, such as PSNR, SSIM, or MSE. IDM-based rewards compare actions $\tilde{\mathbf{a}} = (\tilde{a}_1, \ldots, \tilde{a}_K)$ inferred from the generated video with the conditioning actions $\mathbf{a}$:

$$R^{\text{metric}} = s_D\, D\Big(\hat{\mathbf{I}}, \mathbf{I}\Big), \qquad R^{\text{IDM}} = -\frac{1}{14K} \sum_{k=1}^{K} \sum_{j=1}^{14} \ell_{\text{SL1}}\left(\frac{\tilde{a}_{k,j} - a_{k,j}}{\sigma_j + 10^{-8}}\right). \tag{1}$$

Here, $s_D = 1$ for higher-is-better metrics and $s_D = -1$ otherwise. For R^{IDM}, ℓ_{SL1} is the Smooth-L1 loss with threshold 1, and σ_j is the standard deviation of action dimension j.

To optimize the ACWM with reward R, Flow-GRPO first converts the deterministic sampling ODE into an SDE with matching marginals, enabling stochastic sampling in denoising process. For each c, it samples G videos $\{\hat{\mathbf{I}}_i\}_{i=1}^{G}$ from the rollout policy $p_{\theta_{\text{old}}}$ and scores them using R. The resulting scores R_i are normalized into group-relative advantages $\hat{A}_i = (R_i - \mu_R)/(\sigma_R + \epsilon)$, where μ_R and σ_R are the group reward mean and standard deviation. The ACWM is then updated by maximizing:

$$\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\Big(r_i(\theta)\hat{A}_i, \text{clip}\big(r_i(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_i\Big) - \beta D_{\text{KL}}(p_\theta(\cdot \mid c) \,\|\, p_{\text{ref}}(\cdot \mid c))\right], \tag{2}$$

where $r_i(\theta)$ is the SDE transition likelihood ratio of p_θ to $p_{\theta_{\text{old}}}$, ε is the clipping range, and β controls KL regularization toward the frozen reference model p_{ref}. The clipped objective and KL penalty are averaged over denoising steps; step indices and latent conditioning are omitted for brevity.

4 METHOD

We first describe AVM-Eval construction (Sec. 4.1) and then our ACWM-RM, including AVM-Pref construction, model architecture, and two-stage training (Sec. 4.2).

4.1 ACTION AND VIDEO MISMATCH BENCHMARK

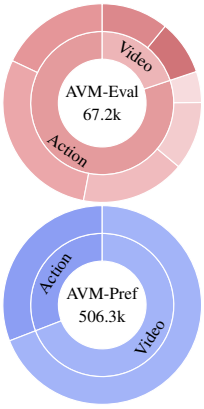
Existing IDMs are normally trained and evaluated on real-world videos. However, IDM-based rewards are applied to generated videos, whose distributions can differ from those of real-world videos. Generated videos may contain visual artifacts and physical inconsistencies, including robot-arm deformation and interpenetration, absent from real videos. To our knowledge, no existing benchmark compares the reliability of IDM-based AF assessment on real and generated videos.

To address this gap, we construct AVM-Eval using a pairwise evaluation protocol (Lambert et al., 2025). AVM-Eval compares the reliability of AF assessment on real and ACWM-generated videos, using **action-mismatch pairs** and **video-mismatch pairs**, respectively. We next describe the source data preparation and benchmark construction.

4.1.1 SOURCE DATA PREPARATION

We use AgiBotWorld (Bu et al., 2025), a large-scale bimanual robot dataset with real-world videos, as our source dataset, and partition its data by task into three subsets: Agi_A, Agi_B, and Agi_C.

Table 1: Statistics of AVM-Eval and AVM-Pref. Abbreviations: N-P/S-P, numerical/semantic perturbations; Intra/Cross, intra-model and cross-model pairs. Video source: Rel., real-world; Rec., simulator-recorded; Gen., generated. Criteria: AF, action-following fidelity; VQ, visual quality.



	Source	Tasks	Episodes	Action Pairs		Video Pairs		Video	Criteria
				N-P	S-P	Intra	Cross		
AVM-Eval	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$	36	2.2k	4.9k	7.2k	–	–	Rel.	AF
	$\mathrm{Agi}_{\mathrm{B}}$	30	10.0k	12.5k	7.0k	–	–	Rel.	AF
	$\mathrm{Robo}_{\mathrm{T}}$	50	2.2k	4.4k	7.0k	–	–	Rec.	AF
	$\mathrm{Robo}_{\mathrm{D}}$	35	1.5k	2.6k	4.9k	–	–	Rel.	AF
	Gen	19	2.3k	1.2k	2.2k	–	–	Rec.	AF
	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$	36	2.2k	–	–	3.0k	3.0k	Gen.	AF
	$\mathrm{Agi}_{\mathrm{B}}$	30	10.0k	–	–	4.3k	3.0k	Gen.	AF
AVM Pref	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$	36	34.0k	63.5k	92.8k	–	–	Rel.	AF
	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$	36	34.0k	–	–	250.0k	100.0k	Gen.	VQ

Covering 36 tasks, $\mathrm{Agi}_{\mathrm{A}}$ is split at the episode level into 34.0k training episodes ($\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$) for IDM, ACWM-RM predictor pretraining, and AVM-Pref construction, and 2.2k testing episodes ($\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$) for constructing seen-task evaluation pairs in AVM-Eval. For unseen-task evaluation, we construct AVM-Eval pairs from $\mathrm{Agi}_{\mathrm{B}}$, which contains 10.0k episodes across 30 tasks and also supports Flow-GRPO fine-tuning. Table 1 summarizes the resulting AVM-Eval pairs. For downstream evaluation of RL-optimized ACWMs, we use $\mathrm{Agi}_{\mathrm{C}}$, containing 1.5k episodes across 10 tasks. Additionally, to evaluate the embodiment and scene generalization of the IDMs, we also use other robot datasets from Genie-Sim (Yin et al., 2026) Gen, RoboDojo (Chen et al., 2026b) $\mathrm{Robo}_{\mathrm{D}}$, and RoboTwin (Mu et al., 2025) $\mathrm{Robo}_{\mathrm{T}}$. More statistics are provided in Appendix H. Across all datasets, we subsample each episode to 8 fps and randomly extract two 13-frame video clips for all experiments, each with corresponding end-effector states $\mathbf{s} = (s_0, \ldots, s_{12})$ and action chunks $\mathbf{a} = (a_1, \ldots, a_{12})$.

4.1.2 AVM-Eval Construction

AVM-Eval comprises two complementary types of pairs: A video-mismatch pair holds the action chunk fixed and contrasts two videos, whereas an action-mismatch pair holds the video fixed and contrasts two action chunks. Together, these two types constitute 67.2k pairs in AVM-Eval.

Action-mismatch pairs (53.9k). To evaluate IDM-based AF assessment on real videos, we construct action-mismatch pairs. Each pair is represented as $(I_0, \mathbf{a}^+, \mathbf{a}^-, \mathbf{I})$, where $\mathbf{a}^+$ is the action chunk recorded with $\mathbf{I}$ and $\mathbf{a}^-$ is a perturbed counterpart. We generate $\mathbf{a}^-$ using two types of perturbations: Numerical Perturbations (N-P) and Semantic Perturbations (S-P). N-P probe sensitivity to low-level numerical errors in end-effector translation and rotation magnitudes or directions, gripper opening degree, and action ordering. S-P probe sensitivity to high-level semantic errors through position offsets during grasping or placement, incorrect gripper timing, and mid-motion drift or freeze. We also perturb action chunks from Gen, $\mathrm{Robo}_{\mathrm{D}}$, and $\mathrm{Robo}_{\mathrm{T}}$ to assess IDM generalization across embodiments and scenes. Detailed perturbation procedures are provided in Appendix A.

Video-mismatch pairs (13.3k). To evaluate AF on generated videos, we considered constructing action-mismatch pairs in the same way. However, ACWMs cannot guarantee that generated videos faithfully follow the action chunk, so a perturbed action chunk may be more consistent with the generated video than the original action chunk, reversing the intended preference label. We therefore construct video-mismatch pairs by fixing the action chunk and ranking generated videos using WorldArena's nDTW (Shang et al., 2026) for AF, with additional filtering to limit the influence of visual quality (Appendix B). We use four ACWMs with varying AF performance: DreamDojo-2B (DD-2B), DreamDojo-14B (DD-14B) (Gao et al., 2026), GE-Sim-2B (GE-2B), and OSCAR-2B (OS-2B) (Wu & Gao, 2026). Under the same I_0 and $\mathbf{a}$, each ACWM independently samples six 12-frame videos using the stochastic SDE sampler adopted by Flow-GRPO. Within each ACWM, we pair a higher-nDTW video $\hat{\mathbf{I}}^+$ with a lower-nDTW video $\hat{\mathbf{I}}^-$ to form an **intra-model pair** $(I_0, \mathbf{a}, \hat{\mathbf{I}}^+, \hat{\mathbf{I}}^-)$. We rank videos generated by any two ACWMs under the same input to construct **cross-model pairs**.

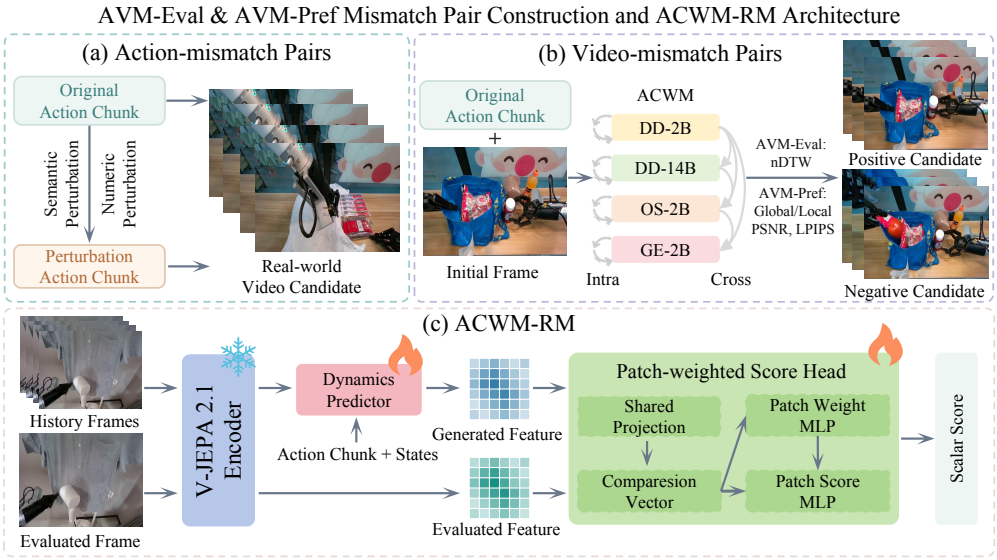


Figure 2: Overview of (a,b) action-mismatch and video-mismatch pair construction for AVM-Eval and AVM-Pref, and (c) ACWM-RM reward computation without reference videos.

4.2 Reward Model for Action-Conditioned World Models

Existing rewards provide only partial supervision, with metric-based rewards also requiring ground-truth videos. We therefore propose ACWM-RM, a reference-free reward model that jointly assesses VQ and AF. We next describe the construction of its training dataset, termed AVM-Pref (Sec. 4.2.1), followed by the architecture and the two-stage training strategy (Sec. 4.2.2).

4.2.1 AVM-Pref Construction

To train ACWM-RM with the Bradley-Terry loss, we construct the Action and Video Mismatch Preference Dataset (AVM-Pref) from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, comprising 506.3k action-mismatch and video-mismatch pairs that provide complementary supervision. **Action-mismatch pairs (156.3k)**: We use the same numerical and semantic action perturbations as AVM-Eval to provide AF supervision. **Video-mismatch pairs (350k)**: We follow AVM-Eval to generate videos from four ACWMs and construct intra-model and cross-model pairs, but use four visual quality metrics (global and local PSNR and LPIPS) instead of nDTW to provide VQ supervision. Since action-mismatch pairs already provide broader AF supervision than nDTW's 2D trajectory alignment, including translation, rotation, and gripper opening, we use video-mismatch pairs for complementary VQ supervision. Although action-mismatch pairs use real videos and thus differ from the generated-video distribution, our empirical results show that combining both types improves AF assessment on generated videos. Finally, to reduce sampling bias, we aim for comparable pair counts across models and tasks for intra-model pairs and across model combinations and tasks for cross-model pairs. Appendix C details how the four metrics are used, while Appendix H provides AVM-Pref statistics.

4.2.2 Reward Model Architecture and Training Strategy

ACWM-RM scores a candidate video conditioned on an action chunk and its associated state sequence, producing a scalar reward for VQ and AF. It consists of a frozen V-JEPA 2.1 encoder E, a bimanual action-conditioned predictor P_ϕ, and a patch-weighted score head h_ψ. Training proceeds in two stages: dynamics pretraining optimizes P_ϕ on real-world videos, while preference training jointly optimizes P_ϕ and h_ψ on AVM-Pref to adapt the learned dynamics for AF assessment.

Dynamics pretraining. Following V-JEPA 2-AC (Assran et al., 2025), we pretrain P_ϕ with E frozen. Unlike the original single-arm setting, our predictor takes bimanual actions. For bimanual inputs, we express actions and states in a coordinate system aligned with the head-camera axes, normalize them, project them to the predictor's hidden dimension, and add a learnable identity embedding for each arm. At each time step t, tokens are ordered as $\left[u^s_{t,L}, u^a_{t,L}, u^s_{t,R}, u^a_{t,R}, U^z_t\right]$,

where $u^s_{t,r}$ and $u^a_{t,r}$ denote state and action tokens for arm $r \in \{L, R\}$, and U^z_t contains projected frame features from E. Appendix D.1 provides further details and loss functions.

Preference training. We jointly train P_ϕ and h_ψ to rank pairs by comparing predicted and encoded frame features and aggregating the resulting patch scores. For each candidate video $\mathbf{I}'$ from either $\mathbf{I}$ or $\hat{\mathbf{I}}$, P_ϕ predicts a layer-normalized representation z^P_t at time step t from the candidate's encoded frames $I'_{0:t-1}$, actions $a'_{1:t}$, and states $s'_{0:t-1}$, while E encodes the evaluated frame as $z^E_t = \mathrm{LN}(E(I'_t))$. Here, $s'_{0:t-1}$ is reconstructed from the candidate's actions and initial state s_0. To avoid accumulating rollout errors, the predictor conditions on the candidate's own history rather than recursively predicted features. For scoring, h_ψ compares z^P_t and z^E_t patch by patch, using the spatial correspondence between their N patch features. Specifically, for each patch n, a shared projection maps both features to a common space, yielding $\bar{z}^P_{t,n} = W_{\mathrm{proj}} z^P_{t,n}$ and $\bar{z}^E_{t,n} = W_{\mathrm{proj}} z^E_{t,n}$. The projected features are then combined into a comparison vector:

$$c_{t,n} = \big[\bar{z}^P_{t,n} - \bar{z}^E_{t,n};\ |\bar{z}^P_{t,n} - \bar{z}^E_{t,n}|;\ (\bar{z}^P_{t,n} - \bar{z}^E_{t,n})^2;\ \bar{z}^P_{t,n} \odot \bar{z}^E_{t,n};\ \cos(\bar{z}^P_{t,n}, \bar{z}^E_{t,n})\big], \tag{3}$$

which combines feature differences, elementwise product, and cosine similarity.

Different regions contribute unequally to VQ and AF assessment. Thus, we use two MLPs to predict patch scores and spatial weight logits, enabling adaptive aggregation of evidence for both criteria:

$$\ell_{t,n} = f_{\mathrm{score}}(c_{t,n}), \qquad \tilde{\alpha}_{t,n} = f_{\mathrm{weight}}(c_{t,n}). \tag{4}$$

We apply softmax over patches to obtain weights, compute each frame's score as a weighted sum of patch scores, and average frame-level scores over K future frames to obtain the candidate reward:

$$R_{\phi,\psi}(\mathbf{I}', \mathbf{a}', \mathbf{s}') = \frac{1}{K} \sum_{t=1}^{K} \sum_{n=1}^{N} \alpha_{t,n} \ell_{t,n}, \qquad \alpha_{t,n} = \frac{\exp(\tilde{\alpha}_{t,n})}{\sum_{n'=1}^{N} \exp(\tilde{\alpha}_{t,n'})}. \tag{5}$$

We score both candidates with shared parameters and minimize the Bradley-Terry loss to favor the preferred candidate, keeping E frozen. Appendix D.2 provides the loss and training details.

5 EXPERIMENTS

We first describe experimental setup (Sec. 5.1) and evaluate IDMs and ACWM-RM on AVM-Eval (Sec. 5.2). We then compare Flow-GRPO reward signals (Sec. 5.3) and conduct ablations (Sec. 5.4).

5.1 EXPERIMENTAL SETUP

Baselines. On AVM-Eval, we compare ACWM-RM with four architecturally distinct IDMs: EVA-IDM ($\mathrm{EVA_I}$) (Wang et al., 2026), which uses a convolutional backbone; GR00T-IDM ($\mathrm{GRT_I}$) (Bjorck et al., 2025), which uses a diffusion-based model; TAP-IDM ($\mathrm{TAP_I}$) (Shi et al., 2026), a vision-language model equipped with an action expert; and J-IDM ($\mathrm{J_I}$) (Li et al., 2026), which recovers actions by inverting a predicted per-pixel Jacobian field. This evaluates how reliably IDMs with different architectures transfer from real-video training to generated-video evaluation.

For Flow-GRPO training of ACWMs, we compare ACWM-RM with IDM-based, metric-based, and VLM-based rewards. The IDM-based baseline uses the best-performing IDM on AVM-Eval. The metric-based baseline combines the four visual quality metrics used in AVM-Pref following the fusion procedure of RLVR-World (Wu et al., 2026b). We further explore whether VLM-as-a-Judge (Hong et al., 2025) can provide effective rewards for ACWM optimization. Specifically, we use Qwen3-VL-30B-A3B-Instruct to score videos using embodiment-specific prompts and criteria from RBench. Prompts and criteria definitions are provided in the Appendix F.

Training setup. We train the four IDMs on $\mathrm{Agi}^{\mathrm{train}}_{\mathrm{A}}$ using their official pretrained backbones and implementations, following the original training paradigm of predicting actions from real-world videos. Each IDM and ACWM-RM has a nominal budget of approximately 7.12M 13-frame video draws, including both ACWM-RM training stages. For Flow-GRPO, we fine-tune DreamDojo-2B on $\mathrm{Agi_B}$ and evaluate it on $\mathrm{Agi_C}$. Implementation details for the IDMs, ACWM-RM, and Flow-GRPO are provided in Appendices E, D, and G, respectively.

Evaluation protocol. We evaluate IDMs using pairwise accuracy on AVM-Eval. For each ACWM optimized with Flow-GRPO, we repeat video generation five times and report the mean and standard

Table 2: Pairwise accuracy on AVM-Eval within the AgiBot domain. $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ evaluates seen tasks and $\mathrm{Agi}_{\mathrm{B}}$ evaluates task generalization. N-P and S-P denote numerical and semantic perturbations, respectively. Intra and Cross denote Intra-model and Cross-model pairs, respectively.

Model	Action-mismatch Pair					Video-mismatch Pair				
	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$		$\mathrm{Agi}_{\mathrm{B}}$		Overall	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$		$\mathrm{Agi}_{\mathrm{B}}$		Overall
	N-P	S-P	N-P	S-P		Intra	Cross	Intra	Cross	
$\mathrm{EVA}_{\mathrm{I}}$	92.1	81.6	90.6	80.2	86.1	69.3	73.8	70.2	66.8	70.0
$\mathrm{GRT}_{\mathrm{I}}$	74.7	70.9	75.1	61.1	70.5	52.4	52.7	51.9	52.3	52.3
$\mathrm{TAP}_{\mathrm{I}}$	79.1	67.3	76.2	63.6	71.6	58.5	58.0	55.7	58.3	57.6
J_{I}	76.7	62.2	77.0	63.7	69.9	58.3	57.9	57.8	55.7	57.4
Ours	**97.9**	**86.1**	**97.7**	**84.7**	**91.6**	**90.5**	**93.3**	**86.2**	**84.3**	**88.6**

Table 3: Pairwise accuracy on action-mismatch pairs, testing embodiment and scene generalization. N-P and S-P denote numerical and semantic perturbations.

Model	Gen		$\mathrm{Robo}_{\mathrm{T}}$		$\mathrm{Robo}_{\mathrm{D}}$		Overall
	N-P	S-P	N-P	S-P	N-P	S-P	
$\mathrm{EVA}_{\mathrm{I}}$	61.5	50.9	69.5	70.6	56.5	59.5	61.4
$\mathrm{GRT}_{\mathrm{I}}$	50.9	49.7	61.6	63.5	59.4	59.5	57.4
$\mathrm{TAP}_{\mathrm{I}}$	58.9	52.4	74.6	54.2	**66.7**	**62.9**	61.6
J_{I}	59.9	54.2	63.5	61.9	58.1	55.5	58.9
Ours	**65.9**	**55.5**	**77.8**	**72.4**	66.2	60.9	**66.5**

Table 4: Action-following fidelity on $\mathrm{Agi}_{\mathrm{C}}$ (nDTW $\times 100$). Colors follow Table 5.

Method	nDTW ↑
Origin	14.84 $\pm$0.24
IDM-based	13.21 $\pm$0.38
VLM-based	14.00 $\pm$0.41
Metric-based	14.05 $\pm$0.33
Ours	**16.41** $\pm$0.22

deviation of the following metrics across runs: (1) VQ, measured by PSNR, LPIPS, SSIM, MSE, and FVD; (2) AF, measured by nDTW, a proxy based on the alignment of generated and ground-truth 2D manipulator trajectories; and (3) physical consistency, assessed by GPT-5.6 sol using the same prompt and criteria as the VLM-based reward, covering Object Consistency (OC), Manipulator Consistency (MC), and Anomaly Check (AC).

5.2 IDM Performance on AVM-Eval

Table 2 examines whether IDMs trained on real videos transfer to AF assessment on generated videos. On action-mismatch pairs from real videos, IDM accuracies range from 69.9 to 86.1, with $\mathrm{EVA}_{\mathrm{I}}$ performing best. On video-mismatch pairs from ACWM-generated videos, however, $\mathrm{EVA}_{\mathrm{I}}$ reaches only 70.0, while the other three IDMs score 52.3 to 57.6. These results show that strong action-mismatch performance on real videos does not ensure reliable AF evaluation on generated videos. Therefore, before using an IDM as a reward for ACWM training, its ability to accurately infer actions from generated videos should be evaluated. Table 3 shows limited IDM generalization across embodiments and scenes: overall action-mismatch accuracy ranges from 57.4 to 61.6.

5.3 Flow-GRPO with Different Reward Signals

We evaluate ACWMs trained with different rewards on $\mathrm{Agi}_{\mathrm{C}}$ along the following dimensions.

Visual quality. ACWM-RM achieves the best results across all five VQ metrics in Table 5, outperforming the full-reference metric-based reward. Unlike this baseline, ACWM-RM requires no ground-truth videos for reward computation during RL. Although its training preferences are derived from PSNR and LPIPS, the gains extend to SSIM, MSE, and FVD, which are not used for preference supervision. In contrast, the IDM-based reward degrades all five metrics relative to Origin, suggesting that optimizing IDM-based AF alone is insufficient to preserve visual quality.

Action-following fidelity. Table 4 shows that ACWM-RM is the only evaluated reward that improves nDTW over Origin, from 14.84 to 16.41. The VLM-based and metric-based rewards yield no improvement, while the IDM-based reward reduces nDTW to 13.21 despite explicitly measur-

Table 5: Visual quality and physical consistency of DreamDojo on Agi_C after Flow-GRPO with different reward signals, compared with the pre-RL model (Origin). LPIPS, SSIM, and MSE are reported $\times 100$, and physical consistency scores range from 1 to 5. Better Worse than Origin.

Method	Visual Quality					Physical Consistency			
	PSNR ↑	LPIPS ↓	SSIM ↑	MSE ↓	FVD ↓	OC ↑	MC ↑	AC ↑	Overall ↑
Origin	23.13 ± 0.06	3.66 ± 0.04	88.67 ± 0.02	0.49 ± 0.0	18.31 ± 1.09	**4.25** ± 0.10	**4.31** ± 0.29	**3.43** ± 0.06	**4.00** ± 0.10
IDM-based	22.82 ± 0.08	4.27 ± 0.04	88.04 ± 0.11	0.58 ± 0.0	31.31 ± 0.07	4.19 ± 0.17	4.16 ± 0.37	3.42 ± 0.09	3.92 ± 0.14
VLM-based	24.03 ± 0.17	3.75 ± 0.12	88.41 ± 0.28	0.49 ± 0.0	18.20 ± 1.08	4.12 ± 0.17	4.07 ± 0.32	3.38 ± 0.06	3.86 ± 0.12
Metric-based	24.20 ± 0.09	3.63 ± 0.06	89.62 ± 0.19	0.48 ± 0.0	17.56 ± 0.63	4.11 ± 0.22	4.07 ± 0.36	3.34 ± 0.12	3.84 ± 0.15
Ours	**24.34** ± 0.06	**3.51** ± 0.05	**89.86** ± 0.13	**0.46** ± 0.0	**17.10** ± 0.56	4.20 ± 0.12	4.17 ± 0.22	3.42 ± 0.05	3.93 ± 0.08

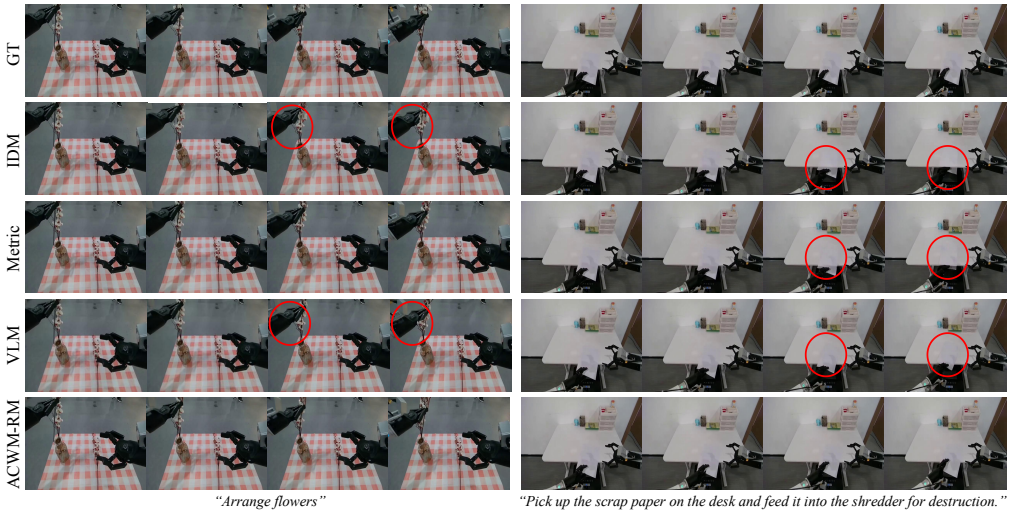


Figure 3: Qualitative comparison of generated videos with different reward signals.

ing consistency between inferred and conditioning actions. Together with the AVM-Eval findings, these results suggest that strong action-mismatch discrimination on real videos does not necessarily translate into an effective RL reward for improving action-following fidelity.

Physical consistency. Although ACWM-RM does not explicitly reward physical consistency, the model optimized with it achieves scores comparable to Origin across all three criteria in Table 5. By contrast, the VLM-based reward yields a lower overall physical-consistency score than Origin (3.86 vs. 4.00), despite directly targeting these criteria. One possible explanation is disagreement between the VLMs used for reward computation and evaluation: optimizing scores assigned by the reward VLM may not improve physical consistency as judged by the evaluation VLM.

5.4 Ablation Study

Score head architecture. We freeze the pretrained predictor and optimize the remaining head parameters on AVM-Pref. Starting from the complete head (Frozen Predictor), we cumulatively replace learned patch weighting with uniform averaging (w/o weight MLP), replace comparison-vector-based scoring with cosine similarity between projected features (w/o comparison), and remove the projection (w/o projection). With the predictor frozen, the complete head improves video-mismatch accuracy from 55.0 to 79.0. Most of this gain comes from learned comparison-based scoring (59.4 to 77.7), while patch weighting provides a further improvement to 79.0 (Table 6).

Complementary preference supervision. Table 6 shows the complementary roles of both pair types (Sec. 4.2.1). Training only on action-mismatch pairs (w/o video-pair) matches Ours at 91.6 on real-video action-mismatch pairs but reaches 71.8 on generated-video pairs. Training only on video-mismatch pairs (w/o action-pair) achieves 82.5 on real-video action-mismatch pairs and 77.3 on generated-video pairs. Combining both achieves 91.6 and 88.6, respectively, suggesting that action-mismatch pairs supervise AF, while video-mismatch pairs help transfer it to generated videos.

Table 6: In-domain ablation results for four aspects of ACWM-RM on AVM-Eval.

Model	Action-mismatch Pair					Video-mismatch Pair				
	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$		$\mathrm{Agi}_{\mathrm{B}}$		Overall	$\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$		$\mathrm{Agi}_{\mathrm{B}}$		Overall
	N-P	S-P	N-P	S-P		Intra	Cross	Intra	Cross	
Frozen Predictor	97.0	78.8	95.0	76.5	86.8	<u>80.0</u>	<u>84.3</u>	<u>77.2</u>	<u>74.5</u>	<u>79.0</u>
w/o weight MLP	96.9	79.2	95.4	76.2	86.9	77.9	82.9	75.8	74.1	77.7
w/o comparison	96.2	76.5	94.2	75.2	85.5	57.8	61.8	58.9	59.1	59.4
w/o projection	96.0	76.4	93.9	75.1	85.4	53.7	55.7	55.7	54.7	55.0
Fresh Predictor	92.0	77.2	92.2	78.4	85.0	61.6	61.8	62.7	61.9	62.0
JEPA-IDM	95.1	84.1	93.1	81.7	<u>88.5</u>	73.2	76.3	73.6	71.5	73.7
Ours	**97.9**	**86.1**	**97.7**	<u>84.7</u>	**91.6**	**90.5**	**93.3**	**86.2**	**84.3**	**88.6**
w/o action-pair	91.5	75.8	89.4	73.1	82.5	78.2	81.4	76.2	73.4	77.3
w/o video-pair	<u>97.7</u>	<u>85.8</u>	<u>97.4</u>	**85.5**	**91.6**	71.9	73.9	70.6	70.8	71.8

Dynamics pretraining and adaptation. We examine whether real-video dynamics pretraining and joint preference training improve AF assessment. We construct two variants using the same score head and AVM-Pref: Fresh Predictor jointly trains a randomly initialized predictor and the head, while Frozen Predictor trains only the head with the pretrained predictor fixed. Compared with Fresh Predictor, Ours improves action-mismatch accuracy from 85.0 to 91.6 and video-mismatch accuracy from 62.0 to 88.6 (Table 6). These gains support the value of learning dynamics before preference training. Compared with Frozen Predictor, Ours improves video-mismatch accuracy from 79.0 to 88.6 under the same preference supervision, showing that adapting the pretrained predictor during preference training further benefits AF assessment on generated videos.

Table 7: Out-of-domain ablation results for four aspects of ACWM-RM on AVM-Eval.

Model	Gen		$\mathrm{Robo}_{\mathrm{T}}$		$\mathrm{Robo}_{\mathrm{D}}$		Overall
	N-P	S-P	N-P	S-P	N-P	S-P	
Frozen Predictor	**72.8**	**61.3**	<u>77.7</u>	<u>71.7</u>	49.0	46.8	<u>63.2</u>
w/o weight MLP	65.4	55.4	76.2	71.0	47.5	47.3	60.5
w/o comparison	62.9	52.4	76.3	69.6	49.8	46.9	59.7
w/o projection	62.4	51.4	75.3	69.5	50.2	46.3	59.2
Fresh Predictor	50.5	46.5	67.0	63.1	64.9	53.2	57.5
JEPA-IDM	63.1	53.1	73.6	66.1	<u>66.1</u>	<u>57.1</u>	63.2
Ours	<u>65.9</u>	<u>55.5</u>	**77.8**	**72.4**	**66.2**	**60.9**	**66.5**
w/o action-pair	62.6	52.5	71.6	61.2	49.3	48.4	57.6
w/o video-pair	54.8	49.7	76.8	71.6	48.0	45.9	57.8

Comparison with an inverse-dynamics architecture. To assess whether ACWM-RM's advantage can be explained solely by exposure to preference data, we construct JEPA-IDM with the same frozen V-JEPA 2.1 encoder and predictor backbone, but use action prediction instead of visual-feature prediction. Following the IDM reward formulation, JEPA-IDM uses action-prediction errors as rewards and undergoes action-prediction pretraining on real videos from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, followed by preference training on AVM-Pref. Implementation details are provided in Appendix I. As shown in Table 6, ACWM-RM outperforms JEPA-IDM by 3.1 points on action-mismatch pairs (91.6 vs. 88.5) and 14.9 points on video-mismatch pairs (88.6 vs. 73.7). The larger gap on generated videos supports the benefit of ACWM-RM's feature prediction and scoring design, even when JEPA-IDM has the same backbone and training data and follows the same two-stage training paradigm.

6 Conclusion, Limitations, and Future Work

Through AVM-Eval, we showed that strong action-mismatch discrimination on real videos does not ensure reliable AF assessment of generated videos. To address this limitation while also assessing VQ, we introduced ACWM-RM, a reward model requiring no ground-truth videos for reward computation. ACWM-RM outperforms IDMs on generated-video AF and, when used to fine-tune DreamDojo-2B, yields better VQ and AF than alternative reward signals.

Limitations and future work. ACWM-RM is trained to assess VQ and AF, but does not explicitly supervise physical consistency (PC). Although RL fine-tuning with ACWM-RM maintains PC scores close to those of the pre-RL model, the reward does not directly target physical violations such as interpenetration. Moreover, downstream RL improvements have so far been demonstrated only for DreamDojo-2B on AgiBotWorld. Future work will incorporate physical-consistency supervision and evaluate ACWM-RM for RL optimization across a broader range of ACWMs.

AI USE STATEMENT

We used generative AI tools to polish the prose and check the grammar of author-written text. We also used vision-language models in our experiments: Qwen3-VL provided the VLM-based reward during Flow-GRPO, and GPT-5.6 sol evaluated the physical consistency of generated videos, as described in Sec. 5.1 and Appendix F. We did not use generative AI to generate research ideas. The authors checked each AI-polished passage to ensure that it preserved the intended technical meaning, and reviewed and tested all AI-assisted code. We take responsibility for the final content of this work, including all AI-assisted text, code, and experimental results.

REPRODUCIBILITY STATEMENT

We describe dataset construction and evaluation protocols in Secs. 4.1, 4.2.1, and 5.1. Training configurations and hyperparameters for ACWM-RM, the baseline IDMs, JEPA-IDM, and Flow-GRPO are provided in Appendices D, E, I, and G, respectively. Additional appendices detail action perturbations, preference-pair construction, evaluation prompts, and dataset statistics.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 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.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3549–3556. IEEE, 2025. doi: 10.1109/IROS60139.2025.11247088.

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

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026b.

Yufan Deng, Zilin Pan, Hongyu Zhang, Xiaojie Li, Ruoqing Hu, Yufei Ding, Yiming Zou, Yan Zeng, and Daquan Zhou. Rethinking video generation model for the embodied world. In *Forty-third International Conference on Machine Learning*, 2026.

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

Shibo Hong, Jiahao Ying, Haiyuan Liang, Mengdi Zhang, Jun Kuang, Jiazheng Zhang, and Yixin Cao. Frabench and ufeval: Unified fine-grained evaluation with task and aspect generalization. *arXiv preprint arXiv:2505.12795*, 2025.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

Sizhe Lester Li, Evan Kim, Xingjian Bai, Tong Zhao, Tao Pang, Max Simchowitz, and Vincent Sitzmann. Turning video models into generalist robot policies. *arXiv preprint arXiv:2605.27817*, 2026.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, et al. Genie Envisioner: A unified world foundation platform for robotic manipulation. *arXiv preprint arXiv:2508.05635*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. *Advances in Neural Information Processing Systems*, 38:40783–40818, 2026a.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, et al. Improving video generation with human feedback. *Advances in Neural Information Processing Systems*, 38:82155–82192, 2026b.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. Robotwin: Dual-arm robot benchmark with generative digital twins. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660. IEEE, 2025.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Baorui Peng, Wenyao Zhang, Liang Xu, Zekun Qi, Jiazhao Zhang, Hongsi Liu, Wenjun Zeng, and Xin Jin. Reworld: Multi-dimensional reward modeling for embodied world models. *arXiv preprint arXiv:2601.12428*, 2026.

Gawon Seo, Dongwon Kim, and Suha Kwak. ACID: Action consistency via inverse dynamics for planning with world models. *arXiv preprint arXiv:2607.02403*, 2026.

Yu Shang, Zhuohang Li, Yiding Ma, Weikang Su, Xin Jin, Ziyou Wang, Lei Jin, Xin Zhang, Yinzhou Tang, Haisheng Su, et al. Worldarena: A unified benchmark for evaluating perception and functional utility of embodied world models. *AI Open*, 7:208–226, 2026.

Junhao Shi, Siyin Wang, Xiaopeng Yu, Li Ji, Jingjing Gong, and Xipeng Qiu. Learning to move before learning to do: Task-agnostic pretraining for vlas. *arXiv preprint arXiv:2607.02466*, 2026.

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

Hao Wu, Yuqi Li, Yuan Gao, Fan Xu, Fan Zhang, Kun Wang, Penghao Zhao, Qiufeng Wang, Yizhou Zhao, Weiyan Wang, et al. RoboAlign-R1: Distilled multimodal reward alignment for robot video world models. *arXiv preprint arXiv:2605.03821*, 2026a.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive VideoGPTs are scalable world models. *Advances in Neural Information Processing Systems*, 37:68082–68119, 2024.

Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. RLVR-World: Training world models with reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 125312–125350, 2026b.

Zhuoyuan Wu and Jun Gao. OSCAR: Omni-embodiment action-conditioned world model for robotics. *arXiv preprint arXiv:2606.04463*, 2026.

Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

Yang Ye, Tianyu He, Shuo Yang, and Jiang Bian. Reinforcement learning with inverse rewards for world model post-training. *arXiv preprint arXiv:2509.23958*, 2025.

Chenghao Yin, Da Huang, Di Yang, Jichao Wang, Nanshu Zhao, Chen Xu, Wenjun Sun, Linjie Hou, Zhijun Li, Junhui Wu, Zhaobo Liu, Zhen Xiao, Sheng Zhang, Lei Bao, Rui Feng, Zhenquan Pang, Jiayu Li, Qian Wang, and Maoqing Yao. Genie sim 3.0: A high-fidelity comprehensive simulation platform for humanoid robot. *arXiv preprint arXiv:2601.02078*, 2026.

Dingxi Zhang, Fangjinhua Wang, Marc Pollefeys, and Haofei Xu. Megaflow: Zero-shot large displacement optical flow. *arXiv preprint arXiv:2603.25739*, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: A fine-grained world model for robot manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9834–9844. IEEE, 2025.

A Action Perturbations for Action-Mismatch Pairs

This appendix details how the two classes of action perturbations are performed. Let a clip be $\mathbf{I} = (I_0, \dots, I_{12})$ and the corresponding end-effector states be $\mathbf{s} = (s_0, \dots, s_{12})$. The clip therefore contains 12 consecutive action transitions $\mathbf{a} = (a_1, \dots, a_{12})$. For arm $r \in \{L, R\}$, state s_t^r consists of a position $x_t^r \in \mathbb{R}^3$, an orientation quaternion q_t^r in (q_x, q_y, q_z, q_w) order, and a scalar gripper value γ_t^r in degrees. The action at transition t, which links I_{t-1} to I_t, is

$$a_t = [\Delta x_t^L, \Delta x_t^R, \Delta\omega_t^L, \Delta\omega_t^R, \Delta\gamma_t^L, \Delta\gamma_t^R] \in \mathbb{R}^{14}, \qquad t = 1, \dots, 12, \tag{6}$$

with

$$\Delta x_t^r = x_t^r - x_{t-1}^r, \qquad \Delta\omega_t^r = \mathrm{axisangle}\big(q_t^r \otimes (q_{t-1}^r)^{-1}\big), \qquad \Delta\gamma_t^r = \gamma_t^r - \gamma_{t-1}^r. \tag{7}$$

We refer to each of the six groups Δx_t^r, $\Delta\omega_t^r$ and $\Delta\gamma_t^r$ ($r \in \{L, R\}$) as a channel, so that every action carries the three channel types: translation, rotation and gripper. Throughout, $t \in \{1, \dots, 12\}$ indexes the transitions of a clip and $j \in \{0, \dots, 13\}$ indexes the action dimensions of Eq. 6, so that $a_{t,j}$ is a single scalar of a_t. The clean action chunk is $\mathbf{a}^+ = (a_1, \dots, a_{12}) \in \mathbb{R}^{12\times 14}$. Both mismatch classes retain the same video and use a modified action chunk $\mathbf{a}^-$ as the negative condition. Appendix D.1 writes the same three components as Δp_t, ω_t and Δg_t after normalization, where the gripper is rescaled from degrees to $[0, 1]$.

A.1 Active Channel Identification

ACWMs predict a stationary arm well but a moving arm poorly, so their prediction error is concentrated in the channels in which an arm is moving. These are therefore the channels a reward model has to discriminate, and we restrict perturbations to them.

To identify these channels, we compute six Boolean activity flags, one per channel. Each flag is set from the spread of the absolute trajectory over the 13 frames rather than summed per-step absolute changes, preventing small mean-reverting servo jitter from being mistaken for sustained motion.

For each arm r, compute

$$\begin{aligned} S_{\mathrm{trans}}^r &= \max_{0\le t,t'\le 12} \|x_t^r - x_{t'}^r\|_2, \\ S_{\mathrm{rot}}^r &= \max_{0\le t,t'\le 12} 2\arccos(|\langle q_t^r, q_{t'}^r\rangle|), \\ S_{\mathrm{grip}}^r &= \max_{0\le t\le 12} \gamma_t^r - \min_{0\le t\le 12} \gamma_t^r. \end{aligned} \tag{8}$$

The first quantity is the diameter of the arm's 3D position trajectory, the second is the largest geodesic orientation difference, and the third is the peak-to-peak gripper excursion. The resulting

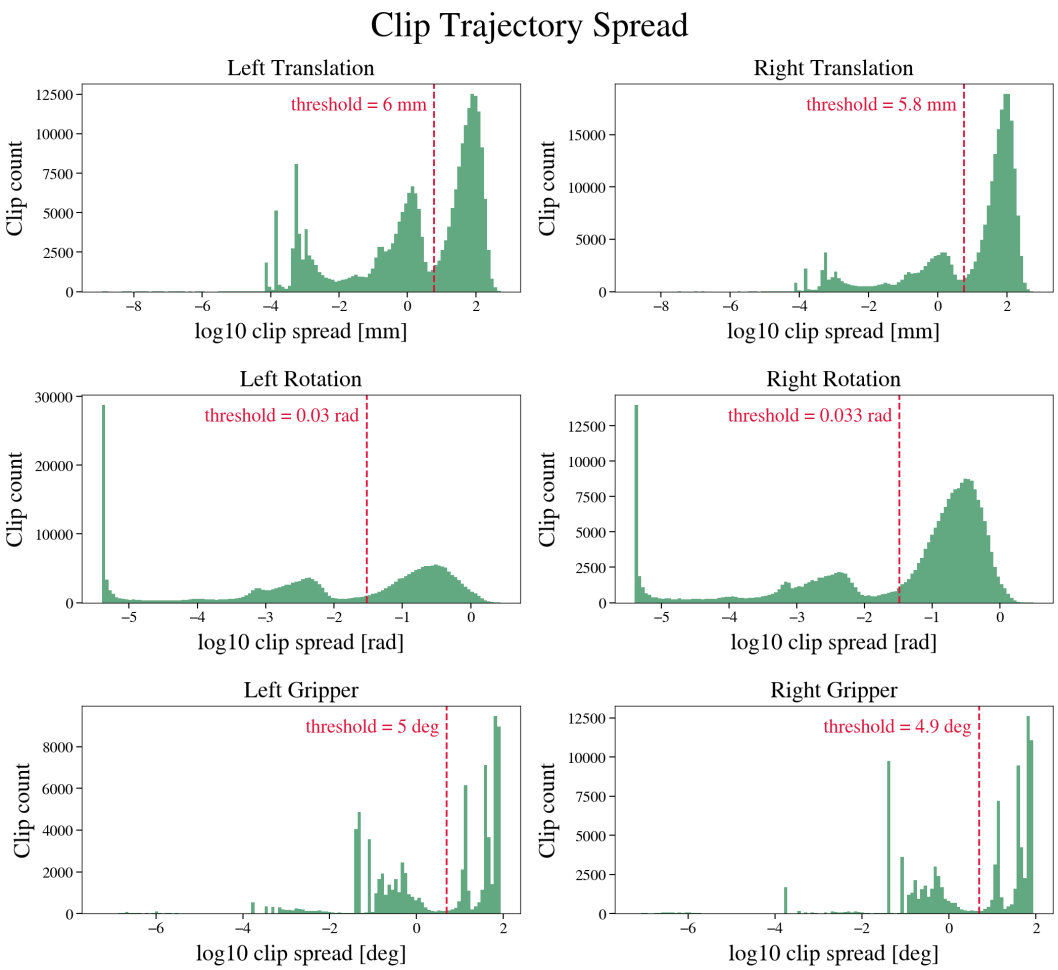


Figure 4: Clip trajectory spread. Each panel histograms the $\log_{10}$ spread statistic of Eq. 8 over the extracted clips, with one column per arm and one row per channel type. The red dashed line marks the threshold calibrated for that channel, placed slightly to the right of the valley separating them.

activity flags are

$$\begin{aligned} \text{active}^r_{\text{trans}} &\iff S^r_{\text{trans}} > \tau^r_{\text{trans}}, \\ \text{active}^r_{\text{rot}} &\iff S^r_{\text{rot}} > \tau^r_{\text{rot}}, \\ \text{active}^r_{\text{grip}} &\iff S^r_{\text{grip}} > \tau_{\text{grip}}. \end{aligned} \tag{9}$$

We calibrate the thresholds offline by collecting a large number of clips and histogramming each spread statistic separately for the left and right arm, as shown in Figure 4. Every histogram is bimodal, with small spreads forming the static and servo-noise population and larger spreads forming the genuine-motion population. Each threshold is placed slightly to the right of the valley separating the two populations, and channels above it are marked active. Biasing the threshold in this direction trades false positives for false negatives: we accept that a few low-amplitude moving channels are missed in order to keep static channels out of the active set. Each arm has its own valley, so the calibration yields six separate values, namely 6 mm and 5.8 mm for left and right translation, 0.03 rad and 0.033 rad for rotation, and 5 and 4.9 degrees for the gripper, held fixed during pair generation.

The flags determine the eligible perturbations. For example, if only the left-translation and right-gripper channels are active, perturbations for that action chunk are drawn only from the left-translation and right-gripper operators. Clips with no active channel are discarded.

A.2 Numerical Action Perturbations

Numerical action perturbations modify $\mathbf{a}^+$ using operators drawn from a perturbation pool whose membership is determined by the six activity flags. We first select the temporal windows within the action chunk to be edited, then assemble the perturbation pool, then apply the drawn perturbations channel by channel, and finally check whether the negative differs sufficiently from the positive.

Temporal Window Sampling. Perturbing every action would make the negative wrong everywhere, which is both easy to detect and unrepresentative of the localized errors an ACWM actually makes. We therefore confine each edit to a few windows of consecutive actions and leave the remaining actions clean. In addition, many different combinations of windows and perturbation operators remain available, so a single action chunk can yield many distinct negatives.

The windows themselves are sampled in one of two ways. With probability 0.3, the whole chunk is taken as a single window. Retaining these fully perturbed negatives keeps a coarse difficulty level in the mix. Otherwise, we sample one to three windows, each covering one to seven consecutive actions, and require neighboring windows to be separated by at least one unedited action so that the edited segments stay distinguishable. If the sampled windows cannot all be placed within the 12 actions under this constraint, we drop windows until the remaining ones fit. In either case, the outcome is a set of windows $\mathcal{W} = \{[w_k^{\mathrm{lo}}, w_k^{\mathrm{hi}}]\}$ marking the actions to be edited, where k indexes the windows and w_k^{lo}, w_k^{hi} are the first and last actions of window k. The actions outside them stay clean.

Perturbation Pool Construction and Sampling. To enlarge the range of perturbation types, we build a perturbation pool for each action chunk to be perturbed. For every arm with an active translation flag, we draw one translation perturbation with equal probability from *Magnitude*, *Direction*, and *Magnitude & Direction*, and add it to the pool. For every arm with an active rotation flag, we draw one rotation perturbation with equal probability from *Orientation Direction*, *Orientation Magnitude*, and *Orientation Magnitude & Direction*, and add it to the pool. For every active gripper flag, we add one *Gripper Perturbation* operator to the pool. The pool further holds a global *Gaussian Perturbation* whenever at least one channel is active, together with a global *Temporal Order Permutation* operator. Each perturbation in the pool is then selected independently with probability 0.5, and if no perturbation is selected, we select one uniformly from the pool, so that every negative carries at least one edit. We initialize $\mathbf{a}^- = \mathbf{a}^+$ before applying the selected perturbations sequentially. All selected perturbations are applied one after another to all sampled windows to produce a single $\mathbf{a}^-$, with the structured perturbations writing their own action dimensions, Gaussian noise added on top, and order permutation applied last.

Perturbation Types. We describe each perturbation type below:

1. **Translation Perturbations.** Translation perturbations act on the translation deltas $\Delta x_t^{r,+}$ and write the perturbed deltas $\Delta x_t^{r,-}$. They comprise the following three types of perturbations.

   (1) *Magnitude.* For each window, draw $\mu \sim U[0.4, 0.8]$ or $\mu \sim U[1.2, 1.6]$ and set

   $$\Delta x_t^{r,-} = \mu\, \Delta x_t^{r,+}, \qquad t \in [w_k^{\mathrm{lo}}, w_k^{\mathrm{hi}}]. \tag{10}$$

   The direction is preserved, but the motion is under- or over-executed.

   (2) *Direction.* Compute the arm's net translation $m^r = \sum_t \Delta x_t^{r,+}$. When $\|m^r\|_2 \geq 10^{-6}$, sample a random unit axis $v^r \perp m^r$ and draw a signed $\eta \sim U[0.15, 0.5]$ radians per window. Then

   $$\Delta x_t^{r,-} = \mathrm{Rot}(v^r, \eta)\, \Delta x_t^{r,+}, \tag{11}$$

   where Rot is Rodrigues rotation. Each action keeps its length but points in a different direction.

   (3) *Magnitude & Direction.* Apply both operations:

   $$\Delta x_t^{r,-} = \mu\, \mathrm{Rot}(v^r, \eta)\, \Delta x_t^{r,+}. \tag{12}$$

   For an active translation channel with $\|m^r\|_2 < 10^{-6}$, both *Direction* and *Magnitude & Direction* fall back to *Magnitude*.

2. **Rotation Perturbations.** Rotation perturbations act on the rotation deltas $\Delta\omega_t^{r,+}$, whose norm is the turn amount and whose direction is the turn axis. The three types are:

   (1) *Orientation Direction.* For each arm with an active rotation channel, define the aggregate rotation direction $m_{\text{rot}}^r = \sum_{t=1}^{12} \Delta\omega_t^{r,+}$ over the entire clip. When $\|m_{\text{rot}}^r\|_2 \geq 10^{-6}$, sample a random unit axis $v^r \perp m_{\text{rot}}^r$ and apply the following perturbation only within the sampled windows:

   $$\Delta\omega_t^{r,-} = \text{Rot}(v^r, \eta)\, \Delta\omega_t^{r,+}, \qquad \eta \sim U[0.3, 0.8]. \tag{13}$$

   When an axis is available, the turn amount is preserved, but the turn axis is changed.

   (2) *Orientation Magnitude.* Draw $\mu \sim U[0.5, 0.8]$ or $\mu \sim U[1.2, 1.5]$ and set

   $$\Delta\omega_t^{r,-} = \mu\, \Delta\omega_t^{r,+}. \tag{14}$$

   The axis is preserved, but the turn amount is too small or too large.

   (3) *Orientation Magnitude & Direction.* Apply both the axis change and the magnitude scale:

   $$\Delta\omega_t^{r,-} = \mu\, \text{Rot}(v^r, \eta)\, \Delta\omega_t^{r,+}. \tag{15}$$

   For an active rotation channel with $\|m_{\text{rot}}^r\|_2 < 10^{-6}$, both *Orientation Direction* and *Orientation Magnitude & Direction* fall back to *Orientation Magnitude*.

3. **Gripper, Gaussian, and Temporal Order Perturbations.**

   (1) *Gripper Perturbation.* Draw $\mu \sim U[0.3, 0.7]$ or $\mu \sim U[1.3, 2.0]$ per window and set

   $$\Delta\gamma_t^{r,-} = \mu'\, \Delta\gamma_t^{r,+}, \qquad \mu' = \begin{cases} -\mu, & \text{with probability } 0.25, \\ \mu, & \text{otherwise.} \end{cases} \tag{16}$$

   A sign flip changes closing into opening or opening into closing.

   (2) *Gaussian Perturbation.* Let $a_{t,j}^-$ denote the current action value after any preceding structured perturbations. We draw $\sigma_w \sim U[0.25, 2.5]$ for each window and add independent noise to every active dimension:

   $$a_{t,j}^- \leftarrow a_{t,j}^- + \epsilon_{t,j}, \qquad \epsilon_{t,j} \sim \mathcal{N}\big(0, (\sigma_w\, \sigma_{\text{act},j})^2\big)\,. \tag{17}$$

   Here $\sigma_{\text{act},j}$ is the empirical standard deviation of dimension j on clips whose corresponding channel is active.

   (3) *Temporal Order Permutation.* Within each window containing at least two actions, reversal, random shuffling, or a cyclic roll reorders only active dimensions; inactive dimensions remain unchanged. This preserves the active values after any preceding perturbations and, when applied alone, changes only their temporal order. Permutations that leave the window unchanged are discarded and resampled.

Filtering Degenerate Negatives. Applying the selected perturbations does not by itself guarantee a usable negative. A perturbation drawn near the mild end of its range, or applied to a window in which the arm barely moves, can leave $\mathbf{a}^-$ almost identical to $\mathbf{a}^+$ while the pair is still labeled a mismatch. We therefore keep a pair only when it is far enough from the positive.

Measuring this distance requires care on two counts. The 14 action dimensions of Eq. 6 carry different units, so the differences $a_{t,j}^+ - a_{t,j}^-$ cannot be summed across dimensions. And the perturbations apply only to the active channels inside the sampled windows, leaving most of these differences at zero, so an average taken over all positions would be diluted by the clean part. We address the first by dividing $a_{t,j}^+ - a_{t,j}^-$ by σ_j, the empirical standard deviation of dimension j over all transitions of the extracted clips,

$$\begin{aligned} \sigma = \big(&\underbrace{0.00316,\ 0.00397,\ 0.00372}_{\text{left trans. [m]}},\ \underbrace{0.00335,\ 0.00504,\ 0.00417}_{\text{right trans. [m]}}, \\ &\underbrace{0.01535,\ 0.01566,\ 0.01552}_{\text{left rot. [rad]}},\ \underbrace{0.01622,\ 0.01437,\ 0.01608}_{\text{right rot. [rad]}},\ \underbrace{3.950,\ 4.287}_{\text{grippers [deg]}}\big), \end{aligned} \tag{18}$$

measured at the 8 fps clips. We address the second by averaging only over the positions (t, j) at which $a^+_{t,j} \neq a^-_{t,j}$, so that the score reflects the average displacement of each edited action rather than the number of actions edited. Dividing per dimension by σ, the resulting distance is

$$\mathrm{dist}(\mathbf{a}^+, \mathbf{a}^-) = \frac{\|(\mathbf{a}^+ - \mathbf{a}^-)/\sigma\|_F}{\sqrt{\#\{(t,j) : a^+_{t,j} \neq a^-_{t,j}\}}}. \tag{19}$$

A pair is kept only if $\mathbf{a}^-$ differs from $\mathbf{a}^+$ and $\mathrm{dist}(\mathbf{a}^+, \mathbf{a}^-) \geq 0.12$. We set this floor from the activity thresholds of Sec. A.1. A channel only counts as active once its 13-frame trajectory spread exceeds τ^r, which corresponds to an average per-transition motion of $\tau^r/12$, or $\tau^r/(12\sigma)$ once expressed in the units of Eq. 19. Across the six channels, this quantity ranges from 0.095 to 0.177 with a median of 0.126, and we round it to 0.12. Negatives whose average per-transition change falls below this level are therefore discarded. The floor deserves particular attention when *Temporal Order Permutation* is applied alone: although it preserves the positive action values, changing their temporal order does not guarantee a sufficiently large action mismatch. When the reordered window moves at near-constant velocity, however, those actions return to almost the same places, so the distance can still fall below the floor and the pair is discarded. A rejected $\mathbf{a}^-$ triggers a fresh draw of the windows, the perturbation pool, and all magnitudes, for at most five attempts, after which the clip yields no pair and is discarded.

A.3 Semantic Action Perturbations

Semantic action perturbations differ from numerical action perturbations in what they edit and where the edits are applied. The semantic procedure never perturbs $\mathbf{a}^+$ directly, but edits the absolute state sequence $\mathbf{s}$ and obtains $\mathbf{a}^-$ by differencing the edited states through Eq. 7. The perturbation is anchored at a grasp, release, or mid-motion event rather than at an individual action, so the error corresponds to a semantic-level mistake. These events can be identified from the gripper events: a gripper closing marks a grasp, a gripper opening marks a release, and the interval between a closing and the following opening marks mid-motion drift or freeze. We therefore locate these events and anchor the semantic perturbations to them.

Applying perturbation at a gripper event requires more context than a single clip provides. We therefore subsample the episode to 8 fps, detect the events and apply the perturbation over this whole subsampled episode, and crop the 13-frame clip afterward.

Gripper Event Detection. We first describe how gripper closing and opening events are extracted from an episode. Let $\gamma^r_{\max}$ and $\gamma^r_{\min}$ be the largest and smallest values it takes over the whole episode, γ^r_t be the gripper opening of arm r, and $\rho^r = \gamma^r_{\max} - \gamma^r_{\min}$ be its range. If $\rho^r < 5$ degrees, the gripper never opens or closes appreciably. Otherwise set

$$\gamma^r_{\mathrm{lo}} = \gamma^r_{\min} + 0.30\rho^r, \qquad \gamma^r_{\mathrm{hi}} = \gamma^r_{\min} + 0.70\rho^r. \tag{20}$$

Here γ^r_{lo} and γ^r_{hi} are the closing and opening thresholds for arm r, placed at 30% and 70% of that arm's observed gripper range, so that the band between them spans the middle 40%. A single pass over the episode gives every frame an open or closed flag. Specifically, a frame is labeled close when its γ^r falls below γ^r_{lo} and open when it rises above γ^r_{hi}, while frames lying between the two thresholds are left unlabeled. Whenever two consecutive labeled frames carry opposite labels, the interval between them is marked as an event, which is a gripper close event when the labels run from open to close and a gripper open event in the reverse case. Because the thresholds are 40% apart, fluctuations near one of them cannot produce an event. Then, for the mid-motion events, we scan each arm's γ^r over the episode and mark a maximal interval as mid-motion if γ^r stays below γ^r_{lo} throughout while the end-effector displacement across the interval exceeds τ^r_{trans}. (Intervals shorter than 25 frames are discarded.) Finally, the detected three events are used by the three semantic perturbations described below.

Perturbation Types. We describe each perturbation type below:

1. **Position offset.** This type covers *grasp offset*, anchored at a gripper close event, and *place offset*, anchored at a gripper open event. The two variants use the same construction, so we describe only one of them here. Let b be the first frame of the event interval, at which the gripper starts to

change state, and r be the arm on which it was detected. We sample a window length L from 10 to 25 frames and set $w_{\mathrm{hi}} = b$ and $w_{\mathrm{lo}} = \max(b-L, 0)$, so the window covers the motion leading up to the event rather than the gripper transition itself. We then take the arm's net displacement over the window, $m^r = x^{r,+}_{w_{\mathrm{hi}}} - x^{r,+}_{w_{\mathrm{lo}}}$, sample a random unit direction $v^r \perp m^r$ as in Eq. 11, and draw an amplitude $\delta \sim U[0.005, 0.050]$. The arm's positions are shifted along v^r by an offset that grows linearly over the window and is then held to the end of the episode, while frames before w_{lo} stay unchanged,

$$x^{r,-}_t = x^{r,+}_t + \frac{t - w_{\mathrm{lo}}}{w_{\mathrm{hi}} - w_{\mathrm{lo}}}\,\delta v^r, \quad t \in [w_{\mathrm{lo}}, w_{\mathrm{hi}}], \qquad x^{r,-}_t = x^{r,+}_t + \delta v^r, \quad t > w_{\mathrm{hi}}. \tag{21}$$

By the frame at which the gripper begins to close or open, the arm has already been displaced by the full δ, so the grasp or release is performed at the wrong position.

2. **Gripper timing.** This type covers four variants, obtained by pairing either gripper event with either direction of shift: *early close*, *late close*, *early open*, and *late open*. Let $[b, e]$ denote the event interval and r the arm on which it was detected. We first sample a shift magnitude d from 3 to 16 and set $\zeta = +d$ for a late variant and $\zeta = -d$ for an early one, which moves the event to $[b + \zeta, e + \zeta]$. The gripper values on $[b, e]$ are copied to $[b + \zeta, e + \zeta]$, preserving the event's gripper-value profile while shifting it in time. The gap between the original and the shifted interval is filled with a constant gripper value. When the event is delayed, the gap lies before the shifted interval and is filled with $\gamma^{r,+}_b$, the gripper value at the first index of the original interval. When the event is advanced, the gap lies after the shifted interval and is filled with $\gamma^{r,+}_e$, the gripper value at the last index of the original interval. For a late variant ($\zeta = +d$),

$$\gamma^{r,-}_t = \begin{cases} \gamma^{r,+}_{t-\zeta}, & b + \zeta \le t \le e + \zeta, \\ \gamma^{r,+}_b, & b \le t < b + \zeta, \\ \gamma^{r,+}_t, & \text{otherwise}, \end{cases} \tag{22}$$

and for an early variant ($\zeta = -d$),

$$\gamma^{r,-}_t = \begin{cases} \gamma^{r,+}_{t-\zeta}, & b + \zeta \le t \le e + \zeta, \\ \gamma^{r,+}_e, & e + \zeta < t \le e, \\ \gamma^{r,+}_t, & \text{otherwise}. \end{cases} \tag{23}$$

The edit spans the original and shifted intervals and any gap between them; gripper states outside this span remain unchanged. When d is smaller than the duration of the event, the two intervals overlap, and the second case of each equation covers only the part of the original interval that the shifted one does not. The constant values used to fill the gap are taken from the corresponding boundaries of the original event interval. The copied event profile is preserved, but continuity is not guaranteed at the joins with the unedited gripper trajectory. These variants change grasp or release timing: the gripper closes before or after the object is in place, or opens before or after the arm has reached the target.

3. **Mid-motion drift or freeze.** This type covers *mid drift* and *mid freeze*, both anchored at a mid-motion event. Let $[b, e]$ be the event interval and r be the arm on which it was detected. We set $w_{\mathrm{lo}} = b$ and $w_{\mathrm{hi}} = e$, so the window covers the transport motion itself. With probability 0.5 we apply the ramped offset of Eq. 21 to this window, with m^r, v^r and δ obtained as before, so the arm drifts sideways while it carries the object. With probability 0.5 we instead freeze the arm at the position it holds at w_{lo},

$$x^{r,-}_t = \begin{cases} x^{r,+}_{w_{\mathrm{lo}}}, & w_{\mathrm{lo}} \le t \le w_{\mathrm{hi}}, \\ x^{r,+}_t - m^r, & t > w_{\mathrm{hi}}, \\ x^{r,+}_t, & \text{otherwise}, \end{cases} \qquad m^r = x^{r,+}_{w_{\mathrm{hi}}} - x^{r,+}_{w_{\mathrm{lo}}}. \tag{24}$$

After the window, the arm moves exactly as in the original trajectory, but from a position that lags it by m^r, the net displacement it should have made while frozen. Both variants edit translation only, leaving orientation and gripper values unchanged. The arm therefore fails to carry the object along the demonstrated path, either drifting off course while transporting it or not transporting it at all.

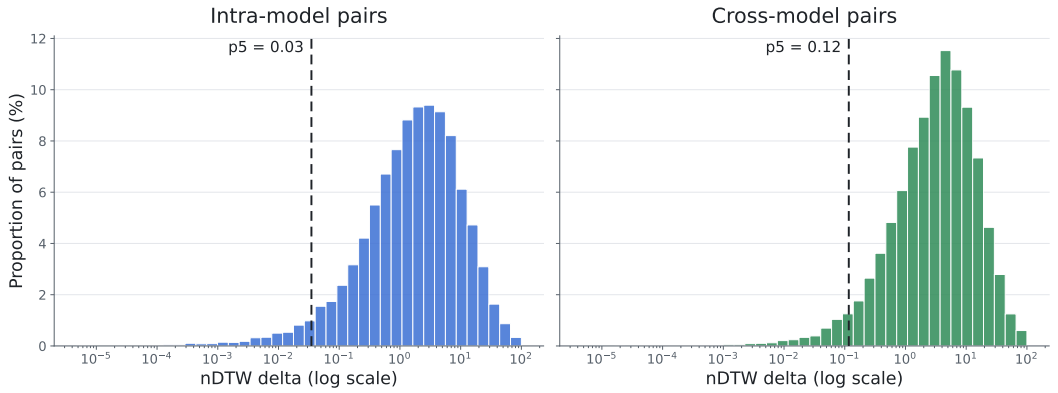


Figure 5: Distributions of absolute nDTW score differences ($\times 100$) for intra-model (left) and cross-model (right) candidate pairs from 5,760 rollouts conditioned on clips from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ and $\mathrm{Agi}_{\mathrm{B}}$. The horizontal axes use a logarithmic scale. Black vertical lines mark the respective 5th-percentile thresholds, 0.03 and 0.12; pairs below the corresponding threshold are discarded.

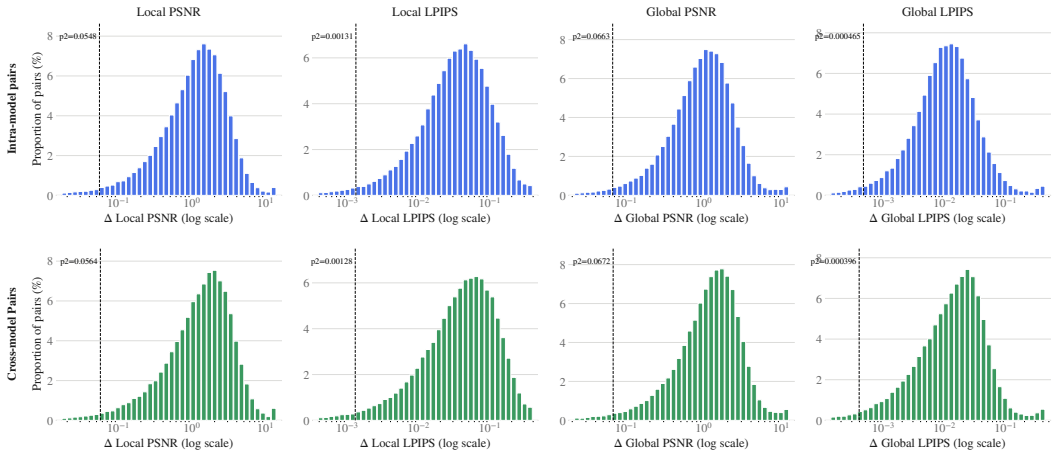


Figure 6: Visual-quality gap distributions for intra-model (top) and cross-model (bottom) candidate pairs from AgiBotWorld. Black dashed lines mark the 2nd-percentile thresholds.

Pair Construction and Filtering. After cropping the 13-frame clip, we take the difference between the clean and the edited state sequences through Eq. 7 to obtain the two action chunks $\mathbf{a}^+$ and $\mathbf{a}^-$. We keep the pair only if at least one of the 12 actions differs between them and the normalized distance of Eq. 19 is at least 0.12. The first condition ensures that the crop overlaps the edited part of the episode, while the actions outside it stay clean and serve as context. The resulting training pair is $(\mathbf{I}, \mathbf{a}^+) \succ (\mathbf{I}, \mathbf{a}^-)$.

B Video-mismatch pair filtering for AVM-Eval

B.1 nDTW-Based Pair Filtering

We compute nDTW scores using the official implementation of WorldArena (Shang et al., 2026) and filter candidate video pairs with small score differences. The threshold statistics are collected from rollouts conditioned on 240 input clips selected from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ and $\mathrm{Agi}_{\mathrm{B}}$, covering 30 tasks and 120 episodes, with eight clips per task. For each clip, DreamDojo-14B, DreamDojo-2B, GE-Sim-2B, and OSCAR-2B each generate six videos under the same initial frame and action chunk, giving $240 \times 4 \times 6 = 5{,}760$ rollouts.

For each input clip, we enumerate all $\binom{6}{2} = 15$ pairs within each model and all $6 \times 6 = 36$ pairs for each of the $\binom{4}{2} = 6$ model combinations. This gives 14,400 intra-model and 51,840 cross-model candidate pairs for the threshold statistics. Let s_i and s_j denote the nDTW scores ($\times 100$) of two candidate videos relative to the same ground-truth video. We compute their absolute score difference as

$$\Delta_{ij} = |s_i - s_j|. \tag{25}$$

We pool these differences across input clips separately for intra-model and cross-model pairs, and set each filtering threshold to the 5th percentile of its corresponding distribution. As shown in Figure 5, the resulting thresholds are $\tau_{\text{intra}} = 0.03$ and $\tau_{\text{cross}} = 0.12$. We retain an intra-model pair only when $\Delta_{ij} \geq \tau_{\text{intra}}$ and a cross-model pair only when $\Delta_{ij} \geq \tau_{\text{cross}}$, assigning the video with the higher score as preferred. This empirical filtering rule excludes approximately the lowest 5% of pairwise score differences in each distribution, removing candidate pairs with similar nDTW scores.

B.2 Visual Quality Control

To limit the influence of visual quality on AF evaluation, we further filter the pairs retained above using global and local PSNR and LPIPS, computed against the shared ground-truth video (Appendix C). To calibrate the thresholds, we construct candidate pairs from AgiBotWorld and order each pair by the quality metric being considered. For each metric m and pair type c (intra-model or cross-model), we set $\tau_{m,c}$ to the 2nd percentile of the resulting positive quality gaps (Figure 6).

For the pairs retained by nDTW filtering, we keep their preference labels fixed and define the signed quality gap δ_m as positive-minus-negative for PSNR and negative-minus-positive for LPIPS. We retain a pair only if $\delta_m \leq \tau_{m,c}$ for all four metrics. This allows the higher-nDTW video to have lower or approximately equal visual quality under these empirical thresholds.

C Video-Mismatch Pair Construction for AVM-Pref

This appendix details the four visual quality metrics and pair selection procedure used to construct video-mismatch pairs for AVM-Pref. A clip consists of a ground-truth (GT) video $\mathbf{I} = (I_0, \ldots, I_K)$ and a generated video $\hat{\mathbf{I}} = (\hat{I}_0, \ldots, \hat{I}_K)$, each with one initial frame and K future frames. The initial frame serves as the conditioning input to the ACWM and is therefore shared, $\hat{I}_0 = I_0$. The two global metrics assign one value to each future frame $t = 1, \ldots, K$ and average these values over the clip. The two local metrics assign one value to each valid transition $t : I_{t-1} \to I_t$, using the optical flow between the two frames to determine the regions to score; these values are then averaged over the valid transitions of the clip.

The four metrics comprise two *global* metrics that evaluate the entire frame—global PSNR $\text{PSNR}^{\text{glob}}$ (higher is better) and global LPIPS $\text{LPIPS}^{\text{glob}}$ (lower is better)—and two *local* metrics that evaluate only the regions containing motion—local PSNR PSNR^{loc} (higher is better) and local LPIPS $\text{LPIPS}^{\text{loc}}$ (lower is better).

C.1 Global Visual Quality Metrics

Both global metrics compare each generated frame with the corresponding GT frame and average over all K future frames. Unlike the local metrics below, they have a value on every future frame, so the average always runs over the full set $\{1, \ldots, K\}$.

Global PSNR. For frame t, MSE is computed over all pixels and RGB channels:

$$\text{MSE}_t = \frac{1}{3HW} \sum_{k=1}^{3} \sum_{i=1}^{H} \sum_{j=1}^{W} \big(I_t[k,i,j] - \hat{I}_t[k,i,j]\big)^2, \qquad \text{PSNR}_t^{\text{glob}} = 10 \log_{10} \frac{255^2}{\text{MSE}_t}, \tag{26}$$

with the convention $\text{PSNR}_t^{\text{glob}} = 99$ whenever $\text{MSE}_t \leq 10^{-12}$, so that an exactly reproduced frame yields a finite score. The global PSNR of the clip is

$$\text{PSNR}^{\text{glob}} = \frac{1}{K} \sum_{t=1}^{K} \text{PSNR}_t^{\text{glob}}. \tag{27}$$

Global LPIPS. We use the official LPIPS (Zhang et al., 2018) implementation with the AlexNet backbone, loaded from a fixed local checkpoint so that every clip in the dataset is scored by identical weights. Frames are converted to RGB, scaled to $[0, 1]$, and then affinely mapped once onto the network input range $[-1, 1]$. Writing d_{LPIPS} for the distance this network produces, the global LPIPS of the clip is

$$\mathrm{LPIPS}^{\mathrm{glob}} = \frac{1}{K} \sum_{t=1}^{K} d_{\mathrm{LPIPS}}\big(\hat{I}_t,\, I_t\big). \tag{28}$$

C.2 Motion Region Extraction

Action-induced changes are confined to the local regions where the robot interacts with objects, and these are also the regions an ACWM is most likely to render incorrectly. Because the two global metrics average over the whole frame, the surrounding static background dilutes this evidence, making them less sensitive to the failures that matter most. The two local metrics therefore restrict their evaluation to these local regions. These regions are located from optical flow, estimated independently on the GT and on the generated video, so that the local metrics penalize both motion the ACWM fails to reproduce and motion it invents.

Optical Flow. We estimate optical flow with MegaFlow (Zhang et al., 2026). For each video, we run the estimator on every pair of adjacent frames, obtaining fields:

$$\Phi_t = \mathrm{MegaFlow}(I_{t-1}, I_t) \in \mathbb{R}^{2\times H\times W}, \qquad \widehat{\Phi}_t = \mathrm{MegaFlow}(\hat{I}_{t-1}, \hat{I}_t), \qquad t = 1, \ldots, K. \tag{29}$$

Motion Bounding Box Extraction. The optical flow field Φ_t maps pixels from frame $t-1$ to frame t. It is defined on the grid of frame $t-1$, where each vector $\Phi_t[:, i, j]$ gives the displacement of pixel (i, j) between the two frames. We identify moving pixels by thresholding the displacement magnitude at $\tau = 2$ pixels:

$$S_t = \big\{(i, j) :\ \|\Phi_t[:, i, j]\|_2 > \tau\big\}. \tag{30}$$

Thus, S_t contains the source positions of moving pixels in frame $t-1$. Their corresponding destination positions in frame t are

$$S_t^{\rightarrow} = \Big\{\big(\mathrm{clip}(i + \lfloor \Phi_t^{\mathrm{v}}[i, j] \rceil),\, \mathrm{clip}(j + \lfloor \Phi_t^{\mathrm{u}}[i, j] \rceil)\big) :\ (i, j) \in S_t\Big\}, \tag{31}$$

where Φ_t^{u} and Φ_t^{v} denote the horizontal and vertical components of the optical flow at time t, respectively. Here, $\lfloor \cdot \rceil$ denotes rounding to the nearest integer, and clip restricts each coordinate to the valid range of the corresponding image axis. Using the same construction, we obtain the pixel sets $\widehat{S}_t$ and $\widehat{S}_t^{\rightarrow}$ for the generated video from its optical flow field $\widehat{\Phi}_t$.

Since the resulting pixel sets may contain isolated or spatially scattered pixels, we group neighboring pixels into rectangular regions so that each local score corresponds to a coherent moving structure. Specifically, for a pixel set S, let $\mathrm{Box}(S)$ denote the collection of bounding boxes of its connected components, with one box assigned to each component. We discard any box whose area is smaller than 5×10^{-4} of the total frame area. The boxes extracted from the GT and generated videos are then merged frame by frame:

$$\mathcal{B}_t = \mathrm{Box}(S_t) \cup \mathrm{Box}(\widehat{S}_t), \qquad \mathcal{B}_t^{\rightarrow} = \mathrm{Box}(S_t^{\rightarrow}) \cup \mathrm{Box}(\widehat{S}_t^{\rightarrow}), \tag{32}$$

where $\mathcal{B}_t$ is defined on frame $t-1$ and $\mathcal{B}_t^{\rightarrow}$ on frame t. Including boxes from both videos captures complementary errors: GT boxes identify motion that the ACWM fails to reproduce, whereas generated-video boxes identify motion that it introduces spuriously.

C.3 Local Visual Quality Metrics

Scored Regions. For each transition t, we evaluate only the bounding boxes constructed above. The boxes in $\mathcal{B}_t$ are evaluated on the source frame $t-1$, and those in $\mathcal{B}_t^{\rightarrow}$ on the destination frame t. The box-level evaluations can be recorded as

$$\mathcal{P}_t = \big\{(b, t-1) : b \in \mathcal{B}_t\big\} \cup \big\{(b, t) : b \in \mathcal{B}_t^{\rightarrow}\big\}, \tag{33}$$

where each pair specifies one box and the frame on which it is evaluated. For each metric, we first aggregate the box-level values within $\mathcal{B}_t$ and within $\mathcal{B}_t^{\rightarrow}$ separately, and then aggregate these two set-level values into the score of transition t. For each transition, we compute local metrics over the available bounding boxes, omitting any empty box set from aggregation. For $t = 1$, the shared initial frame is excluded from local scoring, so only boxes in $\mathcal{B}_1^{\rightarrow}$ contribute. We determine the valid transition set $\mathcal{T}_{\text{valid}}$ separately for each GT-generated video comparison, skipping t if neither video provides a valid scored box and discarding the clip only if this set is empty. Thus, a transition with no valid motion region in either video is omitted, whereas spurious motion in the generated video is still evaluated even when the GT is stationary.

Local PSNR. For box b in frame f, we compute PSNR over all enclosed pixels:

$$\begin{aligned}
\mathrm{PSNR}\big(b, f\big) &= 10 \log_{10} \frac{255^2}{\frac{1}{3|b|} \sum_{k=1}^{3} \sum_{(i,j) \in b} \big(I_f[k,i,j] - \hat{I}_f[k,i,j]\big)^2}, \\
\mathrm{PSNR}_t^{\mathrm{src}} &= \min_{b \in \mathcal{B}_t} \mathrm{PSNR}(b, t-1), \\
\mathrm{PSNR}_t^{\rightarrow} &= \min_{b \in \mathcal{B}_t^{\rightarrow}} \mathrm{PSNR}(b, t), \\
\mathrm{PSNR}_t^{\mathrm{loc}} &= \min\big(\mathrm{PSNR}_t^{\mathrm{src}}, \mathrm{PSNR}_t^{\rightarrow}\big),
\end{aligned} \tag{34}$$

As with global PSNR, we set the box-level PSNR to 99 whenever the corresponding MSE is at most 10^{-12}. The local PSNR of the clip is

$$\mathrm{PSNR}^{\mathrm{loc}} = \frac{1}{|\mathcal{T}_{\text{valid}}|} \sum_{t \in \mathcal{T}_{\text{valid}}} \mathrm{PSNR}_t^{\mathrm{loc}}. \tag{35}$$

Local LPIPS. For each box b evaluated on frame f, we crop the corresponding regions from both videos and, where necessary, pad each spatial dimension to at least 31 pixels, the minimum required by the AlexNet feature extractor used in LPIPS. Writing $I_f|_b$ for the padded crop extracted from frame f using box b,

$$\begin{aligned}
\mathrm{LPIPS}\big(b, f\big) &= d_{\mathrm{LPIPS}}\big(\hat{I}_f|_b,\, I_f|_b\big), \\
\mathrm{LPIPS}_t^{\mathrm{src}} &= \max_{b \in \mathcal{B}_t} \mathrm{LPIPS}(b, t-1), \\
\mathrm{LPIPS}_t^{\rightarrow} &= \max_{b \in \mathcal{B}_t^{\rightarrow}} \mathrm{LPIPS}(b, t), \\
\mathrm{LPIPS}_t^{\mathrm{loc}} &= \max\big(\mathrm{LPIPS}_t^{\mathrm{src}}, \mathrm{LPIPS}_t^{\rightarrow}\big).
\end{aligned} \tag{36}$$

The local LPIPS of the clip is

$$\mathrm{LPIPS}^{\mathrm{loc}} = \frac{1}{|\mathcal{T}_{\text{valid}}|} \sum_{t \in \mathcal{T}_{\text{valid}}} \mathrm{LPIPS}_t^{\mathrm{loc}}. \tag{37}$$

C.4 Pair Selection

A pair is retained only if the positive video strictly beats the negative one on all four metrics: higher global and local PSNR and lower global and local LPIPS. Pairs that do not satisfy all four inequalities are discarded.

We aim for comparable pair counts across models and tasks for intra-model pairs, and across model combinations and tasks for cross-model pairs; the resulting distributions are reported in Appendix H, as shown in Figure 9.

D Reward Model Implementation Details

For dynamics pretraining, preference training, and IDM training, we hold out internal validation subsets from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$ for checkpoint selection and exclude them from parameter optimization. These internal validation subsets are distinct from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$, which is used for AVM-Eval evaluation and does not participate in checkpoint selection.

D.1 DYNAMICS PRETRAINING

This appendix specifies the input processing, predictor architecture, and dynamics-pretraining setup. The input processing and predictor architecture are shared by both training stages.

Input Representation. We use 13-frame clips sampled at 8 fps, with one initial frame and $K = 12$ future frames. Let $\mathbf{I} = (I_0, \ldots, I_K)$ denote a clip and $\mathbf{a} = (a_1, \ldots, a_K)$ its associated action chunk, which consists of relative end-effector actions for both arms. The action a_t corresponds to the transition from frame I_{t-1} to frame I_t. Let $\mathbf{s} = (s_0, \ldots, s_{K-1})$ denote the corresponding state sequence, where s_t records the end-effector states of both arms at frame I_t. For each arm $r \in \{L, R\}$, $a_t^r \in \mathbb{R}^7$ comprises a 3D translation increment, a 3D rotation-vector increment, and a gripper-opening increment. The absolute end-effector state $s_t^r \in \mathbb{R}^{10}$ comprises a 3D position, a continuous 6D rotation representation, and the gripper state. For predictor input, we group components by arm and write $a_t = (a_t^L, a_t^R)$ and $s_t = (s_t^L, s_t^R)$. This arm-wise action layout differs from Eq. 6, which groups the two arms' translations, rotations, and gripper increments by component type. The two action layouts are related by a fixed permutation of components; this permutation changes their arrangement, not their values. The coordinate transformations and normalization used for predictor input are described below.

We first transform the absolute end-effector position and orientation components of each arm's state into the head-camera coordinate frame. For each arm, let p_t and $Q_t \in \mathrm{SO}(3)$ denote the transformed position and orientation. The state stores p_t and the first two columns of Q_t as its position and 6D orientation representation, respectively. We then compute the translation and rotation components of the action from consecutive transformed states. Translation increments are $\Delta p_t = p_t - p_{t-1}$. Rotation increments use left composition, $\Delta Q_t = Q_t Q_{t-1}^\top$, and are encoded as rotation vectors ω_t in radians. For the gripper, we normalize the opening w_t as $g_t = \mathrm{clip}(w_t/100, 0, 1)$. The state stores the absolute normalized opening g_t, while the action stores its increment $\Delta g_t = g_t - g_{t-1}$.

We then normalize the robot inputs by scaling the translation components of both arms' actions by 65.48 and the rotation-vector components by 11.80, leaving the gripper command unchanged. These coefficients are the reciprocals of the median per-step translation magnitude (0.015271 m) and rotation magnitude (0.084724 rad) computed from the training trajectories. This scaling brings the median magnitudes of both types of action increments to approximately one. We apply the same translation scaling factor of 65.48 to the XYZ components of the absolute states, since state positions and action translation increments are expressed in the same head-camera coordinate frame and share the same physical units (meters). The states' 6D orientation representations consist of two orthonormal columns of a rotation matrix, whose dimensionless entries lie in $[-1, 1]$. These representations already have a fixed scale, so we retain them without additional scaling. Gripper states are already normalized to $[0, 1]$, and their increments lie in $[-1, 1]$, so neither needs rescaling.

Each full frame is resized to 384×384 using bilinear interpolation, without cropping. We encode frames independently with the frozen V-JEPA 2.1 ViT-G/16 encoder, which has 48 layers, hidden dimension $D = 1664$, and approximately 1.84B parameters. With a patch size of 16×16, each frame produces $N = 24 \times 24 = 576$ tokens of dimension D. Layer normalization gives the per-frame target features

$$z_t = \mathrm{LN}(E(I_t)) \in \mathbb{R}^{N \times D}. \tag{38}$$

To form the predictor input tokens, we map the normalized states, normalized actions, and visual features to the predictor's hidden dimension $d_P = 1024$ using separate affine projections f_s, f_a, and f_z, respectively. Let $\bar{s}_t^r \in \mathbb{R}^{10}$ and $\bar{a}_{t+1}^r \in \mathbb{R}^7$ denote the normalized state and action inputs for arm r. We then add learnable arm-identity embeddings $e_L, e_R \in \mathbb{R}^{d_P}$ to the corresponding projected state and action features, yielding

$$u_{t,r}^s = f_s(\bar{s}_t^r) + e_r, \qquad u_{t,r}^a = f_a(\bar{a}_{t+1}^r) + e_r. \tag{39}$$

Here, $u_{t,r}^s, u_{t,r}^a \in \mathbb{R}^{d_P}$ denote embedded tokens. The same identity embedding is added to both token types for a given arm. The visual token block is

$$U_t^z = f_z(z_t) \in \mathbb{R}^{N \times d_P}, \tag{40}$$

where f_z is applied independently to each visual patch. Within each time block, we place the left-arm state and action tokens before the right-arm state and action tokens, followed by the visual

tokens, giving

$$\mathbf{X}_t = \left[u^s_{t,L}, u^a_{t,L}, u^s_{t,R}, u^a_{t,R}, U^z_t\right]. \tag{41}$$

Concatenating these blocks in temporal order gives the full predictor input

$$\mathbf{X} = \left[\mathbf{X}_0,\ \ldots,\ \mathbf{X}_{K-1}\right]. \tag{42}$$

Each time block contains two state tokens, two action tokens, and N visual tokens.

Predictor Architecture. The predictor has 24 Transformer layers, hidden dimension $d_P = 1024$, 16 attention heads per layer, and approximately 0.31B parameters. Block-causal attention lets a time block access itself and earlier blocks, but no future-frame blocks. Visual patches can therefore attend to both arms' controls and the preceding visual history. After the Transformer blocks, conditioning tokens are discarded and predicted visual tokens are projected to dimension $D = 1664$ for comparison with the encoder features.

Training Objectives. We combine teacher-forcing and rollout losses, following the training scheme of V-JEPA 2-AC. Under teacher forcing, the input blocks $\mathbf{X}_0, \ldots, \mathbf{X}_{t-1}$ contain encoded features of the observed frames rather than previous model predictions. From these blocks, P_ϕ produces a prediction for frame I_t, which is layer-normalized to obtain $\widehat{z}^{\mathrm{TF}}_t \in \mathbb{R}^{N\times D}$. The corresponding encoder feature z_t serves as the supervision target. For the rollout loss, we first predict $\widehat{z}^{\mathrm{roll}}_1$ from $\mathbf{X}_0$. We then predict $\widehat{z}^{\mathrm{roll}}_2$ using $\mathbf{X}_0$ and $\mathbf{X}_1$, replacing only U^z_1 in $\mathbf{X}_1$ with $f_z(\widehat{z}^{\mathrm{roll}}_1)$. Both rollout predictions are layer-normalized, and gradients pass through the first prediction. The equally weighted losses compare these predictions with the encoder targets z_t,

$$\mathcal{L}_{\mathrm{dyn}}(\phi) = \underbrace{\frac{1}{KND}\sum_{t=1}^{K}\left\|\widehat{z}^{\mathrm{TF}}_t - z_t\right\|_1}_{\mathcal{L}_{\mathrm{TF}}} + \underbrace{\frac{1}{2ND}\sum_{t=1}^{2}\left\|\widehat{z}^{\mathrm{roll}}_t - z_t\right\|_1}_{\mathcal{L}_{\mathrm{roll}}}. \tag{43}$$

Only the predictor is updated, and its learned weights initialize P_ϕ for preference training.

Training Setup. We train for 120 epochs, sampling one clip per episode in each epoch, with a global batch of 256 clips (16 per GPU on 16 GPUs). Based on the reported 34k episodes, this corresponds to approximately 4.08M clip draws and 15.94k optimizer steps. The learning rate warms up from 7.5×10^{-5} to 3×10^{-4} over the first 10% of training steps, remains constant for the next 45%, and decays linearly to 10^{-6} over the final 45%. Both training stages use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.04 (excluding biases and one-dimensional parameters), gradient-norm clipping at 1.0, and bfloat16 mixed precision. After completing the prescribed pretraining budget, we select the checkpoint with the lowest loss $\mathcal{L}_{\mathrm{dyn}}$ on the internal validation subset of $\mathrm{Agi}^{\mathrm{train}}_{\mathrm{A}}$ to initialize preference training.

D.2 Preference Training

We initialize P_ϕ from dynamics pretraining and randomly initialize h_ψ, keeping E frozen. The clip length, input representation, and predictor architecture follow Appendix D.1. We expand on Sec. 4.2.2 by detailing the conditioning states, score head architecture, preference objective, and training setup.

Conditioning States. Following the main-text notation, we represent each candidate as $q = (\mathbf{I}', \mathbf{a}', \mathbf{s}')$, where $\mathbf{I}'$ is a real or generated video, $\mathbf{a}' = (a'_1, \ldots, a'_K)$ is the original or perturbed action chunk, and $\mathbf{s}' = (s'_0, \ldots, s'_{K-1})$ is the corresponding state sequence. For both original and perturbed actions, we construct $\mathbf{s}'$ by integrating the candidate's own actions from the shared initial state s_0:

$$s'_0 = s_0, \qquad s'_t = F(s'_{t-1}, a'_t), \quad t = 1, \ldots, K-1, \tag{44}$$

where F updates the end-effector state using the relative action under the input conventions in Appendix D.1. For action-mismatch pairs, the two candidates share the same real video and initial state, but each uses its own original or perturbed action chunk to construct its conditioning states. This keeps the states consistent with the actions and avoids conditioning the negative candidate on states obtained from the positive actions. For video-mismatch pairs, the candidates instead contain

different generated videos but share the original action chunk and initial state, so they also share the integrated conditioning states.

We use the same state construction during Flow-GRPO on $\mathrm{Agi}_{\mathrm{B}}$, taking the initial state and action chunk from the AgiBotWorld recordings. The world model is conditioned on $(I_0, \mathbf{a})$, and each generated video is scored with $\mathbf{I}' = \hat{\mathbf{I}}$, $\mathbf{a}' = \mathbf{a}$, and the states $\mathbf{s}'$ integrated from s_0 and $\mathbf{a}'$. All candidates sampled under the same condition therefore share the actions and conditioning states. ACWM-RM combines these inputs with the candidate's own visual history and evaluated frames, without requiring ground-truth future frames.

In our offline experiments, actions are derived from differences between recorded end-effector states, so integrating them from s_0 recovers the recorded trajectory. The conditioning states therefore contain trajectory information already encoded in the actions and initial state. Here, *reference-free* means that scoring requires no ground-truth future video or separately supplied future measured states.

Score Head Architecture. Here we specify the dimensions along the path from a pair of patch features to the reward score. The shared projection $W_{\mathrm{proj}} \in \mathbb{R}^{d \times D}$ maps both branches from $D = 1664$ down to $d = 256$ and is bias-free, so the five terms of Equation 3, four of dimension d and one scalar cosine similarity, concatenate into a comparison vector of dimension $4d + 1 = 1025$. Both MLPs of Equation 4 read this vector and share the architecture $1025 \to 256 \to 1$ with a GELU activation between the two affine layers, differing only in whether their output serves as a patch score or a patch weight. Together, W_{proj} and the two MLPs form h_ψ with 951,810 randomly initialized parameters. ACWM-RM has approximately 2.15B parameters in total, including the 1.84B frozen encoder, the 0.31B predictor, and the score head. During preference training, only the pretrained predictor and the randomly initialized score head are updated, giving approximately 0.31B trainable parameters. Gradients reach W_{proj} through both branches, while the encoded features are detached so that no gradient enters E. We use the unbounded scalar output of Equation 5 directly as the reward, without a final sigmoid. The Bradley-Terry loss in Equation 45 applies the sigmoid to the score difference $R^+ - R^-$ rather than to either candidate score individually.

Preference Objective. Let q^+ and q^- denote the preferred and dispreferred candidates, respectively, with each candidate containing the video, actions, and conditioning states defined above. We score both candidates with shared parameters and jointly optimize P_ϕ and h_ψ using

$$\mathcal{L}_{\mathrm{BT}}(\phi, \psi) = -\mathbb{E}_{(q^+, q^-) \sim \text{AVM-Pref}} \left[\log \sigma(R^+ - R^-) \right], \tag{45}$$

where σ is the sigmoid function and $R^\pm = R_{\phi,\psi}(q^\pm)$. For numerical stability, the per-pair loss is implemented as $\mathrm{softplus}(-(R^+ - R^-))$.

Training Setup. We use the shared optimizer settings in Appendix D.1, with a global batch of 256 preference pairs (16 per GPU on 16 GPUs). Training runs for 3 epochs, corresponding to approximately 1.52M preference-pair draws, 3.04M candidate-clip draws, and 5.93k optimizer steps based on the reported 506.3k pairs. The head learning rate warms up from 10^{-5} to 4×10^{-4} over the first 5% of training steps, remains constant for the next 20%, and decays to 4×10^{-5} with a cosine schedule over the final 75%. The predictor learning rate is 0.05 times the head rate throughout training, including warmup. After completing the prescribed preference-training budget, we select the checkpoint with the highest pairwise accuracy on internal validation pairs from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$.

D.3 Training Budget Accounting

We report nominal training budgets in 13-frame clip draws, including repeated samples, based on corpus sizes before internal validation holdout. Each pretraining clip contributes one draw, while each preference pair contributes two candidate-clip draws. For each IDM, one draw likewise corresponds to a complete clip covering all 12 action transitions; predictions within a clip are not counted as separate draws. Pretraining samples one clip per episode per epoch, so 120 epochs over 34k episodes yield $120 \times 34{,}000 = 4.08\mathrm{M}$ clip draws. Preference training traverses 506.3k pairs for 3 epochs, yielding $3 \times 506{,}300 = 1{,}518{,}900$ pair draws, or $3{,}037{,}800 \approx 3.04\mathrm{M}$ candidate-clip draws. Together, the two stages have a nominal budget of

$$120 \times 34{,}000 + 3 \times 506{,}300 \times 2 = 7{,}117{,}800 \approx 7.12\mathrm{M}$$

Table 8: Architectures and observation-action interfaces. Frame counts are per prediction within a 13-frame training clip.

	EVA-IDM	GR00T-IDM	TAP-IDM	J-IDM
Visual backbone	ResNet-50	SigLIP2-L/16	Qwen2.5-VL-3B	VGGT
Backbone initialization	pretrained	pretrained	pretrained	pretrained
Input resolution	512×512	224×224	224×224	128×128
Frames/prediction	3	2	2	2
Action dim for reward	(14)	(14)	(14)	(14)
Objective	smooth-L_1	flow matching	CE	Charbonnier flow
Action recovery	direct	16-step Euler	detokenize	damped least sq

training clip draws, approximately matching the nominal budget of each IDM in Table 9. Actual draw and step counts depend on the held-out subsets and final partial batches. This accounting matches nominal clip-draw budgets and temporal coverage, not computational cost or the number of unique videos or frames seen by each model.

E Training Setup for the Baseline IDMs

We train four baseline IDMs on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, starting from the pretrained backbone checkpoints used in their official implementations. Specifically, EVA-IDM uses pretrained ResNet-50, GR00T-IDM uses pretrained SigLIP2-L/16, TAP-IDM uses pretrained Qwen2.5-VL-3B, and J-IDM uses pretrained VGGT. For each baseline, we follow its official implementation with the bimanual action-interface adaptations described below; the resulting architectures, observation-action interfaces, and hyperparameters are summarized in Tables 8 and 9. TAP-IDM retains its original LM head and shared action-token vocabulary, configuring the FAST tokenizer to jointly decode 14-dimensional bimanual actions. J-IDM uses a 14-dimensional command space and a 28-channel per-pixel Jacobian head, jointly recovering both arms' actions through ridge-regularized inversion. GR00T-IDM retains its 32-dimensional output with 20 valid components and padding masked during training; before reward computation, padding is discarded, and each arm's 6D rotation is converted via quaternions to a 3D rotation-vector increment, yielding the 14-dimensional action layout in Eq. 6. Its action horizon is set to 12, matching the 12 transitions in each 13-frame clip.

For each IDM, we set a nominal budget of approximately 7.12M training clip draws, comparable to ACWM-RM's combined budget (Appendix D.3). Each sampled clip contains 13 frames and 12 action transitions. We apply each IDM's native observation window across the clip, with all 12 actions contributing to its training loss. After completing this budget, we select each IDM's checkpoint by the lowest mean distance between its predicted actions and the ground-truth actions on the internal validation subset of $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$.

Action Standardization and IDM Reward Computation. For the IDM-based Flow-GRPO baseline, we standardize both the predicted and conditioning actions using per-dimension statistics estimated from action samples drawn from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$. We compute the empirical mean μ_j and standard deviation σ_j separately for each of the 14 camera-frame delta-action dimensions and keep these statistics fixed during reward computation. Table 10 reports the standard deviations used in the IDM reward. Following the notation in Section 3, $\tilde{a}_{k,j}$ and $a_{k,j}$ denote the predicted and conditioning actions at transition $k \in \{1, \ldots, K\}$ and dimension $j \in \{1, \ldots, 14\}$. Their standardized values and residuals are

$$\begin{aligned} \tilde{\bar{a}}_{k,j} &= \frac{\tilde{a}_{k,j} - \mu_j}{\sigma_j + 10^{-8}}, & \bar{a}_{k,j} &= \frac{a_{k,j} - \mu_j}{\sigma_j + 10^{-8}}, \\ e_{k,j} &= \tilde{\bar{a}}_{k,j} - \bar{a}_{k,j} = \frac{\tilde{a}_{k,j} - a_{k,j}}{\sigma_j + 10^{-8}}. \end{aligned} \tag{46}$$

The same statistics are applied to both actions, so the mean cancels in the residual.

Table 9: IDM optimization settings. Each model uses 8 GPUs; batch sizes count complete 13-frame clips. Optimizer steps are set to target a nominal budget of approximately 7.12M clip draws per model. Step and draw counts are rounded.

	EVA-IDM	GR00T-IDM	TAP-IDM	J-IDM
Batch / GPU	32	16	16	16
GPUs	8	8	8	8
Global batch	256	128	128	128
Learning rate	3.0×10^{-4}	1.0×10^{-4}	5.0×10^{-5}	5.0×10^{-5}
Steps	27.8k	55.6k	55.6k	55.6k
Clip draws	7.12M	7.12M	7.12M	7.12M
Optimizer	AdamW	AdamW	AdamW	AdamW
Weight decay	–	1×10^{-5}	0.05	1×10^{-5}
Betas	–	$(0.95, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.99)$
Schedule	cosine	cosine	cosine	constant
Warmup	5%	5%	1k steps	–
Precision	bf16	bf16	bf16	fp16
Gradient clip	–	–	–	1.0

Table 10: Per-dimension action standard deviations used for IDM reward computation, estimated from AgiBotWorld.

Action component	Unit	Left arm σ_j	Right arm σ_j
X translation	m	0.003746	0.004671
Y translation	m	0.003482	0.003662
Z translation	m	0.003115	0.003379
X rotation	rad	0.014987	0.013658
Y rotation	rad	0.013690	0.014198
Z rotation	rad	0.015895	0.015860
Gripper-opening change	–	0.078766	0.084997

We then apply the elementwise Smooth-L1 loss with threshold 1 and average over all transitions and action dimensions:

$$\ell_{\mathrm{SL1}}(e) = \begin{cases} \frac{1}{2}e^2, & |e| < 1, \\ |e| - \frac{1}{2}, & |e| \geq 1, \end{cases}$$
$$R^{\mathrm{IDM}} = -\frac{1}{14K}\sum_{k=1}^{K}\sum_{j=1}^{14}\ell_{\mathrm{SL1}}(e_{k,j}). \tag{47}$$

Here, K is the number of action transitions, with $K = 12$ for the 13-frame clips used in our experiments. We average the elementwise Smooth-L1 losses over all (K) transitions and 14 action dimensions with equal weights. The reward scale is 1.0, yielding one nonpositive scalar per video; values closer to zero indicate better action agreement.

Dividing by the per-dimension standard deviation expresses translation, rotation, and gripper errors on a common dimensionless scale relative to their variability in $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$. Smooth-L1 penalizes small standardized residuals quadratically and large residuals linearly, reducing the influence of large errors relative to a squared-error loss.

F Prompt and Criteria for Evaluation

When evaluating ACWM-generated videos for visual artifacts and physical inconsistencies, or using the VLM-based rewards to train the ACWM with Flow-GRPO, we use the evaluation prompt from RBench (Deng et al., 2026) shown in Table 11. The prompt produces three criteria: Object Consistency (OC), Manipulator Consistency (MC), and Anomaly Check (AC), each rated on an integer scale from 1 to 5.

> You are a strict evaluator for a short generated robot-manipulation video. The input is a 13-frame clip; the first frame is the conditioning observation and the remaining 12 frames are generated. Given the task description, inspect every frame and score only the visible evidence. Evaluate: (i) *object consistency*, checking whether the manipulated object preserves its identity, shape, structure, color, material, and count; (ii) *manipulator consistency*, checking whether the robot, arms, hands, and grippers preserve their identity, geometry, connectivity, number of parts, and plausible articulation; and (iii) *anomaly check*, detecting floating or penetrating entities, teleportation, spontaneous appearance or disappearance, duplication, unsupported attachment, and false grasps. Assign each requested dimension a score from 1 (severe failure) to 5 (fully plausible and stable), using only information visible in the clip. Do not evaluate task completion or the final goal. Return a JSON object containing the scores and a brief evidence-based reason for each dimension.

Table 11: Prompt used by Qwen3-VL to compute rewards during Flow-GRPO and by GPT-5.6 sol to evaluate the physical consistency of videos generated by DreamDojo after Flow-GRPO fine-tuning.

For the VLM-based rewards, we normalize each score to $[0, 1]$ and average the three normalized scores with equal weights to obtain the scalar reward used by Flow-GRPO:

$$S_{\mathrm{VLM}} = \frac{1}{3}\left(\frac{\mathrm{OC}-1}{4} + \frac{\mathrm{MC}-1}{4} + \frac{\mathrm{AC}-1}{4}\right).$$

The three criteria are defined as follows:

- *Object Consistency* measures whether the target object preserves its identity and attributes throughout the video, including its shape, appearance, material, and count.
- *Manipulator Consistency* measures the temporal stability of the robot embodiment, including its morphology, link or manipulator connectivity, joint articulation, and structural stability.
- *Anomaly Check* measures physical and commonsense violations, such as floating or unsupported entities, interpenetration, spontaneous appearance or disappearance, duplication, non-contact attachment, and incorrect grasping.

G Flow-GRPO Optimization Details

We fine-tune the policy using LoRA with rank 32 on 16 NVIDIA H800 GPUs. We optimize the LoRA parameters with AdamW at a learning rate of 3×10^{-4} for 120 outer training iterations. Each rollout uses 20 Flow-GRPO denoising steps and a diffusion noise level of 0.7. Each sampled episode from $\mathrm{Agi}_{\mathrm{B}}$ (Table 13) supplies a shared input condition—the initial frame I_0 and action chunk $\mathbf{a}$—for a group of $G = 8$ rollouts. In each outer iteration, we collect four rollout batches with a per-GPU sampling batch size of 8 episodes and group size G, yielding $8 \times 8 = 64$ rollouts per batch and 256 rollouts per GPU per outer iteration. We then perform four inner PPO-style policy-optimization epochs on the collected data and accumulate gradients over 16 rollouts. The PPO-style ratio clipping range is 10^{-3}, the KL penalty coefficient is $\beta = 0.004$, and gradients are clipped to a maximum norm of 1.0. We use seed 42 for policy optimization. ACWM-RM is frozen throughout policy optimization and uses the conditioning state sequence integrated from the initial state and each sampled action chunk.

For evaluation on $\mathrm{Agi}_{\mathrm{C}}$, we generate videos from each optimized ACWM using five video-generation seeds and report the mean and standard deviation across seeds for visual quality, action-following fidelity, and physical consistency.

H Dataset and Preference Pair Statistics

Source datasets. Tables 12, 13, and 14 list the tasks and episode counts in $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, $\mathrm{Agi}_{\mathrm{B}}$, and $\mathrm{Agi}_{\mathrm{C}}$, respectively. Tables 16, 17, and 15 report the corresponding statistics for $\mathrm{Robo}_{\mathrm{T}}$, Gen, and $\mathrm{Robo}_{\mathrm{D}}$ used in AVM-Eval.

Action-mismatch pairs. Tables 18 and 19 summarize pair counts in AVM-Eval and AVM-Pref by numerical perturbation type and semantic perturbation group, respectively. Figure 7 shows the distribution of the number of numerical perturbation types per pair, together with the number of distinct pairs N and the mean number of types per pair $\bar{k}$.

Video-mismatch pairs. Figure 9 reports AVM-Pref statistics on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$. Figures 8 and 10 report AVM-Eval statistics on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ and $\mathrm{Agi}_{\mathrm{B}}$, respectively. Each figure breaks down intra-model and cross-model pairs by generating model or model pair and by task. Task counts in Table 1 refer to the source datasets, whereas the task distributions below include only tasks with retained video-mismatch pairs.

Table 12: Task-level statistics of $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, comprising 36 tasks and 33,981 episodes used for IDM training and ACWM-RM predictor pretraining.

ID	Task	Episodes	ID	Task	Episodes
327	Pickup items in the supermarket	203	377	Packing in e-commerce	790
352	Open the fridge to get food.	1,380	378	Clear table in the restaurant	732
354	Pickup items in the supermarket	507	380	Packing in e-commerce	813
356	Packing in the supermarket	473	384	Insert a book into the bookshelf	805
357	Wash dishes with dishwasher	425	385	Pickup in the supermarket produce section	214
358	Toast bread	469	388	Pickup in the supermarket	89
359	Sort in the warehouse	640	389	Pickup in the supermarket	307
360	Packing in the supermarket	513	390	Checkout and scan barcode in the supermarket	747
361	Flatten shorts	2,544	392	Brush water bottle	956
362	Fold shorts	8,194	398	Sort clothes	173
365	Sort personal care products	501	410	Water Pouring in Restaurant	1,083
366	Sort food	726	414	Hang clothes with hanger	2,043
367	Take toast from toaster	257	421	Pick up the item to wipe away the stain.	1,645
368	Cook vegetables with oven	577	422	Pack items for industrial logistics	2,078
372	Packing in the supermarket	534	424	Clear the countertop waste	910
374	Sort laundry and personal care products	91	440	Iron clothes	1,082
375	Brew tea	419	446	Transport table with another robot	72
376	Sort electronic products	402	475	Ironing Clothes	587
Total: 36 tasks					**33,981**

Table 13: Task-level statistics of $\mathrm{Agi}_{\mathrm{B}}$, comprising 30 tasks and 10,000 episodes used for AVM-Eval construction and Flow-GRPO fine-tuning.

ID	Task	Episodes	ID	Task	Episodes
351	Open the wardrobe and hang the clothes.	608	463	Clean the microwave and range hood.	454
373	Sweep the floor	583	464	Replace the toilet paper roll	428
428	Open drawer and store items	467	465	Wash clothes in the washing machine.	329
429	Install memory module	24	466	Separate dark and light clothes	440
431	Brew coffee with a capsule machine	360	468	Remove clothes from the drying rack and place them in the basket.	454
434	Remove capsules from the coffee machine.	258	471	Organize the condiments on the stove.	294
444	Fold short sleeve	175	477	Fold towels on the table	214
445	Open the fridge to get fruits and vegetables.	454	478	Wipe the mirror cabinet	454
451	Make a salad	151	480	Make sandwiches with salad dressing 1213 edition	454
452	Slice a lemon to make lemon water.	278	483	Place items from the meeting room table into the storage box.	223
453	Make a sandwich	83	486	Remove bottled water from the carton and arrange it neatly on the table.	195
454	Make a sandwich	116	487	Remove bottled water from the carton and arrange neatly on the table.	260
455	Serve meals	285	491	Iron clothes	267
460	Remove the pillowcase from the clothesline and place it in the basket.	454	492	Pack in the supermarket cashier	454
462	Discard the trash bag into the large bin.	330	497	Restock in the supermarket	454
Total: 30 tasks					**10,000**

Table 14: Task-level statistics of $\mathrm{Agi_C}$, comprising 10 tasks and 1,499 episodes used to evaluate ACWM rollouts after Flow-GRPO fine-tuning.

ID	Task	Episodes
363	Open the drawer to store clothes.	150
369	Remove clothes from the washing machine	150
425	Prepare oatmeal porridge	150
433	Make instant coffee	150
438	Place the pen into the pen holder.	150
470	Tidy the tables at the milk tea shop.	149
474	Arrange flowers	150
485	Pick up the scrap paper on the desk and feed it into the shredder for destruction.	150
494	Organize the kitchen counter seasonings	150
498	Make milk tea	150
Total: 10 tasks		**1,499**

Table 15: Task-level statistics of $\mathrm{Robo_D}$, comprising 35 tasks and 1,450 real-world episodes used to construct action-mismatch pairs in AVM-Eval.

Task	Episodes	Task	Episodes
arrange_largest_number	47	pack_objects_into_box	43
build_tower	48	play_Xylophone	45
classify_objects	53	play_stacking_toy	44
cover_blocks	53	play_tic_tac_toe	41
deposit_coin	42	plug_in_charger	41
dlc	43	pour_balls_into_vase	48
fasten_screws	31	pour_liquid_into_cup	48
fill_egg_holder	49	press_by_number	41
fill_pen_holder	43	push_T	39
fold_clothes	42	put_bottles_into_dustbin	46
hang_mugs	30	sort_nesting_dolls_by_size	43
imitate_sorting_sequence	35	stack_blocks	15
insert_key	36	stack_bowls	48
insert_tubes	42	store_laptop_and_headphones	42
make_kong	39	swap_T	14
make_toast	41	swap_blocks	45
match_and_pick_from_conveyor	44	sweep_blocks	45
organize_table	44		
Total: 35 tasks			**1,450**

Table 16: Task-level statistics of $\mathrm{Robo_T}$, comprising 50 tasks and 2,166 simulated episodes used to construct action-mismatch pairs in AVM-Eval.

Task	Episodes	Task	Episodes
adjust_bottle	33	place_can_basket	40
beat_block_hammer	37	place_cans_plasticbox	42
blocks_ranking_rgb	57	place_container_plate	43
blocks_ranking_size	49	place_dual_shoes	41
click_alarmclock	11	place_empty_cup	50
click_bell	11	place_fan	53
dump_bin_bigbin	46	place_mouse_pad	44
grab_roller	31	place_object_basket	44
handover_block	32	place_object_scale	42
handover_mic	52	place_object_stand	51
hanging_mug	45	place_phone_stand	43
lift_pot	45	place_shoe	53
move_can_pot	55	press_stapler	57
move_pillbottle_pad	57	put_bottles_dustbin	36
move_playingcard_away	41	put_object_cabinet	38
move_stapler_pad	41	rotate_qrcode	50
open_laptop	52	scan_object	44
open_microwave	56	shake_bottle	33
pick_diverse_bottles	42	shake_bottle_horizontally	33
pick_dual_bottles	40	stack_blocks_three	43
place_a2b_left	44	stack_blocks_two	59
place_a2b_right	48	stack_bowls_three	43
place_bread_basket	47	stack_bowls_two	47
place_bread_skillet	41	stamp_seal	54
place_burger_fries	47	turn_switch	23
Total: 50 tasks			**2,166**

Table 17: Task-level episode counts for Gen: 19 tasks and 2,254 simulated episodes.

Task	Episodes	Task	Episodes
Pick block by number	170	Pick block by color (camnoise)	78
Pick block by shape	122	Pick block by color (camocgen)	114
Pick object by type	136	Pick block by color (camposgen)	128
Pick specific object	112	Pick block by color (ctrlgen)	112
Pick billiard ball by color	66	Pick block by color (eefgen)	98
Pick block by color	122	Pick block by color (jointgen)	188
Pick block by color (instructgen)	104	Pick block by color (lightgen)	98
Pick block by size	88	Pick block by color (matgen)	128
Pick block by color (basegen)	138	Pick block by color (ref)	120
Pick block by color (camdropgen)	132		
Total: 19 tasks			**2,254**

Table 18: Counts of action-mismatch pairs by numerical perturbation type (Appendix A.2). Each pair is counted once per type, so totals across types may exceed the number of distinct pairs.

Group	Operator	AVM-Eval					AVM-Pref
		$\text{Agi}_{\text{A}}^{\text{test}}$	Agi_{B}	Robo_{T}	Robo_{D}	Gen	$\text{Agi}_{\text{A}}^{\text{train}}$
Translation perturbations	Magnitude	930	2,241	718	472	266	12,020
	Direction	910	2,315	818	494	242	12,021
	Magnitude & Direction	963	2,277	742	473	262	12,283
Rotation perturbations	Orientation Direction	938	2,371	823	452	182	12,014
	Orientation Magnitude	955	2,158	774	433	134	12,033
	Orientation Magnitude & Direction	943	2,167	828	457	160	12,142
Gripper, Gaussian, and temporal order perturbations	Gripper Perturbation	915	1,905	1,919	992	556	11,874
	Gaussian Perturbation	2,549	6,648	1,155	612	504	33,274
	Temporal Order Permutation	2,385	6,289	2,445	1,330	378	31,628
Distinct pairs		**4,905**	**12,504**	**4,429**	**2,616**	**1,220**	**63,522**

Table 19: Counts of action-mismatch pairs by semantic perturbation group (Appendix A.3). Each pair is counted once per group; "–" denotes no available pairs.

Group	AVM-Eval					AVM-Pref
	$\text{Agi}_{\text{A}}^{\text{test}}$	Agi_{B}	Robo_{T}	Robo_{D}	Gen	$\text{Agi}_{\text{A}}^{\text{train}}$
Position offset	3,655	3,429	2,540	2,119	1,084	46,345
Gripper timing	1,728	1,738	2,810	1,503	–	23,272
Mid-motion drift or freeze	1,806	1,810	1,641	1,245	1,094	23,194
Distinct pairs	**7,189**	**6,977**	**6,991**	**4,867**	**2,178**	**92,811**

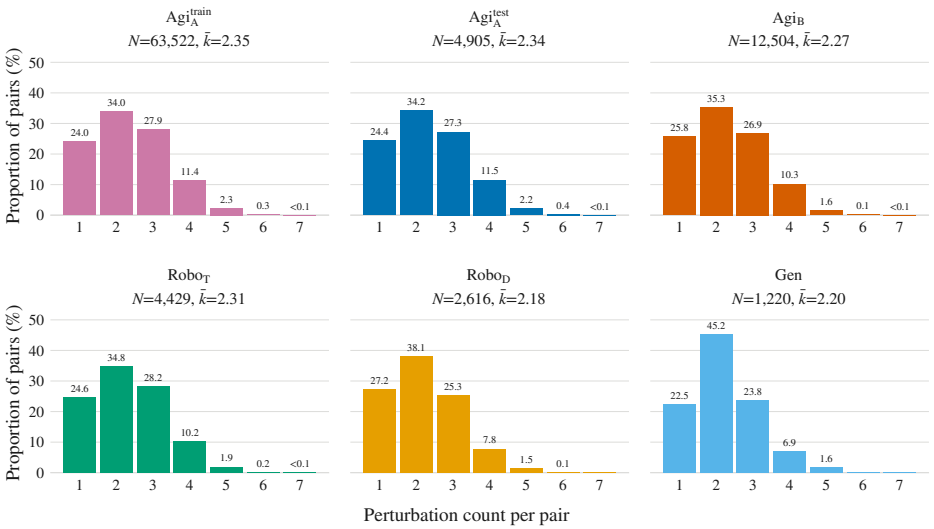


Figure 7: Distribution of numerical perturbation counts per action-mismatch pair. $\text{Agi}_{\text{A}}^{\text{train}}$ shows AVM-Pref; all other panels show AVM-Eval. N is the number of distinct numerical-perturbation pairs, and $\bar{k}$ is the mean number of distinct perturbation types per pair.

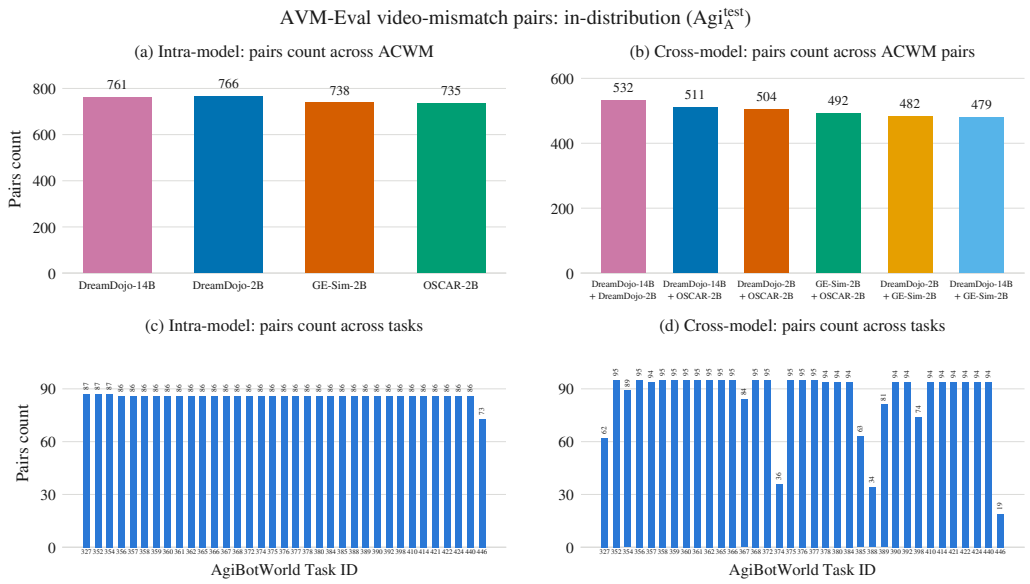


Figure 8: Video-mismatch pair statistics for AVM-Eval on $\text{Agi}_{\text{A}}^{\text{test}}$, with 3,000 intra-model and 3,000 cross-model pairs. Panels (a) and (b) show counts by ACWM and ACWM pair, respectively; (c) and (d) show the corresponding task distributions.

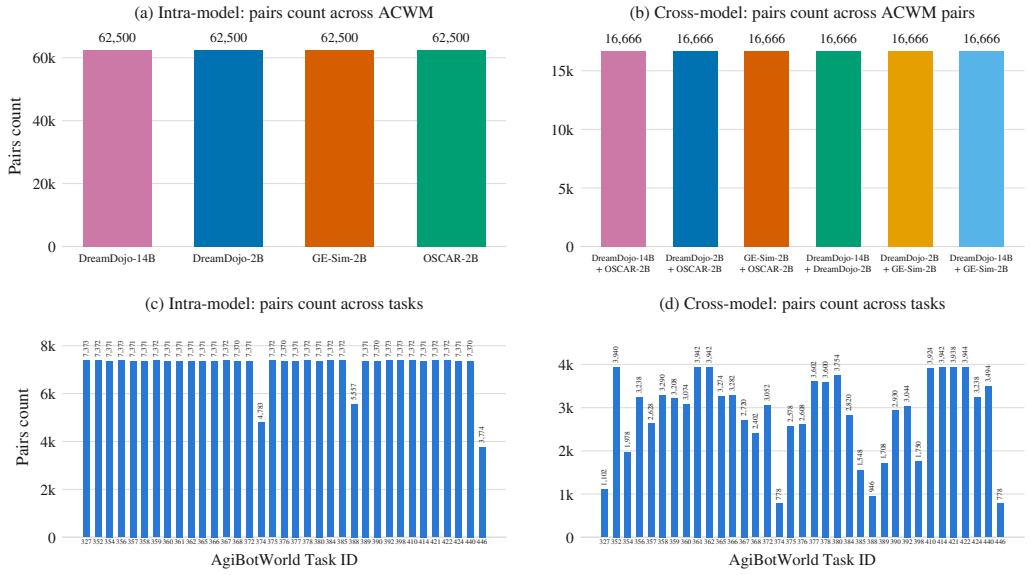


Figure 9: Statistics of video-mismatch pairs in AVM-Pref constructed from $\text{Agi}_{\text{A}}^{\text{train}}$. Panels (a) and (b) show intra-model pair counts by ACWM and cross-model pair counts by ACWM combination, respectively; panels (c) and (d) show the corresponding counts by source task. We aim to balance pair counts across models and tasks. However, task-level balance is limited by the availability of candidate pairs that satisfy all four quality criteria. Cross-model pairing additionally requires roll-outs from both ACWMs under the same input condition, further restricting the available candidates for some tasks. So, tasks with insufficient eligible pairs cannot meet their target quotas, and the remaining quotas are redistributed to tasks with sufficient candidates.

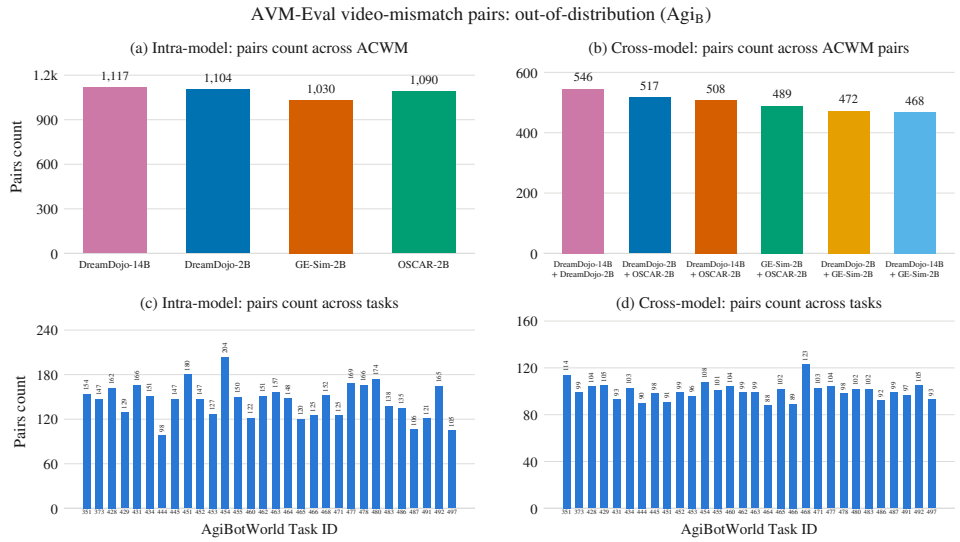


Figure 10: Statistics of video-mismatch pairs in AVM-Eval on $\mathrm{Agi}_{\mathrm{B}}$, comprising 4,341 intra-model and 3,000 cross-model pairs. Panels (a) and (b) show counts by ACWM and ACWM pair, respectively; panels (c) and (d) show the corresponding task distributions.

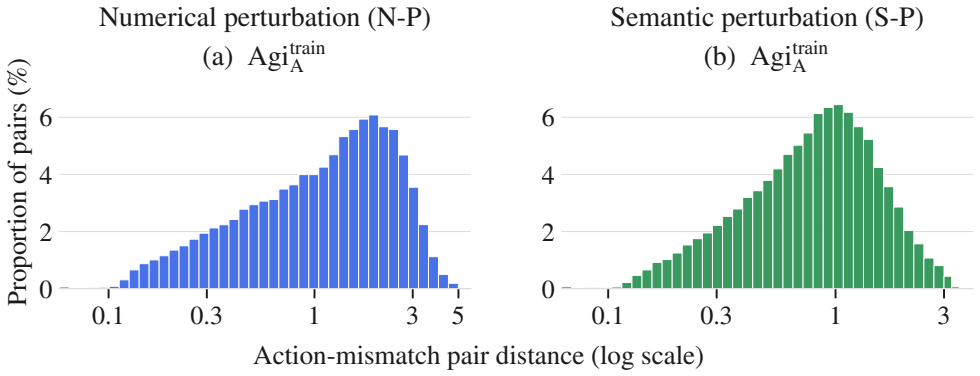


Figure 11: Action-mismatch pair distances in AVM-Pref on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$, computed using Eq. 48. Panels (a) and (b) show numerical and semantic perturbations, respectively. The horizontal axes use a logarithmic scale.

Pairwise Separation Distributions. We characterize the separation between the positive and negative candidates using action differences for action-mismatch pairs and score differences for video-mismatch pairs. The three distance definitions below are used to summarize the pair distributions.

(1) **Action-mismatch pairs in AVM-Eval and AVM-Pref.** Let $\mathbf{a}^+, \mathbf{a}^- \in \mathbb{R}^{K\times 14}$ denote the two action chunks, where $K = 12$ is the number of action transitions in a 13-frame clip. Following Appendix A, $t = 1, \ldots, K$ indexes the transitions and $j = 0, \ldots, 13$ indexes the action dimensions. We divide the difference in each dimension by its fixed standard deviation σ_j, estimated from the corresponding source dataset, and compute the root mean square over the entire chunk:

$$d_{\mathrm{act}}(\mathbf{a}^+, \mathbf{a}^-) = \sqrt{\frac{1}{14K}\sum_{t=1}^{K}\sum_{j=0}^{13}\left(\frac{a^+_{t,j} - a^-_{t,j}}{\sigma_j}\right)^2}. \tag{48}$$

This descriptive distance averages over all entries, whereas the filtering distance in Eq. 19 averages only over modified entries. Figures 13 and 11 show the distributions for numerical and semantic perturbations in AVM-Eval and AVM-Pref, respectively.

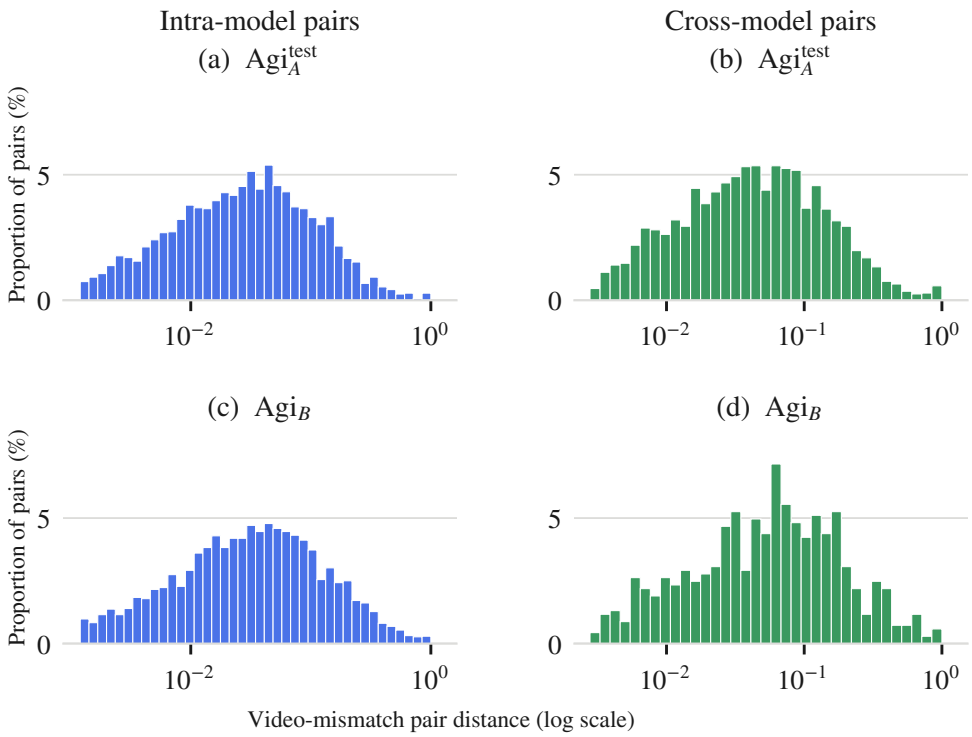


Figure 12: Video-mismatch nDTW gaps in AVM-Eval (Eq. 49). Rows: $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ and $\mathrm{Agi}_{\mathrm{B}}$; columns: intra-model and cross-model pairs. Horizontal axes are logarithmic.

(2) **Video-mismatch pairs in AVM-Eval.** Let $\hat{\mathbf{I}}^+$ and $\hat{\mathbf{I}}^-$ denote the positive and negative generated videos, and let $\mathbf{I}$ denote their shared ground-truth video. We measure separation by the score difference:

$$d_{\mathrm{vid}}^{\mathrm{Eval}}(\hat{\mathbf{I}}^+, \hat{\mathbf{I}}^-) = \left| \mathrm{nDTW}(\hat{\mathbf{I}}^+, \mathbf{I}) - \mathrm{nDTW}(\hat{\mathbf{I}}^-, \mathbf{I}) \right|. \tag{49}$$

Figure 12 reports this gap for intra-model and cross-model pairs on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{test}}$ and $\mathrm{Agi}_{\mathrm{B}}$.

(3) **Video-mismatch pairs in AVM-Pref.** For each candidate video $\hat{\mathbf{I}}$, we collect the four quality metrics from Appendix C, evaluated relative to the shared ground-truth video $\mathbf{I}$:

$$\mathbf{m}(\hat{\mathbf{I}}, \mathbf{I}) = \left[\mathrm{PSNR}^{\mathrm{glob}}, \mathrm{PSNR}^{\mathrm{loc}}, \mathrm{LPIPS}^{\mathrm{glob}}, \mathrm{LPIPS}^{\mathrm{loc}}\right].$$

For each metric $\ell \in \{1, 2, 3, 4\}$, we estimate its standard deviation $\sigma_\ell^{\mathrm{VQ}}$ over all unique video observations appearing on either side of the preference pairs. The same four standard deviations are used for intra-model and cross-model pairs. We standardize each metric difference and take their root mean square:

$$\begin{aligned} \Delta_\ell^{\mathrm{VQ}} &= \frac{m_\ell(\hat{\mathbf{I}}^+, \mathbf{I}) - m_\ell(\hat{\mathbf{I}}^-, \mathbf{I})}{\sigma_\ell^{\mathrm{VQ}}}, \\ d_{\mathrm{vid}}^{\mathrm{Pref}}(\hat{\mathbf{I}}^+, \hat{\mathbf{I}}^-) &= \sqrt{\frac{1}{4}\sum_{\ell=1}^{4} \left(\Delta_\ell^{\mathrm{VQ}}\right)^2}. \end{aligned} \tag{50}$$

Figure 14 shows the distributions of this distance for intra-model and cross-model pairs on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$.

I JEPA-IDM Implementation Details

JEPA-IDM shares ACWM-RM's video preprocessing, camera-frame action parameterization, frozen V-JEPA 2.1 encoder, and predictor backbone (Appendix D.1). It predicts actions instead of

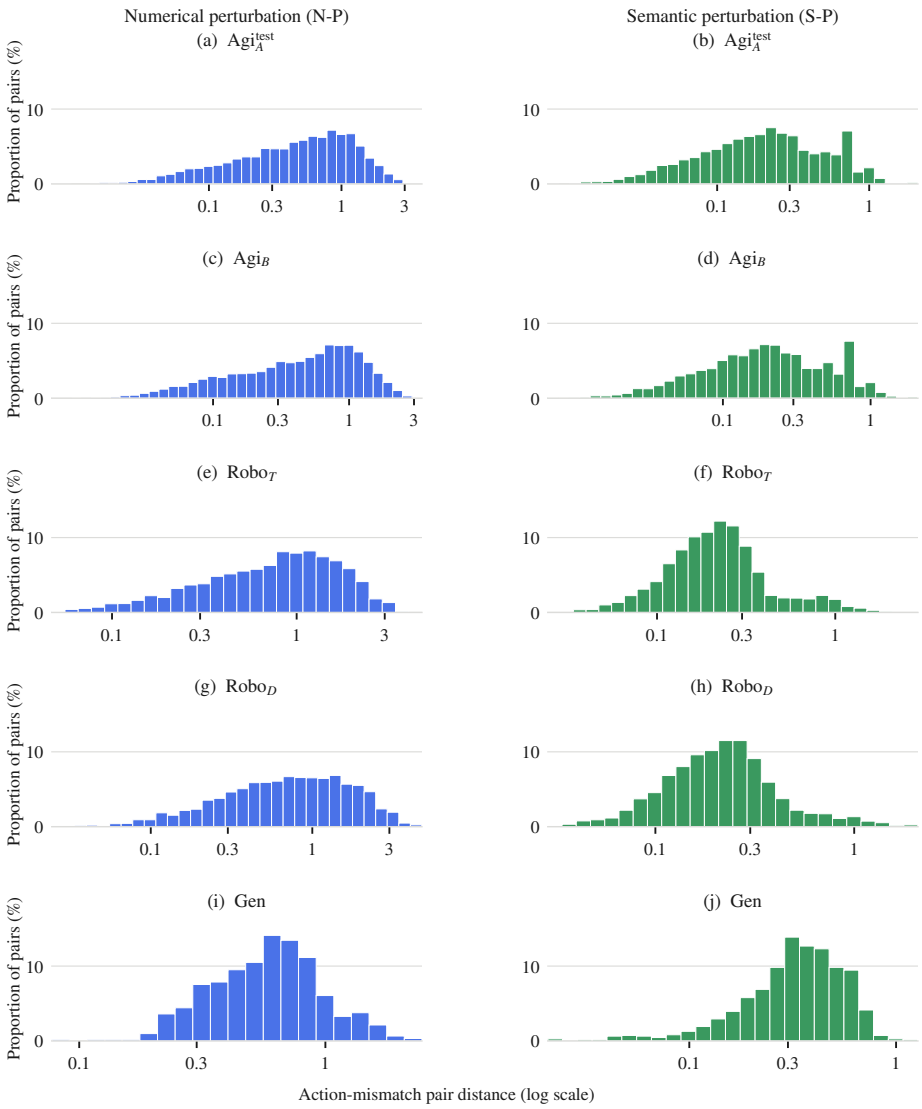


Figure 13: Action-mismatch pair distances in AVM-Eval (Eq. 48). Rows: five source datasets; columns: numerical and semantic perturbations. Horizontal axes are logarithmic.

visual features and scores candidates using the IDM baselines' action-distance formulation instead of a learned scalar score head.

Predictor Inputs. For a clip $\mathbf{I} = (I_0, \ldots, I_K)$, action a_t corresponds to the transition from I_{t-1} to I_t. Following Appendix D.1, we independently encode each frame and project its visual features:

$$z_t = \mathrm{LN}(E(I_t)) \in \mathbb{R}^{N\times D}, \qquad U_t^z = f_z(z_t) \in \mathbb{R}^{N\times d_P}. \tag{51}$$

We use two learnable tokens $e_L, e_R \in \mathbb{R}^{d_P}$ as action readout tokens for the left and right arms, respectively. The initial block contains only the visual tokens of I_0. For each subsequent frame I_t, we append the two readout tokens after its visual tokens:

$$\begin{aligned} \mathbf{X}_0^{\mathrm{IDM}} &= U_0^z, \\ \mathbf{X}_t^{\mathrm{IDM}} &= \big[U_t^z, e_L, e_R\big], \qquad t = 1, \ldots, K, \\ \mathbf{X}^{\mathrm{IDM}} &= \big[\mathbf{X}_0^{\mathrm{IDM}}, \ldots, \mathbf{X}_K^{\mathrm{IDM}}\big]. \end{aligned} \tag{52}$$

For a two-frame input, the sequence is therefore $[U_0^z, U_1^z, e_L, e_R]$, with both arm readout tokens placed after the two frames' visual tokens. Block-causal attention lets block t access blocks $0, \ldots, t$,

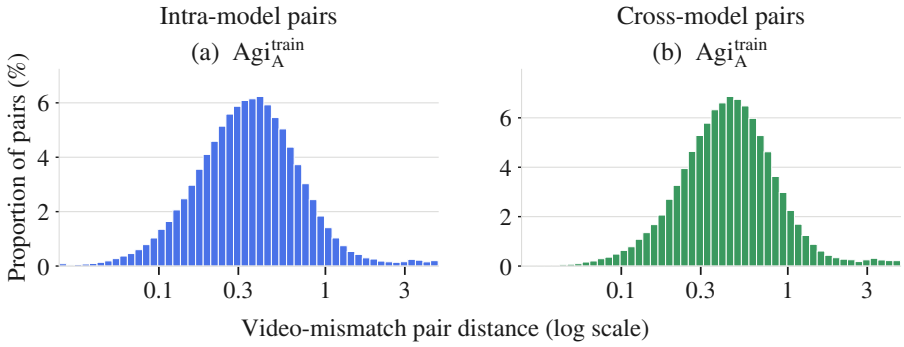


Figure 14: Video-mismatch pair distances in AVM-Pref on $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$ (Eq. 50). Panels (a) and (b): intra-model and cross-model pairs. Horizontal axes are logarithmic.

including both transition frames I_{t-1} and I_t when $t \geq 1$. During preference training and evaluation, all visual blocks come from the candidate's own video; the candidate action chunk is used only to compute its score.

Predictor Outputs. Let $h_{t,r} \in \mathbb{R}^{d_P}$ denote the predictor's output at arm r's readout token appended to block t. A shared MLP $g : \mathbb{R}^{d_P} \to \mathbb{R}^7$ directly maps each readout feature to an action:

$$\tilde{a}_t^r = g(h_{t,r}), \qquad r \in \{L, R\}, \quad t = 1, \ldots, K, \tag{53}$$

The seven components are a 3D translation increment, a 3D rotation-vector increment, and a gripper-opening increment. Translation and rotation increments are expressed in the head-camera coordinate frame, while gripper increments retain their original units. Per-dimension standardization is applied to their residuals in Equations 54 and 55. The initial block has no action readout; each subsequent block produces one action per arm for its incoming transition. Thus, the output for B clips containing $K + 1$ frames has shape $[B, K, 2, 7]$; each 13-frame clip produces 12 bimanual actions.

Action-Prediction Pretraining. We pretrain the predictor on real video-action pairs from $\mathrm{Agi}_{\mathrm{A}}^{\mathrm{train}}$ by minimizing the standardized Smooth-L1 loss between the predicted actions and their ground-truth counterparts:

$$\mathcal{L}_{\mathrm{IDM}} = \frac{1}{14K} \sum_{t=1}^{K} \sum_{r \in \{L,R\}} \sum_{j=1}^{7} \ell_{\mathrm{SL1}} \left(\frac{\tilde{a}_{t,j}^r - a_{t,j}^r}{\sigma_j^r + 10^{-8}} \right). \tag{54}$$

Here, σ_j^r is the training-set standard deviation of the corresponding action component; the same fixed statistics are used in preference training. The visual encoder remains frozen. We use the same training hyperparameters as ACWM-RM's dynamics pretraining (Appendix D.1).

Preference Training. We initialize from action-prediction pretraining and fine-tune the predictor on AVM-Pref, keeping the visual encoder frozen. For the preferred $(+)$ and dispreferred $(-)$ candidates, the reward model infers actions from each candidate's video and computes its score using the associated actions:

$$R^{\pm} = -\frac{1}{14K} \sum_{t=1}^{K} \sum_{r \in \{L,R\}} \sum_{j=1}^{7} \ell_{\mathrm{SL1}} \left(\frac{\tilde{a}_{t,j}^{r,\pm} - a_{t,j}^{r,\pm}}{\sigma_j^r + 10^{-8}} \right). \tag{55}$$

We optimize the Bradley–Terry loss over preference pairs from AVM-Pref:

$$\mathcal{L}_{\mathrm{BT}} = -\mathbb{E}_{(q^+, q^-) \sim \text{AVM-Pref}} \left[\log \sigma \big(R^+ - R^- \big) \right], \tag{56}$$

where $q^{\pm}$ denote the preferred and dispreferred candidates and σ is the sigmoid function. We use the same optimizer settings, batch size, training duration, and schedule as ACWM-RM's preference training (Appendix D.2). All trainable JEPA-IDM parameters use 0.05 times ACWM-RM's score-head learning rate throughout preference training, including warmup, with a peak learning rate of 2×10^{-5}.