PELICAN-SIM 1.0: A GENERAL WORLD MODEL SIMULATOR FOR EMBODIED INTELLIGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot world models must predict not merely plausible videos, but outcomes faithful to specified actions. Across heterogeneous robots, numerical configurations preserve joint-level detail but leave visual consequences implicit, whereas projected visual actions expose motion geometry but obscure depth and configuration information. Our key idea is to let these complementary representations interact throughout generation. We introduce Pelican-Sim 1.0, a shared action-conditioned world model simulator that injects numerical configurations and camera-aligned whole-arm skeleton videos into alternating video-transformer blocks. Each condition shapes hidden states already informed by the other, coupling configuration detail with projected geometry. Sparse mixture-of-experts layers accommodate heterogeneous dynamics, while causal adaptation and few-step distillation support efficient autoregressive rollouts. Trained on approximately one million real and simulated trajectories, Pelican-Sim 1.0 outperforms all evaluated baselines on five video-quality metrics across held-out splits of AgiBotWorld Beta, RoboMIND, and RoboTwin. Ablations demonstrate improvements over single-condition, concatenation, and late-fusion alternatives, supporting the importance of how complementary conditions interact. On RoboTwin, generated-data augmentation raises policy success from 70.0% to 93.0%, while imagined policy optimization with task-specific simulator adaptation improves success from 62.7% to 75.4%. These results support interleaved numerical–visual conditioning as an effective design for shared robot world models and demonstrate their utility for policy learning. We will release model checkpoints and code.

1 INTRODUCTION

World models support robot control by predicting the consequences of interaction (Huang et al., 2025b), enabling planning and policy learning with generalization to new language instructions (Yang et al., 2024; Zhou et al., 2024). As action-conditioned simulators, they can generate policy-training data, compare policies, select candidate actions, and provide environments for policy optimization (Yang et al., 2024; Jiang et al., 2025; Guo et al., 2025). Generated experience can augment scarce demonstrations, while video-based evaluation provides learning signals (Escontrela et al., 2023). Learning from real trajectories also offers a way to capture real-world appearance and interactions without manually reconstructing each scene.

Existing action interfaces broadly fall into three categories. Numerical conditions preserve precise configurations or motion increments, but leave their embodiment- and viewpoint-dependent image-space effects implicit (Zhu et al., 2025; Guo et al., 2025). Visual conditions expose projected motion through end-effector maps, masks, or articulated skeletons (Jiang et al., 2025; Chen et al., 2026b; Wu & Gao, 2026), yet two-dimensional projection can obscure depth and individual joint configurations. Hybrid designs, including EnerVerse-AC and ViPSim, combine geometric inputs with attention-based numerical conditioning (Jiang et al., 2025; Chen et al., 2026a). These designs exploit the complementarity of the two representations, making their interaction within the generative backbone a central architectural choice.

Our key insight is to let numerical precision and explicit geometry jointly shape the evolving video representation. By injecting the two conditions at alternating stages, each can act on hidden states already informed by the other: geometry locates motion in the image, while numerical configu-

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.

rations retain joint-level information that resolves projection ambiguities. This interaction targets the gap between visual plausibility and action fidelity (Huang et al., 2025c). We study this design in a unified action-conditioned world model simulator that shares parameters across heterogeneous robots.

We introduce **Pelican-Sim 1.0**, a world model simulator with synchronized numerical and visual action conditions. A canonical 28-dimensional layout accommodates arm, gripper, and hand configurations for the studied single-arm and bimanual robots. Unified Robot Description Format (URDF) files and camera calibration convert the same trajectory into a camera-aligned, texture-free skeleton video that expresses whole-arm motion in image space. The numerical condition preserves the underlying configurations, including information obscured by projection. Separate pathways inject these conditions into alternating blocks of a video diffusion transformer (DiT) (Peebles & Xie, 2023), integrating modality-specific transformations through shared hidden states throughout generation.

To model heterogeneous dynamics, we equip the DiT with sparse mixture-of-experts (MoE) layers that combine a shared expert with token-dependent routed experts (Fei et al., 2024; Ma et al., 2026). Causal adaptation and few-step distillation enable efficient autoregressive rollouts with four denoising steps per chunk. A fine-tuned vision–language model (VLM) evaluator, frozen during downstream use, converts predicted outcomes into success estimates for policy evaluation, utilities for action selection, and rewards for policy optimization.

We train on approximately one million real and simulated trajectories. On held-out splits of AgiBotWorld Beta (AgiBot-World Contributors et al., 2025), RoboMIND (Wu et al., 2025), and RoboTwin (Mu et al., 2024; Chen et al., 2025), PSNR improves over the strongest evaluated baseline by 4.636, 2.080, and 10.343 dB, respectively. On RoboTwin, generated-data augmentation raises policy success from 70.0% to 93.0%. With task-specific simulator adaptation, action selection reaches 63.8% versus 52.3% for Ctrl-World under shared candidate pools and the same evaluator, while imagined policy optimization improves success from 62.7% to 75.4%. Ablations support complementary action conditioning and the sparse MoE configuration.

Our contributions are:

1. **We introduce Pelican-Sim 1.0, a unified cross-embodiment world model simulator.** It can generate interactive videos accurately followed by visual actions. As a backbone, its distilled autoregressive variant enables efficient rollouts, while a fine-tuned VLM evaluator converts predicted outcomes into feedback for policy evaluation, action selection, and policy optimization.
2. **We demonstrate that action alignment and sparse expert capacity are both beneficial for heterogeneous dynamics.** Interleaving numerical and visual conditions through shared hidden states improves prediction over single-condition, concatenation, and late-fusion baselines. With the action interface and mixed real–simulated corpus fixed, sparse MoE further improves prediction over the dense backbone. Both can be adopted as a general mechanism for modeling video dynamics.
3. **Pelican-Sim 1.0 achieve SOTA performance and will be open-source.** Pelican-Sim 1.0 outperforms all evaluated baselines on five video-quality metrics across three held-out test sets and supports four downstream policy applications. On RoboTwin, data augmentation and imagined policy optimization raise policy success from 70.0% to 93.0% and from 62.7% to 75.4%, respectively. Ablations validate the benefit of visual action conditioning, while qualitative studies show action-consistent predictions under trajectory, scene, embodiment, object, and viewpoint shifts.

2 Related Work

Predictive models for control. Latent dynamics models such as PlaNet (Hafner et al., 2019) support planning from pixels, while video-generative models emphasize visual prediction and interactive generation (Ho et al., 2022; Alonso et al., 2024; Bruce et al., 2024). In robotics, UniSim (Yang et al., 2024) learns from heterogeneous data, RoboDreamer (Zhou et al., 2024) uses compositional language for planning, and VIPER (Escontrela et al., 2023) derives rewards from video prediction. IRASim, EnerVerse-AC, and Ctrl-World (Zhu et al., 2025; Jiang et al., 2025; Guo et al., 2025)

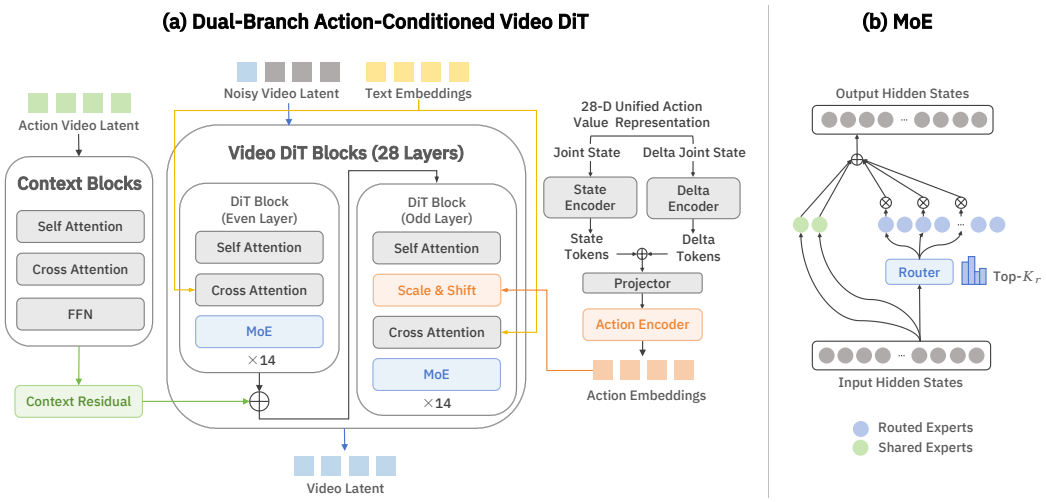


Figure 1: **Dual action conditioning.** Numerical embeddings modulate odd-indexed blocks of a 28-layer Video DiT; a parallel visual context branch adds residuals after even-indexed blocks (zero-based indexing). Sparse MoE replaces the backbone feed-forward layers. Expert counts are schematic; the model uses one shared and eight routed experts, with two routed experts active per token.

study action-conditioned robot simulation. Our focus is a shared action interface and its use across prediction and policy applications.

Cross-embodiment and visual action interfaces. Open X-Embodiment and HPT (Open X-Embodiment Collaboration, 2024; Wang et al., 2024) motivate sharing representations across robot platforms. Visual Action Prompts and OSCAR (Wang et al., 2025; Wu & Gao, 2026) expose articulated motion through skeletons; BridgeV2W, GeniWorld, and Masked Visual Actions (Chen et al., 2026b; Gu et al., 2026; Alzayer et al., 2026) use projected masks or partially revealed trajectories. Hybrid interfaces also appear in EnerVerse-AC and ViPSim (Jiang et al., 2025; Chen et al., 2026a). We build on this line by pairing a fixed bilateral numerical layout with whole-arm skeletons and testing alternating injection against single-condition and fusion alternatives. Appendix A compares related action interfaces.

Sparse diffusion models and efficient generation. DiTs (Peebles & Xie, 2023) provide scalable diffusion backbones. Sparse expert architectures (Lepikhin et al., 2021; Riquelme et al., 2021; Fei et al., 2024; Ma et al., 2026) increase capacity through conditional computation. Causal and distribution-matching distillation (Yin et al., 2024; Huang et al., 2025a; Zhu et al., 2026) reduce iterative generation cost. We use sparse MoE to model heterogeneous real and simulated dynamics, and causal adaptation with few-step distillation to enable efficient autoregressive rollouts.

3 Method

3.1 Prediction Objective

Given an initial RGB observation $\mathbf{x}_0$, a frame-aligned configuration sequence $\mathbf{A} = (\mathbf{a}_0, \dots, \mathbf{a}_T) \in \mathbb{R}^{(T+1)\times 28}$, and an optional instruction $\mathbf{y}$, we model $p_\theta(\mathbf{x}_{1:T} \mid \mathbf{x}_0, \mathbf{A}, \mathbf{y})$. Known URDF and camera metadata determine the visual action condition. The backbone is the 28-layer Cosmos-Predict 2.5 Video DiT (NVIDIA et al., 2025). Its video tokenizer maps RGB observations to an initial conditioning latent $\mathbf{Z}^{\text{cond}}$ and future target latents $\mathbf{Z}_0^{\text{tar}}$.

We keep the conditioning latent clean and train the future latents by flow matching (Lipman et al., 2023). For flow time $\tau \in [0, 1]$ and Gaussian noise $\boldsymbol{\epsilon}$, define

$$\mathbf{Z}_\tau^{\text{tar}} = (1-\tau)\mathbf{Z}_0^{\text{tar}} + \tau\boldsymbol{\epsilon}, \qquad \mathbf{U}^{\text{tar}} = \boldsymbol{\epsilon} - \mathbf{Z}_0^{\text{tar}}. \tag{1}$$

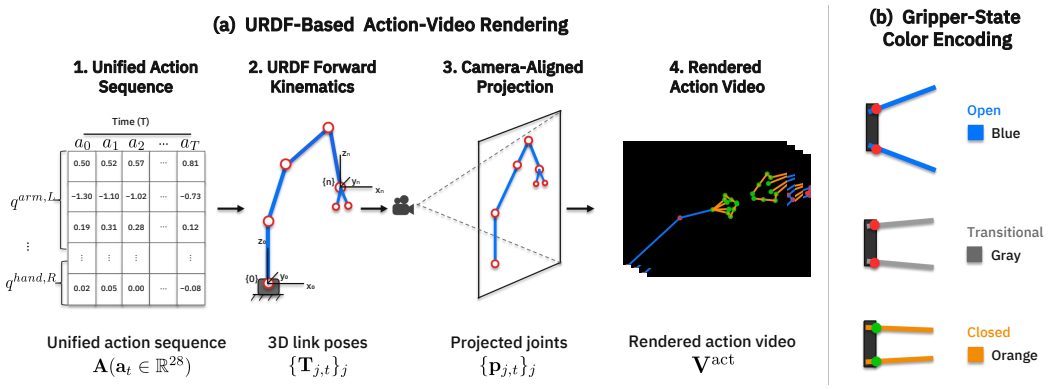


Figure 2: **From numerical configurations to visual action conditions.** A robot-specific mapping and URDF forward kinematics produce link poses; the paired camera calibration projects them into image coordinates. Rasterization preserves link connectivity and color-codes gripper states. Both action conditions are synchronized with the RGB sequence.

The denoiser receives $\widetilde{\mathbf{Z}}_\tau = [\mathbf{Z}^{\text{cond}}; \mathbf{Z}^{\text{tar}}_\tau]$, a temporal conditioning mask $\mathbf{M}$, numerical embeddings $\mathbf{E}^{\text{val}}$, and visual action latents $\mathbf{Z}^{\text{act}}$. Its target-only objective is

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\left[\left\| f_\theta(\widetilde{\mathbf{Z}}_\tau, \mathbf{M}, \tau \mid \mathbf{E}^{\text{val}}, \mathbf{Z}^{\text{act}}, \mathbf{y})_{\text{tar}} - \mathbf{U}^{\text{tar}} \right\|_2^2\right]. \tag{2}$$

At inference, integration from $\tau = 1$ to 0 generates future latents while the initial observation remains fixed. The tokenizer decodes the result into RGB frames. Appendix B gives the full notation and computation.

3.2 Synchronized Numerical and Visual Conditions

Numerical configurations. Each configuration has two 14-dimensional sides. For side $b \in \{L, R\}$, seven entries encode arm joints, one encodes parallel-gripper opening, and six encode dexterous-hand joints:

$$\mathbf{a}_t = [\mathbf{a}_t^L; \mathbf{a}_t^R], \qquad \mathbf{a}_t^b = [\mathbf{q}_t^{\text{arm},b}; g_t^b; \mathbf{q}_t^{\text{hand},b}] \in \mathbb{R}^{14}. \tag{3}$$

Unused joint slots, absent end effectors, and an absent side are zero-filled. This layout accommodates the studied single-arm and bimanual robots; it does not imply coverage of arbitrary kinematic structures. The numerical encoder separately embeds the initial configuration $\mathbf{s} = \mathbf{a}_0$ and scaled increments $\boldsymbol{\delta}_t = (\mathbf{a}_t - \mathbf{a}_{t-1}) \odot \boldsymbol{\eta}$. It groups increments to match the tokenizer's temporal compression and uses MLPs and a residual projector to align embeddings with the latent time grid. Spatial broadcasting produces $\mathbf{E}^{\text{val}}$.

Camera-aligned skeletons. A robot-specific map $\Psi_{\mathcal{U}}$ converts each $\mathbf{a}_t$ into URDF joint order. Forward kinematics produces link poses; camera intrinsics $\mathbf{K}$ and the reference-to-camera transform $\mathbf{T}_{\text{cam}}$ project their origins into the paired RGB view:

$$\mathbf{p}_{j,t} = \Pi(\mathbf{K}, \mathbf{T}_{\text{cam}}, \text{FK}_j(\mathcal{U}, \Psi_{\mathcal{U}}(\mathbf{a}_t))), \tag{4}$$

where $\mathcal{U}$ is the robot description, j indexes links, and Π denotes perspective projection. As illustrated in Figure 2, rasterizing the projected kinematic edges yields a texture-free skeleton, with gripper-link colors encoding opening states. Applying this procedure to every configuration gives a synchronized action video $\mathbf{V}^{\text{act}}$. The RGB tokenizer encodes the normalized video into 16-channel action latents $\mathbf{Z}^{\text{act}}$.

The skeleton video provides camera-aligned geometric guidance, while numerical configurations retain information obscured by projection. Its frames are synchronized with the RGB sequence and rendered solely from configurations, URDF geometry, and camera calibration.

3.3 INTERLEAVED INJECTION AND SPARSE EXPERT CAPACITY

Alternating conditioning pathways. Using zero-based block indices $\ell = 0, \ldots, 27$, numerical conditions enter the 14 odd blocks and visual conditions enter the 14 even blocks (Figure 1). In an odd block, a residual adapter and scale–shift head transform $\mathbf{E}^{\mathrm{val}}$ into $\boldsymbol{\gamma}^\ell$ and $\boldsymbol{\beta}^\ell$. These modulate the post-self-attention feature $\overline{\mathbf{H}}^\ell$ before text cross-attention:

$$\widehat{\mathbf{H}}^\ell = (\mathbf{1} + \boldsymbol{\gamma}^\ell) \odot \overline{\mathbf{H}}^\ell + \boldsymbol{\beta}^\ell, \qquad \ell \in \{1, 3, \ldots, 27\}. \tag{5}$$

The visual pathway uses 14 auxiliary Context Blocks initialized from the corresponding pretrained backbone blocks. Its aligned residual $\mathbf{R}^{\ell/2}$ is added after the even block's MoE and gated residual update:

$$\mathbf{H}^{\ell+1} = \mathbf{H}^\ell_{\mathrm{moe}} + \mathbf{R}^{\ell/2}, \qquad \ell \in \{0, 2, \ldots, 26\}. \tag{6}$$

Interleaving couples the two conditions through an evolving shared representation. Numerical modulation acts on states enriched by visual geometry, while subsequent visual residuals update states already informed by robot configurations. This repeated interaction integrates configuration precision and camera-aligned geometry throughout generation while preserving modality-specific transformations.

Sparse MoE. Each backbone feed-forward layer has one shared expert and eight routed experts, of which two are selected per token (Dai et al., 2024). For an adaptively normalized token $\mathbf{u}$, sigmoid affinities ρ_i and a selection bias determine the top-two set $\mathcal{J}$ (DeepSeek-AI et al., 2024). Mixture weights use unbiased affinities, $g_i = \kappa_r \rho_i / \sum_{j \in \mathcal{J}} \rho_j$, with $\kappa_r = 2.5$. The update is

$$\Delta\mathbf{h} = \mathcal{F}_{\mathrm{shared}}(\mathbf{u}) + s_r(q) \sum_{i \in \mathcal{J}} g_i \mathcal{F}_i(\mathbf{u}), \tag{7}$$

where $\Delta\mathbf{h}$ is the feed-forward token update, $\mathcal{F}_{\mathrm{shared}}$ and $\mathcal{F}_i$ denote the shared and routed expert networks, and $s_r(q) \in [0, 1]$ scales the routed contribution at training iteration q. A load-balancing penalty (Fedus et al., 2022) is added to $\mathcal{L}_{\mathrm{flow}}$; Appendix B.3 gives the routing and training details.

3.4 EFFICIENT AUTOREGRESSIVE ROLLOUTS

To enable efficient autoregressive rollouts, we adapt the 35-step bidirectional model to chunk-wise causal generation and distill it to four denoising steps per chunk. Block-causal attention lets a chunk attend to itself and preceding visual chunks while accessing its known action conditions. Teacher-forced flow matching with clean histories first trains a 35-step causal teacher. Its probability-flow ODE trajectories initialize the four-step student. Self-forcing then exposes the student to generated histories, with distribution matching against the frozen bidirectional teacher and a small fraction of teacher-forced updates (Yin et al., 2024; Huang et al., 2025a; Zhu et al., 2026). Key-value caching within a rolling context reuses historical keys and values across chunks, reducing redundant computation during rollouts. In the recorded 21-frame benchmark, this variant takes 2.2 seconds per clip, with a reported $5.67\times$ speedup. All downstream applications use this distilled autoregressive model. The current results do not isolate the prediction-quality cost of each acceleration stage.

4 EXPERIMENTS

We ask whether the shared action interface improves prediction, which components account for the gains, and whether predicted outcomes are useful for policy learning and selection. Full protocols and additional results appear in Appendix C.

4.1 EXPERIMENTAL SETUP

Data and splits. Our training corpus contains approximately one million trajectories (8,000 hours), combining real-world data from AgiBotWorld Beta, RealSource World, and RoboMIND with simulated data from LIBERO, ManiSkill2, RoboTwin, and RoboCasa (AgiBot-World Contributors et al., 2025; RealSource, 2025; Wu et al., 2025; Liu et al., 2023; Gu et al., 2023; Mu et al., 2024; Nasiriany et al., 2024). Video prediction uses held-out test trajectories from AgiBotWorld Beta, RoboMIND, and RoboTwin, with each source split $90\%/5\%/5\%$ for training, validation, and testing before clip extraction. Appendix C.1 details data curation and source-specific split protocols.

Table 1: Video-quality comparison on three held-out test splits. For each dataset and metric, the best and second-best results are bold and underlined, respectively.

Method	Reconstruction		Perceptual / distribution		
	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	FVD ↓
AgiBotWorld Beta					
IRASim	15.796	0.544	0.028	64.418	62.548
Ctrl-World	17.550	0.682	<u>0.022</u>	46.600	45.100
OSCAR	16.644	0.608	0.023	27.805	32.241
Wan-Move	12.610	0.429	0.447	88.496	95.260
EnerVerse-AC	<u>17.640</u>	<u>0.769</u>	0.023	<u>19.510</u>	<u>21.390</u>
Ours	**22.276**	**0.796**	**0.013**	**14.640**	**10.850**
RoboMIND					
IRASim	20.102	0.709	0.019	44.597	43.414
Ctrl-World	<u>21.770</u>	<u>0.716</u>	<u>0.015</u>	<u>21.800</u>	<u>16.700</u>
EnerVerse-AC	17.930	0.680	0.293	67.350	63.690
Ours	**23.850**	**0.835**	**0.012**	**17.610**	**15.930**
RoboTwin					
IRASim	17.121	0.666	0.025	69.777	57.350
Ctrl-World	<u>20.040</u>	<u>0.804</u>	<u>0.016</u>	<u>31.400</u>	<u>26.600</u>
EnerVerse-AC	19.680	0.779	0.221	32.450	29.700
Ours	**30.383**	**0.942**	**0.006**	**8.910**	**4.980**

Training and comparison. The model uses 93-frame clips at 480×640, conditioning on the first frame and instruction to predict 92 future frames. Training runs for 500K iterations on 32 NVIDIA B200 GPUs with global batch size 32, learning rate 3.05×10^{-5}, and 1,000 warmup steps followed by a constant schedule. IRASim, Ctrl-World, and EnerVerse-AC are evaluated on all three test datasets; OSCAR and Wan-Move (Chu et al., 2025) are additionally evaluated on AgiBotWorld Beta. Baselines are retrained on the same mixed-domain corpus using their respective action representations. We report PSNR, SSIM, LPIPS, FID, and FVD (Wang et al., 2004; Zhang et al., 2018; Heusel et al., 2017; Unterthiner et al., 2018), and an adapted EWMBench (Hu et al., 2025) for action alignment and scene consistency.

4.2 Prediction quality and component analysis

Prediction quality. Table 1 shows that Pelican-Sim 1.0 leads the evaluated methods on all five video-quality metrics. Relative to the strongest baseline per dataset, PSNR improves by 4.636, 2.080, and 10.343 on AgiBotWorld Beta, RoboMIND, and RoboTwin, respectively. On RoboTwin, FVD decreases from 26.600 to 4.980. Adapted EWMBench overall scores are 4.997, 4.666, and 5.044, compared with the best baseline scores of 4.431, 4.276, and 3.971. The advantage is not uniform across component metrics: RoboMIND HSD/nDTW and AgiBotWorld Beta/RoboTwin scene consistency and diversity remain below their best baselines (Table 7). Higher reconstruction quality therefore does not establish superiority on every aspect of dynamics or diversity.

Action representation and injection. Table 2 tests the main architectural choices. Visual actions outperform numerical actions alone, while their combination further increases PSNR by 1.000 on AgiBotWorld Beta and 0.748 on RoboMIND. Both interleaving orders improve on concatenation and late fusion. Injecting both branches at every block gives the highest scores, but exceeds the selected interleaving by only 0.075 and 0.051. Interleaving therefore retains most of the measured benefit with one conditioning pathway per block. Without uncertainty estimates, the small gap should not be interpreted as statistical equivalence. Full metrics are in Tables 9 and 10.

Data composition and expert capacity. The cumulative ablation in Table 3 holds the conditioning interface and training schedule fixed. Adding simulated trajectories to dense-model training improves PSNR by 0.548 and reduces FVD by 3.030; adding sparse MoE under mixed-domain

Table 2: Action-conditioning ablations. Higher PSNR and lower LPIPS are better. The selected model is bold; dual injection at every block gives slightly higher scores. Full SSIM results are in the appendix.

	AgiBotWorld Beta		RoboMIND	
Setting	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
No action	17.684	0.0287	19.640	0.0248
Numerical only	19.238	0.0214	21.356	0.0176
Visual only	21.276	0.0158	23.102	0.0132
Dual, interleaved	**22.276**	**0.0125**	**23.850**	**0.0115**
Token concatenation	21.214	0.0164	21.903	0.0170
Late fusion	21.372	0.0155	22.931	0.0138
Reverse interleaving	22.241	0.0126	23.821	0.0116
Dual every block	22.351	0.0123	23.901	0.0114

Table 3: Cumulative data and capacity ablation on AgiBotWorld Beta. Conditioning and training schedule are fixed. Full five-metric results appear in Table 8.

Training configuration	PSNR ↑	LPIPS ↓	FVD ↓
Real only, dense	20.514	0.0186	20.410
Real + simulation, dense	21.062	0.0169	17.380
Real + simulation, sparse MoE	**22.276**	**0.0125**	**10.850**

training yields a further PSNR gain of 1.214 and FVD reduction of 6.530. The former changes both corpus size and diversity; the latter changes model capacity. These comparisons support the combined configuration, but do not isolate data diversity or routing at fixed capacity.

4.3 Downstream utility on RoboTwin

All four applications use the distilled autoregressive model with four denoising steps per chunk. Evaluation, selection, and policy improvement additionally use a fine-tuned Qwen3-VL-2B-Instruct evaluator (Bai et al., 2025) with LoRA (Hu et al., 2022) and success, progress, and visual-validity heads. It receives an instruction, a task checklist, and 16 ordered video frames, and remains frozen during downstream use. Validation accuracy is 92.6% on 1,000 simulator videos and 91.4% on 700 generated videos. Generated-video labels use majority vote from three annotators; each video is judged on its depicted outcome. This tests evaluator accuracy separately from the correctness of the world-model prediction (Appendix C.5).

Reliability of learned feedback. Evaluator training uses simulator and generated rollouts with manual success, progress, and validity annotations and same-task ranking supervision. Only the LoRA adapters and prediction heads are trained. Table 5 reports validation beyond accuracy: on generated videos, success AUROC is 0.951, expected calibration error (ECE) (Guo et al., 2017) is 0.048, and progress MAE is 0.078. Validity AUROC is 0.953 (Table 14). Progress references are median five-level human ratings normalized to $[0, 1]$; threshold-dependent metrics use disjoint calibration data. These measurements support the use of the learned feedback while exposing residual judgment errors that can affect selection and optimization.

Data generation. We augment each original trajectory with ten rollouts, using GPT-Image-2 (OpenAI, 2026) to vary the initial observation while retaining its actions. A $\pi_{0.5}$ policy (Physical Intelligence et al., 2025) is trained on original or augmented data and evaluated on 200 unseen randomized RoboTwin seeds per task (Chen et al., 2025). At 10, 30, and 50 original demonstrations per task, success rises from 28.5%, 57.0%, and 70.0% to 64.5%, 87.0%, and 93.0%, respectively. This experiment starts from randomized demonstrations. A separate clean-to-randomized study obtains 72.0% with 50 clean plus 500 generated trajectories, versus 70.0% with 50 directly collected

Table 4: Downstream RoboTwin success rates (%). Each panel has its own policy and protocol; rates are comparable within panels. Selection uses shared pools of ten candidates. Simulator oracles are separated from learned methods.

Setting	Success ↑
Data augmentation ($\pi_{0.5}$; 50 original trajectories/task)	
Original only	70.0
Original + 500 generated trajectories/task	**93.0**
Action selection (Diffusion Policy; five tasks)	
Single sample / Random-of-10	43.2 / 43.5
VLM-direct / Ctrl-World selection	48.7 / 52.3
Pelican-Sim 1.0 selection	**63.8**
Oracle Best-of-10 (simulator outcomes)	68.1
Policy improvement (OpenVLA-OFT; three training runs)	
Initial SFT / Iter-SFT	62.7 / 67.0
Pelican-Sim 1.0-GRPO (terminal only)	68.5
Pelican-Sim 1.0-GRPO (full reward)	**75.4**
RoboTwin-GRPO (simulator feedback)	83.8

Table 5: **Evaluator reliability and adaptation requirements.** Top: judging depicted outcomes in simulator videos and a score-stratified subset of generated videos; ECE measures success-confidence calibration. Bottom: agreement of the world-model–VLM pipeline with RoboTwin across five policy checkpoints, with 200 initial conditions each. The evaluator is frozen as the world-model adaptation budget changes; the selected 500-rollout setting is bold.

Evaluation input	N	Acc. (%) ↑	AUROC ↑	ECE ↓	Progress MAE ↓
RoboTwin videos	1,000	92.6	0.964	0.041	0.061
Generated videos	700	91.4	0.951	0.048	0.078

Adaptation rollouts	Pearson r ↑	Spearman ρ ↑	SR MAE (pp) ↓
0	0.902	0.800	8.6
100	0.958	0.900	5.3
200	0.976	1.000	4.0
500	**0.989**	**1.000**	**2.9**
1,000	0.994	1.000	2.4

randomized trajectories. The differing data budgets preclude a matched-sample-efficiency claim (Appendix C.4).

Policy evaluation and ranking. We compare five checkpoints from one VLA training run on 200 held-out initial conditions each. Identical action trajectories are executed in RoboTwin and supplied to the world model; this assesses fixed-trajectory outcome prediction rather than closed-loop policy execution. Thresholded VLM success estimates recover all five checkpoint rankings after adaptation on 200 task-specific rollouts (Spearman $\rho = 1$, Kendall $\tau = 1$). Table 5 separates this ranking result from absolute rate estimation: MAE falls from 8.6 percentage points without adaptation to 2.9 at 500 rollouts and 2.4 at 1,000; Pearson r reaches 0.994 at 1,000. Nonzero adaptation budgets are balanced across the five policies, with the evaluator held fixed. We use the 500-rollout variant for the next two applications: doubling that budget preserves ranking agreement and reduces MAE by only 0.5 points. These results concern the tested checkpoint set, not arbitrary policy populations.

Action selection. Diffusion Policy (Chi et al., 2023) samples ten candidate trajectories per initial condition. The model predicts each outcome; the frozen evaluator selects the maximum utility $u_\phi = a_\phi[q_\phi + \lambda(1-q_\phi)p_\phi]$, where q_ϕ, p_ϕ, and a_ϕ denote success, progress, and validity. Only the selected

candidate is executed, open-loop. Five tasks use 200 unseen conditions each and shared candidate pools across selectors. Success increases from 43.2% for a single sample to 63.8%, exceeding direct VLM scoring by 15.1 points and Ctrl-World selection by 11.5 points. Oracle selection, using simulator outcomes unavailable to learned selectors, reaches 68.1%. Increasing the candidate count from 8 to 10 adds 1.3 success points while reported latency increases from 10.7 to 15.9 seconds (Appendix C.6).

Policy improvement. An OpenVLA-OFT policy (Kim et al., 2025; 2024) with a token-based action head generates chunks from predicted observations. Each chunk is converted to 21 configurations and rendered into paired action conditions; the final predicted frame conditions the next decision. With the world model and evaluator frozen, GRPO (Shao et al., 2024) uses validity-weighted progress increments and terminal-success feedback to update the policy (Sharma et al., 2026). Specifically, for imagined trajectory k, the evaluator scores J successive prefixes and assigns the return

$$R_k = \sum_{j=1}^{J} \gamma^{j-1} a_{\phi,j,k} \big(p_{\phi,j,k} - p_{\phi,j-1,k}\big) + \beta a_{\phi,J,k} q_{\phi,k}, \tag{8}$$

where $p_{\phi,0,k} = 0$, γ discounts progress increments, and β weights terminal success. Normalizing returns within each rollout group yields the advantage $\widehat{A}_k = (R_k - \mu_R)/(\sigma_R + \epsilon)$, with group mean μ_R, standard deviation σ_R, and stabilizer $\epsilon > 0$. This advantage is assigned to every action chunk in the trajectory. Low-validity rollouts are excluded, and groups with insufficient valid samples or negligible return variance are resampled. The clipped policy objective and full protocol are given in Appendix C.7.

All variants share supervised initialization; final evaluation uses the RoboTwin task checker on unseen seeds, averaged over three training runs. Mean success increases from 62.7% to 75.4%, versus 67.0% for filtered iterative SFT and 68.5% for terminal-only GRPO. The full method remains 8.4 points below simulator-based GRPO. The reward comparison changes several components jointly and does not isolate progress from validity.

4.4 Qualitative distribution shifts

Appendix C.8 examines reversed trajectories, six edited scene variants, three held-out embodiments (Sawyer, IIWA, and UR5e), five edited object variants, and five unseen viewpoints. Retargeted embodiments use inverse kinematics and target-camera skeleton rendering. The examples show action-consistent motion under these tested changes. They neither establish quantitative OOD robustness nor imply that image-edited objects preserve the original physical properties. Additional multi-source predictions are in Appendix D.

5 Discussion and Conclusion

Pelican-Sim 1.0 is a unified world model simulator for heterogeneous robots that integrates numerical precision and camera-aligned geometry through interleaved action conditioning. Experiments demonstrate complementary gains in video prediction from action alignment and sparse expert capacity. Efficient autoregressive rollouts support data generation, while VLM-based feedback enables policy evaluation, action selection, and policy optimization. Qualitative studies show action-consistent predictions under the tested distribution shifts. We will release all trained model checkpoints and code as reusable resources for robot learning.

Future work will explore broader interaction data, including egocentric human videos, and assess generalization and physical consistency through quantitative evaluation of long-horizon closed-loop interaction across unseen robots and tasks.

AI Use Statement

During manuscript preparation, AI tools were used to assist with writing and language polishing and to retrieve and discover relevant literature. Research uses include GPT-Image-2 for editing initial observations in synthetic data augmentation and visual variation studies, and a vision–language

model for data-alignment checks, as described in Section 4.3 and Appendix C. The authors take responsibility for the final text, claims, and artifacts.

Reproducibility Statement

Section 3 and Appendix B specify the action interface, conditioning architecture, and training objective. Section 4.1 and Appendix C.1 describe data curation, splits, and recorded training settings. Appendix C retains the downstream protocols, evaluator validation, and full result tables. The present record does not specify every implementation and evaluation parameter; results should be interpreted with the limitations stated in Section 5. We plan to release model checkpoints and inference code.

Ethics Statement

Predicted videos can misrepresent contacts, object motion, or task completion. The reported downstream success rates are measured in RoboTwin and do not establish safe deployment on physical robots. Use on hardware requires independent validation and appropriate execution safeguards. Synthetic initial-frame edits and VLM judgments can introduce appearance or evaluation biases. Data use and any future release must respect source-dataset terms, privacy obligations, and annotation consent; this paper does not establish additional release rights over third-party data.

References

AgiBot-World Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in Atari. In *Advances in Neural Information Processing Systems*, 2024.

Hadi Alzayer, Wenlong Huang, Haonan Chen, Christopher Luey, Lvmin Zhang, Maneesh Agrawala, Gordon Wetzstein, Li Fei-Fei, Yilun Du, Jiajun Wu, and Jia-Bin Huang. Masked visual actions for unified world modeling. *arXiv preprint arXiv:2607.19343*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Longyu Chen, Heng Li, Wei Yang, Manqi Zhao, and Dongsheng Jiang. ViPSim: Collaborating visual and parameter spaces for consistent long-horizon embodied world models. *arXiv preprint arXiv:2606.28804*, 2026a.

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

Yixiang Chen, Peiyan Li, Jiabing Yang, Keji He, Xiangnan Wu, Yuan Xu, Kai Wang, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. BridgeV2W: Bridging video generation models to embodied world models via embodiment masks. *arXiv preprint arXiv:2602.03793*, 2026b.

Ke Cheng, Hanqiao Ye, Lei Shi, Yahui Liu, Yunhan Shen, Jingtao Dong, Zhenke Wang, Wenxuan Ao, Weixiang Xu, Kaining Huang, and Shuhan Shen. M4World: A multi-view multimodal driving world model for interactive object manipulation and minute-long streaming. *arXiv preprint arXiv:2607.14005*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Ruihang Chu, Yefei He, Zhekai Chen, Shiwei Zhang, Xiaogang Xu, Bin Xia, Dingdong Wang, Hongwei Yi, Xihui Liu, Hengshuang Zhao, Yu Liu, Yingya Zhang, and Yujiu Yang. Wan-Move: Motion-controllable video generation via latent trajectory guidance. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*, 2024.

Youhe Feng, Hansen Shi, Haoyang Li, Xinlei Guo, Yang Wang, Chengyang Zhang, Jinkai Zhang, Xiaohan Zhang, Jie Tang, and Jing Zhang. ProcVLM: Learning procedure-grounded progress rewards for robotic manipulation. *arXiv preprint arXiv:2605.08774*, 2026.

Chenghao Gu, Hanyang Yu, Jingbo Zhang, Haitao Lin, Wenyao Zhang, Jinghe Wang, Hanglei Jin, Shuzhao Xie, Jingyan Jiang, and Zhi Wang. GeniWorld: A generalizable interactive world model for robotic manipulation via visual actions. *arXiv preprint arXiv:2608.06332*, 2026.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL `https://proceedings.mlr.press/v70/guo17a.html`.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Songjia He, Zixuan Chen, Hongyu Ding, Dian Shao, Jieqi Shi, Chenxu Li, Jing Huo, and Yang Gao. V-Dreamer: Automating robotic simulation and trajectory synthesis via video generation priors. *arXiv preprint arXiv:2603.18811*, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yue Hu, Siyuan Huang, Yue Liao, Shengcong Chen, Pengfei Zhou, Liliang Chen, Guanghui Ren, and Maoqing Yao. EWMBench: Evaluating scene, motion, and semantic quality in embodied world models. In *British Machine Vision Conference*, 2025.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train–test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. LaDi-WM: A latent diffusion-based world model for predictive manipulation. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1726–1743. PMLR, 2025b. URL https://proceedings.mlr.press/v305/huang25a.html.

Yuhang Huang, Shilong Zou, Jiazhao Zhang, Xinwang Liu, Ruizhen Hu, and Kai Xu. AdaPower: Specializing world foundation models for predictive manipulation. *arXiv preprint arXiv:2512.03538*, 2025c. URL https://arxiv.org/abs/2512.03538.

Byeongguk Jeon, Seonghyeon Ye, JaeHyeok Doo, Sungdong Kim, Minjoon Seo, Hyungmok Son, and Kimin Lee. RoboWorld: Fast and reliable neural simulators for generalist robot policy evaluation. *arXiv preprint arXiv:2607.01060*, 2026.

Yuxin Jiang, Shengcong Chen, Siyuan Huang, Liliang Chen, Pengfei Zhou, Yue Liao, Xindong He, Chiming Liu, Hongsheng Li, Maoqing Yao, and Guanghui Ren. EnerVerse-AC: Envisioning embodied environments with action condition. In *Conference on Robot Learning*, 2025.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

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

Shuailei Ma, Jiaqi Liao, Xinyang Wang, Jingjing Wang, Chaoran Feng, Zijing Hu, Chong Bao, Zichen Xi, Yuqi Gan, Yanhong Zeng, et al. Scaling mixture-of-experts video pretraining for embodied intelligence. *arXiv preprint arXiv:2607.07675*, 2026.

Yao Mu, Tianxing Chen, Shijia Peng, Zanxin Chen, Zeyu Gao, Yude Zou, Lunkai Lin, Zhiqiang Xie, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. *arXiv preprint arXiv:2409.02920*, 2024.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

NVIDIA, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Yu-Wei Chao, et al. World simulation with video foundation models for physical AI. *arXiv preprint arXiv:2511.00062*, 2025.

Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024.

OpenAI. GPT-Image-2 model documentation. `https://developers.openai.com/api/docs/models/gpt-image-2`, 2026. Accessed: 2026-09-03.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040/`.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

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

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. WorldGym: World model as an environment for policy evaluation. *arXiv preprint arXiv:2506.00613*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL `https://proceedings.mlr.press/v139/radford21a.html`.

RealSource. RealSource World: A large-scale real-world dual-arm manipulation dataset. Hugging Face dataset repository, 2025. Dataset release. Accessed September 8, 2026.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, Andre Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, 2021.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. World-Gymnast: Training robots with reinforcement learning in a world model. *arXiv preprint arXiv:2602.02454*, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL `https://arxiv.org/abs/2307.09288`.

Wei-Cheng Tseng, Jinwei Gu, Qinsheng Zhang, Hanzi Mao, Ming-Yu Liu, Florian Shkurti, and Lin Yen-Chen. Scalable policy evaluation with video world models. *arXiv preprint arXiv:2511.11520*, 2025.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *Advances in Neural Information Processing Systems*, 2024.

Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.152.

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. In *International Conference on Learning Representations*, 2024.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024.

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.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning compositional world models for robot imagination. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

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

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026.

A Comparison of Action Interfaces

Table 6 summarizes the numerical and visual action inputs of the related models in Section 2. Figure 3 illustrates the geometric cues provided by different visual interfaces.

Table 6: Numerical and visual action interfaces in related predictive models.

Method	Numerical input	Visual geometry input
IRASim (Zhu et al., 2025)	Yes	None
Ctrl-World (Guo et al., 2025)	Yes	None
EnerVerse-AC (Jiang et al., 2025)	Yes	End-effector map
Visual Action Prompts (Wang et al., 2025)	No	Hand/gripper skeleton
OSCAR (Wu & Gao, 2026)	No	Arm skeleton
BridgeV2W (Chen et al., 2026b)	No	Robot mask
GeniWorld (Gu et al., 2026)	No	Robot mask
ViPSim (Chen et al., 2026a)	Yes	End-effector map, mask
Masked Visual Actions (Alzayer et al., 2026)	No	Partially revealed trajectories
Pelican-Sim 1.0	Yes	Whole-arm URDF skeleton

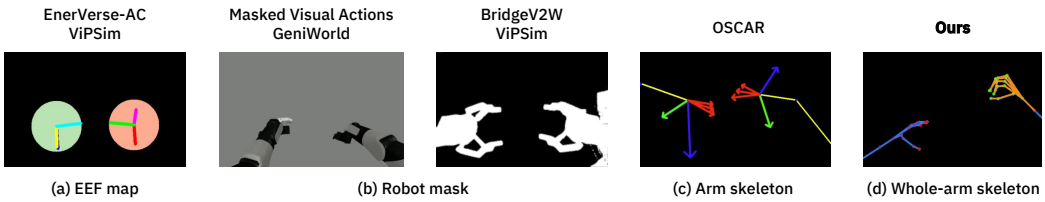


Figure 3: Geometric cues in visual action conditions: end-effector maps, robot masks, and articulated skeletons. Numerical configurations complement image projections that can obscure depth or joints.

B Model Implementation Details

B.1 Latent Conditioning and Sampling

We use the prediction objective and notation of Section 3.1, with $\mathbf{X}^{\mathrm{tar}} = \mathbf{x}_{1:T}$ denoting the future video. The batch dimension is omitted here. Configuration $\mathbf{a}_0$ aligns with the initial observation, and $\mathbf{a}_{1:T}$ aligns with the target frames.

During training, the encoder $\mathcal{E}$ of the pretrained video tokenizer maps the complete clip to a latent sequence. We partition it into the initial-frame conditioning latent $\mathbf{Z}^{\mathrm{cond}}$ and the clean future target latent $\mathbf{Z}_0^{\mathrm{tar}}$:

$$[\mathbf{Z}^{\mathrm{cond}}; \mathbf{Z}_0^{\mathrm{tar}}] = \mathcal{E}\big([\mathbf{x}_0; \mathbf{X}^{\mathrm{tar}}]\big)\,. \tag{9}$$

Here, $[\cdot;\cdot]$ concatenates sequences along time. The subscript 0 on $\mathbf{Z}_0^{\mathrm{tar}}$ denotes the clean flow endpoint; RGB frames use index t and flow time uses τ. In Equation 1, Gaussian noise has the target latent's shape and identity covariance after vectorization. The binary temporal mask $\mathbf{M}$ identifies conditioning and target positions, while the subscript tar in Equation 2 selects only future latent outputs. The loss norm covers all target entries, and its expectation is over paired training examples, Gaussian noise, and flow timesteps. Action encodings are specified in Section B.3.

At inference, $\mathbf{Z}^{\mathrm{cond}} = \mathcal{E}(\mathbf{x}_0)$ remains fixed while the velocity field is integrated from $\tau = 1$ to 0. The decoder reconstructs $[\mathbf{Z}^{\mathrm{cond}}; \widehat{\mathbf{Z}}_0^{\mathrm{tar}}]$; discarding the conditioning frame gives $\widehat{\mathbf{X}}^{\mathrm{tar}}$. Hats denote generated predictions.

B.2 Action Layout and Skeleton Rendering

Unified action value. Equation 3 fixes the left-side slots before the right-side slots. Within each side, arm joints precede the gripper opening and dexterous-hand joints. A 6-DoF arm uses six of the seven arm slots; the remaining slot is zero. Missing end effectors and all 14 entries of an absent side are also zero-filled. Concatenation is along the feature dimension for configuration vectors.

URDF-rendered action video. The rendering pipeline in Figure 2 first maps the unified configuration to the joint order of robot URDF $\mathcal{U}$:

$$\mathbf{q}_t^{\mathcal{U}} = \Psi_{\mathcal{U}}(\mathbf{a}_t), \tag{10}$$

where $\mathbf{q}_t^{\mathcal{U}}$ is the configuration in URDF joint order and $\Psi_{\mathcal{U}}$ is the robot-specific conversion from the unified layout. For rendered link index j, the forward-kinematics function FK_j returns the homogeneous link-pose transform $\mathbf{T}_{j,t} \in \mathbb{R}^{4\times4}$:

$$\mathbf{T}_{j,t} = \mathrm{FK}_j(\mathcal{U}, \mathbf{q}_t^{\mathcal{U}}). \tag{11}$$

Let $\mathbf{K} \in \mathbb{R}^{3\times3}$ and $\mathbf{T}_{\mathrm{cam}} \in \mathbb{R}^{4\times4}$ denote the intrinsics and reference-to-camera transform of the paired RGB view, with all link poses expressed in the same reference frame. The link origins are projected onto the image plane as

$$\mathbf{p}_{j,t} = \Pi(\mathbf{K}, \mathbf{T}_{\mathrm{cam}}, \mathbf{T}_{j,t}), \tag{12}$$

where Π denotes perspective projection and $\mathbf{p}_{j,t} \in \mathbb{R}^2$ is the image coordinate of link origin j. Let $\mathcal{K}_{\mathcal{U}}$ be the URDF kinematic edge set connecting these projected nodes. The rasterizer $\mathcal{R}$ produces an action-video frame $\mathbf{v}_t^{\mathrm{act}}$ and the full action video $\mathbf{V}^{\mathrm{act}}$:

$$\mathbf{v}_t^{\mathrm{act}} = \mathcal{R}(\{\mathbf{p}_{j,t}\}_j, \mathcal{K}_{\mathcal{U}}, \mathbf{g}_t), \qquad \mathbf{V}^{\mathrm{act}} = (\mathbf{v}_0^{\mathrm{act}}, \ldots, \mathbf{v}_T^{\mathrm{act}}), \tag{13}$$

where $\mathbf{g}_t = (g_t^L, g_t^R)$ contains the gripper openings. Gripper-link colors distinguish open, transitional, and closed states. Frame $\mathbf{v}_t^{\mathrm{act}}$ is synchronized with RGB frame $\mathbf{x}_t$ across the complete $T+1$-frame sequence.

B.3 Dual-Branch Conditioning and Sparse MoE Layers

We detail the encoders and routing used in Section 3.3. The numerical and visual block sets are $\mathcal{I}_{\mathrm{val}} = \{1, 3, \ldots, 27\}$ and $\mathcal{I}_{\mathrm{vid}} = \{0, 2, \ldots, 26\}$, respectively. Features $\mathbf{H}_{\mathrm{moe}}^{\ell}$ are taken after the backbone block's MoE and gated residual, before visual conditioning is added.

We now include the batch dimension B. Let T_{lat} be the complete latent sequence length, including the conditioning position, $H_{\mathrm{tok}} \times W_{\mathrm{tok}}$ the spatial token grid after patch embedding, and D the DiT hidden width. Hidden features and aligned action embeddings are represented on the grid $B \times T_{\mathrm{lat}} \times H_{\mathrm{tok}} \times W_{\mathrm{tok}} \times D$, with an equivalent flattened token representation used inside transformer blocks.

Action-value encoding and injection. As shown in Figure 1(a), the action-value pathway represents the trajectory by its initial configuration $\mathbf{s}$ and scaled frame-to-frame changes $\boldsymbol{\delta}_t$:

$$\mathbf{s} = \mathbf{a}_0, \qquad \boldsymbol{\delta}_t = (\mathbf{a}_t - \mathbf{a}_{t-1}) \odot \boldsymbol{\eta}, \quad t = 1, \ldots, T, \tag{14}$$

where $\boldsymbol{\eta} \in \mathbb{R}^{28}$ is a dimension-wise scaling vector and $\odot$ denotes element-wise multiplication. Let r_t be the tokenizer's temporal compression factor and $T_{\mathrm{a}} = 1 + \lceil T/r_t \rceil$ the number of action tokens. We group the deltas into $T_{\mathrm{a}} - 1$ non-overlapping chunks of length r_t, padding the final chunk when necessary. Two multilayer perceptrons (MLPs), $\mathcal{M}_{\mathrm{state}}$ and $\mathcal{M}_{\mathrm{delta}}$, encode the initial state and the delta chunks:

$$\mathbf{e}^{\mathrm{state}} = \mathcal{M}_{\mathrm{state}}(\mathbf{s}), \qquad \mathbf{e}_k^{\mathrm{delta}} = \mathcal{M}_{\mathrm{delta}}\big(\mathrm{Concat}(\boldsymbol{\delta}_{kr_t+1:(k+1)r_t})\big), \quad k = 0, \ldots, T_{\mathrm{a}} - 2, \tag{15}$$

Here, $\mathbf{e}^{\mathrm{state}}$ is the initial-state token, $\mathbf{e}_k^{\mathrm{delta}}$ is delta-chunk token k, and Concat concatenates the chunk's r_t delta vectors along the feature dimension. A learned projector and residual action encoder align these tokens to T_{lat} latent positions and width D. Spatial broadcasting then yields $\mathbf{E}^{\mathrm{val}} \in \mathbb{R}^{B\times T_{\mathrm{lat}}\times H_{\mathrm{tok}}\times W_{\mathrm{tok}}\times D}$.

For each odd block $\ell \in \mathcal{I}_{\text{val}}$, a residual adapter and scale–shift head produce the modulation parameters of Equation 5:

$$\begin{aligned} \mathbf{E}^{\text{inj},\ell} &= \mathbf{E}^{\text{val}} + \mathcal{I}_2^\ell\big(\text{SiLU}\big(\mathcal{I}_1^\ell(\mathbf{E}^{\text{val}})\big)\big)\,, \\ (\boldsymbol{\gamma}^\ell, \boldsymbol{\beta}^\ell) &= \mathcal{S}^\ell(\mathbf{E}^{\text{inj},\ell}). \end{aligned} \tag{16}$$

Here, $\mathbf{E}^{\text{inj},\ell}$ is the block-adapted embedding, $\mathcal{I}_1^\ell$ and $\mathcal{I}_2^\ell$ are adapter layers, and $\mathcal{S}^\ell$ is the scale–shift head. The modulation tensors match the hidden-feature grid. Text cross-attention and MoE follow the modulation, yielding $\mathbf{H}^{\ell+1} = \mathbf{H}^\ell_{\text{moe}}$ for odd blocks.

Action-video encoding and injection. As shown in Figure 1(a), let $\overline{\mathbf{V}}^{\text{act}}$ denote the action video normalized to $[-1, 1]$. The same video tokenizer $\mathcal{E}$ used for RGB observations produces

$$\mathbf{Z}^{\text{act}} = \mathcal{E}(\overline{\mathbf{V}}^{\text{act}}) \in \mathbb{R}^{B\times C_z\times T_{\text{lat}}\times H'\times W'}, \qquad C_z = 16, \tag{17}$$

where C_z is the latent-channel dimension and $H' \times W'$ is the latent grid before patch embedding. After patch embedding and spatial alignment, parallel Context Blocks initialized from the corresponding pretrained backbone blocks process $\mathbf{Z}^{\text{act}}$. Context Block $k \in \{0, \dots, 13\}$ produces the hidden-grid residual $\mathbf{R}^k$ added after even block $\ell = 2k$ according to Equation 6.

Sparse MoE layers. Each MoE layer has $N_s = 1$ shared and $N_r = 8$ routed feed-forward networks (FFNs), with $K_r = 2$ routed experts selected per token (Dai et al., 2024). Routing uses token features without predefined embodiment or domain assignments.

Let n index flattened spatiotemporal tokens, and let $\mathbf{h}_n^{\ell,\text{ca}} \in \mathbb{R}^D$ be token n after text cross-attention in block ℓ. The router and experts operate on its adaptive layer-normalized (AdaLN) representation (Peebles & Xie, 2023) $\mathbf{u}_n^\ell \in \mathbb{R}^D$, conditioned on flow timestep τ. Following sigmoid token-choice routing with selection bias (DeepSeek-AI et al., 2024; Ma et al., 2026), we define

$$\begin{aligned} \mathbf{u}_n^\ell &= \text{AdaLN}^\ell_{\text{mlp}}(\mathbf{h}_n^{\ell,\text{ca}}; \tau), \\ \boldsymbol{\rho}_n^\ell &= \text{sigmoid}(\mathbf{W}_r^\ell \mathbf{u}_n^\ell), \\ \mathcal{J}_n^\ell &= \text{TopK}_{K_r}\big(\boldsymbol{\rho}_n^\ell + \mathbf{b}_r^\ell\big)\,, \\ g_{i,n}^\ell &= \kappa_r \frac{\rho_{i,n}^\ell}{\sum_{j\in\mathcal{J}_n^\ell} \rho_{j,n}^\ell}, \quad i \in \mathcal{J}_n^\ell. \end{aligned} \tag{18}$$

Here, $\mathbf{W}_r^\ell \in \mathbb{R}^{N_r\times D}$ is the router projection, $\boldsymbol{\rho}_n^\ell \in \mathbb{R}^{N_r}$ contains its element-wise sigmoid affinities, and $\mathbf{b}_r^\ell \in \mathbb{R}^{N_r}$ is the selection bias. The operator TopK_{K_r} returns the index set $\mathcal{J}_n^\ell$ of the K_r highest-scoring experts. For routed expert index i, $\rho_{i,n}^\ell$ is the corresponding affinity and $g_{i,n}^\ell$ is its mixture weight, normalized over the selected experts and scaled by κ_r. Once the expert set is selected, mixture weights use the unbiased affinities.

The MoE layer returns a feed-forward update, while the original gated residual remains outside the module:

$$\begin{aligned} \Delta\mathbf{h}_n^\ell &= \sum_{m=1}^{N_s} \mathcal{F}_m^{\text{shared},\ell}(\mathbf{u}_n^\ell) + s_r(q) \sum_{i\in\mathcal{J}_n^\ell} g_{i,n}^\ell \mathcal{F}_i^{\text{routed},\ell}(\mathbf{u}_n^\ell), \\ \mathbf{h}_n^{\ell,\text{moe}} &= \mathbf{h}_n^{\ell,\text{ca}} + \mathbf{g}_{\text{mlp}}^\ell \odot \Delta\mathbf{h}_n^\ell. \end{aligned} \tag{19}$$

Here, $\mathcal{F}_m^{\text{shared},\ell}$ and $\mathcal{F}_i^{\text{routed},\ell}$ are FFNs mapping $\mathbb{R}^D$ to $\mathbb{R}^D$, with m indexing shared experts and i indexing routed experts. At training iteration q, the multiplier $s_r(q) \in [0, 1]$ ramps up the routed branch as specified in Section C.1. The vector $\mathbf{g}_{\text{mlp}}^\ell \in \mathbb{R}^D$ is the original DiT's adaptive residual gate, distinct from expert mixture weights $g_{i,n}^\ell$. The update $\Delta\mathbf{h}_n^\ell$ produces the residual-stream token $\mathbf{h}_n^{\ell,\text{moe}}$; collecting and reshaping these tokens gives $\mathbf{H}_{\text{moe}}^\ell$ used in both injection pathways.

For load balancing (Fedus et al., 2022; DeepSeek-AI et al., 2024), let f_i^ℓ and P_i^ℓ denote the assignment fraction and mean normalized routing affinity of routed expert i in layer ℓ. With $L_{\text{M}} = 28$ MoE layers and auxiliary-loss weight λ_{MoE}, the balancing loss $\mathcal{L}_{\text{MoE}}$ and total training objective $\mathcal{L}$ are

$$\mathcal{L}_{\text{MoE}} = \frac{\lambda_{\text{MoE}}}{L_{\text{M}}} \sum_{\ell=0}^{L_{\text{M}}-1} N_r \sum_{i=1}^{N_r} f_i^\ell P_i^\ell, \qquad \mathcal{L} = \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{MoE}}. \tag{20}$$

Let N_{tok} be the number of tokens included in the routing statistics and $\mathbb{I}[\cdot]$ the indicator function, equal to one when its condition holds and zero otherwise. Then $f_i^\ell = (N_{\text{tok}} K_r)^{-1} \sum_{n=1}^{N_{\text{tok}}} \mathbb{I}[i \in \mathcal{J}_n^\ell]$ is normalized over all top-K_r assignments, and $P_i^\ell = N_{\text{tok}}^{-1} \sum_{n=1}^{N_{\text{tok}}} \rho_{i,n}^\ell / \sum_{j=1}^{N_r} \rho_{j,n}^\ell$ averages affinities normalized over all routed experts. Thus, both $\sum_i f_i^\ell$ and $\sum_i P_i^\ell$ equal one.

B.4 Efficient Autoregressive Rollouts

The distilled autoregressive model in Section 3.4 retains the task, numerical-action, and visual-action interfaces. We obtain it through causal adaptation and few-step distillation (Cheng et al., 2026; Huang et al., 2025a; Zhu et al., 2026).

Causal adaptation and few-step distillation. We initialize the causal model from the bidirectional weights and replace full temporal attention with block-causal attention. Each chunk accesses itself, prior visual chunks, and its known actions. Training proceeds through teacher-forced flow matching on clean histories, four-step student initialization from the causal teacher's ODE trajectories, and self-forcing on generated histories. Distribution matching uses the frozen bidirectional teacher (Yin et al., 2024; Huang et al., 2025a); a small fraction of teacher-forced updates is retained in the final stage.

Autoregressive inference with KV caching. The distilled model generates each chunk in four denoising steps and appends it to a rolling context window. Cached keys and values for historical tokens are reused across chunks. All downstream applications use this model. The clip-level timing result in Section 3.4 and the complete-trajectory measurements in Section C.4 use different timing scopes.

C Extended Experimental Protocols and Results

C.1 Experimental Setup

This section supplements the datasets and training configuration in Section 4.1 with data curation, split construction, and evaluator details.

Data quality curation. We discard trajectories with incomplete observations, invalid actions, missing or invalid URDFs, missing camera calibration, or inconsistent timestamps. A VLM compares rendered action videos with synchronized RGB observations for whole-arm pose, motion direction, gripper state, temporal alignment, and robot visibility. Low-scoring samples undergo automatic alignment correction, rerendering, and reassessment; those that remain inconsistent are discarded.

Evaluation datasets and splits. The within-source partitions in Section 4.1 group AgiBotWorld Beta and RoboMIND trajectories by collection session and stratify by task and embodiment. RoboTwin uses the randomized configuration (Chen et al., 2025), with disjoint randomization seeds and initial object configurations across splits. Validation data serve model selection; reported video-generation metrics use the fixed test partitions.

Implementation details. Training uses no gradient accumulation. In the MoE configuration of Section B.3, the route scale is $\kappa_r = 2.5$. The routed-branch multiplier $s_r(q)$ increases linearly from 0 to 1 over the first 2,000 iterations and remains 1 thereafter. Clip length, resolution, hardware, batch size, and the learning-rate schedule are given in Section 4.1.

Evaluation protocol and metrics. All methods use the same test partitions and their respective action representations. The video-quality metrics are specified in Section 4.1. Adapted EWM-Bench (Hu et al., 2025) additionally measures motion correctness (DYN, HSD, nDTW), semantic alignment (BLEU (Papineni et al., 2002), CLIP (Radford et al., 2021)), scene consistency (SceneC), and variation among task-consistent rollouts (Diversity).

Table 7: Adapted EWMBench results on the same test splits. Primitive scores are normalized to $[0, 1]$; higher is better. Overall denotes the adapted EWMBench overall score. The best and second-best results are bold and underlined, respectively.

Method	Motion			Semantics			Scene	Overall
	DYN ↑	HSD ↑	nDTW ↑	BLEU ↑	CLIP ↑	Div. ↑	SceneC ↑	↑
AgiBotWorld Beta								
IRASim	0.435	0.673	0.639	0.234	0.827	0.138	0.626	3.571
Ctrl-World	<u>0.556</u>	0.904	0.889	0.300	0.848	0.197	<u>0.707</u>	4.401
OSCAR	0.488	0.823	0.811	<u>0.400</u>	<u>0.904</u>	0.192	0.685	4.303
Wan-Move	0.220	0.637	0.553	0.227	0.829	0.182	0.597	3.244
EnerVerse-AC	0.436	<u>0.915</u>	<u>0.904</u>	0.365	0.886	**0.202**	**0.724**	<u>4.431</u>
Ours	**0.715**	**0.959**	**0.962**	**0.524**	**0.933**	<u>0.201</u>	0.703	**4.997**
RoboMIND								
IRASim	<u>0.301</u>	0.394	0.318	0.310	0.856	0.112	0.788	3.079
Ctrl-World	0.214	**0.938**	**0.853**	<u>0.419</u>	<u>0.873</u>	<u>0.142</u>	<u>0.837</u>	<u>4.276</u>
EnerVerse-AC	0.170	0.419	0.393	0.257	0.839	0.127	0.776	2.982
Ours	**0.591**	<u>0.838</u>	<u>0.810</u>	**0.495**	**0.926**	**0.157**	**0.849**	**4.666**
RoboTwin								
IRASim	<u>0.395</u>	0.612	0.551	0.277	0.785	0.122	0.640	3.383
Ctrl-World	0.155	<u>0.891</u>	<u>0.856</u>	<u>0.374</u>	0.826	**0.163**	**0.708**	<u>3.971</u>
EnerVerse-AC	0.293	0.675	0.615	0.333	<u>0.844</u>	0.152	<u>0.699</u>	3.610
Ours	**0.821**	**0.972**	**0.976**	**0.531**	**0.918**	<u>0.160</u>	0.666	**5.044**

VLM evaluator. The Qwen3-VL-2B-Instruct evaluator (Bai et al., 2025) uses LoRA (Hu et al., 2022) and success, progress, and validity heads, following rollout evaluation and progress estimation (Quevedo et al., 2025; Tseng et al., 2025; Wu & Gao, 2026; Feng et al., 2026). Parameters ϕ comprise the adapters and heads; the pretrained backbone is frozen. Training combines manual annotations on simulator and generated videos with same-task ranking supervision. The heads produce terminal-success confidence q_ϕ, five-level progress normalized to $[0, 1]$ as p_ϕ, and visual validity a_ϕ. The 16 input frames are uniformly sampled in temporal order, including both endpoints, and accompanied by an instruction and success/failure checklist. All evaluator weights remain fixed downstream. The selection utility in Section 4.3 uses λ to weight progress; annotation and validation protocols follow in Section C.5.

C.2 Additional World-Model Comparisons

Table 7 extends the video-quality comparison in Table 1 with the full adapted EWMBench metrics under the same baseline protocol.

Compared with the strongest baseline per dataset, DYN improves by 0.159, 0.290, and 0.426 on AgiBotWorld Beta, RoboMIND, and RoboTwin. The component scores also reveal the trade-offs behind the overall ranking: HSD/nDTW on RoboMIND and SceneC/Diversity on AgiBotWorld Beta and RoboTwin remain below their best baselines.

C.3 Ablation Studies and Design Analysis

Mixed-domain training and sparse MoE. Table 8 adds SSIM and FID to the cumulative ablation in Table 3. Action conditions, injection architecture, optimization schedule, and training iterations are fixed. The comparison changes corpus size and composition, then model capacity, as discussed in Section 4.2.

Contribution of complementary action conditions. Table 9 reports all five metrics for the conditioning variants summarized in Table 2. The no-action variant removes both inputs. Visual con-

Table 8: Cumulative ablation of mixed-domain training and sparse MoE on the held-out test split of AgiBotWorld Beta. Dense and Sparse MoE denote the backbone variants with dense feed-forward layers and sparse MoE layers, respectively. All variants use the same action conditions, injection architecture, optimization schedule, and number of training iterations. The best results are highlighted in bold.

Training setting	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**FID ↓**	**FVD ↓**
Real only (Dense)	20.514	0.7512	0.0186	23.870	20.410
Real + Simulation (Dense)	21.062	0.7668	0.0169	20.620	17.380
Real + Simulation + Sparse MoE	**22.276**	**0.7960**	**0.0125**	**14.640**	**10.850**

Table 9: Ablation of the complementary action conditions on the held-out test splits of AgiBotWorld Beta and RoboMIND. The no-action baseline evaluates video generation without numerical or visual action inputs. All variants use the same backbone, training data, and evaluation protocol. The best results are highlighted in bold.

Setting	**Action value**	**Action video**	**AgiBotWorld Beta**			**RoboMIND**		
			PSNR ↑	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
No action condition	×	×	17.684	0.6513	0.0287	19.640	0.7195	0.0248
Action value only	✓	×	19.238	0.7126	0.0214	21.356	0.7768	0.0176
Action video only	×	✓	21.276	0.7668	0.0158	23.102	0.8209	0.0132
Full dual condition	✓	✓	**22.276**	**0.7960**	**0.0125**	**23.850**	**0.8350**	**0.0115**

ditioning outperforms numerical conditioning alone, while their combination improves all reported metrics on both datasets.

Injection architecture. Table 10 extends the injection comparison to all five metrics. Dual injection achieves the highest scores; the selected interleaving uses one conditioning pathway per block. Section 4.2 quantifies the small observed quality difference.

C.4 Data Generation

We provide generation timings and protocols for the two data-augmentation studies in Section 4.3, following generated-data policy training (He et al., 2026). Trajectory counts are per task. Within each policy study, all variants share 200 unseen evaluation seeds per task under RoboTwin's randomized configuration, disjoint from training-data construction.

Table 10: Ablation of action-condition injection architectures on the held-out test splits of AgiBotWorld Beta and RoboMIND. Dual injection instantiates both conditioning branches at every block, whereas interleaved variants alternate the branches across blocks. The best results are highlighted in bold, and the selected configuration is identified by its bold method name.

Injection architecture	**AgiBotWorld Beta**			**RoboMIND**		
	PSNR ↑	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
Token concat.	21.214	0.7601	0.0164	21.903	0.7815	0.0170
Late fusion	21.372	0.7664	0.0155	22.931	0.8108	0.0138
Reverse interleave	22.241	0.7953	0.0126	23.821	0.8344	0.0116
Interleaved (ours)	22.276	0.7960	0.0125	23.850	0.8350	0.0115
Dual every block	**22.351**	**0.7976**	**0.0123**	**23.901**	**0.8362**	**0.0114**

Table 11: Data generation efficiency of Pelican-Sim 1.0 compared with manual collection and RoboTwin simulation for complete trajectories at matched horizons. The distilled autoregressive model uses four denoising steps per chunk in both comparisons. Times cover generation or execution and recording; nominal throughput excludes upstream preparation and the separately reported manual reset time. Zero reset time denotes no manual intervention between episodes.

Data source	Resource	Sec./traj. ↓	Traj./hour ↑	Reset time (min./traj.) ↓
Real-world comparison				
Manual collection	Robot + operator	60	60	0.4
Pelican-Sim 1.0	GPU	24	150	0
Simulation comparison				
RoboTwin simulator	Simulator + GPU	55	65.5	0
Pelican-Sim 1.0	GPU	19	189.5	0

Table 12: Cross-domain data augmentation for downstream $\pi_{0.5}$ policy learning. Per task, Pelican-Sim augmentation combines 50 clean demonstrations with 500 generated rollouts from edited initial observations; direct collection uses 50 randomized simulator trajectories. Identically configured policies are evaluated under RoboTwin's randomized configuration, with success averaged over five tasks and 200 unseen seeds per task.

Training data	Seed traj.	Aug. / seed	Total traj.	Success (%)
RoboTwin randomized	50	0	50	70.0
Pelican-Sim augmentation	50	10	550	72.0

Generation efficiency. We compare the distilled autoregressive Pelican-Sim 1.0 model, RoboTwin simulation, and manual collection for complete task trajectories, rather than individual video clips, at a matched trajectory horizon. Timing includes inference and video serialization for Pelican-Sim 1.0, stepping, rendering, and serialization for RoboTwin, and physical execution and recording for manual collection. Nominal throughput is derived from these times, excluding manual reset and upstream preparation such as model training, simulator construction, and initial-frame editing. We separately report reset time for manual scene reset and preparation between episodes, excluding teleoperation during execution.

Table 11 reports the two comparisons separately. The measured nominal throughput is $2.50\times$ that of manual collection and $2.89\times$ that of RoboTwin. Manual reset time is listed separately and excluded from both ratios.

Cross-domain augmentation. We compare two data-construction strategies through downstream policy performance: augmenting clean demonstrations with Pelican-Sim 1.0 and collecting randomized demonstrations directly in RoboTwin. Starting from 50 trajectories per task in the non-randomized `demo_clean` configuration (Chen et al., 2025), we use GPT-Image-2 (OpenAI, 2026) to produce ten initial-frame variants resembling `demo_randomized` scenes. Pelican-Sim 1.0 generates a rollout from each variant using the original actions. We train a $\pi_{0.5}$ VLA policy (Physical Intelligence et al., 2025) on the 50 clean and 500 generated trajectories, and compare it with an identically configured policy trained on 50 directly collected randomized trajectories per task.

Table 12 reports the resulting policy success rates. The two strategies use unequal training-data budgets.

Scaling with limited demonstrations. We next evaluate augmentation with 10, 30, or 50 original randomized trajectories per task. Using the same strategy as in the cross-domain study, we generate ten additional rollouts per original trajectory. We compare $\pi_{0.5}$ policies trained on the originals alone with those trained on augmented sets of 110, 330, and 550 trajectories per task.

Table 13: Effect of Pelican-Sim 1.0 augmentation on $\pi_{0.5}$ policy success at three demonstration budgets. Original-only training is compared with adding ten generated rollouts per original trajectory. Counts are per task; evaluation uses RoboTwin's randomized configuration with 200 unseen seeds per task. Parenthesized values report percentage-point gains over the corresponding original-only baseline.

Setting	Original traj.	Generated traj.	Total traj.	Success (%)
Original only	10	0	10	28.5
Original + Pelican-Sim 1.0	10	100	110	64.5 (↑ 36.0)
Original only	30	0	30	57.0
Original + Pelican-Sim 1.0	30	300	330	87.0 (↑ 30.0)
Original only	50	0	50	70.0
Original + Pelican-Sim 1.0	50	500	550	93.0 (↑ 23.0)

Table 13 gives the complete data budgets and gains. The largest absolute success-rate gain occurs at the lowest demonstration budget.

C.5 Policy Evaluation and Ranking

Following world-model-based policy evaluation (Quevedo et al., 2025; Tseng et al., 2025; Jeon et al., 2026), we use five checkpoints $\{\pi_k\}_{k=1}^{5}$ from different stages of one VLA training run. Their trajectories support both evaluator validation and comparison of the complete world-model–VLM pipeline with RoboTwin.

Validation of the VLM evaluator. For each policy, we obtain paired RoboTwin and Pelican-Sim 1.0 rollouts on held-out initial conditions using the same action sequences and initial observations. The validation set contains 1,000 simulator rollouts across five tasks, with binary task-success labels provided by the RoboTwin task checker and human-annotated progress references. Three annotators label a score-stratified subset of 700 corresponding generated videos, with majority vote determining task success and visual validity. Progress labels for simulator and generated videos follow the same annotation protocol: annotators rate how fully the task requirements are satisfied at the end of each video on a five-level scale, and the median rating is normalized to $[0, 1]$ as the reference score. Each video is assessed on its own depicted outcome; shared annotation criteria do not imply identical labels for paired rollouts. We report accuracy, balanced accuracy, F1, AUROC, and expected calibration error (ECE) (Guo et al., 2017) for task-success prediction; MAE and Spearman correlation for progress estimation; and AUROC for visual validity.

Table 14 extends Table 5 with balanced accuracy, F1, progress correlation, and validity metrics. It assesses the evaluator's judgments of depicted outcomes, independently of prediction fidelity.

Policy evaluation and ranking protocol. Each policy is evaluated on the same $N = 200$ held-out RoboTwin initial conditions. For each condition, the same policy-predicted action trajectory is executed in RoboTwin and supplied to Pelican-Sim 1.0. Success is determined by the RoboTwin checker for simulator trials and by the frozen VLM for generated trials. This matched-action protocol evaluates outcome prediction for fixed trajectories, rather than closed-loop policy execution under generated observations. We vary the world-model adaptation budget over 0, 100, 200, 500, and 1,000 task-specific RoboTwin rollouts, with nonzero budgets balanced across the five policies.

For policy π_k, the estimated success rate is $\mathrm{SR}_k^{\mathrm{wm}} = N^{-1} \sum_{i=1}^{N} z_\phi(\widehat{\mathbf{V}}_{k,i}, \mathbf{y}_i)$, where $\widehat{\mathbf{V}}_{k,i}$ is the generated rollout including the initial frame for evaluation condition i, and $\mathbf{y}_i$ is its instruction. The binary decision is $z_\phi = \mathbb{I}[q_\phi \geq \tau_{\mathrm{succ}}]$, with τ_{succ} denoting the terminal-success threshold. We compare the estimate with the corresponding RoboTwin success rate using Pearson r for linear correlation, Spearman ρ and Kendall τ for ranking consistency, mean maximum rank violation (MMRV) (Li et al., 2025) for ordering errors, and MAE in percentage points for absolute success-rate error (Tseng et al., 2025).

Table 14: Validation of the fine-tuned Qwen3-VL-2B-Instruct evaluator on simulator and generated videos. Upper: terminal-success prediction; lower: progress estimation and visual validity. Success references use the RoboTwin task checker or human majority vote for generated videos; progress and validity use human medians and majority vote, respectively. Threshold-dependent metrics use disjoint calibration data. Each video is judged on its depicted outcome, independently of world-model prediction accuracy.

Evaluation input	N	Accuracy (%) ↑	Balanced acc. (%) ↑	F1 (%) ↑	AUROC ↑	ECE ↓
RoboTwin videos	1,000	92.6	91.8	92.1	0.964	0.041
Pelican-Sim 1.0 videos	700	91.4	90.8	91.0	0.951	0.048

Evaluation input	N	Progress MAE ↓	Progress ρ ↑	Validity AUROC ↑
RoboTwin videos	1,000	0.061	0.926	–
Pelican-Sim 1.0 videos	700	0.078	0.901	0.953

Table 15: Policy evaluation and ranking with the Pelican-Sim 1.0–VLM pipeline across five world-model adaptation budgets. Agreement with RoboTwin is measured for five policy checkpoints using matched action trajectories on 200 held-out initial conditions each, with the VLM frozen. MAE reports success-rate error in percentage points; MMRV measures ordering errors using success rates in $[0, 1]$, with zero indicating no ranking violation.

Adaptation rollouts	Pearson r ↑	Spearman ρ ↑	Kendall τ ↑	MMRV ↓	MAE (pp) ↓
0	0.9020	0.800	0.600	0.162	8.6
100	0.9580	0.900	0.800	0.074	5.3
200	0.9760	**1.000**	**1.000**	**0.000**	4.0
500	0.9890	**1.000**	**1.000**	**0.000**	2.9
1,000	**0.9940**	**1.000**	**1.000**	**0.000**	**2.4**

Policy evaluation and ranking results. Table 15 adds Kendall τ and MMRV to the adaptation results in Table 5. Ranking agreement saturates before absolute success-rate accuracy. Subsequent selection and optimization studies use the 500-rollout adapted model.

C.6 Action Selection

Candidate generation and selection. Given an initial observation $\mathbf{x}_0$ and instruction $\mathbf{y}$, Diffusion Policy (Chi et al., 2023) samples $M = 10$ action sequences over a common horizon using independent diffusion noise. Following Best-of-N selection (Alzayer et al., 2026), the distilled autoregressive Pelican-Sim 1.0 model, adapted on 500 rollouts, predicts each candidate's outcome, and the frozen VLM evaluator ranks the resulting videos using u_ϕ from Section 4.3:

$$m^* = \underset{m\in\{1,\dots,M\}}{\operatorname{argmax}} \; u_\phi\Big(\widehat{\mathbf{V}}^{(m)}, \mathbf{y}\Big), \qquad \widehat{\mathbf{X}}^{\mathrm{tar},(m)} \sim p_\theta\Big(\cdot \mid \mathbf{x}_0, \mathbf{A}^{(m)}, \mathbf{y}\Big). \tag{21}$$

Here, $\mathbf{A}^{(m)}$ is candidate m in the unified configuration layout, $\widehat{\mathbf{X}}^{\mathrm{tar},(m)}$ is its predicted future video, and $\widehat{\mathbf{V}}^{(m)} = [\mathbf{x}_0; \widehat{\mathbf{X}}^{\mathrm{tar},(m)}]$ is the complete video scored by the evaluator. The selected index is m^*, and only $\mathbf{A}^{(m^*)}$ is executed in RoboTwin, open-loop without replanning.

Evaluation protocol and baselines. We evaluate five RoboTwin tasks using 200 held-out initial conditions per task, disjoint from world-model adaptation and VLM-evaluator calibration. All methods share candidate pools. Baselines include *Single sample* (the first candidate), *Random-of-10* (uniform selection), *VLM-direct* (scoring the initial observation and rendered action condition without future prediction), and *Ctrl-World selection* (Guo et al., 2025) (replacing Pelican-Sim 1.0 as the rollout generator). The two world-model selectors share the adaptation budget, prediction

Table 16: Action selection results averaged over five RoboTwin tasks. All methods use the same Diffusion Policy candidate pools and 200 held-out initial conditions per task. Ctrl-World and Pelican-Sim 1.0 use the same adaptation budget and VLM evaluator; Oracle Best-of-10 uses simulator outcomes only as an upper bound.

Method	Predicted future	Success (%) ↑	ΔSR (pp) ↑	Oracle gap (pp) ↓	Selection recall (%) ↑
Single sample	–	43.2	0.0	24.9	63.4
Random-of-10	–	43.5	+0.3	24.6	63.9
VLM-direct	–	48.7	+5.5	19.4	71.5
Ctrl-World selection	✓	<u>52.3</u>	<u>+9.1</u>	<u>15.8</u>	<u>76.8</u>
Pelican-Sim 1.0 selection	✓	**63.8**	**+20.6**	**4.3**	**93.7**
Oracle Best-of-10	RoboTwin	68.1	+24.9	0.0	100.0

Table 17: Test-time scaling of Pelican-Sim 1.0 action selection with the number of Diffusion Policy candidates. Candidate sets are nested across M. Latency covers world-model generation and VLM scoring under a fixed four-GPU setup. Any-success is the oracle success rate within each candidate pool; selection recall is conditional on the pool containing a successful candidate.

M	Any-success (%) ↑	Selected success (%) ↑	Selection recall (%) ↑	Sec./decision ↓
1	43.2	43.2	100.0	5.2
2	52.6	51.2	97.3	5.3
4	60.4	58.0	96.0	5.5
8	66.4	62.5	94.1	10.7
10	**68.1**	**63.8**	93.7	15.9

horizon, output resolution, and VLM evaluator. *Oracle Best-of-10* provides a candidate-pool upper bound using task-checker outcomes from executions of all candidates in cloned simulator states; these outcomes are unavailable to the learned selectors.

We report task-checker success rate, improvement over Single sample, and oracle gap, with differences in percentage points. Any-success@M is the fraction of pools containing a successful candidate, while selection recall is the success rate conditional on such a pool.

Table 16 supplements the success rates in Table 4 with selection recall and oracle gaps.

Test-time scaling. We vary $M \in \{1, 2, 4, 8, 10\}$ using nested candidate pools and measure world-model generation and VLM scoring latency on four GPUs. Table 17 reports candidate-pool coverage, selection recall, success, and latency. Recall at $M = 1$ is 100% by construction; its decline for larger pools reflects missed successful candidates despite increasing overall success.

C.7 Policy Improvement

Following World-Gymnast (Sharma et al., 2026), we initialize from a task-specific OpenVLA-OFT checkpoint (Kim et al., 2025; 2024) and replace its L1 regression head with a Llama-2 (Touvron et al., 2023) language-model head to obtain action-token probabilities while retaining action-chunk prediction. The distilled autoregressive world model, adapted on 500 rollouts, and the VLM evaluator remain frozen throughout policy optimization.

Policy-to-world-model action interface. The policy predicts fixed-length chunks of joint-angle targets and gripper or hand states. We decode and denormalize the actions, then pack them into the 28-dimensional layout with zero-filled unused entries (Section B.2). Each chunk is interpolated to 21 configurations, including the current configuration at the conditioning frame, and rendered through the URDF pipeline in Figure 2. This produces synchronized numerical and visual action conditions. The final commanded configuration initializes the next transition; no privileged object or simulator state is provided to the policy.

Closed-loop world-model rollouts. Let ϑ denote the trainable policy parameters, distinct from world-model parameters θ and evaluator parameters ϕ, and let ϑ_{old} denote the behavior-policy parameters used to collect a rollout group. For each instruction $\mathbf{y}$ and initial observation $\mathbf{x}_0$, we sample K imagined trajectories over H decision steps. At decision index $h \in \{0, \dots, H-1\}$ in trajectory $k \in \{1, \dots, K\}$, the policy samples an action chunk $\mathbf{c}_{h,k} \sim \pi_{\vartheta_{\text{old}}}(\cdot \mid \widehat{\mathbf{x}}_{h,k}, \mathbf{y})$, with $\widehat{\mathbf{x}}_{0,k} = \mathbf{x}_0$. The chunk is converted into frame-aligned configurations $\mathbf{A}$ as described above; unlike $\mathbf{a}_t$ in Section B.1, $\mathbf{c}_{h,k}$ denotes an entire policy action chunk rather than one configuration. Pelican-Sim 1.0 predicts the corresponding observation segment, whose final frame becomes $\widehat{\mathbf{x}}_{h+1,k}$. The imagined trajectories $\widehat{\tau}_k = (\mathbf{x}_0, \mathbf{c}_{0,k}, \widehat{\mathbf{x}}_{1,k}, \dots, \mathbf{c}_{H-1,k}, \widehat{\mathbf{x}}_{H,k})$ support closed-loop policy optimization without additional RoboTwin interaction.

VLM-based dense reward. Returns R_k follow Equation 8. The evaluator scores J successive prefixes: $p_{\phi,j,k}$ and $a_{\phi,j,k}$ are progress and validity for prefix j of trajectory k, while $q_{\phi,k}$ is full-trajectory success confidence. We set $p_{\phi,0,k} = 0$, reject rollouts below a terminal-validity threshold, and resample groups with insufficient valid samples or negligible return variance.

GRPO policy optimization. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) normalizes returns within each rollout group to obtain trajectory-level advantages $\widehat{A}_k$:

$$\widehat{A}_k = \frac{R_k - \mu_R}{\sigma_R + \epsilon}, \qquad \mu_R = \frac{1}{K}\sum_{k=1}^{K} R_k, \tag{22}$$

where μ_R and σ_R are the within-group mean and standard deviation, respectively, and $\epsilon > 0$ ensures numerical stability. Each action chunk receives the trajectory-level advantage. We maximize the clipped objective

$$\mathcal{J}_{\text{GRPO}}(\vartheta) = \mathbb{E}\left[\frac{1}{KH}\sum_{k=1}^{K}\sum_{h=0}^{H-1} \min\Big(\rho_{h,k}(\vartheta)\widehat{A}_k, \text{clip}(\rho_{h,k}(\vartheta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\,\widehat{A}_k\Big)\right], \tag{23}$$

where $\rho_{h,k}(\vartheta) = \pi_\vartheta(\mathbf{c}_{h,k} \mid \widehat{\mathbf{x}}_{h,k}, \mathbf{y}) / \pi_{\vartheta_{\text{old}}}(\mathbf{c}_{h,k} \mid \widehat{\mathbf{x}}_{h,k}, \mathbf{y})$ is the action-chunk probability ratio. The operator clip restricts this ratio to $[1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}]$, with ϵ_{low} and ϵ_{high} controlling the lower and upper clipping margins. The expectation is over rollout groups sampled from the behavior policy and the training initial conditions and instructions.

Baselines and evaluation. We evaluate five RoboTwin tasks: *Open Laptop*, *Open Microwave*, *Place Container Plate*, *Stack Bowls Three*, and *Press Stapler*. All variants share the same supervised initialization. We compare the initial SFT policy, Iter-SFT on VLM-filtered successful rollouts, Pelican-Sim 1.0-GRPO with binary terminal reward, and the full reward in Equation 8. Training variants use matched tasks and optimization budgets, with matched world-model rollout budgets where applicable. RoboTwin-GRPO uses simulator task-checker feedback as an oracle reference. Final policies are evaluated in RoboTwin's randomized configuration on shared unseen seeds, using the task checker rather than the training VLM. We report per-task and mean success rates, averaged over three independent policy-training seeds.

Table 18 provides the per-task results behind the averages in Table 4. Gains over supervised initialization occur on all five tasks, with the largest on *Stack Bowls Three*. The terminal-only comparison tests the combined reward design rather than isolating progress and validity.

C.8 Generalization under Distribution Shifts

We provide the protocols and examples for the five qualitative shifts summarized in Section 4.4. Task semantics and action horizon remain fixed while trajectory, scene, embodiment, object, or viewpoint changes.

Trajectory OOD. We reverse a rollout's frame-aligned configuration sequence and use its terminal observation as the new conditioning frame. Figure 4 shows motions following the reversed commands toward the initial robot configuration. This tests sensitivity to the supplied trajectory without assuming exact physical reversibility of contact interactions.

Table 18: Policy improvement on five RoboTwin tasks under the randomized configuration. Success rates are measured on shared unseen seeds and averaged over three policy-training runs; RoboTwin-GRPO is a simulator-based oracle reference.

Method	RoboTwin success (%) ↑					
	Open Laptop	Open Microwave	Place Container Plate	Stack Bowls Three	Press Stapler	Mean ↑
Initial OpenVLA-OFT (SFT)	67.8	58.3	71.2	53.7	62.5	62.7
Iter-SFT (VLM-filtered)	71.5	62.7	74.8	58.8	67.3	67.0
RoboTwin-GRPO (oracle)	88.2	79.5	90.0	76.8	84.3	83.8
Pelican-Sim 1.0-GRPO (terminal only)	72.8	63.4	76.2	59.7	70.4	68.5
Pelican-Sim 1.0-GRPO (full)	**79.2**	**71.0**	**82.3**	**68.0**	**76.5**	**75.4**

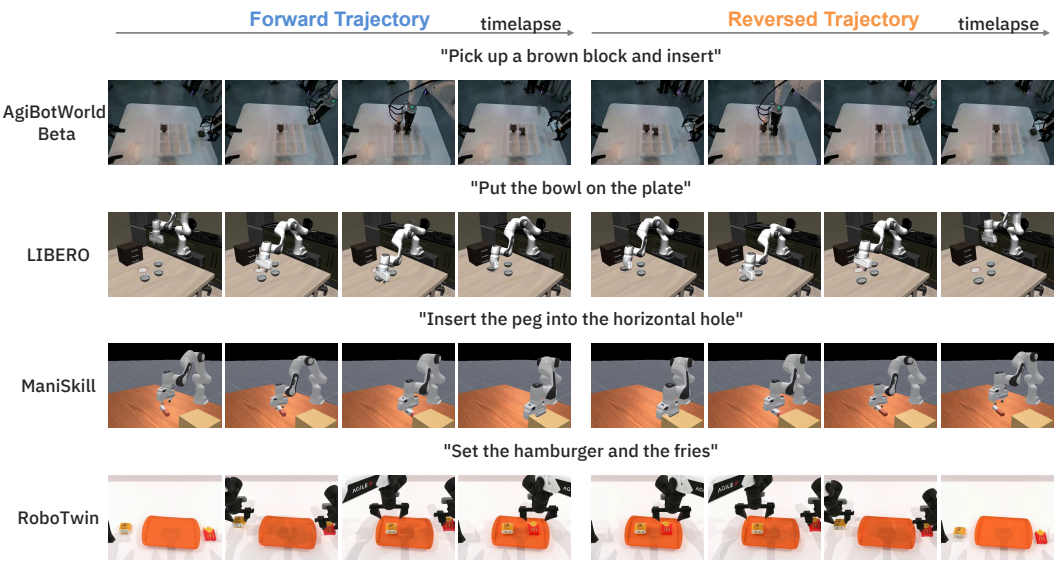


Figure 4: **Trajectory OOD.** Forward and reversed action-conditioned rollouts across four real-world and simulated benchmarks.

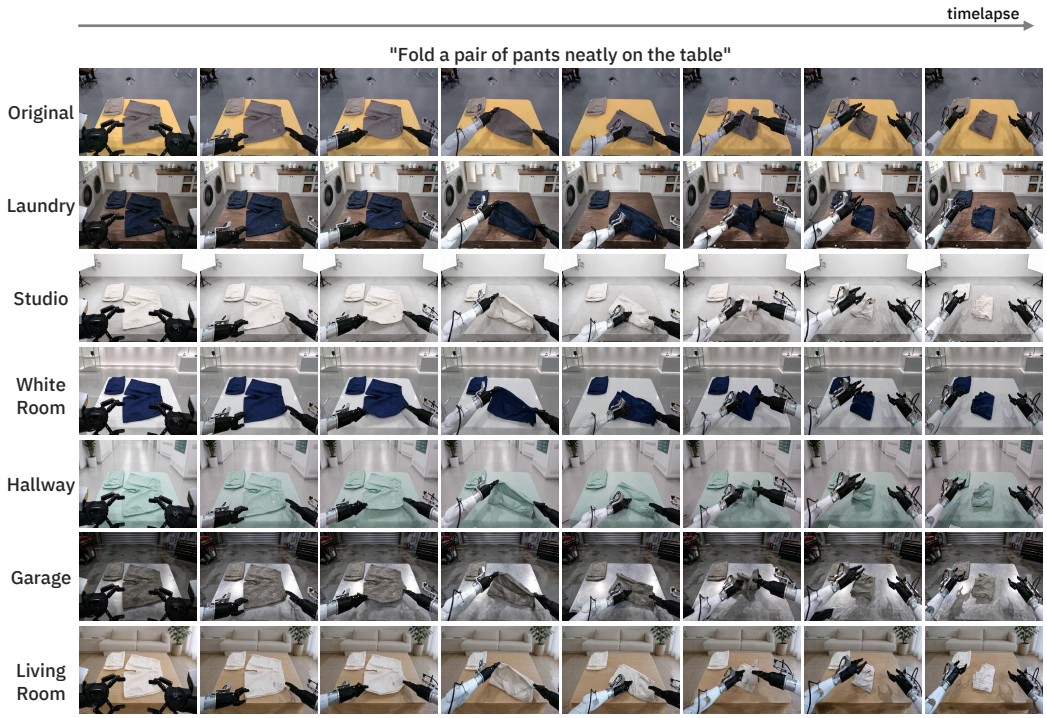


Figure 5: **Scene/appearance OOD.** Pants-folding rollouts in the original scene and six image-edited environments.

Scene/appearance OOD. GPT-Image-2 (OpenAI, 2026) edits the conditioning frame while preserving the task, robot pose, and spatial layout. We reuse the original actions for six variants spanning background, illumination, texture, and local object appearance. Figure 5 shows consistent folding behavior under these combined appearance changes.

Robot-embodiment OOD. We extract the source Panda end-effector pose trajectory and solve inverse kinematics for Sawyer, IIWA, and UR5e, which are excluded from training. The resulting joint trajectories are converted to unified configurations and target-camera skeletons. Figure 6 shows block-stacking motions consistent with the retargeted trajectories.

Object OOD. GPT-Image-2 (OpenAI, 2026) replaces the manipulated object while preserving robot configuration, coarse task-relevant spatial relations, and actions. Figure 7 shows five variants in appearance, shape, and scale being moved toward the table center with largely consistent visual identity. The edits specify visual variation, not equivalent physical properties.

Viewpoint OOD. For each of five viewpoints excluded from training, we render action conditions using the target camera calibration. Figure 8 varies elevation, azimuth, and distance while retaining the task of moving the black bowl from between the plate and ramekin onto the plate. The displayed rollouts remain consistent with the commanded motion.

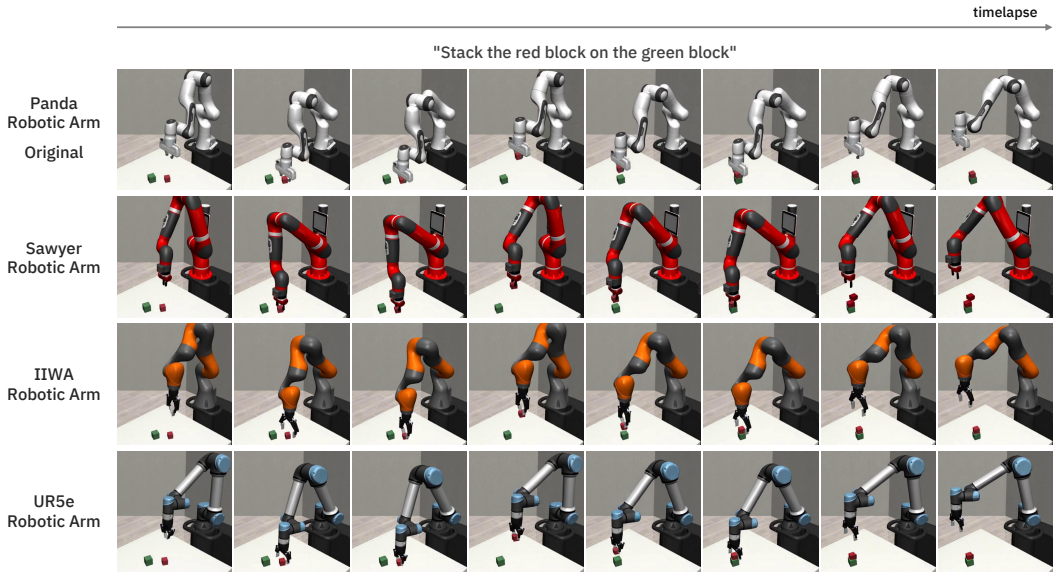


Figure 6: **Robot-embodiment OOD.** Block-stacking rollouts retargeted from Panda to held-out Sawyer, IIWA, and UR5e embodiments.

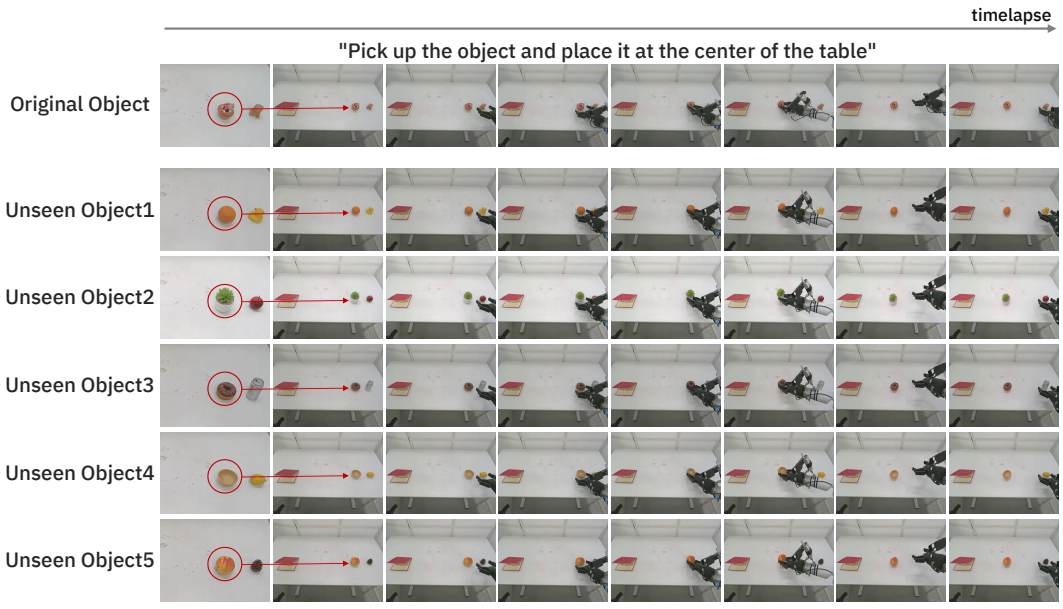


Figure 7: **Object OOD.** Pick-and-place rollouts for the original object and five image-edited object variants.

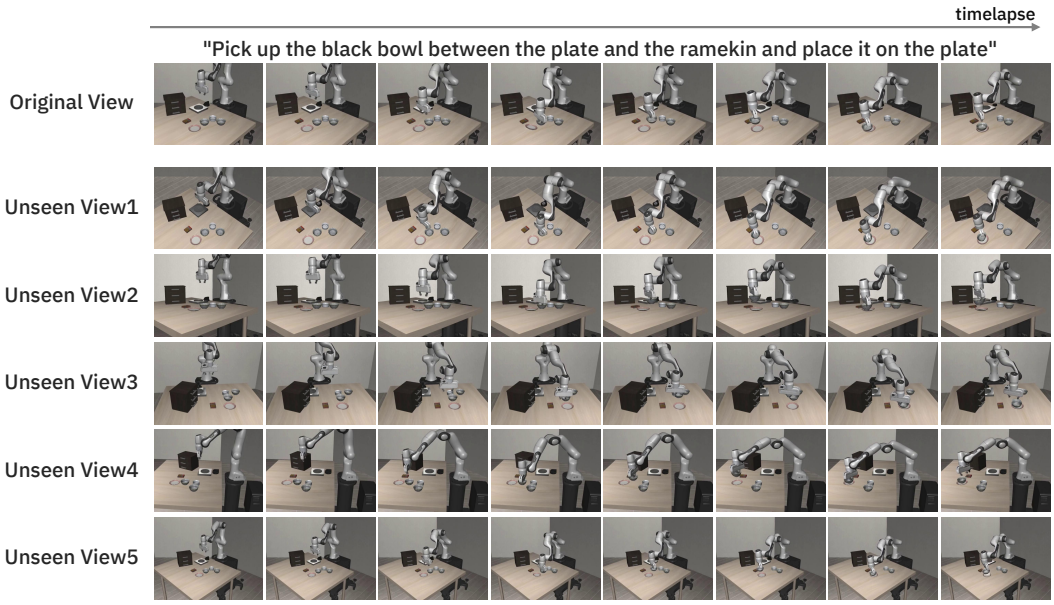


Figure 8: **Viewpoint OOD.** Action-conditioned rollouts from the original camera and five unseen viewpoints.

D MULTI-SOURCE QUALITATIVE RESULTS

Figures 9 and 10 compare predicted rollouts with synchronized action videos and ground truth across the seven training sources, covering single-arm and bimanual robots with parallel grippers and dexterous hands.

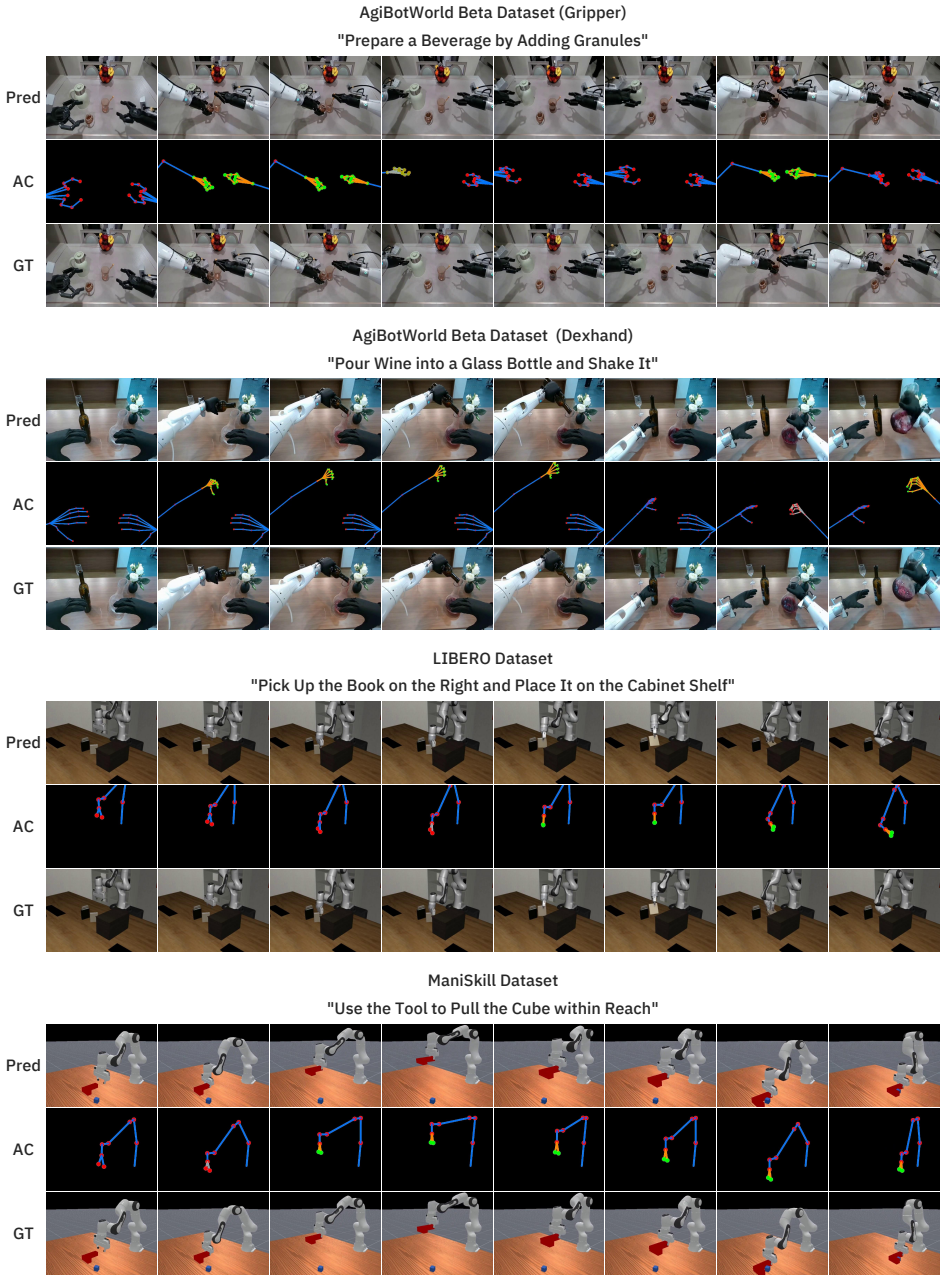


Figure 9: **Multi-source rollouts (Part I).** AgiBotWorld Beta, LIBERO, and ManiSkill. Rows: prediction (Pred), action video (AC), and ground truth (GT); time advances left to right.

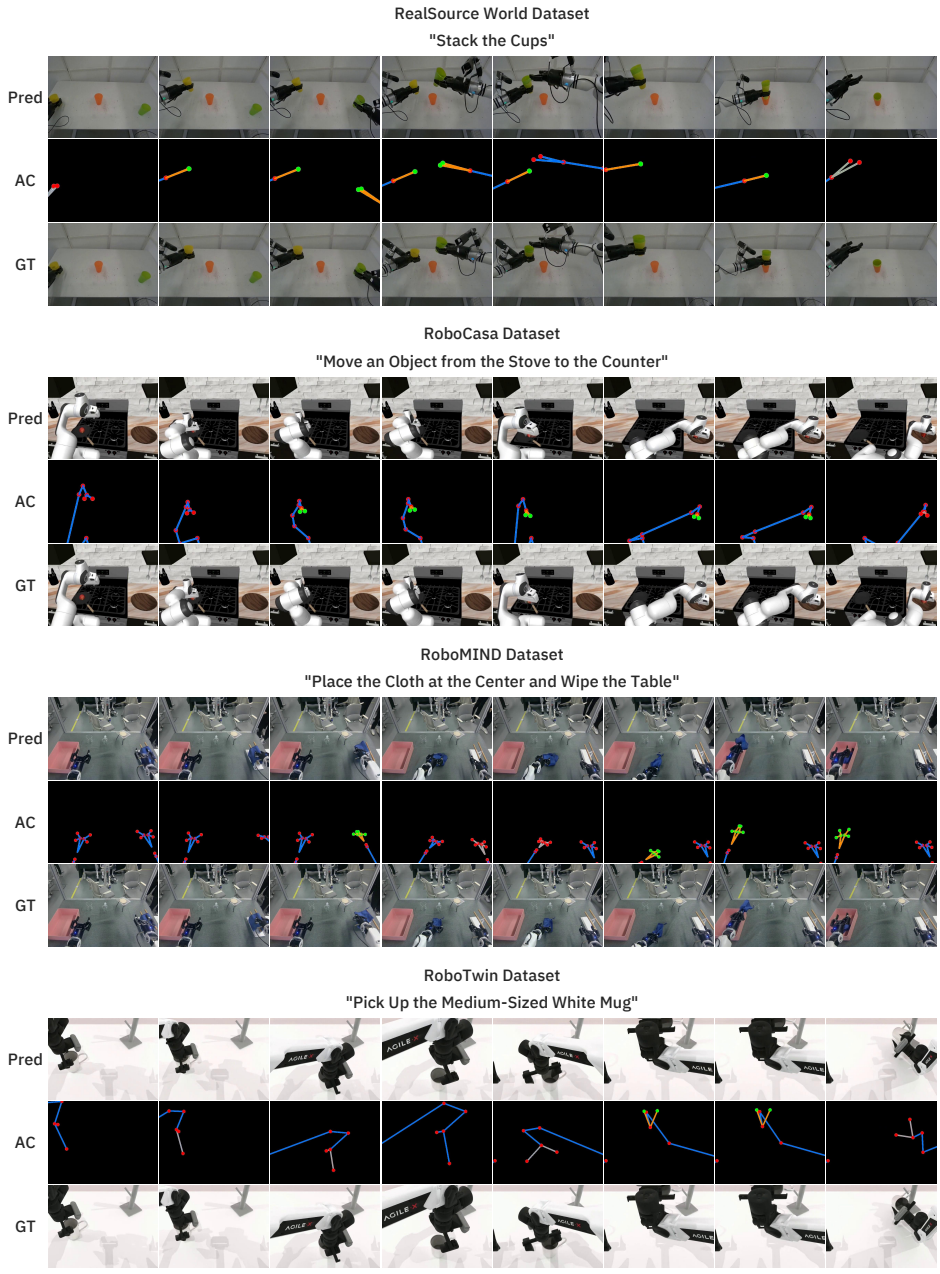


Figure 10: **Multi-source rollouts (Part II).** RealSource World, RoboCasa, RoboMIND, and RoboTwin. Rows: Pred (prediction), AC (action video), GT (ground truth); time runs left to right.