

Pelican-Sim 1.0: A General World Model Simulator for Embodied Intelligence

Beijing Innovation Center of Humanoid Robotics (X-Humanoid)

WFM System Group

{jack.zou,vito.dai,jian.tang,jason.ju}@x-humanoid.com

GitHub: [Open-X-Humanoid/Pelican-Sim1.0](https://github.com/Open-X-Humanoid/Pelican-Sim1.0)

Project page: Pelican-Sim1.0.github.io

September 11, 2026



Figure 1: **Overview of Pelican-Sim 1.0.** **Top:** Trained on approximately one million real-world and simulated trajectories, Pelican-Sim 1.0 predicts future observations from an initial RGB image, unified 28-dimensional action values, and camera-aligned action videos using a Video DiT with sparse MoE layers. Four downstream applications are illustrated: data generation for policy training, policy evaluation and ranking, action selection, and policy improvement. A fine-tuned VLM evaluator supplies success estimates, candidate scores, and rewards for the latter three applications, respectively; policy improvement uses closed-loop imagined interaction. **Middle:** Qualitative examples illustrate generalization across five dimensions: reversed trajectories, edited scenes and object variants, held-out robot embodiments, and novel viewpoints. **Bottom:** Adapted EWM-Bench overall scores summarize generation performance. Separate RoboTwin studies evaluate data generation through demonstration augmentation (Data Generation), action selection, and policy improvement against their respective baselines.

Abstract

In this technical report, we propose **Pelican-Sim 1.0**, a general world model simulator for embodied intelligence that predicts future observations from visual context and robot actions to support downstream learning and decision making. The model incorporates four key design features: **(1) Unified action representation**: a 28-dimensional action value space covering most mainstream embodiments, keeping one model valid across heterogeneous devices. **(2) Action-visual injection**: URDF- and camera-rendered action videos bridge actions and pixels, giving markedly better controllability across embodiments, scenes, and tasks (PSNR +0.904 over alternative fusion baselines). **(3) Sparse mixture-of-experts (MoE)**: sparse MoE layers add capacity for heterogeneous dynamics and absorb the action modality while reducing inter-modality conflict (FVD -6.530 vs. the dense backbone). **(4) Efficient rollout generation**: causal adaptation and few-step distillation yield a four-step autoregressive simulator, achieving a $5.67\times$ speedup over the 35-step model.

Benefiting from these designs, we train on approximately one million real-world and simulated trajectories and obtain large gains in action controllability and video quality: PSNR improves over the strongest evaluated baselines by 4.636 on AgiBot-World Beta, 2.080 on RoboMIND, and 10.343 on RoboTwin, with the adapted EWMBench DYN score up 0.426 on RoboTwin. Relying on this, four downstream applications on RoboTwin succeed: 500 generated trajectories added to 50 demonstrations per task raise policy success from 70% to 93%; policy evaluation reaches a Pearson correlation of 0.994 across five checkpoints; and relative success gains reach 47.7% for action selection and 20.3% for policy improvement. Qualitative generalization across trajectory, scene, object, embodiment, and viewpoint shifts highlights its potential as a general-purpose world model simulator.

1. Introduction

Action-conditioned world models predict future observations from visual context and candidate robot actions, providing learned simulators for evaluating behavior before execution. Such models can support data generation, policy evaluation, and policy improvement while reducing reliance on additional environment interaction (Yang et al., 2024; Escontrela et al., 2023). Extending this capability across heterogeneous robots requires more than visually realistic prediction: the model must interpret numerical commands, account for embodiment geometry, and predict their effects in diverse scenes. A central challenge is therefore to represent robot motion in a form that preserves precise configuration information while providing transferable geometric guidance.

Existing methods use numerical, visual, or hybrid action interfaces (Table 1). IRASim (Zhu et al., 2025) and Ctrl-World (Guo et al., 2025) condition video generation on low-dimensional actions. Numerical configurations preserve precise joint and gripper states but lack explicit image-space motion guidance (Wang et al., 2025); even with a unified layout, the model must learn their visual effects for each robot and camera. Visual conditions make motion explicit in image space but emphasize different information: end-effector maps provide local pose guidance (Jiang et al., 2025), embodiment masks encode projected occupancy (Chen et al., 2026b; Gu et al., 2026), and skeletons expose articulated structure (Wang et al., 2025; Wu & Gao, 2026). Figure 2 illustrates these geometric cues. End-effector maps leave whole-arm articulation underconstrained, while binary robot masks emphasize projected occupancy rather than explicit joint connectivity. Skeletons expose articulated structure, but, like masks, do not generally determine the underlying numerical configuration uniquely from image-space projections. Hybrid methods such as EnerVerse-AC (Jiang et al., 2025) and ViPSim (Chen et al., 2026a) motivate combining these complementary sources. These complementary properties suggest that cross-embodiment action conditioning should jointly preserve precise numerical configurations and explicit image-space motion structure.

To this end, we propose **Pelican-Sim 1.0**, an action-conditioned world model simulator that integrates synchronized numerical and visual action representations. A unified 28-dimensional action value specifies arm-joint and parallel-gripper or dexterous-hand states for single-arm and bimanual robots. The corresponding action video renders whole-arm joint locations and link connectivity using the robot URDF and camera calibration, providing geometric guidance grounded in nominal forward kinematics and camera projection. The action-value and action-video pathways inject numerical and visual features, respectively, into alternating blocks of the same Video Diffusion Transformer (DiT). By jointly preserving precise configurations and their projected motion structure, this design provides a common conditioning interface for training on heterogeneous robot data and enables action-conditioned video generation across embodiments.

The unified action interface makes heterogeneous trajectories compatible as model inputs, but their dynamics still vary across embodiments, tasks, and scenes. We equip the Video DiT with sparse mixture-of-experts (MoE) layers to expand its capacity for modeling heterogeneous robot dynamics (Lepikhin et al., 2021; Riquelme et al., 2021; Fei et al., 2024; Ma et al., 2026). Routed experts provide transformations selected according to token features, while shared experts provide a common processing pathway across the heterogeneous inputs. We train the resulting model on a curated corpus of approximately one million trajectories spanning real-world and simulated manipulation. Inference acceleration further supports efficient rollout generation without changing the action interface. Together, these components address action representation, modeling capacity, and the computational demands of repeated simulator use. Pelican-Sim 1.0 will serve as a key simulation module in future versions of the Pelican-Unify model family (Zhang et al., 2026).

We conduct a systematic evaluation of Pelican-Sim 1.0 that progresses from video prediction and design analysis, to data

Table 1: Comparison of action-conditioning interfaces, the use of token-choice routing in sparse MoE layers, and reported evidence of out-of-distribution (OOD) generalization across robotic world models. OOD entries summarize method-specific evaluations rather than results under a shared protocol.

Method	Action Value	Visual Action Type	Cross-Embod. Interface	Token-choice MoE	Reported OOD Evidence				
					Traj.	Object	Embod.	View.	Scene/App.
IRASim (Zhu et al., 2025)	✓	None	✓	×	✓	–	–	–	–
Ctrl-World (Guo et al., 2025)	✓	None	×	×	✓	–	–	✓	–
EnerVerse-AC (Jiang et al., 2025)	✓	EEF map	×	×	✓	–	–	–	–
Visual Action Prompts (Wang et al., 2025)	×	Hand/gripper skeleton	✓	×	✓	△	–	△	△
OSCAR (Wu & Gao, 2026)	×	Arm skeleton	✓	×	–	–	△	–	–
BridgeV2W (Chen et al., 2026b)	×	Robot mask	✓	×	–	–	△	✓	✓
GeniWorld (Gu et al., 2026)	×	Robot mask	✓	×	–	✓	–	–	✓
ViPSim (Chen et al., 2026a)	✓	EEF map & Robot mask	✓	×	△	△	✓	✓	△
Masked Visual Actions (Alzayer et al., 2026)	×	Robot mask	✓	×	–	–	✓	–	–
Pelican-Sim 1.0 (Ours)	✓	Whole-arm skeleton (URDF)	✓	✓	✓	✓	✓	✓	✓

Interface/architecture: ✓ supported; △ limited support; × not used. OOD: ✓ directly demonstrated, qualitatively or quantitatively; △ limited or indirect evidence; – not reported, rather than unsupported. Evidence may involve joint shifts and does not imply comparable robustness. “Cross-Embod.” denotes interface reuse, not unseen-robot transfer; “Scene/App.” denotes scene or appearance changes. “Token-choice MoE” denotes sparse feed-forward expert selection at the token level; this criterion does not require shared experts. OOD is assessed under each method’s reported training and evaluation setup.



Figure 2: Geometric cues in visual action representations. (a) End-effector maps provide local pose guidance. (b) Robot masks represent projected occupancy. (c) Arm skeletons depict the kinematic chain and gripper-state cues. (d) Our camera-aligned, URDF-rendered skeleton videos provide whole-arm motion guidance with gripper or hand-state cues. Method names indicate representative approaches.

generation, policy evaluation and ranking, action selection, policy improvement, and finally OOD generalization. Experiments on the held-out test splits of AgiBotWorld Beta (AgiBot-World Contributors et al., 2025), RoboMIND (Wu et al., 2025), and RoboTwin (Mu et al., 2024) show that Pelican-Sim 1.0 achieves the best results on all five video-quality metrics and the highest adapted EWMBench overall score on each dataset among compared methods with available results. On RoboTwin, for example, it reduces FVD from 26.60 to 4.98 relative to Ctrl-World, the strongest baseline on this metric. Ablation studies support the contributions of mixed-domain training, complementary action conditioning, and the MoE architecture. Beyond video prediction, we evaluate data generation through policy training with augmented demonstrations and combine the simulator with a fine-tuned vision language model (VLM) evaluator for policy evaluation and ranking, action selection, and policy improvement. The evaluator supplies task-conditioned scores from which these downstream procedures obtain success estimates, selection utilities, and rewards. Qualitative studies further examine generalization across trajectory, scene appearance, embodiment, object, and viewpoint shifts. Together, these experiments evaluate Pelican-Sim 1.0 as a general-purpose simulator supporting the full pipeline from future prediction to downstream policy learning and decision making.

In summary, our contributions are as follows:

- **Unified cross-embodiment action conditioning.** We introduce Pelican-Sim 1.0, a general action-conditioned world model simulator that pairs a unified 28-dimensional numerical action representation with camera-aligned, URDF-rendered whole-arm skeleton videos. Interleaved action-value and action-video pathways integrate precise configuration information and explicit image-space motion geometry within a single Video DiT, enabling joint learning across heterogeneous embodiments. This complementary conditioning improves video prediction over either modality alone; the interleaved architecture improves PSNR by 0.904 dB over late fusion on AgiBotWorld Beta.
- **Heterogeneous dynamics modeling and efficient rollouts.** We equip the Video DiT with sparse MoE layers that combine shared processing with token-dependent expert selection to increase capacity for heterogeneous robot dynamics. Under the same mixed-domain training setup, this design reduces FVD by 6.530 relative to the dense backbone on AgiBotWorld Beta. We further combine causal adaptation with few-step distillation to obtain a four-step autoregressive simulator, achieving a $5.67\times$ speedup over the 35-step model in the reported 21-frame benchmark and supporting repeated queries for downstream decision making.
- **Comprehensive evaluation and downstream utility.** Trained on approximately one million real-world and simulated trajectories, Pelican-Sim 1.0 improves PSNR over the strongest evaluated baselines by 4.636, 2.080, and 10.343 dB on AgiBotWorld Beta, RoboMIND, and RoboTwin, respectively. We demonstrate four downstream applications on

RoboTwin: augmenting 50 demonstrations per task with 500 generated trajectories raises policy success from 70% to 93%; VLM-assisted policy evaluation achieves a Pearson correlation of 0.994 across five checkpoints with 1,000 task-specific adaptation rollouts; and action selection and policy improvement yield relative success gains of 47.7% and 20.3% over single-sample execution and supervised initialization, respectively. Qualitative studies across trajectory, scene, object, embodiment, and viewpoint shifts further demonstrate the model’s generalization potential.

- **Open-source resources.** We will release Pelican-Sim 1.0 model checkpoints and inference code to support reproducible evaluation and further research on general-purpose world model simulators for embodied intelligence.

2. Related Work

2.1. World Models and Action-Conditioned Video Generation

World models learn predictive dynamics for planning and policy learning. PlaNet (Hafner et al., 2019) plans from pixels through compact latent dynamics. Video Diffusion Models (Ho et al., 2022) provide a diffusion-based framework for video synthesis, while interactive world models additionally model the effects of actions. DIAMOND (Alonso et al., 2024) studies the importance of visual fidelity for downstream control, and Genie (Bruce et al., 2024) learns interactive environments and latent actions from unlabelled Internet videos. In robotics, world models have been explored for planning, predictive rewards, and action-conditioned simulation. UniSim (Yang et al., 2024) learns an interactive simulator from heterogeneous data with high- and low-level controls. RoboDreamer (Zhou et al., 2024) factorizes video generation using compositional language structure to support planning with unseen combinations of objects and actions. VIPER (Escontrela et al., 2023) instead uses video prediction likelihood as a reinforcement-learning reward. For action-conditioned simulation, IRASim (Zhu et al., 2025) generates real-robot rollouts, EnerVerse-AC (Jiang et al., 2025) supports multi-level action conditioning and multi-view generation, and Ctrl-World (Guo et al., 2025) combines multi-view prediction with memory for long-horizon rollouts.

In contrast to these approaches, our work targets end-to-end action-conditioned predictive simulation across heterogeneous robots, using a synchronized numerical–visual action interface to preserve precise configurations and make whole-arm motion explicit in image space, thereby supporting video prediction, data generation, policy evaluation, and policy improvement within a unified framework.

2.2. Cross-Embodiment Learning and Visual Action Interfaces

Cross-embodiment learning seeks to share policy representations across heterogeneous robots. Open X-Embodiment (Open X-Embodiment Collaboration, 2024) aggregates robot data for policy learning, while HPT (Wang et al., 2024) maps embodiment-specific inputs into tokens for a shared policy backbone. For predictive simulation, a unified numerical interface retains configuration information but leaves the mapping to visible robot motion implicit. Visual action interfaces address this limitation by providing explicit geometric guidance in image space. Visual Action Prompts (VAP) (Wang et al., 2025) combines human and robot data using hand skeletons estimated from videos and gripper skeletons rendered from robot state logs. OSCAR (Wu & Gao, 2026) uses URDF-rendered arm skeletons with gripper-state cues, whereas BridgeV2W (Chen et al., 2026b) and Geni-World (Gu et al., 2026) render embodiment masks. Masked Visual Actions (Alzayer et al., 2026) uses partially revealed robot or object trajectories. Although these representations make articulated structure or projected occupancy explicit, they do not generally retain the full underlying numerical configuration. Hybrid approaches therefore combine numerical actions with visual geometric cues. EnerVerse-AC (Jiang et al., 2025) and ViPSim (Chen et al., 2026a), for example, combine numerical actions with visual geometry conditions.

Motivated by this complementarity, Pelican-Sim 1.0 pairs unified numerical configurations with synchronized, camera-aligned, URDF-rendered whole-arm skeleton videos through interleaved conditioning pathways. This interface combines precise configuration information with explicit motion guidance in a shared model trained across heterogeneous embodiments and domains. Table 1 and Figure 2 summarize the compared representations.

2.3. Diffusion Transformers and Sparse Mixture-of-Experts

Diffusion Transformers (DiTs) use transformer denoisers for scalable diffusion modeling (Peebles & Xie, 2023). Sparse Mixture-of-Experts (MoE) expands feed-forward capacity through conditional expert selection. GShard (Lepikhin et al., 2021) studies sparse expert scaling, and V-MoE (Riquelme et al., 2021) applies conditional expert selection to vision transformers. DiT-MoE (Fei et al., 2024) introduces sparse diffusion transformers with shared expert routing and expert-level balancing, while LingBot-Video (Ma et al., 2026) scales MoE video pretraining for embodied intelligence.

Building on these architectures, our focus is on modeling heterogeneous robot dynamics under a unified action interface. Pelican-Sim 1.0 combines a sparse MoE backbone with synchronized numerical and visual action conditions, bringing shared and input-dependent processing to joint training on diverse real-world and simulated robot trajectories.

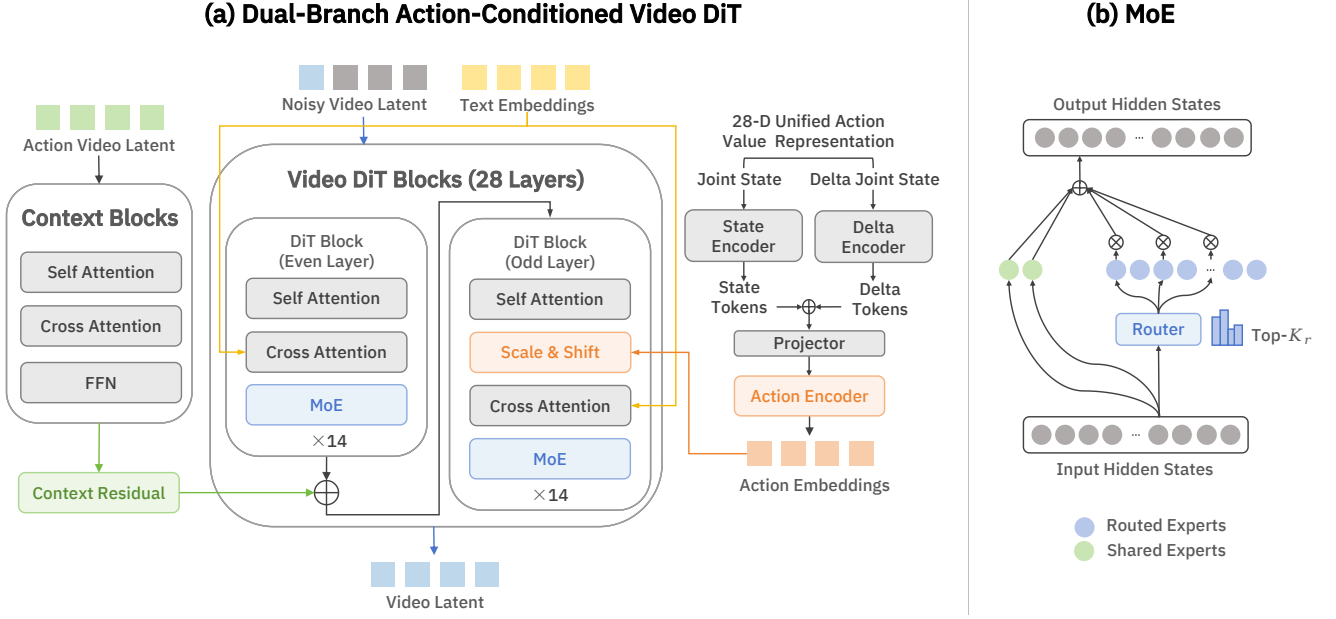


Figure 3: **Architecture of Pelican-Sim 1.0.** (a) A 28-layer Video DiT receives both action conditions: action-value embeddings provide scale-and-shift modulation in odd blocks, while Context Blocks inject action-video residuals at even-block outputs (zero-based indexing). (b) Sparse MoE layers combine shared expert outputs with weighted outputs from token-selected routed experts. Expert counts are schematic; adapters, normalization, and gated residual paths are omitted.

3. Method

We present Pelican-Sim 1.0, an action-conditioned latent video model that predicts future visual observations from an initial observation and a frame-aligned robot-configuration trajectory. As illustrated in Figure 3, Pelican-Sim 1.0 is built on Cosmos-Predict 2.5 (NVIDIA et al., 2025) and augments its Video DiT with two complementary action-conditioning pathways. A low-dimensional *action value* pathway preserves precise numerical configurations, while an *action video* pathway provides embodiment- and viewpoint-aware geometric guidance. Both action value and action video conditions the same Video DiT: the action-value pathway modulates features within odd-indexed blocks, while the action-video pathway uses auxiliary Context Blocks to supply residuals at even-indexed block outputs. The Video DiT uses sparse MoE layers that combine shared and routed experts.

3.1. Action-Conditioned Video Generation

We consider single-frame conditioning and omit the batch dimension in this subsection. Let T be the number of future frames, $t \in \{0, \dots, T\}$ the frame index, and $\mathbf{x}_t$ the RGB observation at frame t . Given the initial observation $\mathbf{x}_0$, define

$$\mathbf{X}^{\text{tar}} = (\mathbf{x}_1, \dots, \mathbf{x}_T), \quad \mathbf{A} = (\mathbf{a}_0, \dots, \mathbf{a}_T) \in \mathbb{R}^{(T+1) \times 28} \quad (1)$$

as the future video and the frame-aligned robot-configuration trajectory, respectively. Each $\mathbf{a}_t \in \mathbb{R}^{28}$ encodes the robot configuration at frame t using the layout in Section 3.2; $\mathbf{a}_0$ describes the conditioning frame and $\mathbf{a}_{1:T}$ denotes the subsequence aligned with the target frames. Given an optional task instruction $\mathbf{y}$, the world model with trainable parameters θ learns the conditional distribution of future observations:

$$p_{\theta}(\mathbf{X}^{\text{tar}} | \mathbf{x}_0, \mathbf{A}, \mathbf{y}), \quad (2)$$

The robot URDF and camera calibration are known metadata used to construct the action conditions; we omit them from the distribution notation for brevity.

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 $\mathbf{Z}^{\text{cond}}$ and the clean future target latent $\mathbf{Z}_0^{\text{tar}}$:

$$[\mathbf{Z}^{\text{cond}}; \mathbf{Z}_0^{\text{tar}}] = \mathcal{E}([\mathbf{x}_0; \mathbf{X}^{\text{tar}}]). \quad (3)$$

Here, $[\cdot; \cdot]$ concatenates video or latent sequences along time. The subscript 0 on $\mathbf{Z}_0^{\text{tar}}$ denotes the clean endpoint of the flow path, not RGB frame zero. The conditioning latent remains clean, while only the target latent is perturbed. Let $\tau \in [0, 1]$ denote the flow timestep, distinct from frame index t , and let $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be standard Gaussian noise with the same shape as the target latent, where $\mathbf{I}$ is the identity covariance after vectorization. We use the linear path

$$\mathbf{Z}_{\tau}^{\text{tar}} = (1 - \tau)\mathbf{Z}_0^{\text{tar}} + \tau\epsilon, \quad \mathbf{U}_{\tau}^{\text{tar}} = \epsilon - \mathbf{Z}_0^{\text{tar}}, \quad (4)$$

where $\mathbf{U}_\tau^{\text{tar}} = \partial \mathbf{Z}_\tau^{\text{tar}} / \partial \tau$ is the target velocity. We form the partially noised latent $\tilde{\mathbf{Z}}_\tau = [\mathbf{Z}_\tau^{\text{cond}}, \mathbf{Z}_\tau^{\text{tar}}]$ and use a binary temporal mask $\mathbf{M}$ to identify the clean conditioning positions and noisy target positions. Let $\mathbf{E}^{\text{val}}$ and $\mathbf{Z}^{\text{act}}$ denote the encoded numerical and visual action conditions, constructed in Section 3.3. The velocity predictor f_θ is trained with the target-only flow-matching objective (Lipman et al., 2023):

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\mathbf{Z}_0^{\text{tar}}, \epsilon, \tau} \left[\left\| f_\theta \left(\tilde{\mathbf{Z}}_\tau, \mathbf{M}, \tau \mid \mathbf{E}^{\text{val}}, \mathbf{Z}^{\text{act}}, \mathbf{y} \right)_{\text{tar}} - \mathbf{U}_\tau^{\text{tar}} \right\|_2^2 \right]. \quad (5)$$

The subscript tar selects the predictor outputs at future latent positions, and $\|\cdot\|_2$ is the Euclidean norm over vectorized target entries. The expectation is over training examples with their paired conditions, Gaussian noise, and sampled flow timesteps. At inference, we keep $\mathbf{Z}^{\text{cond}} = \mathcal{E}(\mathbf{x}_0)$ fixed and integrate the learned velocity field from noise at $\tau = 1$ to the generated target latent $\hat{\mathbf{Z}}_0^{\text{tar}}$ at $\tau = 0$. The tokenizer decoder reconstructs the complete clip from $[\mathbf{Z}^{\text{cond}}, \hat{\mathbf{Z}}_0^{\text{tar}}]$; discarding its conditioning frame yields the predicted future video $\hat{\mathbf{X}}^{\text{tar}}$. Hats on video and latent variables denote generated predictions.

3.2. Complementary Action Representations

Pelican-Sim 1.0 represents the robot-configuration trajectory in two synchronized forms. The action value preserves precise numerical configuration information in a unified layout, while the action video exposes projected joint locations and link connectivity using the corresponding robot URDF and camera calibration. These are complementary representations of the same trajectory, rather than independently specified controls: the numerical branch retains configuration information that image projection may obscure, and the visual branch makes whole-arm motion explicit in image space. Both representations span the complete $T + 1$ -frame clip: their first element aligns with the conditioning frame and the remaining elements align with the target frames.

Unified action value. To accommodate single-arm and bimanual robots equipped with either parallel grippers or dexterous hands, we pack each frame-aligned robot configuration into a fixed 28-dimensional bilateral layout. The first 14 dimensions describe the left side and the remaining 14 dimensions describe the right side. For side $b \in \{L, R\}$, denoting left or right, we define

$$\mathbf{a}_t^b = [\mathbf{q}_t^{\text{arm}, b}; g_t^b; \mathbf{q}_t^{\text{hand}, b}] \in \mathbb{R}^{14}, \quad (6)$$

where $\mathbf{q}_t^{\text{arm}, b} \in \mathbb{R}^7$ contains the arm joint angles, $g_t^b \in \mathbb{R}$ is the parallel-gripper opening, and $\mathbf{q}_t^{\text{hand}, b} \in \mathbb{R}^6$ contains the dexterous-hand joint values. For configuration vectors, $[\cdot; \cdot]$ denotes concatenation along the feature dimension. The complete action value is ordered as

$$\mathbf{a}_t = [\mathbf{a}_t^L; \mathbf{a}_t^R] = \left[\underbrace{\mathbf{q}_t^{\text{arm}, L}, g_t^L, \mathbf{q}_t^{\text{hand}, L}}_{14 \text{ dimensions}}; \underbrace{\mathbf{q}_t^{\text{arm}, R}, g_t^R, \mathbf{q}_t^{\text{hand}, R}}_{14 \text{ dimensions}} \right] \in \mathbb{R}^{28}. \quad (7)$$

For a 6-DoF arm, the unused seventh arm dimension is set to zero. Missing grippers or dexterous hands are represented by zero-filled slots, and all 14 dimensions of an absent side are set to zero for single-arm embodiments. This canonical ordering preserves the original numerical commands while presenting a fixed action interface across heterogeneous robot embodiments.

URDF-rendered action video. Although the unified action values specify robot configurations precisely, their fixed-dimensional layout does not explicitly describe the resulting image-space motion. Without a visual action condition, the model must learn the mapping from numerical configurations to visible robot motion. We therefore use the robot URDF and camera calibration to render each numerical action sequence as a skeleton video aligned with the RGB view, making this geometric mapping explicit in the conditioning input. As illustrated in Figure 4, the unified action at time t is first mapped to the joint configuration required by the corresponding URDF $\mathcal{U}$:

$$\mathbf{q}_t^{\mathcal{U}} = \Psi_{\mathcal{U}}(\mathbf{a}_t), \quad (8)$$

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 $\mathbf{T}_{j,t} \in \mathbb{R}^{4 \times 4}$:

$$\mathbf{T}_{j,t} = \text{FK}_j(\mathcal{U}, \mathbf{q}_t^{\mathcal{U}}). \quad (9)$$

Let $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{T}_{\text{cam}} \in \mathbb{R}^{4 \times 4}$ 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

$$\mathbf{p}_{j,t} = \Pi(\mathbf{K}, \mathbf{T}_{\text{cam}}, \mathbf{T}_{j,t}), \quad (10)$$

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 $\mathbf{v}_t^{\text{act}}$ and the full action video $\mathbf{V}^{\text{act}}$:

$$\mathbf{v}_t^{\text{act}} = \mathcal{R}(\{\mathbf{p}_{j,t}\}_j, \mathcal{K}_{\mathcal{U}}, \mathbf{g}_t), \quad \mathbf{V}^{\text{act}} = (\mathbf{v}_0^{\text{act}}, \dots, \mathbf{v}_T^{\text{act}}), \quad (11)$$

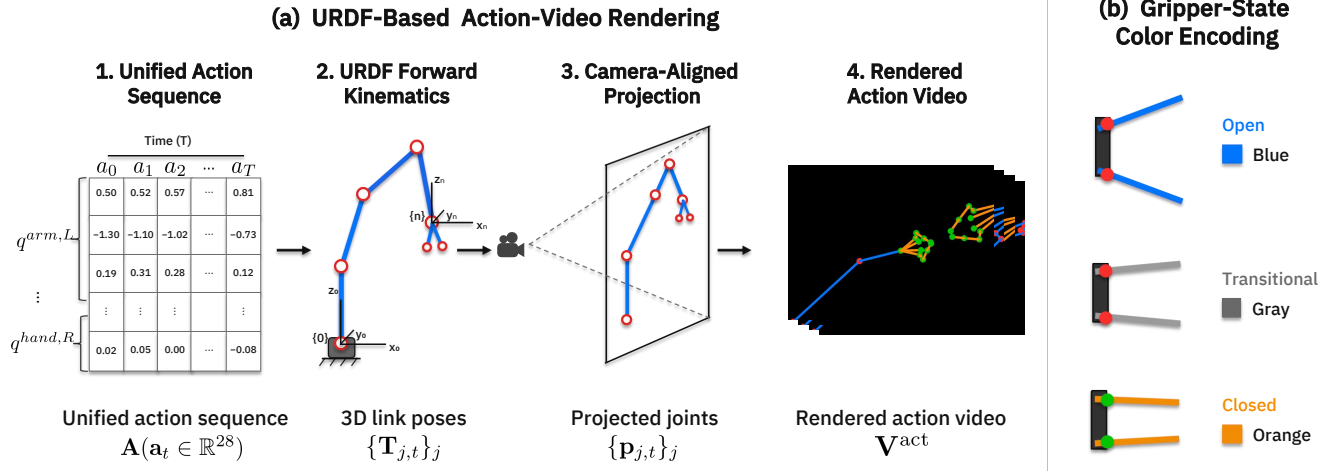


Figure 4: **URDF-based action-video rendering.** (a) The frame-aligned configuration sequence is mapped to robot-specific URDF joint configurations, converted into 3D link poses through forward kinematics, projected using the paired RGB camera calibration, and rasterized as a synchronized skeleton action video. (b) Gripper links are color-coded to indicate open, transitional, and closed states.

where $\mathbf{g}_t = (g_t^L, g_t^R)$ contains the gripper openings used for state coloring. Arm links and joints form a texture-free skeleton, while gripper links use different colors to distinguish open, transitional, and closed states. This representation provides image-space motion guidance consistent with the nominal robot kinematics and camera projection, while leaving scene appearance to the RGB condition and the generative backbone. This consistency applies to the rendered condition; it does not impose a hard kinematic constraint on the generated RGB observations. By construction, $\mathbf{v}_t^{\text{act}}$ is synchronized with RGB frame $\mathbf{x}_t$. The first action-video frame anchors the initial robot configuration, while $\mathbf{v}_{1:T}^{\text{act}}$ provides frame-aligned geometric guidance for $\mathbf{X}^{\text{tar}}$.

3.3. Dual-Branch Conditioning and Sparse MoE Layers

The Video DiT follows the 28-layer latent video architecture of Cosmos-Predict 2.5 (NVIDIA et al., 2025). In addition to $\tilde{\mathbf{Z}}_\tau$, $\mathbf{M}$, τ , and optional instruction $\mathbf{y}$, it receives both action-value and action-video conditions through separate pathways. Both conditions act on the same sequence of Video DiT hidden states; the Context Blocks form an auxiliary action-video conditioning branch. Using zero-based block indices $\ell \in \{0, \dots, 27\}$, action values modulate the 14 odd-indexed blocks $\mathcal{I}_{\text{val}} = \{1, 3, \dots, 27\}$, while action-video features enter the 14 even-indexed blocks $\mathcal{I}_{\text{vid}} = \{0, 2, \dots, 26\}$ through a parallel context branch. Let $\mathbf{H}^\ell$ and $\mathbf{H}^{\ell+1}$ denote the input and output hidden features of block ℓ in the Video DiT, and let $\mathbf{H}_{\text{moe}}^\ell$ denote its features after the MoE layer and gated residual update, before any action-video residual is added.

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

Action-value encoding and injection. As shown in Figure 3(a), the action-value pathway represents the trajectory by its initial configuration $\mathbf{s}$ and scaled frame-to-frame changes δ_t :

$$\mathbf{s} = \mathbf{a}_0, \quad \delta_t = (\mathbf{a}_t - \mathbf{a}_{t-1}) \odot \boldsymbol{\eta}, \quad t = 1, \dots, T, \quad (12)$$

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 $T_a = 1 + \lceil T/r_t \rceil$ the number of action tokens. We group the deltas into $T_a - 1$ non-overlapping chunks of length r_t , padding the final chunk when necessary. Two multilayer perceptrons (MLPs), $\mathcal{M}_{\text{state}}$ and $\mathcal{M}_{\text{delta}}$, encode the initial state and the delta chunks:

$$\mathbf{e}^{\text{state}} = \mathcal{M}_{\text{state}}(\mathbf{s}), \quad \mathbf{e}_k^{\text{delta}} = \mathcal{M}_{\text{delta}}(\text{Concat}(\delta_{kr_t+1:(k+1)r_t})), \quad k = 0, \dots, T_a - 2, \quad (13)$$

Here, $\mathbf{e}^{\text{state}}$ is the initial-state token, $\mathbf{e}_k^{\text{delta}}$ 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 $\mathbf{E}^{\text{val}} \in \mathbb{R}^{B \times T_{\text{lat}} \times H_{\text{tok}} \times W_{\text{tok}} \times D}$.

For each odd block $\ell \in \mathcal{I}_{\text{val}}$ of the Video DiT, a block-specific residual adapter transforms the shared action embeddings before producing the scale γ^ℓ and shift β^ℓ . These parameters modulate the post-self-attention feature $\bar{\mathbf{H}}^\ell$ before text cross-

attention:

$$\begin{aligned} \mathbf{E}^{\text{inj},\ell} &= \mathbf{E}^{\text{val}} + \mathcal{I}_2^\ell(\text{SiLU}(\mathcal{I}_1^\ell(\mathbf{E}^{\text{val}}))), \\ (\gamma^\ell, \beta^\ell) &= \mathcal{S}^\ell(\mathbf{E}^{\text{inj},\ell}), \\ \hat{\mathbf{H}}^\ell &= (\mathbf{1} + \gamma^\ell) \odot \bar{\mathbf{H}}^\ell + \beta^\ell, \quad \ell \in \mathcal{I}_{\text{val}}. \end{aligned} \quad (14)$$

Here, $\mathbf{E}^{\text{inj},\ell}$ is the block-adapted action embedding; $\mathcal{I}_1^\ell$ and $\mathcal{I}_2^\ell$ are adapter layers with the SiLU activation between them, and $\mathcal{S}^\ell$ is the scale-shift head. The all-ones tensor $\mathbf{1}$ supplies the identity scale, and $\hat{\mathbf{H}}^\ell$ denotes the modulated hidden feature. Scale and shift are aligned with the hidden-feature grid. The modulated feature passes through text cross-attention and the MoE layer, yielding $\mathbf{H}^{\ell+1} = \mathbf{H}_{\text{moe}}^\ell$ for odd blocks.

Action-video encoding and injection. As shown in Figure 3(a), let $\bar{\mathbf{V}}^{\text{act}}$ denote the action video normalized to $[-1, 1]$. The same video tokenizer $\mathcal{E}$ used for RGB observations produces

$$\mathbf{Z}^{\text{act}} = \mathcal{E}(\bar{\mathbf{V}}^{\text{act}}) \in \mathbb{R}^{B \times C_z \times T_{\text{lat}} \times H' \times W'}, \quad C_z = 16, \quad (15)$$

where C_z is the tokenizer’s latent-channel dimension and $H' \times W'$ is the latent spatial grid before patch embedding, distinct from the token grid $H_{\text{tok}} \times W_{\text{tok}}$. After patch embedding and spatial alignment, $\mathbf{Z}^{\text{act}}$ is processed by parallel Context Blocks initialized from the corresponding pretrained Video DiT blocks. Context Block $k \in \{0, \dots, 13\}$ produces a residual $\mathbf{R}^k$ with the same shape as the Video DiT hidden features. It is added to the output of even block $\ell = 2k$ after the MoE layer and its gated residual update:

$$\mathbf{H}^{\ell+1} = \mathbf{H}_{\text{moe}}^\ell + \mathbf{R}^{\ell/2}, \quad \ell \in \mathcal{I}_{\text{vid}}, \quad (16)$$

Thus, action value and action video are injected into the same Video DiT at alternating blocks: numerical features modulate intermediate states within odd blocks, while visual features contribute spatially and temporally aligned residuals after even blocks. The Context Blocks provide visual conditioning rather than a separate video-prediction backbone.

Sparse MoE layers. As illustrated in Figure 3(b), we replace the dense feed-forward network (FFN) in each block of the Video DiT with a token-choice sparse MoE layer (Fei et al., 2024; Ma et al., 2026). Let N_s and N_r denote the numbers of shared and routed experts, respectively, and let $K_r \leq N_r$ be the number of routed experts selected per token (Dai et al., 2024). Each token passes through all shared experts and its top- K_r routed experts. The shared branch provides a common transformation, while the routed branch supplies a token-dependent combination of expert outputs. Expert selection depends on token features rather than predefined embodiment or domain assignments. We use $N_s = 1$, $N_r = 8$, and $K_r = 2$ in all experiments.

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 $\mathbf{u}_n^\ell \in \mathbb{R}^D$, conditioned on flow timestep τ . Following sigmoid token-choice routing with selection bias (DeepSeek-AI et al., 2024; Ma et al., 2026), we define

$$\begin{aligned} \mathbf{u}_n^\ell &= \text{AdaLN}_{\text{mlp}}^\ell(\mathbf{h}_n^{\ell, \text{ca}}; \tau), \\ \rho_n^\ell &= \text{sigmoid}(\mathbf{W}_r^\ell \mathbf{u}_n^\ell), \\ \mathcal{J}_n^\ell &= \text{TopK}_{K_r}(\rho_n^\ell + \mathbf{b}_r^\ell), \\ g_{i,n}^\ell &= \kappa_r \frac{\rho_{i,n}^\ell}{\sum_{j \in \mathcal{J}_n^\ell} \rho_{j,n}^\ell}, \quad i \in \mathcal{J}_n^\ell. \end{aligned} \quad (17)$$

Here, $\mathbf{W}_r^\ell \in \mathbb{R}^{N_r \times D}$ is the router projection, $\rho_n^\ell \in \mathbb{R}^{N_r}$ 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:

$$\begin{aligned} \Delta \mathbf{h}_n^\ell &= \sum_{m=1}^{N_s} \mathcal{F}_m^{\text{shared}, \ell}(\mathbf{u}_n^\ell) + s_r(q) \sum_{i \in \mathcal{J}_n^\ell} g_{i,n}^\ell \mathcal{F}_i^{\text{routed}, \ell}(\mathbf{u}_n^\ell), \\ \mathbf{h}_n^{\ell, \text{moe}} &= \mathbf{h}_n^{\ell, \text{ca}} + \mathbf{g}_{\text{mlp}}^\ell \odot \Delta \mathbf{h}_n^\ell. \end{aligned} \quad (18)$$

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 4.1. 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 $L_M = 28$ MoE layers and auxiliary-loss weight λ_{MoE} , the balancing loss $\mathcal{L}_{\text{MoE}}$ and total training objective $\mathcal{L}$ are

$$\mathcal{L}_{\text{MoE}} = \frac{\lambda_{\text{MoE}}}{L_M} \sum_{\ell=0}^{L_M-1} N_r \sum_{i=1}^{N_r} f_i^\ell P_i^\ell, \quad \mathcal{L} = \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{MoE}}. \quad (19)$$

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.

3.4. Causal Few-Step Distillation for Efficient Rollout

The bidirectional model prioritizes generation quality, but its 35-step sampler is costly for downstream applications that require large numbers of rollouts. We therefore derive a four-step causal student while preserving the task, action-value, and action-video interfaces of Pelican-Sim 1.0. Following recent streaming world models (Cheng et al., 2026; Huang et al., 2025; Zhu et al., 2026), we adopt a progressive procedure that separates causal adaptation from few-step distillation.

Progressive causal distillation. We partition the target latent video into temporal chunks and initialize the causal model from the bidirectional Pelican-Sim 1.0 weights. Full temporal self-attention is replaced by block-causal attention, such that each chunk attends only to itself and preceding visual chunks, while the known action conditions for the current chunk remain fully accessible. We first perform teacher-forced flow matching using clean historical chunks, producing a 35-step causal teacher. Its probability-flow ODE trajectories are then used to initialize a four-step causal student (Zhu et al., 2026). Finally, we expose the student to its own autoregressive history and apply distribution matching distillation (Yin et al., 2024; Huang et al., 2025) against the frozen bidirectional teacher. A small fraction of teacher-forced updates is retained during this self-forcing stage to stabilize training and preserve action controllability.

Accelerated streaming inference. During inference, the student generates each latent chunk in four denoising steps and appends it to a rolling context window. Block-causal attention enables key-value caching, avoiding repeated computation over historical tokens. In a 21-frame inference benchmark, the accelerated model requires 2.2 seconds per clip and achieves a measured $5.67\times$ speedup over the original 35-step model. We use this accelerated four-step causal model for all downstream applications in Sections 4.5–4.8.

4. Experiments

Our experiments follow a progressive evaluation from **prediction quality**, to **data utility**, **policy evaluation**, **action selection**, **policy improvement**, and finally **OOD generalization**. Specifically, Sections 4.2–4.4 establish predictive performance, Section 4.5 studies the utility of generated rollouts for policy learning, Section 4.6 evaluates their use for policy assessment and ranking, Section 4.7 examines whether predicted futures can guide candidate action selection, Section 4.8 investigates whether closed-loop imagined rollouts can directly improve policies through reinforcement learning, and Section 4.9 assesses robustness under controlled distribution shifts in trajectory, scene appearance, embodiment, object, and viewpoint.

4.1. Experimental Setup

Training datasets. Our training corpus combines real-world demonstrations from AgiBotWorld Beta (AgiBot-World Contributors et al., 2025), RealSource World (RealSource, 2025), and RoboMIND (Wu et al., 2025) with simulated trajectories from LIBERO (Liu et al., 2023), ManiSkill2 (Gu et al., 2023), RoboTwin (Mu et al., 2024), and RoboCasa (Nasiriany et al., 2024). These sources cover single-arm and bimanual manipulation with parallel grippers and dexterous hands. All trajectories are converted into unified action values and camera-aligned action videos as described in Section 3.2.

Data quality curation. We discard trajectories with incomplete observations, invalid actions, missing or invalid URDF files, missing camera calibration, or inconsistent timestamps. A vision–language model checks the rendered action videos against 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. The curated corpus contains approximately **1.0 million trajectories**, totaling **8,000 hours** of robot interaction data.

Evaluation datasets and splits. We evaluate video prediction on the held-out test splits of AgiBotWorld Beta, RoboMIND, and RoboTwin. Each source is partitioned into training, validation, and test splits at a 90%/5%/5% ratio before temporal clip extraction. For AgiBotWorld Beta and RoboMIND, we group trajectories by collection session and stratify by task and embodiment. For RoboTwin, we use the randomized configuration with disjoint randomization seeds and initial object configurations across splits. Validation splits are used only for model selection; video-generation metrics are computed on the fixed test splits. These are within-source partitions rather than entirely unseen datasets.

Implementation details. Unless otherwise stated, Pelican-Sim 1.0 processes 93-frame clips at 480×640 resolution with frame-aligned action values and action videos, conditioning on the first RGB frame and task instruction to predict the remaining 92 frames. We train for 500K iterations on 32 NVIDIA B200 GPUs with a global batch size of 32 and no gradient accumulation. The learning rate is 3.05×10^{-5} , with a 1,000-step linear warmup followed by a constant schedule. Each MoE layer uses one shared expert ($N_s = 1$), eight routed experts, and sigmoid top-2 routing with route scale $\kappa_r = 2.5$. The routed-branch multiplier $s_r(q)$ increases linearly from 0 to 1 over the first 2,000 iterations and remains 1 thereafter.

Evaluation protocol and metrics. On each dataset, all compared methods are evaluated on the same test split using their respective action-conditioning representations. We report PSNR and SSIM (Wang et al., 2004) for frame-level reconstruction quality, LPIPS (Zhang et al., 2018) for perceptual similarity, FID (Heusel et al., 2017) for image-distribution quality, and FVD (Unterthiner et al., 2018) for spatiotemporal video quality. We further adapt EWMBench (Hu et al., 2025) to assess motion correctness (DYN, HSD, and nDTW), semantic alignment (BLEU and CLIP), and scene consistency (SceneC), and report Diversity to characterize variation among task-consistent rollouts.

VLM evaluator. The world model predicts visual observations; a fine-tuned VLM evaluator converts these predictions into task-conditioned feedback for downstream policy evaluation and ranking, action selection, and policy improvement. Building on VLM-based rollout evaluation (Quevedo et al., 2025; Tseng et al., 2025; Wu & Gao, 2026) and learned progress estimation (Feng et al., 2026), we adapt Qwen3-VL-2B-Instruct (Bai et al., 2025) with LoRA (Hu et al., 2022). Training uses simulator and world-model-generated rollouts from the downstream task domains, with manual annotations of terminal success, task progress, and visual validity, together with same-task ranking supervision. With the pretrained backbone weights frozen, we optimize the LoRA adapters and three prediction heads, whose trainable parameters are denoted by ϕ , for terminal-success confidence q_ϕ , five-level normalized progress p_ϕ , and visual validity a_ϕ . The evaluator takes the task instruction, a success/failure checklist, and 16 uniformly sampled, temporally ordered RGB frames including both endpoints, and is frozen during downstream use. For action selection, we define the utility $u_\phi = a_\phi[q_\phi + \lambda(1 - q_\phi)p_\phi]$, where λ weights the progress contribution. Policy-evaluation and reward formulations are detailed in Sections 4.6 and 4.8.

4.2. Comparison with State-of-the-Art World Models

We compare against IRASim (Zhu et al., 2025) and Ctrl-World (Guo et al., 2025), which use numerical actions, and the hybrid method EnerVerse-AC (Jiang et al., 2025) on all three evaluation datasets. On AgiBotWorld Beta, we additionally compare against OSCAR (Wu & Gao, 2026), which uses visual skeletons, and Wan-Move (Chu et al., 2025), a motion-controllable video model guided by point trajectories. All evaluated baselines are retrained on the same mixed-domain training corpus.

Tables 2 and 3 summarize video quality and action-conditioned generation on AgiBotWorld Beta, RoboMIND, and RoboTwin. Pelican-Sim 1.0 achieves the best scores on all five video-quality metrics and the highest adapted EWMBench overall score among the methods evaluated on each dataset. Relative to the strongest baseline for each metric, PSNR improves by 4.636, 2.080, and 10.343 dB, respectively, while the adapted EWMBench overall score increases by 0.566, 0.390, and 1.073. The largest PSNR and overall-score gains occur on RoboTwin, where FVD also decreases from 26.600 for Ctrl-World to 4.980. These results indicate improved visual fidelity and aggregate action-conditioned generation performance, although individual adapted EWMBench metrics remain mixed: HSD and nDTW on RoboMIND, and SceneC and Diversity on AgiBotWorld Beta and RoboTwin, remain below the corresponding best baselines.

4.3. Qualitative Generation Results

Figures 5 and 6 present representative action-conditioned rollouts across real-world and simulated data sources. For each task, the rows show the generated video, the frame-aligned action-video condition, and the ground-truth video, while columns progress from left to right in time. The examples cover single-arm and bimanual systems, parallel grippers and dexterous hands, and diverse object interactions. Across these settings, Pelican-Sim 1.0 closely follows the commanded whole-arm trajectories and gripper-state transitions while preserving temporal coherence and scene appearance. The predicted object motions are also consistent with the robot actions, indicating effective action control across heterogeneous embodiments and domains.

4.4. Ablation Studies and Design Analysis

Mixed-domain training and sparse MoE. Table 4 presents a cumulative ablation on AgiBotWorld Beta, comparing a dense baseline trained on real data, the same dense model trained on mixed-domain data, and an MoE variant trained on mixed-domain

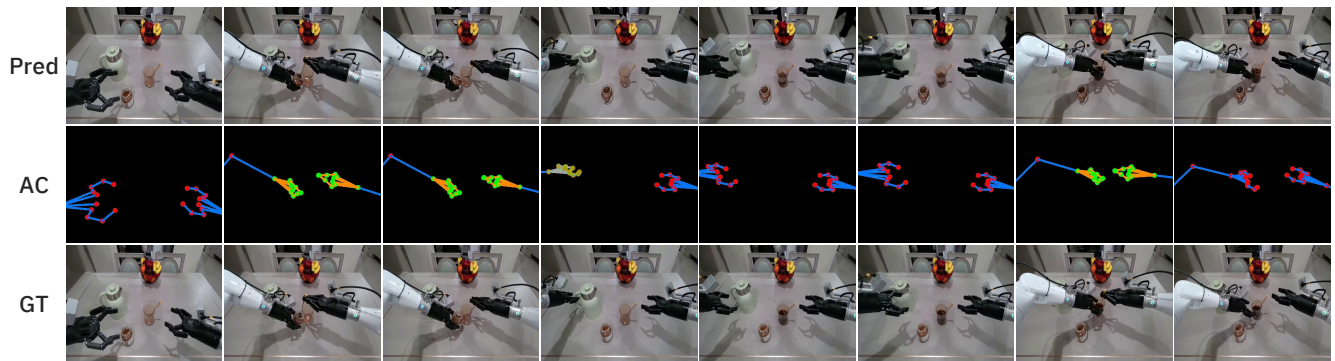
Table 2: Video-quality comparison on three held-out test splits. For each dataset and metric, the best and second-best results are bold and underlined, respectively.

Method	Computation-based		Model-based		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	FID $\downarrow$	FVD $\downarrow$
<i>AgiBotWorld Beta</i>					
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>
Pelican-Sim 1.0	<u>22.276</u>	<u>0.796</u>	0.013	14.640	10.850
<i>RoboMIND</i>					
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
Pelican-Sim 1.0	23.850	0.835	0.012	17.610	15.930
<i>RoboTwin</i>					
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
Pelican-Sim 1.0	30.383	0.942	0.006	8.910	4.980

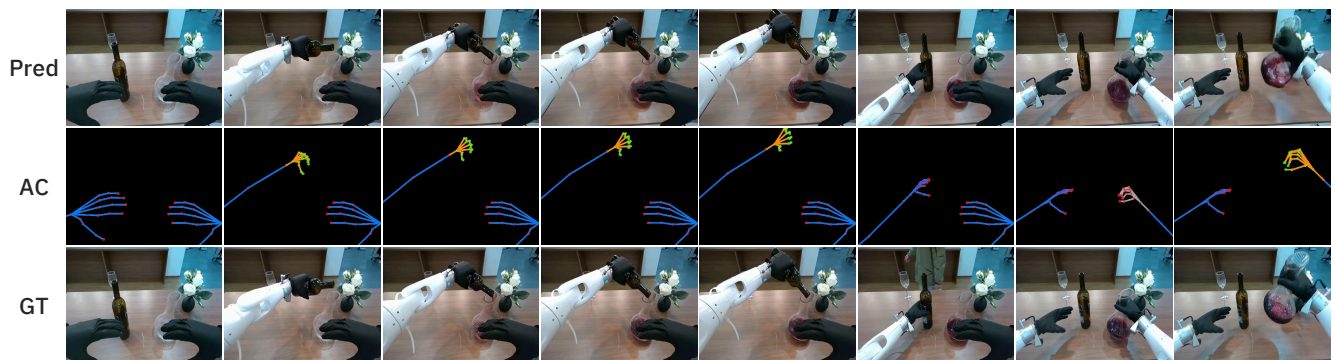
Table 3: Adapted EWMBench results on the same test splits. Primitive scores are normalized to $[0, 1]$; higher is better. Overall denotes the adapted EWMBench overall score. The best and second-best results are bold and underlined, respectively.

Method	Motion			Semantics			Scene	Overall ↑
	DYN ↑	HSD ↑	nDTW ↑	BLEU ↑	CLIP ↑	Div. ↑	SceneC ↑	
<i>AgiBotWorld Beta</i>								
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>
Pelican-Sim 1.0 (Ours)	0.715	0.959	0.962	0.524	0.933	<u>0.201</u>	0.703	4.997
<i>RoboMIND</i>								
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
Pelican-Sim 1.0 (Ours)	0.591	<u>0.838</u>	<u>0.810</u>	0.495	0.926	0.157	0.849	4.666
<i>RoboTwin</i>								
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
Pelican-Sim 1.0 (Ours)	0.821	0.972	0.976	0.531	0.918	<u>0.160</u>	0.666	5.044

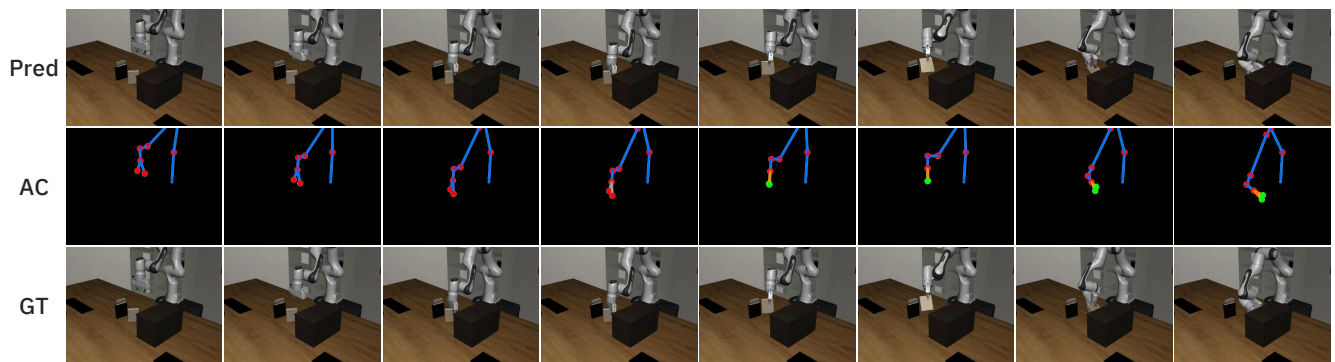
AgiBotWorld Beta Dataset (Gripper)
"Prepare a Beverage by Adding Granules"



AgiBotWorld Beta Dataset (Dexhand)
"Pour Wine into a Glass Bottle and Shake It"



LIBERO Dataset
"Pick Up the Book on the Right and Place It on the Cabinet Shelf"



ManiSkill Dataset
"Use the Tool to Pull the Cube within Reach"

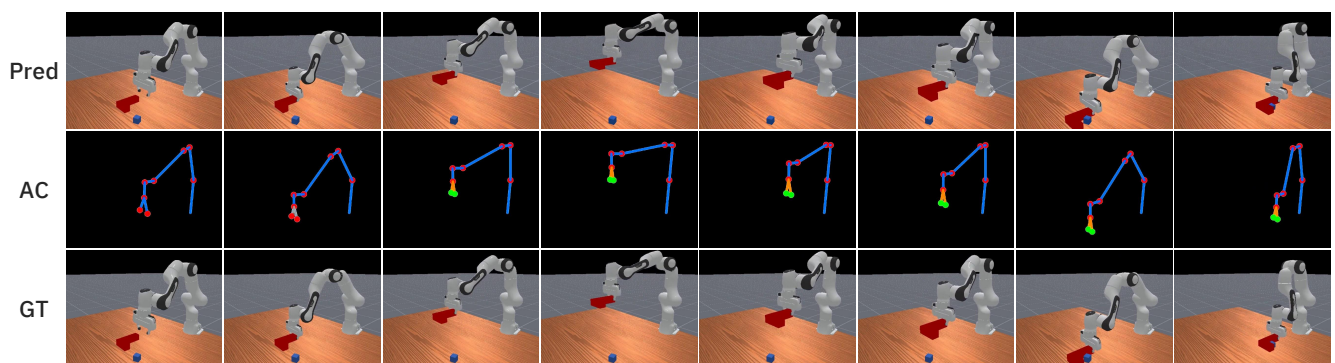
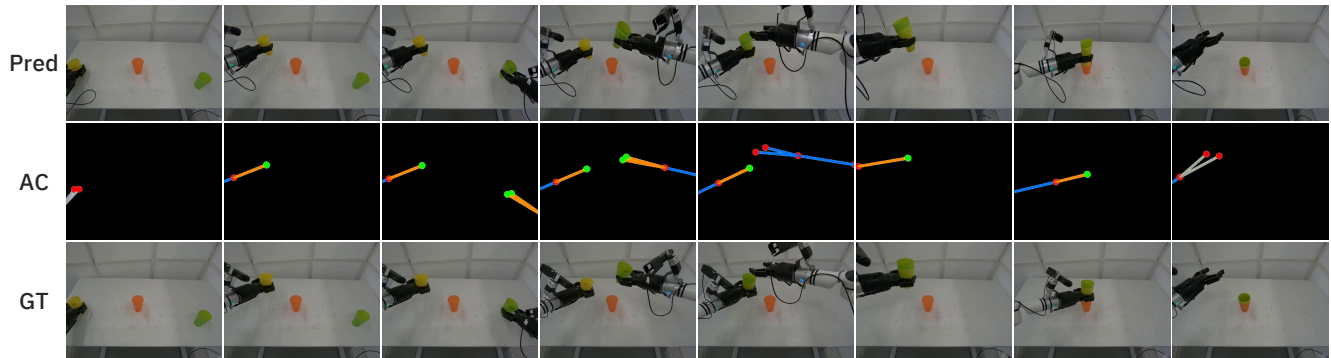


Figure 5: **Multi-source rollouts (Part I)**. AgiBotWorld Beta, LIBERO, and ManiSkill. Rows: prediction (Pred), action video (AC), and ground truth (GT); time advances left to right.

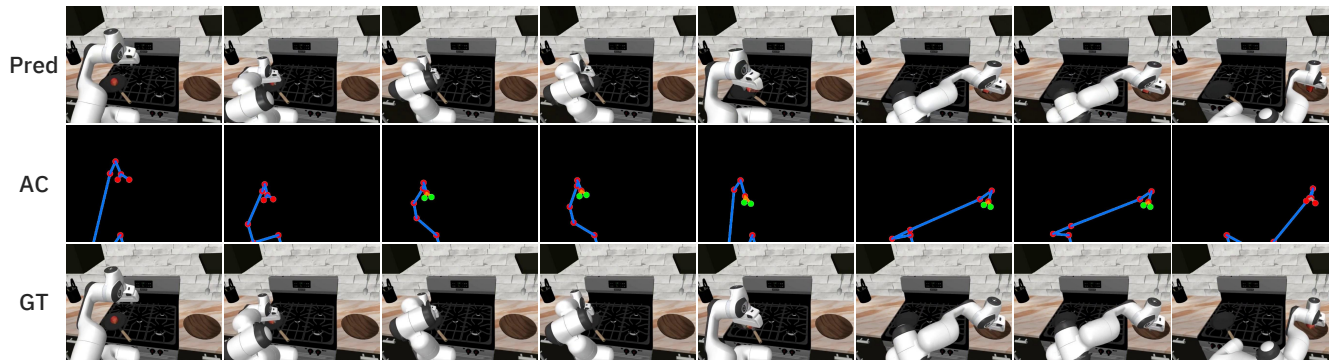
RealSource World Dataset

"Stack the Cups"



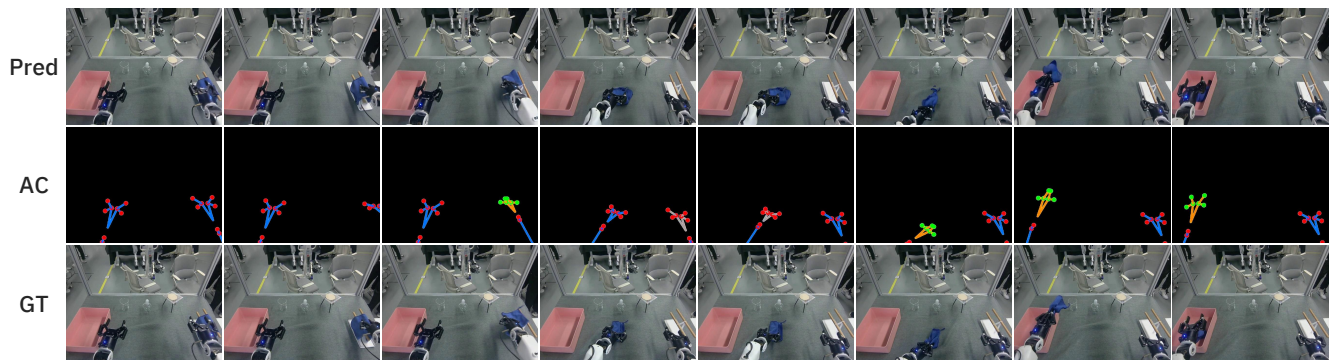
RoboCasa Dataset

"Move an Object from the Stove to the Counter"



RoboMIND Dataset

"Place the Cloth at the Center and Wipe the Table"



RoboTwin Dataset

"Pick Up the Medium-Sized White Mug"

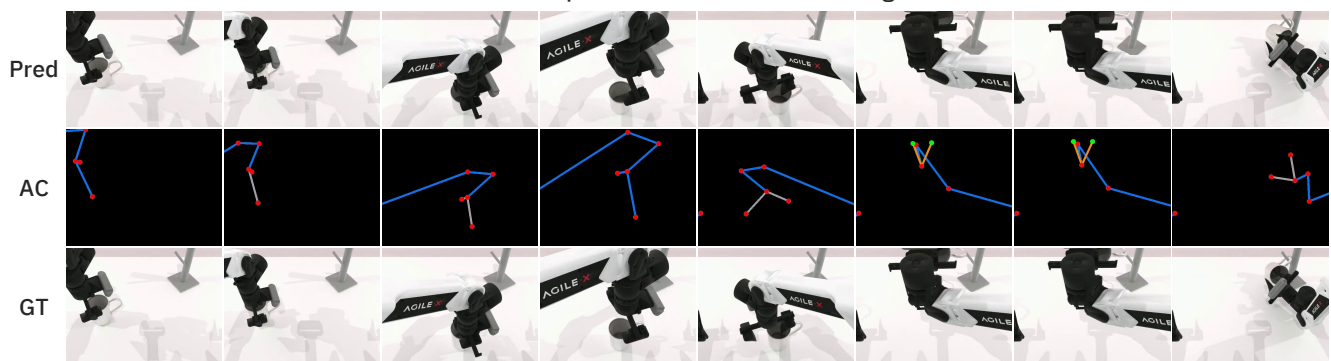


Figure 6: **Multi-source rollouts (Part II).** RealSource World, RoboCasa, RoboMIND, and RoboTwin. Rows: Pred (prediction), AC (action video), GT (ground truth); time runs left to right.

Table 4: 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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	FID $\downarrow$	FVD $\downarrow$
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 5: 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 $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
No action condition	$\times$	$\times$	17.684	0.6513	0.0287	19.640	0.7195	0.0248
Action value only	$\checkmark$	$\times$	19.238	0.7126	0.0214	21.356	0.7768	0.0176
Action video only	$\times$	$\checkmark$	21.276	0.7668	0.0158	23.102	0.8209	0.0132
Full dual condition	$\checkmark$	$\checkmark$	22.276	0.7960	0.0125	23.850	0.8350	0.0115

data. All variants share the action conditions, injection architecture, optimization schedule, and training iterations. Adding simulated trajectories improves PSNR by 0.548 dB and reduces FVD by 3.030; replacing the dense feed-forward layers with sparse MoE layers under mixed-domain training yields further gains of 1.214 dB and 6.530, respectively. These results support the two successive design choices, while the mixed-domain improvement reflects changes in both data diversity and corpus size rather than an isolated effect of domain composition.

Contribution of complementary action conditions. Table 5 compares four action-conditioning configurations on AgiBotWorld Beta and RoboMIND. The no-action baseline evaluates the video generation model without numerical or visual action inputs. Both single-condition variants improve over this baseline, with action videos outperforming numerical actions on all reported metrics in this ablation. Combining both conditions yields the best prediction quality, increasing PSNR over action video alone by 1.000 and 0.748 dB on the two datasets, respectively, with corresponding LPIPS reductions from 0.0158 to 0.0125 and from 0.0132 to 0.0115. These results support combining explicit image-space motion guidance with numerical configuration information, rather than replacing numerical actions with visual conditions alone.

Injection architecture. Table 6 compares token concatenation, late fusion, two opposite interleaving orders, and dual injection into every block of the Video DiT. Both interleaved variants outperform token concatenation and late fusion, with similar scores across the reported metrics. Dual injection achieves the highest scores but improves PSNR over the selected interleaving by only 0.075 dB on AgiBotWorld Beta and 0.051 dB on RoboMIND. This suggests that repeatedly injecting both conditions at every block provides limited additional benefit. Since numerical and visual actions encode complementary aspects of the same trajectory, their information can instead be progressively integrated through the shared hidden representations across network depth. We therefore adopt interleaved injection, which alternates the two conditioning pathways across blocks of the same Video DiT and achieves nearly the same prediction quality as dual injection with fewer conditioning modules.

4.5. Data Generation

We assess Pelican-Sim 1.0 as a data-generation engine across three progressively demanding settings: **generation efficiency**, examining whether trajectories can be generated efficiently; **cross-domain data construction**, evaluating whether generated rollouts provide useful supervision for policy training; and **policy learning with limited demonstrations**, assessing whether augmentation remains beneficial when collected data are scarce. In both policy studies, trajectory counts are specified per task. Within each study, compared policies share a fixed set of 200 previously unseen evaluation seeds per task under the RoboTwin randomized configuration, disjoint from training-data construction.

Generation efficiency. Following robot data-generation evaluations (He et al., 2026), we compare the accelerated four-step 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,

Table 6: 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 7: Data generation efficiency of Pelican-Sim 1.0 compared with manual collection and RoboTwin simulation for complete trajectories at matched horizons. The accelerated four-step causal model is used 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.) ↓
<i>Real-world comparison</i>				
Manual collection	Robot + operator	60	60	0.4
Pelican-Sim 1.0	GPU	24	150	0
<i>Simulation comparison</i>				
RoboTwin simulator	Simulator + GPU	55	65.5	0
Pelican-Sim 1.0	GPU	19	189.5	0

Table 8: Cross-domain data augmentation for downstream $\pi_{0.5}$ policy learning. Per task, Pelican-Sim 1.0 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 1.0 augmentation	50	10	550	72.0

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 7 reports generation efficiency separately for the real-world and simulation comparisons. In the real-world comparison, Pelican-Sim 1.0 generates a complete task trajectory in 24 seconds, compared with 60 seconds for manual collection, yielding nominal throughputs of 150 and 60 trajectories per hour, respectively ($2.50\times$). In the simulation comparison, Pelican-Sim 1.0 requires 19 seconds per trajectory, compared with 55 seconds for RoboTwin, yielding approximately 189.5 and 65.5 trajectories per hour, respectively ($2.89\times$). These throughput ratios exclude manual reset time. Manual collection additionally requires 0.4 minutes of reset time per trajectory, whereas neither automated method requires manual reset between episodes.

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 8 reports mean success rates of 72.0% for Pelican-Sim 1.0 augmentation and 70.0% for direct simulator collection across five tasks. The similar observed performance supports Pelican-Sim 1.0-based data construction as an alternative for

Table 9: 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)

downstream policy training.

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 9 shows success-rate gains of 36.0, 30.0, and 23.0 percentage points at the three demonstration budgets, respectively. Augmentation with only 10 original trajectories per task reaches 64.5%, compared with 70.0% for 50 originals without augmentation. These results indicate that generated rollouts can reduce the need for additional collected demonstrations in the evaluated setting, with the largest absolute gain at the smallest demonstration budget.

4.6. Policy Evaluation and Ranking

Following world-model-based policy evaluation (Quevedo et al., 2025; Tseng et al., 2025; Jeon et al., 2026), we next examine whether a pipeline combining Pelican-Sim 1.0 with the fine-tuned VLM evaluator reproduces policy success rates and rankings measured in RoboTwin. The world model predicts visual outcomes for supplied action trajectories, and the evaluator judges task completion from the generated videos. To assess evaluation robustness across policy performance levels, we select five checkpoints $\{\pi_k\}_{k=1}^5$ at different stages of the same VLA training run, representing policies with varying task success rates. These policies provide trajectories for both validation of the frozen VLM evaluator from Section 4.1 and assessment of the complete world-model–VLM pipeline.

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) for task-success prediction; MAE and Spearman correlation for progress estimation; and AUROC for visual validity.

Table 10 separately reports evaluator performance on simulator and generated videos. The evaluator achieves terminal accuracies of 92.6% and 91.4%, respectively, with a progress MAE of 0.078 and validity AUROC of 0.953 on generated videos. These results assess the VLM’s ability to judge depicted outcomes, separately from the world model’s ability to reproduce simulator outcomes.

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 $\text{SR}_k^{\text{wm}} = N^{-1} \sum_{i=1}^N z_\phi(\hat{\mathbf{V}}_{k,i}, \mathbf{y}_i)$, where $\hat{\mathbf{V}}_{k,i}$ is the generated rollout including the initial frame for evaluation condition i , and $\mathbf{y}_i$ is its instruction. The binary decision is $z_\phi = \mathbb{I}[q_\phi \geq \tau_{\text{succ}}]$, with τ_{succ} denoting the terminal-success threshold. We compare the estimate with the corresponding RoboTwin success rate using Pearson

Table 10: 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 (%) $\uparrow$	Balanced acc. (%) $\uparrow$	F1 (%) $\uparrow$	AUROC $\uparrow$	ECE $\downarrow$
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 $\downarrow$	Progress ρ $\uparrow$	Validity AUROC $\uparrow$
RoboTwin videos	1,000	0.061	0.926	–
Pelican-Sim 1.0 videos	700	0.078	0.901	0.953

Table 11: 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 $\uparrow$	Spearman ρ $\uparrow$	Kendall τ $\uparrow$	MMRV $\downarrow$	MAE (pp) $\downarrow$
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

r for linear correlation, Spearman ρ and Kendall τ for ranking consistency, mean maximum rank violation (MMRV) for ordering errors, and MAE in percentage points for absolute success-rate error (Tseng et al., 2025).

Policy evaluation and ranking results. Table 11 summarizes agreement across the five adaptation budgets. With 200 world-model adaptation rollouts, the world-model–VLM pipeline reproduces the RoboTwin ranking of all five evaluated checkpoints, achieving Spearman $\rho = 1.0$, Kendall $\tau = 1.0$, and MMRV = 0. Larger budgets preserve this ranking agreement while improving absolute success-rate estimates: MAE decreases from 4.0 percentage points at 200 rollouts to 2.9 at 500 and 2.4 at 1,000. We use the 500-rollout variant in subsequent experiments, balancing estimation accuracy against adaptation data requirements; doubling this budget yields a further MAE reduction of 0.5 points. These results distinguish accurate policy ranking from accurate success-rate estimation within the evaluated checkpoint set.

4.7. Action Selection

Beyond policy-level evaluation, we test whether VLM-guided selection among outcomes predicted by Pelican-Sim 1.0 improves task success. A frozen policy proposes candidate action sequences, the world model predicts their outcomes, and the selector chooses a candidate using the VLM-derived utility.

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 accelerated 500-rollout adapted Pelican-Sim 1.0 model predicts each candidate’s outcome, and the frozen VLM evaluator ranks the resulting videos using u_ϕ from Section 4.1:

$$m^* = \operatorname{argmax}_{m \in \{1, \dots, M\}} u_\phi(\widehat{\mathbf{V}}^{(m)}, \mathbf{y}), \quad \widehat{\mathbf{X}}^{\text{tar}, (m)} \sim p_\theta(\cdot \mid \mathbf{x}_0, \mathbf{A}^{(m)}, \mathbf{y}). \quad (20)$$

Here, $\mathbf{A}^{(m)}$ is candidate m in the unified configuration layout, $\widehat{\mathbf{X}}^{\text{tar}, (m)}$ is its predicted future video, and $\widehat{\mathbf{V}}^{(m)} = [\mathbf{x}_0; \widehat{\mathbf{X}}^{\text{tar}, (m)}]$ 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.

Table 12: 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 (%) $\uparrow$	Δ SR (pp) $\uparrow$	Oracle gap (pp) $\downarrow$	Selection recall (%) $\uparrow$
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 13: 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 (%) $\uparrow$	Selected success (%) $\uparrow$	Selection recall (%) $\uparrow$	Sec./decision $\downarrow$
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

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 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 12 shows that the selector combining Pelican-Sim 1.0 with the frozen VLM evaluator achieves 63.8% success, exceeding *Single sample* and *Ctrl-World selection* by 20.6 and 11.5 percentage points. Its 93.7% selection recall leaves a 4.3-point gap to the oracle. The 15.1-point gain over *VLM-direct* supports the utility of predicted visual outcomes for selection, whereas *Random-of-10* performs similarly to *Single sample*.

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 13 shows that increasing M from 1 to 10 improves success from 43.2% to 63.8%, but recall falls from 100% to 93.7% as selection misses some available successful candidates; recall at $M = 1$ is 100% by construction. Increasing M from 8 to 10 yields only 1.3 additional percentage points of success while latency rises from 10.7 to 15.9 seconds, highlighting the success–latency trade-off.

4.8. Policy Improvement

We finally evaluate policy improvement through closed-loop rollouts from Pelican-Sim 1.0 and rewards derived from the fine-tuned VLM evaluator. The policy generates actions from predicted observations, the evaluator supplies task-related scores, and GRPO uses returns constructed from these scores to update the policy. Following World-Gymnast (Sharma et al., 2026), we initialize from a task-specific OpenVLA-OFT checkpoint (Kim et al., 2025, 2024) and use a Llama-2 language-model head in place of the L1 regression head to obtain action-token probabilities for GRPO, while retaining action-chunk prediction. Policy optimization uses the accelerated four-step causal version of the 500-rollout adapted model from Section 4.6. The world model and VLM evaluator remain frozen.

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 3.2). Each chunk is interpolated to 21 configurations, including the current configuration at the conditioning frame, and rendered through the URDF pipeline in Figure 4. 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 $\mathbf{c}_{h,k} \sim \pi_{\vartheta_{\text{old}}}(\cdot \mid \hat{\mathbf{x}}_{h,k}, \mathbf{y})$, with $\hat{\mathbf{x}}_{0,k} = \mathbf{x}_0$. The chunk is converted into frame-aligned configurations $\mathbf{A}$ as described above; unlike $\mathbf{a}_t$ in Section 3.1, $\mathbf{c}_{h,k}$ denotes an entire policy action chunk rather than one configuration. Pelican-Sim 1.0 predicts the corresponding observation segment, whose final frame becomes $\hat{\mathbf{x}}_{h+1,k}$. The imagined trajectories $\hat{\tau}_k = (\mathbf{x}_0, \mathbf{c}_{0,k}, \hat{\mathbf{x}}_{1,k}, \dots, \mathbf{c}_{H-1,k}, \hat{\mathbf{x}}_{H,k})$ support closed-loop policy optimization without additional RoboTwin interaction.

VLM-based dense reward. The frozen evaluator scores J successive prefixes of each rollout, assigning task-completion progress $p_{\phi,j,k}$ and visual validity $a_{\phi,j,k}$ to prefix $j \in \{1, \dots, J\}$ of trajectory k , together with terminal-success confidence $q_{\phi,k}$ for the full trajectory. We define the trajectory return R_k as

$$R_k = \sum_{j=1}^J \gamma^{j-1} a_{\phi,j,k} (p_{\phi,j,k} - p_{\phi,j-1,k}) + \beta a_{\phi,J,k} q_{\phi,k}, \quad (21)$$

where $p_{\phi,0,k} = 0$, γ discounts progress increments, and β weights the terminal-success bonus. This reward combines changes in task completion with terminal feedback, weighted by visual validity. We exclude 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 $\hat{A}_k$:

$$\hat{A}_k = \frac{R_k - \mu_R}{\sigma_R + \epsilon}, \quad \mu_R = \frac{1}{K} \sum_{k=1}^K R_k, \quad (22)$$

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

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

where $\rho_{h,k}(\vartheta) = \pi_{\vartheta}(\mathbf{c}_{h,k} \mid \hat{\mathbf{x}}_{h,k}, \mathbf{y}) / \pi_{\vartheta_{\text{old}}}(\mathbf{c}_{h,k} \mid \hat{\mathbf{x}}_{h,k}, \mathbf{y})$ 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. 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 21. 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 across five tasks, averaged over three independent policy-training seeds.

Table 14 shows that the policy trained with the full world-model-VLM optimization pipeline achieves a mean success rate of 75.4%, exceeding initial SFT, Iter-SFT, and terminal-only GRPO by 12.7, 8.4, and 6.9 percentage points, respectively. Gains over initial SFT occur on all five tasks, with the largest on T4 (53.7% to 68.0%); the full method remains 8.4 points below RoboTwin-GRPO. The terminal-only comparison evaluates the combined reward design, rather than isolating individual progress and validity contributions.

4.9. Generalization under Distribution Shifts

A general-purpose world-model simulator should accurately capture action-conditioned dynamics beyond its training distribution. We therefore evaluate Pelican-Sim 1.0 along five controlled axes: action trajectory, scene appearance, robot embodiment, manipulated object, and camera viewpoint. The task semantics and action horizon remain fixed while the designated factor is varied.

Table 14: 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 (%) $\uparrow$					
	T1	T2	T3	T4	T5	Mean $\uparrow$
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

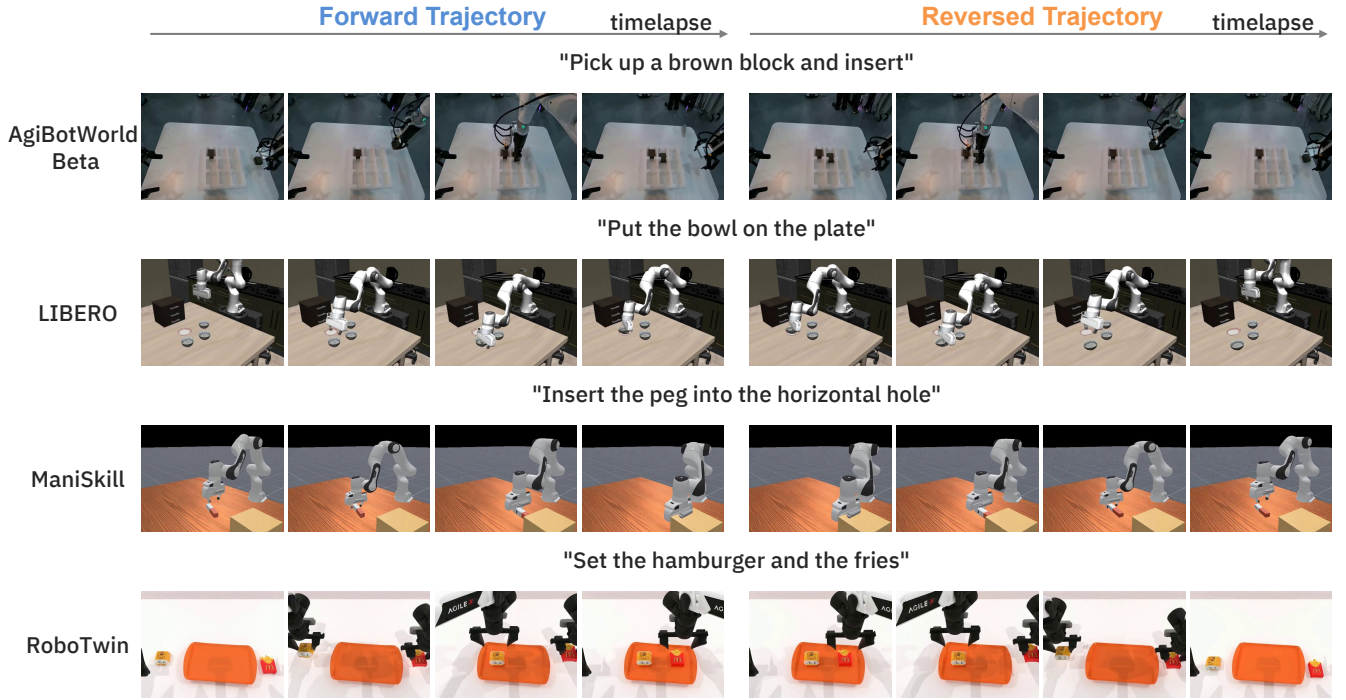


Figure 7: **Trajectory OOD.** Forward and reversed action-conditioned rollouts across four real-world and simulated benchmarks.

Trajectory OOD. We evaluate trajectory-level generalization using reversed robot motions. Given a forward rollout, we reverse its frame-aligned robot-configuration sequence and use the terminal observation as the new conditioning frame. As shown in Figure 7, the generated motions remain consistent with the reversed commands across both real-world and simulated tasks, driving the robot toward its original initial configuration. This setting assesses the sensitivity of the model to the supplied trajectory rather than to dominant forward-motion patterns in the training data. It evaluates action-conditioned trajectory reversal without assuming exact time reversibility in contact-rich dynamics.

Scene/appearance OOD. We use GPT-Image-2 (OpenAI, 2026) to synthesize OOD scene variants from the conditioning frame while preserving the task, robot pose, and spatial layout, and then reuse the same action trajectory for generation. Figure 8 includes six edited variants in addition to the original scene. Despite substantial changes in background, illumination, texture, and local object appearance, the generated rollouts remain temporally coherent and consistent with the specified folding behavior. This experiment evaluates broad scene and appearance variation rather than an isolated background shift.

Robot-embodiment OOD. To assess embodiment-level generalization, we retarget a manipulation trajectory from a source robot to embodiments excluded from training. Specifically, we extract the Cartesian end-effector pose trajectory of the source execution and solve inverse kinematics for each target robot, yielding embodiment-specific joint trajectories that are subsequently converted into unified action values and camera-aligned action videos. As shown in Figure 9, Pelican-Sim 1.0 generates block-stacking motions consistent with the source Panda trajectory for unseen Sawyer, IIWA, and UR5e embodiments. The qualitative results suggest that the model retains task-level motion semantics while accommodating changes in robot kinematics and projected geometry.

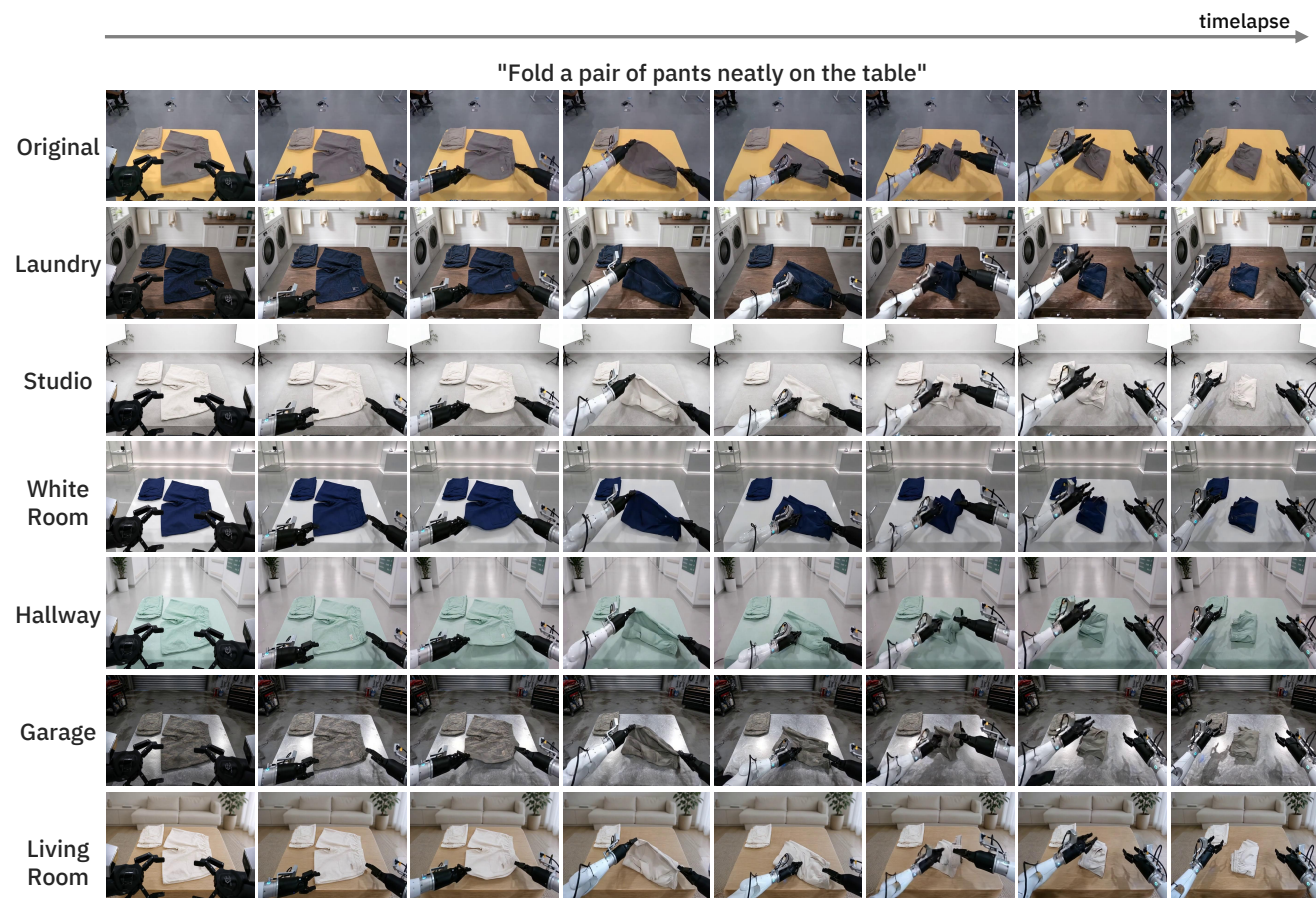


Figure 8: **Scene/appearance OOD**. Pants-folding rollouts in the original scene and six image-edited environments.

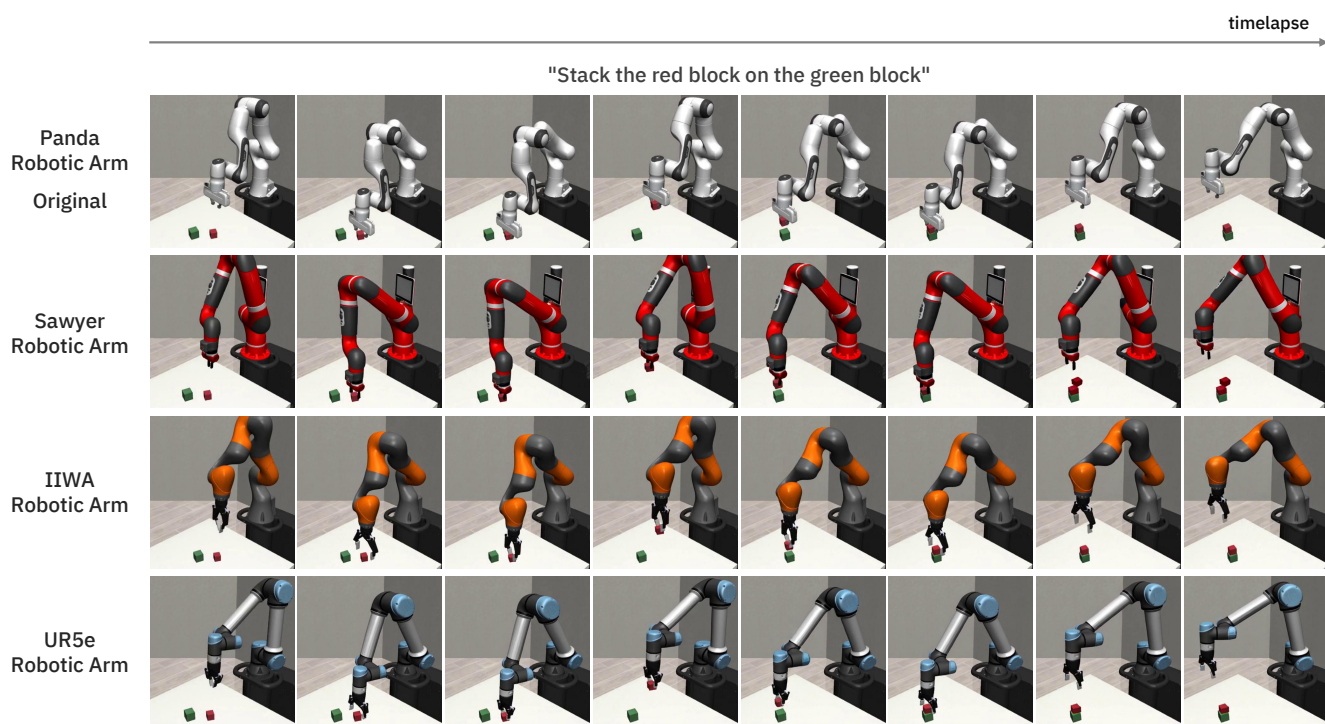


Figure 9: **Robot-embodiment OOD**. Block-stacking rollouts retargeted from Panda to held-out Sawyer, IIWA, and UR5e embodiments.

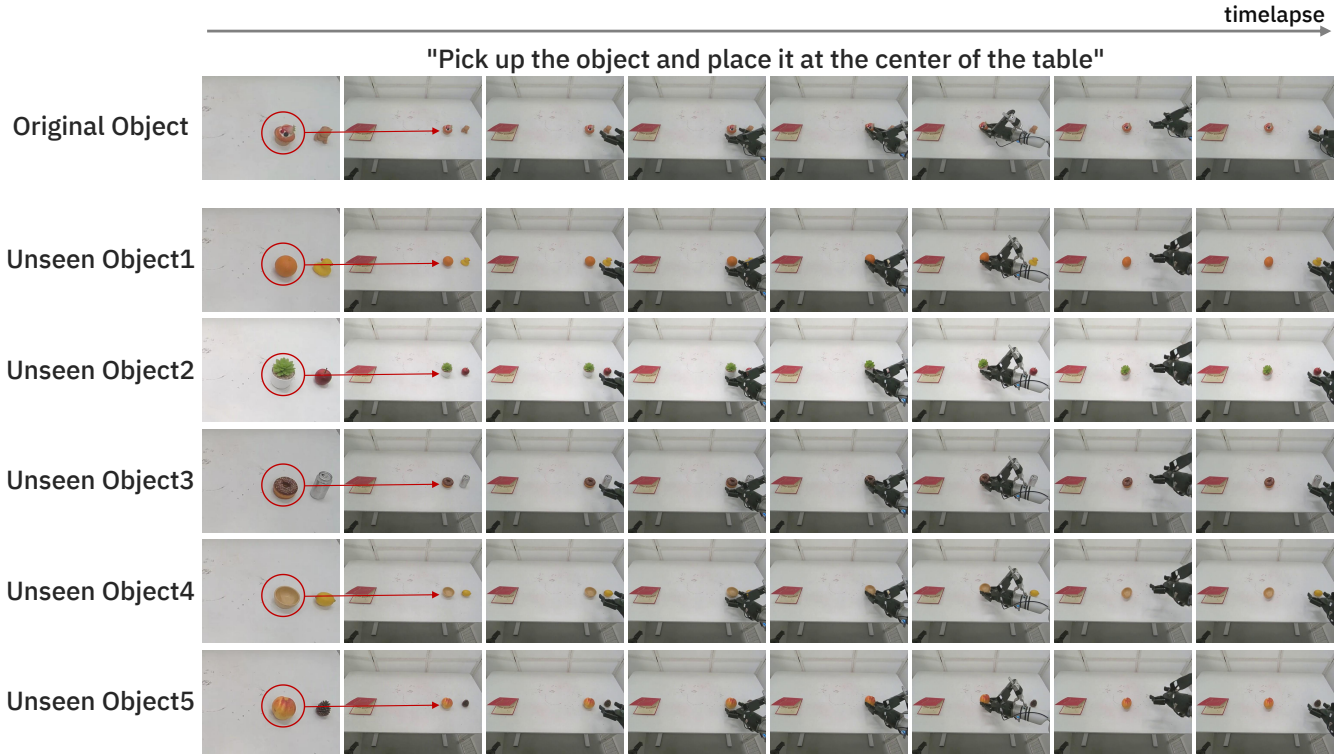


Figure 10: **Object OOD.** Pick-and-place rollouts for the original object and five image-edited object variants.

Object OOD. We replace the manipulated object in the initial observation using GPT-Image-2 (OpenAI, 2026), while preserving the robot configuration, coarse task-relevant spatial relations, and action trajectory. Figure 10 compares the original object with five edited variants that differ in appearance, shape, and scale. Given the same pick-and-place command, the generated motions remain directed toward the edited object, move it toward the center of the table, and largely preserve its visual identity over time. These observations suggest reduced reliance on the appearance of specific training objects. This setting represents controlled visual object variation and does not assume that the edited objects share identical physical properties.

Viewpoint OOD. We further evaluate initial observations captured from camera viewpoints excluded from training. For each target view, the corresponding camera calibration is used to render an action video spatially aligned with the novel viewpoint. Figure 11 compares the original camera with five unseen viewpoints for the task of picking up the black bowl between the plate and the ramekin and placing it on the plate. Across substantial changes in elevation, azimuth, and distance, the generated sequences remain consistent with the commanded robot motion and the intended interaction, suggesting robustness to camera configurations not observed during training.

Taken together, these controlled studies demonstrate broad qualitative generalization across temporal, visual, embodiment, and geometric distribution shifts. The results suggest that the unified numerical and visual action representations capture transferable motion structure, supporting Pelican-Sim 1.0 as a general-purpose world-model simulator for diverse robotic manipulation settings.

5. Conclusion and Future Work

We presented Pelican-Sim 1.0, an action-conditioned world-model simulator for robotic manipulation across heterogeneous embodiments and environments. Pelican-Sim 1.0 combines a unified 28-dimensional robot-configuration representation with URDF-rendered action videos, providing precise numerical specifications and explicit image-space motion guidance through complementary conditioning pathways in a Video DiT with sparse MoE layers for modeling heterogeneous robot dynamics. A multi-stage causal distillation procedure further reduces sampling from 35 to four steps for rollout-intensive applications. Experiments on held-out test splits from real-world and simulated datasets demonstrate improved video quality and action consistency and validate the contributions of the action representations, mixed-domain training, and the MoE design. Generated rollouts support data generation for policy training, while their integration with a fine-tuned VLM evaluator supports policy evaluation and ranking, action selection, and policy improvement. In these downstream procedures, the simulator predicts visual outcomes, the evaluator supplies task-conditioned feedback, and selection or optimization algorithms use that feedback to choose actions or update policies. The model also retains task-consistent motion under controlled changes in trajectory,

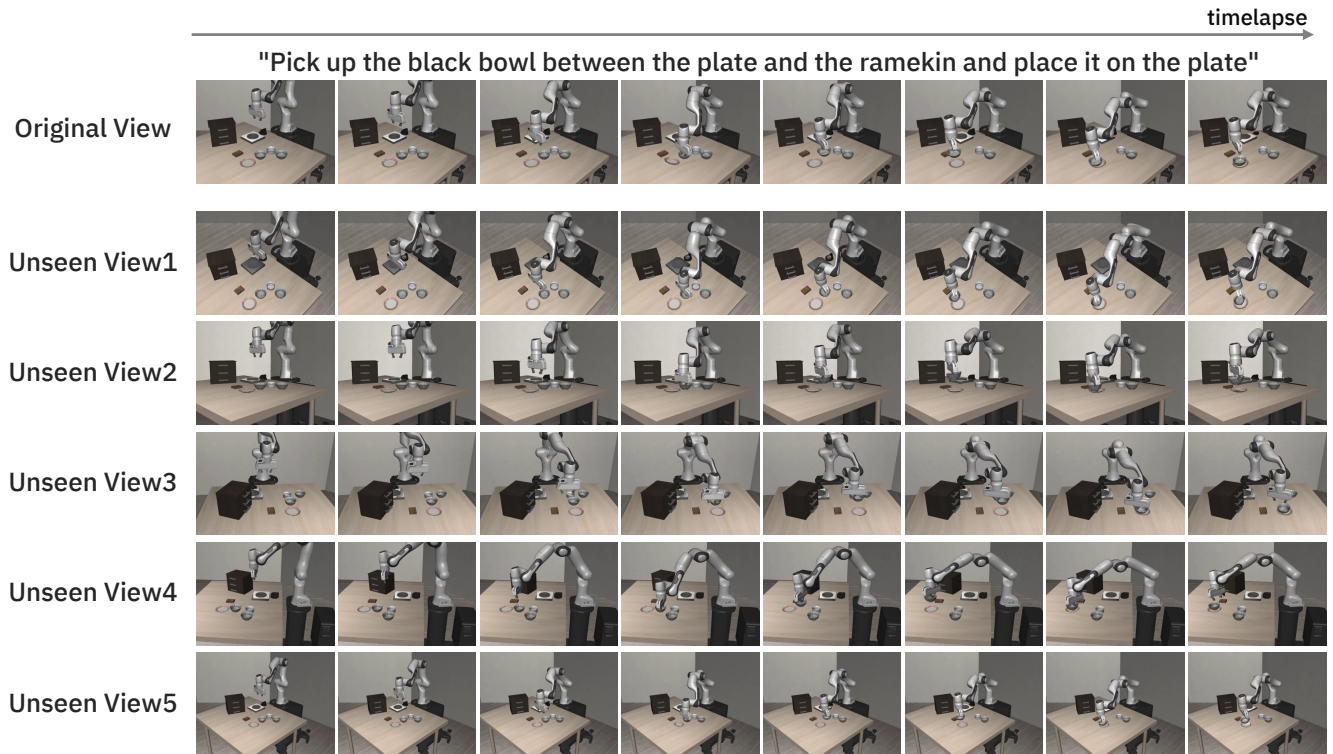


Figure 11: **Viewpoint OOD.** Action-conditioned rollouts from the original camera and five unseen viewpoints.

scene appearance, embodiment, object identity, and viewpoint, positioning Pelican-Sim 1.0 as a practical step toward a general-purpose world-model simulator for embodied intelligence.

Future work will pursue three directions. First, we aim to expand the curated training corpus to cover a wider range of embodiments, object interactions, sensing conditions, and long-horizon behaviors, and assess how this broader coverage affects transfer across domains. Second, we will systematically characterize the quality-efficiency trade-offs of causal adaptation and few-step distillation, particularly their effects on fine-grained dynamics, rollout stability, and generalization. Adaptive computation and hybrid teacher-student inference are potential approaches to reducing latency while preserving predictive quality. Third, we aim to make *generality* more precisely measurable through evaluation protocols that distinguish within-source prediction from transfer under explicitly defined distribution shifts. Such protocols should assess generalization across tasks, embodiments, action spaces, environments, and temporal horizons using complementary measures of perceptual fidelity, action consistency, physical plausibility, and downstream utility. Together, these directions would help establish the capabilities and limitations of general-purpose world-model simulators beyond broad training-data coverage.

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, Zanzin 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, Wen Yao 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.
- 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, Yanzhi 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. EWM-Bench: 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, 2025.
- 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.
- 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, Zanzin 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, Jiabin 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.
- 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.
- 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.
- 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.
- Yi Zhang, Yinda Chen, Che Liu, Zeyuan Ding, Jin Xu, Shilong Zou, Junwei Liao, Jiayu Hu, Xiancong Ren, Xiaopeng Zhang, et al. Pelican-unify 1.0: A unified embodied intelligence model for understanding, reasoning, imagination and action. *arXiv preprint arXiv:2605.15153*, 2026.
- 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.

6. Contributions

Contributors and their roles in the development of Pelican-Sim 1.0 are listed below.

6.1. Core Contributors

Shilong Zou contributed to data preparation, model architecture design, world-model training, and policy training for downstream tasks.

Shilin Zhang contributed to data preparation, policy training for downstream tasks, and visualization of experimental results.

6.2. Contributors

Yingji Zhang, Yuhang Huang, Yi Zhang, Zeyuan Ding, Han Dong, Junwei Liao

6.3. Tech Lead

Yong Dai, Shilong Zou

6.4. Corresponding Authors

Jian Tang, Xiaozhu Ju