CROSSWAM: IMAGINE, EXECUTE, REFINE FOR ZERO-SHOT CROSS-EMBODIMENT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalist robot policies, including vision-language-action (VLA) models and world-action models (WAMs), are typically pre-trained on data from many embodiments and then fine-tuned for one fixed robot, camera, and control interface. This coupling makes a trained policy difficult to reuse once the robot morphology or the camera configuration changes, even when the task and the underlying object dynamics are unchanged. We therefore study zero-shot cross-embodiment transfer, in which a policy trained on a single source embodiment must control robots never seen during training, under camera configurations never seen either, without collecting any target demonstration. Our starting point is that, how a task should unfold (how objects move and how contacts form) is largely independent of the robot that executes it, whereas the source robot's appearance is tightly bound to one particular arm and camera. Building on this observation, CrossWAM comprises three components. First, we make the imagined future of a world-action model embodiment-agnostic, removing the source arm by video inpainting and re-expressing its motion as an abstract gripper trace. Second, because a viewpoint change reprojects the same world state onto different pixels and thereby biases the predicted actions, we close the loop with an *imagination–observation refiner* that compares the imagined future against what actually happened after execution and corrects the actions not yet executed. Third, an asynchronous denoising schedule assigns fewer denoising steps to near-future frames, reducing first-video latency by 48.5%. On the bimanual RoboTwin 2.0 benchmark, which jointly varies embodiment and camera pose, CrossWAM improves success rate by up to 22.1% over the strongest prior method, and real-robot experiments on two platforms confirm that it transfers under simultaneous embodiment and viewpoint shift.

1 INTRODUCTION

Robot foundation models, such as vision-language-action (VLA) models and world-action models (WAMs), have substantially advanced generalist robot control by translating open-vocabulary language goals into executable motions and, increasingly, predicting how those actions will change the world (Ma et al., 2026; Wang et al., 2026b). These models are commonly pretrained on data collected from multiple robots and then fine-tuned on the target platform, so the deployment robot is usually already represented during training. In practice, however, the hardware and deployment environment keep changing, a general-purpose policy should therefore transfer to a robot and camera configuration that were never seen during training. Recent work has begun to address this cross-embodiment setting Chen et al. (2024b); Yan et al. (2026b), but it typically assumes that the camera configuration stays fixed. This assumption is restrictive for a general-purpose policy: real deployments often change the camera height, viewpoint, or intrinsics, and a relocated camera alters the image even when the scene and the task are unchanged. In this work, we study zero-shot cross-embodiment manipulation under simultaneous changes in both the robot embodiment and the camera configuration, without any target-robot demonstration or fine-tuning.

This setting presents two fundamental challenges. First, robots with different link geometries and appearances are rendered differently in the image, even when they interact with the same scene and perform the same task. Second, robot datasets are usually collected with a fixed robot and a few fixed cameras, so the mapping from image observations to spatial locations stays nearly constant through-

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

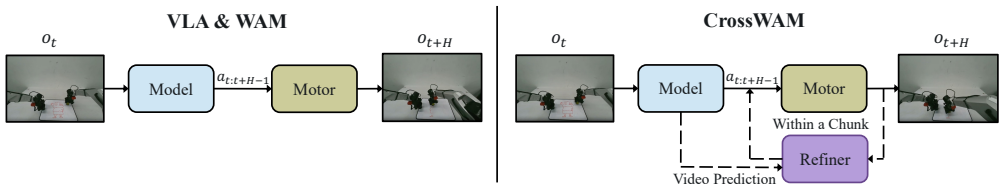

Figure 1: A conventional policy predicts a within-chunk action and executes it open loop, so errors caused by embodiment and camera shifts cannot be corrected until the chunk ends. CrossWAM retains the predicted future video and feeds the post-action observation to a refiner, which updates the remaining actions within the chunk and closes the control loop.

out training; a higher or lower camera at deployment silently invalidates this mapping, biasing the predicted actions without producing any explicit failure signal within an action chunk.

To tackle these challenges, world-action models are a natural starting point, because their imagined futures explicitly encode task progress and object interactions—semantics that are largely independent of the particular robot executing the motion (Ye et al., 2026b). Our key observation is that such a predicted future can serve as a more transferable interface than raw robot actions. Recent work such as Fast-WAM (Yuan et al., 2026) shows that video supervision is useful mainly during training and that the costly future generation can be dropped at inference to reduce latency. Our observation offers a complementary perspective: in the cross-embodiment setting, retaining the imagined future at inference provides an additional, transferable interface that helps bridge unseen embodiments and viewpoints. The difficulty is that an unconstrained future video also reproduces the source robot's links, textures, and camera-specific projection, precisely the appearance cues that impede transfer to unseen embodiments and viewpoints.

Building on these observations, we introduce CrossWAM, a world-action model for zero-shot cross-embodiment manipulation, with three components. First, we train the model to predict an embodiment-agnostic future by explicitly removing the source robot from the video. To preserve motion information, we render a trace connecting the robot's end-effector positions over time. Second, unlike approaches that use future prediction only as an auxiliary training objective, CrossWAM retains this representation at inference time and uses it to guide action generation. Specifically, because task progress and object interactions modeled by a world-action model are largely independent of the robot embodiment and camera configuration, we introduce an imagination–observation refiner. The refiner compares the predicted future with the real observation and corrects the actions not yet executed. As illustrated in Figure 1, rather than being emitted once as a feed-forward signal for an open-loop chunk, the imagination becomes a reference that the robot is continually corrected against, which is what provides robustness to embodiment and viewpoint shifts. Third, to reduce inference latency, we propose an asynchronous denoising mechanism that allocates more denoising steps to distant and uncertain frames while emitting near-future actions early.

In summary, our contributions are threefold:

1. We formulate a strict zero-shot cross-embodiment manipulation setting in which both the robot embodiment and the camera configuration can differ from those during training.
2. We propose CrossWAM, which combines embodiment-agnostic future imagination, an imagination–observation refiner, and an asynchronous denoising mechanism for closed-loop adaptation under embodiment and viewpoint shifts.
3. On the RoboTwin 2.0 benchmark and on real ARX and Piper platforms, CrossWAM improves success under embodiment shift, viewpoint shift, and their combination, outperforming prior VLA and WAMs by a clear margin.

2 Related Work

Robot foundation models. Robot foundation models aim to transfer the broad capabilities of large-scale visual and language models to physical interaction. Vision-language-action (VLA) models combine visual and language understanding with low-level control by adapting pretrained vision-

language models to robot trajectories. Early systems demonstrated that large-scale vision-language pretraining can transfer semantic knowledge to manipulation (Brohan et al., 2022; 2023), while later generalist policies extended training across heterogeneous robot platforms (Team et al., 2024; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025; Bjorck et al., 2025).

Another line of work augments action prediction with explicit modeling of the future (Kim et al., 2026; Ye et al., 2026b; Zhang et al., 2026b; Wang et al., 2026b). World-action models predict future observations conditioned on the current scene, language instruction, thereby exposing task progress and object interactions as an intermediate predictive signal. Recent studies have explored alternative supervision targets (Guo et al., 2026; Bi et al., 2026; Yan et al., 2026a; Yang et al., 2026; Chen et al., 2026) and model designs (Cai et al., 2026; Zhao et al., 2026; Li et al., 2026a) to improve both training efficiency and inference speed. In particular, Fast-WAM (Yuan et al., 2026) shows that future imagination can be omitted during inference to reduce computational latency. Our work shares the premise that future prediction is useful for robot control, but differs in how the future is represented and used.

Despite their strong capabilities, most existing robot foundation models follow a paradigm of pre-training on large-scale, diverse robot datasets followed by fine-tuning on a specific target platform. Their performance can degrade substantially when deployed with robot embodiments or camera configurations that were not observed during training (Yan et al., 2026b; Heo et al., 2026).

Cross-embodiment transfer. Existing cross-embodiment research can be broadly divided into two directions. The first studies how cross-embodiment data can improve the learning of a general robot policy. A common strategy is to construct a shared action space that reduces the dependence on robot-specific action coordinates. These spaces may be hand-crafted action space (Liu et al., 2025; 2026b), or learned from demonstrations (Ye et al., 2025; Zheng et al., 2025; Bu et al., 2025b; Li et al., 2026c; Kang et al., 2026). Other methods preserve a shared backbone while introducing embodiment-specific interfaces (Wang et al., 2024; Zheng et al., 2026; Wu et al., 2026).

The second direction focuses on cross-embodiment deployment under few-shot (Ren et al., 2025; Zhi et al., 2026), one-shot (He et al., 2026; Kedia et al., 2025), or zero-shot protocols (Chen et al., 2024b; Liu et al., 2026a; Zha et al., 2026; Yan et al., 2026b; Wang et al., 2026a; Piseno et al., 2026). (Chen et al., 2024b;c) proposed visual transformation methods that inpaint and repaint the source robot to make observations from different robots more compatible to training datasets. RDT2 (Liu et al., 2026a) and XRZero-G0 (Wang et al., 2026a) collect large-scale embodiment-agnostic UMI (Chi et al., 2024) data. ZETA (Yan et al., 2026b) has done empirical analysis and shown that a uniform state-action representations and embodiment diversity in pretraining datasets improves cross-embodiment transfer. However, most of these methods assume that the relative placement of the robot and camera remains unchanged, which is difficult to satisfy in practice.

Camera-viewpoint robustness. Existing methods address viewpoint shifts through three broad strategies. The first learns geometry-aware or viewpoint-invariant representations. Depth, point clouds, and learned three-dimensional features can provide a more stable description of the scene than raw RGB observations (Ze et al., 2024; Abouzeid et al., 2025). The second strategy augments training trajectories with camera perturbations or synthesized views (Tian et al., 2024; Chen et al., 2024c; Pang et al., 2025). The third strategy performs view conversion at inference time. AnyCamVLA (Heo et al., 2026) applies transform test-time images toward the viewpoint expected by the VLA, enabling zero-shot adaptation. However, these methods may require additional sensors, increased latency, while exhaustive coverage of all possible viewpoints remains impractical.

3 METHOD

3.1 PROBLEM FORMULATION AND PRELIMINARY

Let $\mathcal{E}_{\text{train}} = \{e_1, e_2, \ldots\}$ denote the set of source embodiments and $\mathcal{C}_{\text{train}} = \{c_1, c_2, \ldots\}$ the set of camera configurations seen during training, each comprising the camera intrinsic and extrinsic parameters. Let e' and c' denote the embodiment and the camera configuration at deployment. The policy is trained on a dataset $\mathcal{D}^{(\mathcal{E}_{\text{train}}, \mathcal{C}_{\text{train}})}$ consisting of images, language instructions, and actions. The training objective is

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta; \mathcal{D}^{(\mathcal{E}_{\text{train}}, \mathcal{C}_{\text{train}})}). \tag{1}$$

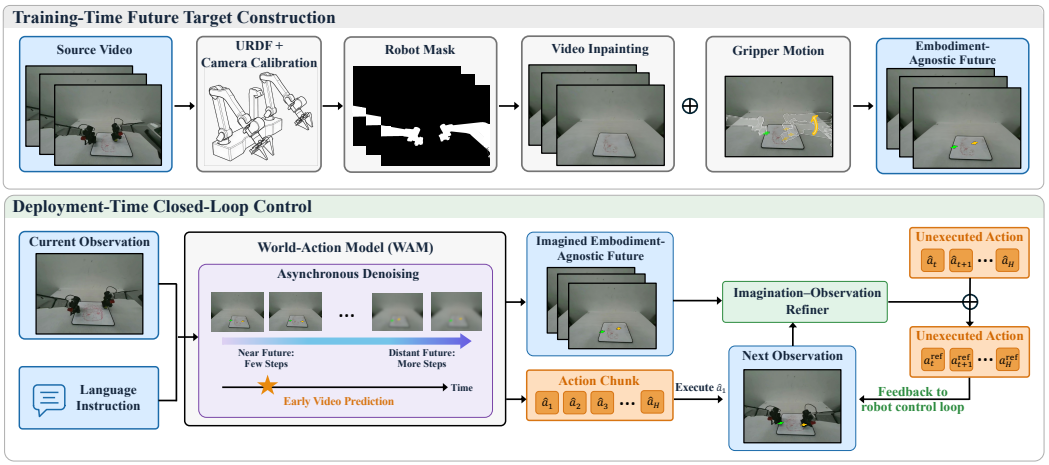

Figure 2: **CrossWAM pipeline.** During training, source demonstration videos are converted into embodiment-agnostic future targets. At deployment, the WAM predicts the imagined future and an action chunk; asynchronous denoising emits near-future actions early, and the imagination–observation refiner compares the imagined future with observation to correct the unexecuted action.

At deployment, the policy generates an action chunk of chunk H for the target embodiment and camera configuration:

$$a_{t:t+H-1} \sim \pi_{\theta^*}(o_t^{(e',c')}, \ell), \tag{2}$$

where $o_t^{(e',c')}$ is the observation and ℓ is the language instruction. Crucially, we do *not* require (e', c') to appear in the training data.

World-action model. CrossWAM builds on the world-action model that couples future prediction with action generation. Given the current observation o_t and instruction ℓ, the world-action model jointly predicts a future observation sequence and the corresponding action chunk, and is trained with a flow-matching objective on both branches.

3.2 Embodiment-Agnostic Future Targets

An unconstrained future prediction reproduces the source robot's links, texture, and camera-specific projection, giving the model a shortcut that ties the imagined future to one particular arm and thus hurts cross-embodiment transfer. To make the WAM learn embodiment-agnostic dynamics, we replace the raw future with a target in which the robot is removed but its task-relevant motion is preserved. As shown in Figure 2, given a source video $I_{1:T}$, we first render the source URDF under the camera calibration to obtain a binary robot mask $M_{1:T}$, and then apply a video inpainting operator, DiffuEraser (Li et al., 2025), to fill the masked region from neighboring frames:

$$B_t = \mathrm{Inpaint}(I_{1:T}, M_{1:T})_t. \tag{3}$$

Removing the robot entirely would also remove the interaction cue needed to predict control. We therefore project the end-effector pose into a small set of keypoints q_t and draw a canonical gripper trace $\mathcal{T}(q_t)$. The embodiment-agnostic future target is constructed as:

$$I'_t = \mathrm{Composite}\big(B_t, \mathcal{T}(q_t)\big), \tag{4}$$

where $\mathrm{Composite}(\cdot,\cdot)$ overlays the trace onto the inpainted frame. The trace records where the gripper is, where it is moving, and how it approaches an object, while discarding the source link visual identity. For bimanual robots, left and right traces use separate colors. Given the observation and instruction, CrossWAM jointly predicts a future video $\widehat{I}'_{t+1:t+H}$ and an action chunk $\widehat{a}_{t:t+H-1}$.

3.3 Imagination–Observation Action Refinement

Conventional VLA and WAM policies predict an action chunk and execute it open-loop, receiving no feedback until the entire chunk has been consumed. Executing a chunk this way lets errors from

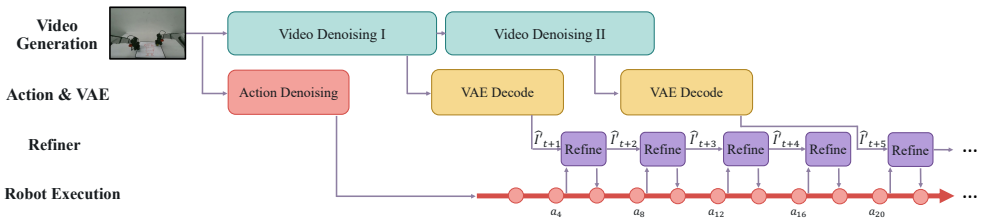

Figure 3: **Asynchronous denoising and closed-loop deployment.** Near-future video latents receive fewer denoising steps than distant-future latents, allowing robot execution and action refinement to begin before all future frames have been generated.

action prediction and execution accumulate, and deploying on an unseen embodiment aggravates the problem: a different arm geometry and a relocated camera both make the same world state project to different pixels, so an action that is optimal for the source setup is systematically biased for the target. CrossWAM instead closes an imagination–observation feedback loop around the chunk.

Throughout, each action is written in the world-frame end-effector space detailed in Section 3.5, $\hat{a}_\tau = \big(\hat{a}_\tau^{\mathrm{L}}, \hat{a}_\tau^{\mathrm{R}}\big)$ with $\hat{a}_\tau^m = (\hat{p}_\tau^m, \hat{q}_\tau^m, \hat{g}_\tau^m)$ for $m \in \{\mathrm{L}, \mathrm{R}\}$, where $\hat{p}_\tau^m \in \mathbb{R}^3$ is a translation, $\hat{q}_\tau^m \in \mathbb{R}^4$ a rotation represented as a unit quaternion, and $\hat{g}_\tau^m \in \mathbb{R}$ a scalar gripper command. Concretely, after the first h actions of the chunk have been executed, the target robot observes o_{t+h}, and a refiner $\mathcal{R}_\psi$ compares this observation against the imagined future $\widehat{I}'_{t+h}$ predicted for the same time index,

$$\big(\delta^{\mathrm{L}},\, \delta^{\mathrm{R}}\big) = \mathcal{R}_\psi\big(\widehat{I}'_{t+h},\, o_{t+h}\big), \qquad \delta^m \in \mathbb{R}^3 \tag{5}$$

which emits one translation offset per arm. We correct the translation only, because an embodiment or viewpoint change corrupts the pixel-to-position mapping and therefore displaces where an action lands much more than how the gripper is oriented on arrival. The offset is applied to every action that has not yet been executed, leaving the rotation and gripper channels untouched,

$$a_\tau^{\mathrm{ref},m} = \big(\hat{p}_\tau^m + \delta^m,\ \hat{q}_\tau^m,\ \hat{g}_\tau^m\big), \qquad m \in \{\mathrm{L}, \mathrm{R}\}, \qquad \tau = t+h, \ldots, t+H-1. \tag{6}$$

The refiner thus turns the discrepancy between what should happen and what actually happened into an error signal that corrects the remaining actions. This converts chunk-level execution from an open-loop rollout into closed-loop trajectory tracking, with the imagined future as the reference, the post-action observation as the measurement, and the residual δ as the correction.

Refiner training. We construct refiner training examples directly from the source-embodiment action trajectories. For each trajectory time t, we form $2H+1$ ordered pairs, (t, t), $(t, t+h)$, and $(t+h, t)$ for $h \in \{1, \ldots, H\}$, where the first index gives the state from which the imagined future is generated and the second gives the observation used as the feedback target. The refiner is trained to predict the relative action that moves from the first state to the second. The matched pair (t, t) represents an aligned imagination–observation condition, whereas the offset pairs expose the refiner to progressively larger temporal discrepancies in both directions; this teaches it to correct sizeable end-effector deviations while staying stable when the imagined and observed states already agree.

Embodiment-agnostic feedback. During cross-embodiment deployment, the imagined future and the real observation can contain visually different robot bodies even when the underlying task state is similar. We address the two inputs to the refiner symmetrically. On the imagination side, the WAM predicts future motion using the embodiment-agnostic targets introduced above, where robot motion is represented by end-effector keypoints and canonical traces. On the observation side, we render the same canonical end-effector traces on images which aid the refiner model to realize the robot pose. The refiner can therefore compare the contact-point and end-effector discrepancies in a common visual representation, instead of learning to match the appearance of different robot bodies, which reduces the difficulty of residual prediction and makes the feedback signal less sensitive to embodiment-specific geometry and texture.

3.4 Asynchronous Denoising

Explicit future generation dominates inference latency, because the video branch must be denoised before action refinement. To accelerate this step, we adopt diffusion forcing (Chen et al., 2024a)

and, under its per-token noise-level formulation, denoise the near future ahead of the far future. The temporal down-sampling of the video VAE encodes the predicted future into two latents, whose noise levels at denoising step s we denote by $k_1(s)$ and $k_2(s)$. We denoise the near latent ahead of the far latent, keeping $k_1 \leq k_2$ so the near latent is never noisier. During training we sample:

$$k_2 \sim \mathcal{U}[0, 1], \qquad k_1 \sim \mathcal{U}[0,\, k_2], \tag{7}$$

and denoise the two latents jointly at $\tau_1 = \phi(k_1), \tau_2 = \phi(k_2)$, where ϕ is the flow-matching shift. During inference we run T steps, driving the far level linearly to 0 while advancing the near level r times as fast,

$$k_2(s) = 1 - \frac{s}{T}, \qquad k_1(s) = \max\Big(0,\; 1 - \frac{r\, s}{T}\Big), \qquad s = 0, \ldots, T. \tag{8}$$

The near latent reaches $k_1 = 0$ at $s = T/r$ and is clamped there afterwards, so the near-future frames become available for the refiner, while the WAM keeps refining the distant future. We set $r = 2$ in all experiments, which roughly halves the number of denoising steps for near future prediction.

3.5 Overall

For the action representation, we operate in a world-frame end-effector space rather than in robot-specific joint coordinates; each action consists of a three-dimensional end-effector translation, a four-dimensional rotation represented by quaternion, and one-dimensional gripper command. A robot-specific inverse-kinematics solver maps these commands to native controls, removing action-dimension and coordinate mismatches across embodiments.

Putting the components together, as shown in Figure 3, at each deployment step, CrossWAM follows an *imagine—execute—refine* loop. Given the current observation, the action branch produces an initial action chunk in the shared world-frame interface, while the video branch concurrently denoises the future target sequence. The first action is dispatched as soon as it is available, without waiting for the full future video to finish. After the action is executed, the robot captures the next observation, and the refiner compares it with the asynchronously generated imagined future to predict residuals for the remaining chunk and these residuals are added to the unexecuted actions.

4 Experiment

4.1 Implementation Details

Model architecture. Both the WAM and refiner are initialized from Wan2.2 (Wan et al., 2025). The WAM predicts 8 future images, which under a temporal down-sampling factor of 4 correspond to an action chunk of 32 steps. Following Fast-WAM (Yuan et al., 2026), the action and video streams form a mixture-of-experts (MoE) that shares a common backbone but keeps separate expert parameters for the two modalities. We also mask the attention between the action and video experts, so the action stream can be decoded without attending to the full video stream, which reduces inference latency. The refiner shares the same backbone architecture to predict residual actions.

Training details. Following prior work (Guo et al., 2026), we pre-train the model on 2400 hours of robot data collected on the Genie G1 embodiment from AgiBotWorld-Beta (Bu et al., 2025a) and InternData-A1 (Tian et al., 2026), with a batch size of 2048 for 40k steps. We then post-train the model for the simulation and real world settings: on RoboTwin2.0 with a batch size of 512 for 3 epochs, and on our real-robot data with a batch size of 512 for 20k steps. All experiments are run on 32 NVIDIA H20 GPUs.

4.2 Simulation Results

Benchmark and protocol. We evaluate on the bimanual **RoboTwin 2.0** benchmark. The model is trained on the source **Aloha-AgileX** embodiment under the default camera and tested on the seen Aloha-AgileX embodiment as well as two held-out target embodiments, **ARX** and **Franka**, with the camera set to the **default**, **raised**, or **lowered** configuration. The raised and lowered views are obtained by translating the camera vertically within $[-0.15,\, 0.15]$ m of the default height and

Table 1: Randomized-scene test-time cross-embodiment results on the bimanual RoboTwin 2.0 benchmark (success rate, %). The policy is trained on the source **Aloha-AgileX** embodiment under the default camera; **ARX** and **Franka** are unseen. Best per column in **bold**.

Method	**Aloha-AgileX** (source)			**ARX** (unseen)			**Franka** (unseen)		
	Default	Raised	Lowered	Default	Raised	Lowered	Default	Raised	Lowered
$\pi_{0.5}$	85.4	64.8	54.9	54.4	38.2	34.6	17.8	12.8	13.1
RotVLA	88.5	33.5	26.0	50.9	16.9	12.2	11.5	10.8	6.4
StarVLA-α	88.3	37.2	36.2	52.4	23.8	18.0	20.2	12.5	11.8
Hy-VLA	90.1	52.1	42.1	60.2	31.9	29.3	12.9	8.6	7.2
Fast-WAM	**91.8**	57.2	54.0	53.9	29.2	33.2	22.4	14.6	13.7
LingBot-VA	91.5	48.7	40.9	64.5	39.8	34.6	3.9	3.4	3.2
X-WAM	90.7	70.4	63.9	43.6	33.2	34.3	17.0	13.5	13.5
CrossWAM (ours)	91.5	**81.1**	**72.0**	**71.2**	**60.7**	**56.7**	**33.0**	**26.4**	**28.5**

adjusting its pitch so that the upper edge of the field of view stays aligned with the top of the workspace. Crossing the three views with the three embodiments yields nine embodiment–camera combinations, and we run 100 episodes per combination for all 50 tasks. We compare against both vision-language-action and world-action models: four VLA baselines, $\pi_{0.5}$ (Intelligence et al., 2025), StarVLA-α (Ye et al., 2026a), RotVLA (Li et al., 2026c), and Hy-VLA (Zhang et al., 2026a); and three WAM baselines, Fast-WAM (Yuan et al., 2026), LingBot-VA (Li et al., 2026b), and X-WAM (Guo et al., 2026). Further evaluation details are provided in the Appendix B.1.

Table 1 reports the randomized-scene results, and the clean-scene results in Table 4 follow the same trend. The per-axis breakdown separates the two sources of shift. Under a pure viewpoint change on the *seen* Aloha-AgileX embodiment, CrossWAM improves over the best baseline by 10.7% and 8.1% for the raised and lowered cameras, showing that the closed-loop refinement helps even when the robot is unchanged. Under a pure embodiment change with the default camera, it improves by 6.7% on ARX, supporting our hypothesis that removing source-robot appearance from the imagined future aids cross-embodiment transfer. The margins are largest when the two shifts combine: on unseen ARX with a shifted camera, CrossWAM leads by 20.9% and 22.1%, and averaged over all unseen-embodiment columns it exceeds every baseline by at least 17.6%.

4.3 Real-World Results

Platform and tasks. Figure 4 summarizes the real-world setup along the three axes we vary: the task, the embodiment, and the camera viewpoint. All training data are collected on a bimanual ARX X5 platform with RGB observations, language instructions, and end-effector actions. We consider two tasks that probe complementary capabilities. *Pick-and-Place* requires the robot to grasp tabletop objects and place them into a box. *Whiteboard Wiping* requires one arm to hold a whiteboard while the other arm grasps an eraser and wipes the board, which demands sustained contact and bimanual coordination. Beyond the source ARX X5, we deploy on an unseen Piper platform that differs in link geometry, appearance, and camera mounting. Each task is evaluated under four deployment conditions that progressively increase the gap from training. On the source ARX embodiment, we test the training camera (in-distribution) and a raised camera (a pure viewpoint shift). On the unseen Piper embodiment, we test two camera poses that differ from training, jointly shifting embodiment and viewpoint. For a fair comparison, we evaluate against Fast-WAM, fine-tuned from the same pre-trained weights as CrossWAM. We run 10 trials per condition.

Metrics. We report two complementary metrics per setting. *Progress* measures how far the task advanced, in 25% increments, whereas *success* requires completing the task in full. For Pick-and-Place, half of the trials place four objects and the other half place three. Progress is the fraction of these objects delivered into the box. In a three-object trial, the first two contribute 25% each and the third contributes 50%. For Whiteboard Wiping, progress is the percentage of the board wiped clean, also scored in 25% increments, and success requires the board to be fully wiped.

Table 2: Real-world results. *Progress* is the percentage of the task completed, scored in four levels (0, 25, 50, 75, 100%); *success* requires completing the whole task. Best per column in **bold**.

Method	Pick-and-Place					Whiteboard Wiping				
	ARX (seen)		Piper (unseen)		Avg.	ARX (seen)		Piper (unseen)		Avg.
	Train	Rais.	Low.	Rais.		Train	Rais.	Low.	Rais.	
Success rate										
Fast-WAM (Pretrained)	**90**	70	50	60	67.5	50	30	20	20	30.0
CrossWAM w/o refiner	**90**	60	60	70	70.0	**60**	40	**40**	30	42.5
CrossWAM (ours)	**90**	**90**	**80**	**80**	**85.0** ↑17.5	50	**50**	**40**	**40**	**45.0** ↑15.0
Progress										
Fast-WAM (Pretrained)	**97.5**	77.5	72.5	77.5	81.2	**85.0**	52.5	45.0	50.0	58.1
CrossWAM w/o refiner	95.0	82.5	80.0	82.5	85.0	**85.0**	57.5	52.5	57.5	63.1
CrossWAM (ours)	**97.5**	**95.0**	**87.5**	**92.5**	**93.1** ↑11.9	82.5	**75.0**	**62.5**	**70.0**	**72.5** ↑14.4

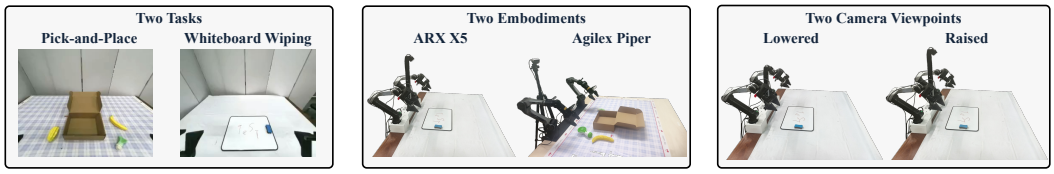

Figure 4: Real-world evaluation setup. **Left:** the two tasks, Pick-and-Place and Whiteboard Wiping. **Middle:** the two embodiments, the source ARX X5 and the unseen Piper. **Right:** the two camera viewpoints. Embodiment and viewpoint are varied independently and jointly.

Results. Table 2 shows that the advantage of CrossWAM appears once the deployment condition departs from training. Averaged over the four conditions, success rises from 67.5% to 85.0% on Pick-and-Place and from 30.0% to 45.0% on Whiteboard Wiping. Training with the embodiment-agnostic future alone (CrossWAM w/o refiner) helps mainly where the embodiment changes, for instance from 20 to 40 on the lowered Piper view of Whiteboard Wiping. The refiner further improves the results from 60 to 90 on the raised view of ARX on Pick-and-Place, and it also raises both Piper views further, for example from 30 to 40 on the raised view of Whiteboard Wiping. This holds even though the refiner is trained only on source-embodiment data and never sees the Piper.

4.4 Analysis

Component ablation. To verify that the gains originate from the proposed representation, we hold the backbone, training data, and action horizon fixed and add one component at a time to the Fast-WAM baseline (Table 3). Initializing Fast-WAM from our pre-trained backbone already improves robustness to camera changes. Training with embodiment-agnostic future accounts for the largest share of the improvement on unseen embodiments, supporting the view that source-robot appearance acts as a shortcut that the policy otherwise exploits. Without embodiment-agnostic future, refiner *degrades* performance. The reason is what the refiner is asked to compare. Its correction is the discrepancy between the imagined future and the observation obtained after execution, which is informative only if the two are expressed in the same terms. Without the embodiment-agnostic target, the imagined future contains the source robot, while the observation contains a different arm. The dominant difference between the two inputs is the appearance of the robot body, not the task-relevant displacement of the end-effector. Once the embodiment-agnostic future is incorporated for training, this appearance mismatch is removed and the refiner instead delivers a marked improvement under shifted views. Finally, adding asynchronous denoising leaves success essentially unchanged while reducing latency. The latency we report is the time from receiving an observation to emitting the first predicted video, which is the quantity that gates how soon the refiner can act. It is measured on the deployment described in Appendix C, where the model is split across two RTX 4090 GPUs, with the video generation and the refiner on one and the action expert and the VAE on the other.

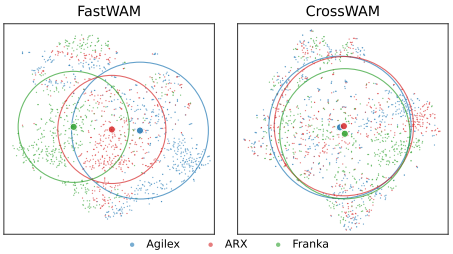

Figure 5: t-SNE of middle-layer features for the same task across embodiments.

Table 3: Component ablation on ARX. D, R and L represent for Default, Raised and Lowered

Variant	D	R	L	Latency
Fast-WAM	53.9	29.2	33.2	559ms
Fast-WAM (Pretrained)	61.9	36.4	42.0	559ms
+ Agnostic Future	69.3	44.0	48.5	559ms
+ Refiner	58.7	31.5	39.9	559ms
+ Agnostic Future + Refiner	71.0	**60.9**	56.4	559ms
+ Async (full)	**71.2**	60.7	**56.7**	**288ms**

Embodiment-agnostic representation. The ablation establishes that inpainted supervision improves transfer, but not why. We select trajectories of the same task performed by different embodiments from RoboTwin, encode each observation with the frozen model, and visualize the features of middle layer by t-SNE (Figure 5). Under raw-future supervision, features separate into compact clusters that coincide with embodiment identity. Under inpainted supervision, the same projection yields overlapping distributions in which embodiment is no longer the dominant factor of variation. We also calculate the quantitative metric that compute the inter-embodiment centroid distance and the intra-embodiment spread (mean distance of each embodiment's points to its own centroid). We report their ratio, cross / within, where a lower value indicates more embodiment-invariant. Training with embodiment-agnostic future achieve better score of 0.316 than with raw future 0.923.

What the refiner corrects. To make the correction mechanism concrete, Figure 6 visualizes the refiner's two inputs side by side: the imagined future and the post-action observation, on which we overlay arrows indicating the residual correction predicted by the refiner. When the embodiment or the camera differs from training, the same predicted action lands somewhere else in the world: in the figure, the executed motion leaves the block offset to one side, or the gripper reaches too far along the approach direction. Such offsets are small, but they decide whether a grasp, a placement, or a contact succeeds. The proposed refiner predicts a residual pointing back toward the imagined trajectory, and adding this residual to the unexecuted actions brings the rest of the chunk back onto the intended contact point.

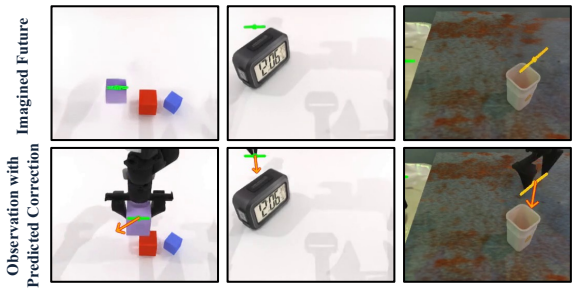

Figure 6: Qualitative visualization of the imagination–observation refiner. **Top:** the imagined future. **Bottom:** the corresponding post-action observation overlaid with arrows denoting the prediction of the refiner.

5 Conclusion

We presented CrossWAM, a world-action model for zero-shot cross-embodiment manipulation under simultaneous embodiment and camera-pose shift. It integrates three components: an embodiment-agnostic future target that removes the source robot by video inpainting and re-expresses its motion as a canonical gripper trace, so the imagined future encodes task dynamics rather than robot appearance; an imagination–observation refiner that closes a feedback loop around the executed chunk, correcting the remaining actions from the discrepancy between the imagined future and the realized observation; and an asynchronous denoising schedule that emits near-future frames early to reduce inference latency. On the RoboTwin 2.0 benchmark and on real ARX and Piper platforms, CrossWAM improves success under embodiment shift, viewpoint shift, and their combination. More broadly, our results suggest a new way to exploit a world-action model's video-prediction capability at test time, rather than treating it merely as training-time supervision.

Ethics statement

This work studies the transfer of manipulation policies across robot embodiments and camera configurations, evaluated in the RoboTwin 2.0 simulator and on two real robot platforms in a controlled laboratory setting. No human subjects or personal data are involved.

Reproducibility statement

Section 3 describes the embodiment-agnostic future target, the imagination–observation refiner, and the asynchronous denoising schedule, and Section 4.1 gives the model architecture, initialization, and training configuration. The appendix provides further details, including baseline adaptation for cross-embodiment deployment, the real-robot data collection, camera setup, and deployment procedure, and the asynchronous-denoising evaluation.

AI use statement

We used a large language model as a writing aid to improve the clarity and grammar of the text. We did not use generative AI to generate research ideas, to design the method, or to produce any experimental results; all reported numbers come from runs executed and verified by the authors. We have reviewed all AI-assisted text and take full responsibility for the final content of this work.

References

Ali Abouzeid, Malak Mansour, Qinbo Sun, Zezhou Sun, and Dezhen Song. Geoaware-vla: Implicit geometry aware vision-language-action model. *arXiv preprint arXiv:2509.14117*, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

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

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

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024a.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *arXiv preprint arXiv:2402.19249*, 2024b.

Lawrence Yunliang Chen, Chenfeng Xu, Karthik Dharmarajan, Muhammad Zubair Irshad, Richard Cheng, Kurt Keutzer, Masayoshi Tomizuka, Quan Vuong, and Ken Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. *arXiv preprint arXiv:2409.03403*, 2024c.

Yixiang Chen, Peiyan Li, Yuan Xu, Qisen Ma, Jiabing Yang, Kai Wang, Jianhua Yang, Dong An, He Guan, Gaoteng Liu, et al. Flowwam: Optical flow as a unified action representation for world action models. *arXiv preprint arXiv:2607.13017*, 2026.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Jingyang He, Guangrun Li, Jieyu Zhang, Chengkai Hou, Zhengping Che, and Shanghang Zhang. Jepa: Joint-embedding predictive architecture for one-shot cross-embodiment imitation. *arXiv preprint arXiv:2605.20811*, 2026.

Hyeongjun Heo, Seungyeon Woo, Sang Min Kim, Junho Kim, Junho Lee, Yonghyeon Lee, and Young Min Kim. Anycamvla: Zero-shot camera adaptation for viewpoint robust vision-language-action models. *arXiv preprint arXiv:2603.05868*, 2026.

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

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, et al. X-tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026.

Kushal Kedia, Prithwish Dan, Angela Chao, Maximus A Pace, and Sanjiban Choudhury. One-shot imitation under mismatched execution. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15649–15656. IEEE, 2025.

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

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

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, et al. Efficient-wam: A 1b-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026a.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

Qiwei Li, Xicheng Gong, Xinghang Li, Peiyan Li, Quanyun Zhou, Hangjun Ye, Jiahuan Zhou, and Yadong Mu. Rotvla: Rotational latent action for vision-language-action model. *arXiv preprint arXiv:2605.13403*, 2026c.

Xiaowen Li, Haolan Xue, Peiran Ren, and Liefeng Bo. Diffueraser: A diffusion model for video inpainting. *arXiv preprint arXiv:2501.10018*, 2025.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Songming Liu, Bangguo Li, Kai Ma, Lingxuan Wu, Hengkai Tan, Xiao Ouyang, Hang Su, and Jun Zhu. Rdt2: Exploring the scaling limit of umi data towards zero-shot cross-embodiment generalization. *arXiv preprint arXiv:2602.03310*, 2026a.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, et al. G0. 5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026b.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Jing-Cheng Pang, Nan Tang, Kaiyuan Li, Yuting Tang, Xin-Qiang Cai, Zhen-Yu Zhang, Gang Niu, Masashi Sugiyama, and Yang Yu. Learning view-invariant world models for visual robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 54853–54876, 2025.

Michael Piseno, Guy Tevet, and C Karen Liu. Cloak: Zero-shot cross-embodiment manipulation by masking the end-effector from the vla. *arXiv preprint arXiv:2606.22836*, 2026.

Juntao Ren, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg. Motion tracks: A unified representation for human-robot transfer in few-shot imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8802–8810. IEEE, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Stephen Tian, Blake Wulfe, Kyle Sargent, Katherine Liu, Sergey Zakharov, Vitor Guizilini, and Jiajun Wu. View-invariant policy learning via zero-shot novel view synthesis. *arXiv preprint arXiv:2409.03685*, 2024.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

James Wang, Primo Pu, Zephyr Fung, Alex Wang, Sam Wang, Bender Deng, Kevin Wang, Zivid Liu, Chris Pan, Panda Yang, et al. Xrzero-g0: Pushing the frontier of dexterous robotic manipulation with interfaces, quality and ratios. *arXiv preprint arXiv:2604.13001*, 2026a.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in neural information processing systems*, 37:124420–124450, 2024.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, et al. World action models: The next frontier in embodied ai. *arXiv preprint arXiv:2605.12090*, 2026b.

Tong Wu, Shoujie Li, Junhao Gong, Changqing Guo, Xingting Li, Shilong Mu, and Wenbo Ding. Cei: A unified interface for cross-embodiment visuomotor policy learning in 3d space. *IEEE Robotics and Automation Letters*, 2026.

Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π_0: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026a.

Mi Yan, Wenhao Zhang, Zhiqi Zhang, Yu Peng, Tangxinyu Wang, Lingfei Zhai, Jiayi Su, Shengliang Deng, Lin Peng, Yaowei Liu, et al. Zeta: A controlled study of zero-shot cross-embodiment vla transfer for tabletop manipulation. *arXiv preprint arXiv:2609.02546*, 2026b.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, et al. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields. *arXiv preprint arXiv:2608.08023*, 2026.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-$alpha$: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026a.

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

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b.

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

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Lihan Zha, Asher J Hancock, Mingtong Zhang, Tenny Yin, Yixuan Huang, Dhruv Shah, Allen Z Ren, and Anirudha Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.10556*, 2026.

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, et al. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*, 2026a.

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

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026.

Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22508–22519. IEEE, 2025.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Heng Zhi, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. Motif: Learning action motifs for few-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.13764*, 2026.

Huayi Zhou, Wei Gao, Dekun Lu, Ruiji Liu, Zhanqi Zhang, Ziyang Zhang, Jian Chen, Wenlve Zhou, Sheng Xu, Shumin Li, et al. Dexterity-bev: Aligning 3d world and actions for generalizable robot policies learning. *arXiv preprint arXiv:2606.02274*, 2026.

A LIMITATIONS

Target-side setup cost. Our protocol is zero-shot in the sense that no target-robot demonstration is collected and no parameter is updated, but it is not setup-free. Rendering the canonical gripper trace onto the target observation, which is what lets the refiner compare imagination and observation in a common representation, requires the URDF of the target robot together with the intrinsic and extrinsic parameters of the deployment camera. Both are easy to obtain: a URDF is released by the manufacturer of essentially every commercial arm, and the camera parameters follow from a standard calibration. Although neither is difficult to acquire, together they still add to the cost of deploying the model on a new platform.

Stale imagination. The reference against which execution is corrected is refreshed once per control cycle rather than continuously. Within a cycle the imagination is therefore held fixed while the scene keeps evolving, and under fast scene dynamics it can become stale before the next refresh, so the refiner compares the current observation against a reference that no longer describes the present state. This bounds how quickly the closed loop can react: disturbances that arise and resolve inside a single cycle are not reflected in the feedback signal at all. Extending the loop to update the imagination online, so that the reference tracks the scene continuously, is a natural next step.

Inference cost. Every control cycle requires denoising a future video and decoding it through the VAE, which dominates inference latency. Our asynchronous denoising schedule reduces the latency of the first video, but the remaining cost is still substantial, and in our real-robot deployment we split the model across two GPUs to meet the control rate (Appendix C). The cost can be absorbed by more capable hardware, yet doing so raises the hardware requirement for deployment rather than removing it, which limits applicability on platforms with a constrained onboard compute budget.

Trajectory discontinuity. Because the refiner revises the remaining actions during execution, the commanded trajectory can be discontinuous at the moment a correction is applied. Real-robot deployment therefore needs extra treatment: we ramp the predicted residual in with a linearly increasing weight over 3 steps and constrain the commanded actions, instead of applying the correction directly. These measures are not embodiment-specific and do not prevent transfer to a new robot, but they add deployment complexity and introduce a trade-off between trajectory smoothness and the responsiveness of the correction.

B SIMULATION EXPERIMENTS

B.1 EMBODIMENT AND CAMERA CONFIGURATIONS

Figure 7 shows the observations the policy actually receives across the two axes we vary. Each row is one embodiment—the source Aloha-AgileX and the unseen ARX and Franka—and each column is one camera height, from -15 cm to $+15$ cm relative to the training pose, with the pitch adjusted so that the upper edge of the field of view stays aligned with the top of the workspace. The columns therefore preserve the visible extent of the tabletop while changing how the scene projects onto the image.

Two aspects of the shift are visible in the grid. Along a row, raising or lowering the camera compresses and re-positions the workspace in the image, so the same world location maps to different pixels and the mapping the policy learned during training no longer holds. Down a column, the three embodiments differ in link geometry, thickness, and color, and they occlude different parts of the scene from the same camera pose. The **default** view used in Tables 1 and 4 is the 0 cm column. The **raised** and **lowered** views are not single poses: for each episode the camera height is drawn uniformly from 0 to $+15$ cm and from 0 to -15 cm respectively, so each reported score aggregates over a continuous range of displacements rather than one calibrated offset. The columns in the figure are representative samples from these ranges.

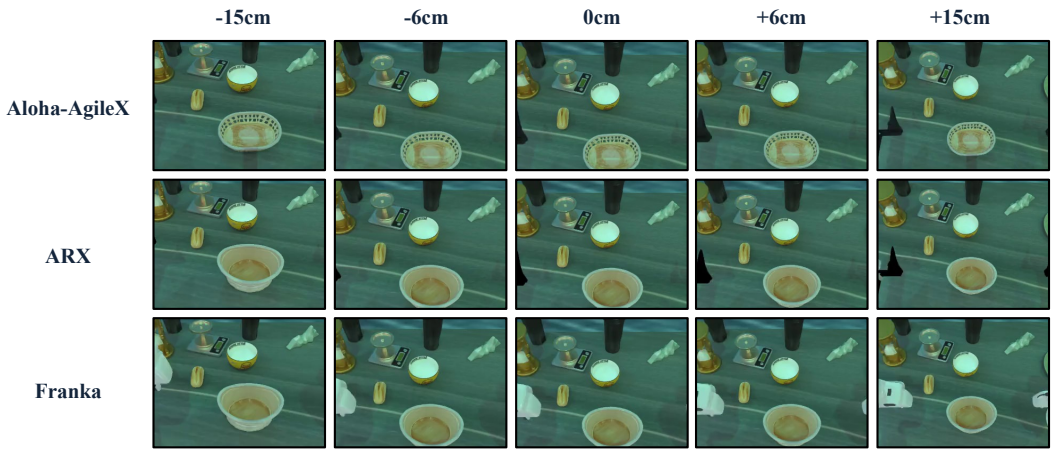

Figure 7: Simulation evaluation configurations on RoboTwin 2.0. **Rows:** the source Aloha-AgileX embodiment and the two unseen embodiments, ARX and Franka. **Columns:** camera height relative to the training pose, from -15 cm to $+15$ cm, with the pitch adjusted to keep the workspace in view. Only Aloha-AgileX at 0 cm matches the training condition.

B.2 Baseline evaluation details and clean-scene results

For all baselines we use the official code and released weights. A fully matched comparison at the pre-training stage is not possible, because the methods are pre-trained on different corpora and none of these corpora is public: LingBot-VA (Li et al., 2026b) uses 16k hours of robot data and X-WAM (Guo et al., 2026) uses 5800 hours, against the 2400 hours we use (Section 4.1). We therefore evaluate baseline from its officially released RoboTwin post-trained weights, so that every method is tested in the configuration its authors optimized and report, rather than one we would have to reproduce from an unavailable corpus. Baselines whose action space is defined in joint-pose coordinates cannot be deployed on target robots with a different kinematic structure. For these methods (StarVLA-α, Fast-WAM) we retrain a variant with an end-effector-pose action representation, keeping the official training setting unchanged and replacing only the action representation. End-effector pose provides an embodiment-agnostic interface that can be executed on the other arms through their inverse-kinematics solvers. Table 4 reports the clean-scene test-time cross-embodiment results on RoboTwin 2.0, complementing the randomized-scene results in the main text; the two settings follow the same trend.

Table 4: Clean-scene test-time cross-embodiment results on the bimanual RoboTwin 2.0 benchmark (success rate, %). Same protocol as Table 1 but without scene randomization. Best per column in **bold**.

Method	**Aloha-AgileX** (source)			**ARX** (unseen)			**Franka** (unseen)		
	Default	Raised	Lowered	Default	Raised	Lowered	Default	Raised	Lowered
$\pi_{0.5}$	84.1	66.0	51.2	55.9	40.1	34.3	20.6	14.1	14.2
RotVLA	89.6	32.5	24.8	46.5	16.7	12.8	13.6	12.0	7.1
StarVLA-α	88.2	35.3	31.9	55.5	27.9	19.7	19.2	13.4	12.4
Hy-VLA	90.9	50.8	40.3	61.6	33.7	27.1	14.7	12.9	9.1
Fast-WAM	91.9	57.8	51.1	57.2	31.7	34.2	25.5	17.4	16.2
LingBot-VA	**92.9**	53.1	41.7	70.6	46.8	38.7	4.2	3.8	3.8
X-WAM	89.8	70.0	58.3	42.3	33.0	31.2	19.6	16.9	17.1
CrossWAM (ours)	92.0	**81.4**	**68.3**	**73.4**	**62.2**	**53.7**	**32.5**	**26.9**	**28.1**

Table 5: Comparison of different baselines against CrossWAM (success rate, %) on the source Aloha-AgileX and the unseen ARX and Franka embodiments, under default, raised, and lowered camera views, in clean and randomized scenes. *Mirage operates purely at test time by replacing the robot in the observation, so on the source embodiment it is inactive and reduces exactly to Fast-WAM. Best per column in **bold**.

Method	**Aloha-AgileX** (source)			**ARX** (unseen)			**Franka** (unseen)		
	Default	Raised	Lowered	Default	Raised	Lowered	Default	Raised	Lowered
Clean scene									
Fast-WAM	91.9	57.8	51.1	57.2	31.7	34.2	25.5	17.4	16.2
+ Mirage	91.9*	57.8*	51.1*	62.7	36.4	38.9	28.5	19.9	18.8
+ Dex-BEV	91.5	64.8	58.1	57.6	36.7	39.2	25.9	22.4	21.2
CrossWAM (ours)	**92.0**	**81.4**	**68.3**	**73.4**	**62.2**	**53.7**	**32.5**	**26.9**	**28.1**
Randomized scene									
Fast-WAM	**91.8**	57.2	54.0	53.9	29.2	33.2	22.4	14.6	13.7
+ Mirage	**91.8***	57.2*	54.0*	59.4	33.9	37.9	25.4	17.1	16.2
+ Dex-BEV	91.4	64.2	61.0	54.3	34.2	38.2	22.8	19.6	18.7
CrossWAM (ours)	91.5	**81.1**	**72.0**	**71.2**	**60.7**	**56.7**	**33.0**	**26.4**	**28.5**

B.3 Comparison with cross-embodiment and camera-robustness methods

A natural alternative to our design is to keep a standard WAM and attach an existing technique that targets one axis of the shift. We test this on top of the Fast-WAM backbone: **Fast-WAM + Mirage** adds the cross-embodiment method Mirage (Chen et al., 2024b), which repaints the robot to bridge morphology gaps, and **Fast-WAM + Dex-BEV** adds the camera-robustness method Dex-BEV (Zhou et al., 2026). Table 5 reports their results on all three embodiments and camera views, in both clean and randomized scenes.

Mirage acts purely at test time: it detects the robot in the observation and replaces it with a rendering of the training embodiment, so the policy sees an input closer to its training distribution. On the source Aloha-AgileX embodiment there is nothing to replace, and the method reduces exactly to Fast-WAM. On the unseen embodiments, it does narrow the appearance gap, but only partially, because the substituted robot never matches the training embodiment exactly—the replacement is rendered at the target robot's joint configuration and under the deployment camera, so residual differences in link geometry, occlusion, and contact appearance survive the substitution. Being a test-time procedure also makes the result contingent on two upstream steps that the policy does not control: the robot must first be segmented correctly in the observation, and the replacement must be rendered plausibly enough for the policy to treat it as in-distribution. Where either step degrades, so does the correction, and no part of this pipeline addresses the camera shift at all.

Dex-BEV attacks the other axis by projecting the scene into a fixed canonical view, so that a camera at an unfamiliar pose is mapped back to the viewpoint the policy was trained on. The table shows this does recover part of the loss under the raised and lowered cameras. Two properties limit how far it goes. First, projecting a single view into a canonical frame discards information: regions visible from the deployment camera but not representable in the canonical view, and regions the projection leaves unfilled, both produce artifacts that never occur during training, where the projection is applied to the training view and is therefore nearly lossless. Second, the network consumes both the projected view and the raw observation; the projected stream is canonicalized, but the observation stream still moves with the camera, so a viewpoint-dependent input remains in the model.

B.4 How much of the cross-embodiment gap is closed?

To quantify how far the cross-embodiment results are from an in-domain oracle, Table 6 reports *default-camera* results on the two unseen embodiments (ARX and Franka) in both clean and randomized scenes. The first row, **Fast-WAM-ID**, is an *in-domain* reference: for each embodiment it is trained and tested on that embodiment's own data, so it never faces an embodiment shift and

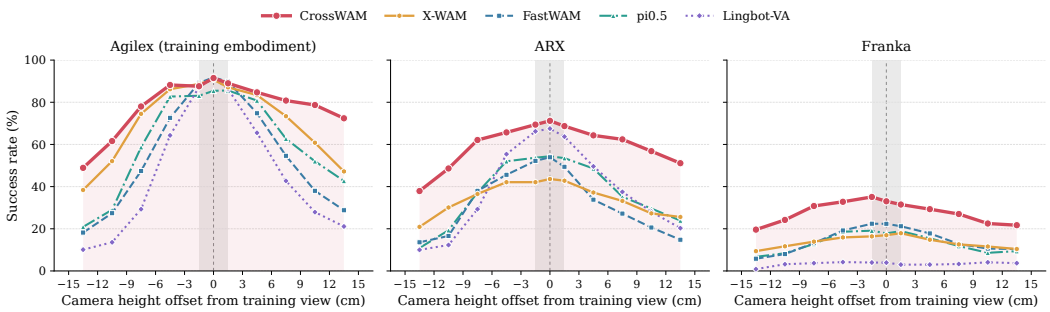

Figure 8: Random-scene success rate versus camera-pose shift on RoboTwin 2.0 for the three embodiments (left: Aloha-AgileX; middle: ARX; right: Franka). All methods degrade as the camera moves away from the training pose, but CrossWAM achieves the best accuracy on the unseen ARX and Franka embodiments and exhibits the smallest degradation across the range.

upper-bounds what a WAM of this capacity can achieve on the target. All other methods are trained only on the source Aloha-AgileX embodiment and transferred zero-shot; besides the Fast-WAM baseline, we also list the best result achieved by any existing method on each setting (*Best existing method*). The gap between Fast-WAM-ID and these zero-shot methods measures the cost of cross-embodiment transfer, and the gap between Fast-WAM-ID and CrossWAM measures how much of that cost remains after our method. Results show that CrossWAM closes a large fraction of this gap on both embodiments.

Table 6: Default-camera cross-embodiment results on the unseen ARX and Franka embodiments (success rate, %). Fast-WAM-ID is an in-domain reference trained and tested on each target embodiment's own data (an upper bound); all other methods are trained only on the source Aloha-AgileX embodiment and transferred zero-shot. *Best existing method* is the highest score reached by any prior method on each setting. Best zero-shot result per column in **bold**.

Method	**ARX**		**Franka**	
	Clean	Random	Clean	Random
Fast-WAM-ID (in-domain)	81.9	77.8	50.9	46.7
Fast-WAM	57.2	53.9	25.5	22.4
Best existing method	70.6	64.5	25.5	22.4
CrossWAM (ours)	**73.4**	**71.2**	**32.5**	**33.0**

B.5 Robustness to camera-pose shift

To characterize how performance changes as the camera moves, Figure 8 plots random-scene success rate against a continuously varying camera pose for each embodiment (Aloha-AgileX, ARX, and Franka). Two observations stand out. First, on the unseen ARX and Franka embodiments CrossWAM attains the highest success across the whole range, consistent with the aggregate results in the main text. Second, and more importantly, while every method degrades as the camera departs from the training pose, CrossWAM degrades the least: its curve is the flattest across all three embodiments. This indicates that the embodiment-agnostic imagination and the imagination–observation refiner make the policy substantially less sensitive to viewpoint change, rather than merely raising performance at a single calibrated pose.

C Real-World Experiments

Supplementary videos. The supplementary material contains rollout videos of CrossWAM on the real robots, covering both tasks (Pick-and-Place and Whiteboard Wiping), both embodiments (the

source ARX and the unseen Piper), and both camera heights, so that each deployment condition reported in Table 2 is shown at least once. All videos are played at $2\times$ speed.

Data and tasks. We collect 10 hours of demonstrations for Pick-and-Place and 5 hours for Whiteboard Wiping, all on the source ARX embodiment under the training camera configuration.

Robot configuration. Figure 9 shows the hardware setup for both platforms, including the arm layout, the workspace, and the placement of the head and wrist cameras used during data collection and deployment. The source ARX platform is used for data collection and for the ARX evaluation conditions, while the unseen Piper platform, which differs in link geometry and camera mounting, is used only at deployment.

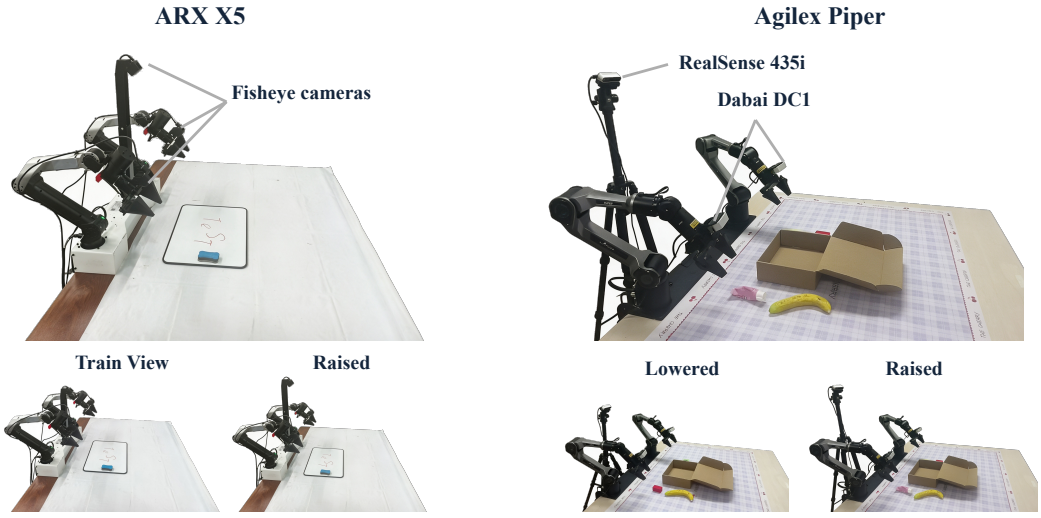

Figure 9: Real-world experimental setup. We show the ARX source platform used for data collection and the unseen Piper platform used only at deployment, together with the workspace and the head and wrist camera placements for the two tasks.

Deployment settings. We evaluate four deployment conditions, matching the columns of Table 2. **ARX, train view** uses the original data-collection robot and camera. **ARX, raised** keeps the ARX embodiment but raises the camera by 15 cm and rotates it 13° downward, so that the visible range of the tabletop is preserved despite the viewpoint change. **Piper, Lowered** and **Raised** deploy the unseen Piper embodiment under two camera configurations, where the head camera is placed between the two arms and the two configurations differ by a 15 cm spacing. On ARX we use three fisheye cameras, and we undistort the images before feeding them to the model at both training and test time. On Piper, the wrist cameras are Dabai cameras and the head camera is a RealSense D435i. To accelerate inference, we split the model across two GPUs as shown in Figure 3: the video generation and the refiner run on one RTX 4090, while the action expert and the VAE run on a second RTX 4090. The actions produced by the model are executed after $2\times$ temporal interpolation.

Refiner deployment. The refiner introduces two practical issues on the real robot: its inference has non-negligible latency, and its correction makes the executed trajectory discontinuous. We address both. First, we run the refiner *asynchronously*: when a correction is due at step t, we send the current image to the refiner while actions continue to execute uninterrupted, and once the refiner returns we apply the correction to the subsequent actions. Compared with synchronously waiting for the refiner, this delays the affected actions by only 1–2 steps. Second, to mitigate the discontinuity introduced by the correction, we apply the predicted residual gradually with a linearly increasing weight over 3 steps rather than all at once. Together these modifications substantially improve the smoothness of the executed trajectory.



(a) Missed Grasp (b) Placement Failure (c) Eraser Grasp Failure (d) Incomplete Wiping

Figure 10: Representative real-world failure cases of CrossWAM. **Pick-and-Place** (a–b): a *missed grasp*, where the gripper closes without securing the object and leaves it on the table, and a *placement failure*, where a grasped object is released outside the container. **Whiteboard Wiping** (c–d): an *eraser grasp failure*, where the eraser is missed or grasped off-center so that its contact face is misaligned with the board, and *incomplete wiping*, where the arm sweeps the board but the episode ends with part of the surface still marked. All four modes originate from a residual spatial offset at the moment of contact that the refiner reduces but does not fully eliminate.

Table 7: Effect of shortening the executed action horizon at inference (success rate, %, randomized scene). The model is frozen and identical in all rows; only the number of actions executed before replanning changes. Executing 12 instead of 24 actions doubles the replanning frequency yet degrades performance everywhere.

Variant	Default	Raised	Lowered
Fast-WAM (Pretrained, 24 actions)	61.9	36.4	42.0
Fast-WAM (Pretrained, 12 actions)	57.5	27.9	29.8
Cross WAM (24 actions)	**71.2**	**60.7**	**56.7**

Failure cases. We observe distinct failure modes for the two tasks, illustrated in Figure 10. For *Pick-and-Place*, the first mode is a *missed grasp* (Figure 10(a)): the gripper is displaced from the object, so it closes on empty space beside the object or contacts it without securing it, and the object is left on the table. The second mode is a *placement failure* (Figure 10(b)): the object is grasped successfully but is not delivered into the container, being released before the box is reached or at its edge. For *Whiteboard Wiping*, the first mode is an *eraser grasp failure* (Figure 10(c)): the eraser is missed entirely, or it is grasped away from its center; the contact patch is then misaligned and the wiping quality degrades even when the subsequent sweeping motion is executed correctly. The second mode is *incomplete wiping* (Figure 10(d)): the arm keeps sweeping but never covers the whole board, and the episode ends with part of the surface still marked. All four modes share a common origin: a small spatial offset at the moment of contact. This is consistent with the mechanism the method targets, since a new embodiment and a new camera pose both perturb the mapping from pixels to contact positions, and the refiner reduces but does not fully remove the resulting error.

D Additional Method Analysis

D.1 More ablation study

In this section, we show that shortening the action horizon is not a substitute for refinement. A natural objection to our design is that the observation only goes stale because the chunk is long, so simply executing fewer actions before replanning should recover the same benefit without any feedback mechanism. We test this directly. Keeping the model frozen and unchanged, we execute only the first 12 of the 32 predicted actions at each step and then replan from the new observation, which doubles the replanning frequency relative to our default setting. Table 7 reports the result: performance does not improve but drops, and it drops in every camera configuration.

Two mechanisms explain the drop. The first is a consequence of how an action chunk is distributed in time. The actions near the beginning of a chunk are predicted for states close to the conditioning observation, so they carry small displacements, whereas the later actions cover the larger motion

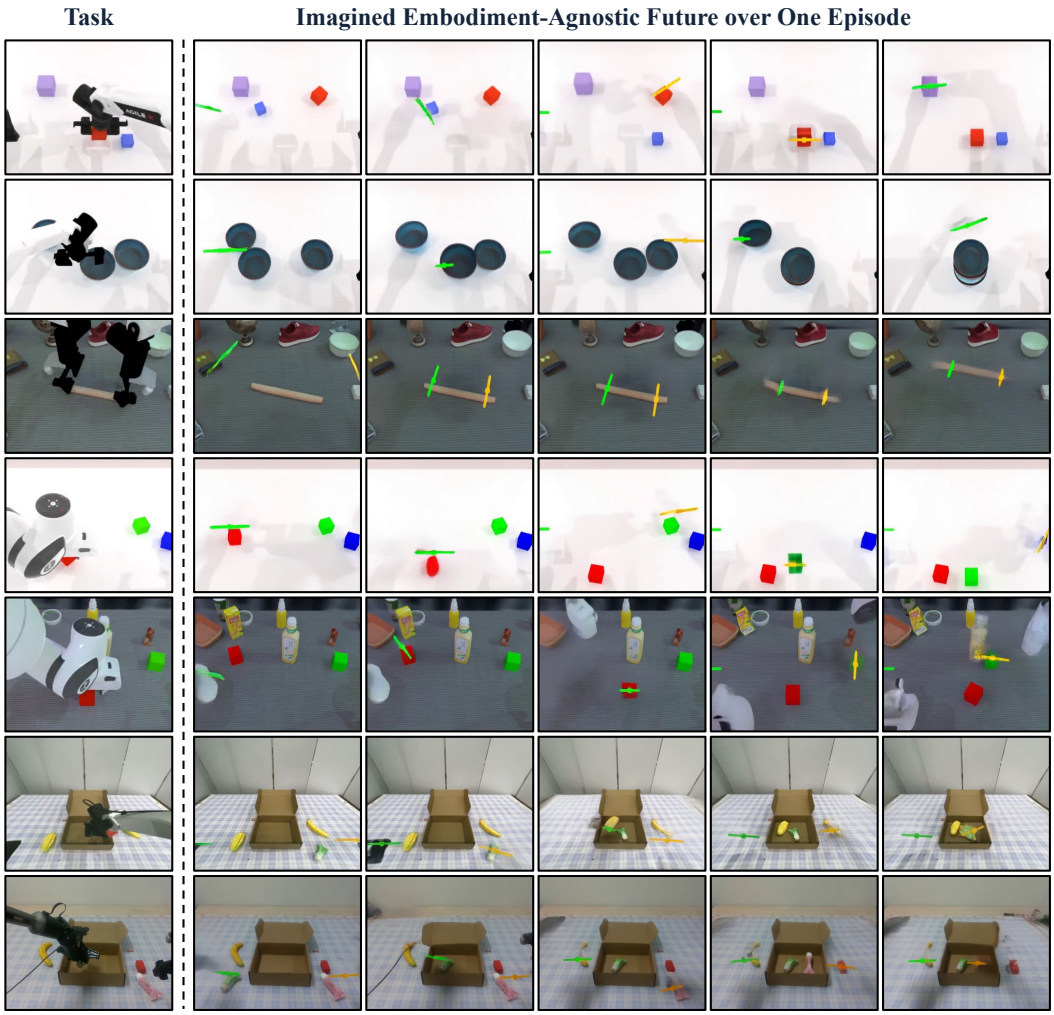

Figure 11: Embodiment-agnostic future prediction. The left panel illustrates the task; the remaining panels are frames sampled along a single episode of the imagined future, ordered in time.

that actually advances the task. Truncating the chunk keeps only the small-displacement prefix and discards the rest, so each replanning cycle advances the end-effector by a short distance. The executed trajectory becomes slow, and where a phase of the task requires deliberate low-speed motion the commanded displacement can fall below what is needed to make progress at all, leaving the arm effectively stationary until the episode times out.

The second is that replanning more often does give the policy a fresher observation, but it does not change what the policy infers from that observation. Under an embodiment or viewpoint shift the mapping from pixels to spatial positions is systematically biased, so every replanning step re-applies the same biased mapping to a new image and introduces a fresh offset of its own. Error therefore still accumulates, a shorter horizon slows this accumulation down, but it does not stop it, and nothing in an action chunk ever measures how far execution has drifted from what was intended. That is what the imagination–observation refiner supplies: by comparing the imagined future against the realized observation it estimates the accumulated discrepancy and subtracts it, which is a different function from predicting more often and cannot be obtained by any choice of horizon.

D.2 Visualizing the embodiment-agnostic future

To verify that the imagined future is genuinely embodiment-agnostic, we visualize the future images the model generates at inference along a single episode (Figure 11). Throughout the rollout the

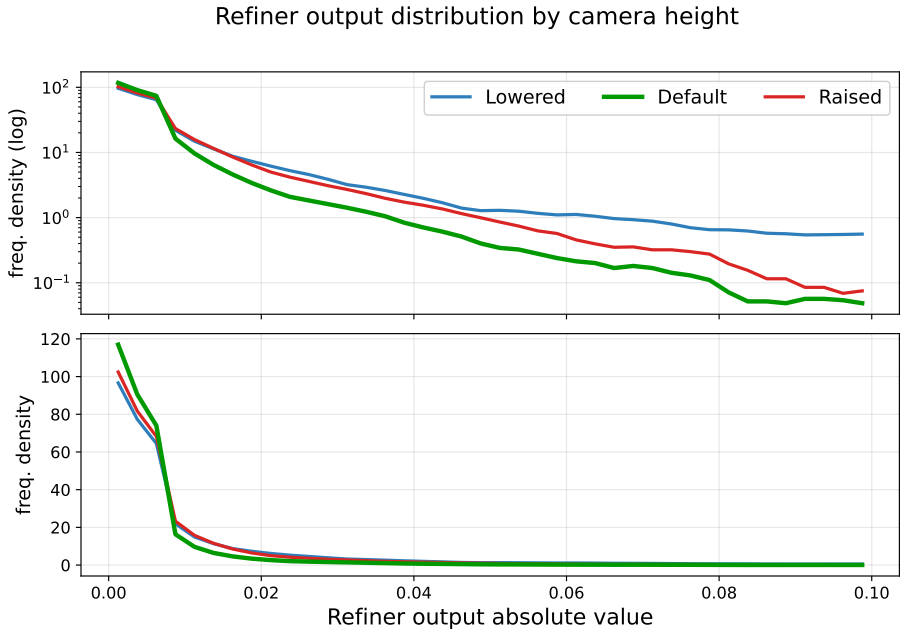

Figure 12: Frequency of the refiner's absolute output magnitude $|\delta|$ on the simulation test set under the default, raised, and lowered camera poses, shown with a linear (top) and a logarithmic (bottom) vertical axis. Default-view corrections concentrate near zero, whereas the raised and lowered views produce substantially more mass at larger magnitudes (roughly $|\delta| \geq 0.01$), showing that the refiner scales its correction with the severity of the viewpoint shift.

imagined frames contain no robot body: the arm is absent and its motion is expressed only through the canonical gripper trace, so the predicted future encodes object motion, contact, and task progress rather than the identity of the executing robot. Because nothing in these frames reveals which arm produced them, the same imagined future can be shared across embodiments. This confirms that the target defined in Section 3 makes the imagined future depend on the task rather than the embodiment, which is the property that enables the cross-embodiment transfer reported in the main results.

D.3 Magnitude of the refiner's correction under viewpoint shift

To examine how the refiner responds to a viewpoint change, we collect its outputs over the simulation test set and plot the frequency of the absolute translation correction magnitude $|\delta|$ under the default, raised, and lowered camera poses (Figure 12); we show the same histogram with a linear and a logarithmic vertical axis, as the log axis makes the tails easier to read. Under the default camera the corrections concentrate near zero, indicating that when the viewpoint matches training the imagined future and the observation already agree and little correction is needed. Under the raised and lowered cameras, the distribution shifts to larger magnitudes: beyond $|\delta| \approx 0.01$m the frequency is markedly higher than under the default view. This shows that the refiner does not emit a fixed offset but scales its correction with the severity of the viewpoint-induced discrepancy, applying larger residuals precisely when the camera shift biases the open-loop actions the most.

D.4 Asynchronous denoising requires matched training

We evaluate future-prediction quality on held-out RoboTwin 2.0 episodes that are unseen during training, reporting PSNR, SSIM, and LPIPS separately for the near future frames (Near) and the far future frames (Far), together with the latency of the first video prediction (Table 8). We compare four configurations: the synchronously trained baseline decoded synchronously steps; the same baseline forced to decode asynchronously at test time; and our model trained with the asynchronous schedule and decoded asynchronously. Applying the fast schedule to the synchronously trained baseline

Table 8: Future-prediction quality and latency under asynchronous denoising on unseen RoboTwin 2.0 episodes. Metrics are reported for the near future frames (Near) and far future frames (Far). *Training* is the noise schedule the model was trained with and *Inference* the schedule used to decode; the third row is the mismatched case, where the fast schedule is applied to a synchronously trained model. Best per column in **bold**.

Training	**Inference**	PSNR ↑		SSIM ↑		LPIPS ↓		Latency ↓
		Near	Far	Near	Far	Near	Far	
Synchronous	sync	29.93	28.14	0.924	0.912	0.062	0.078	559ms
Synchronous	*async*	29.93	26.66	0.923	0.899	0.062	0.087	**288ms**
Asynchronous (ours)	async	**29.98**	**28.20**	**0.925**	**0.914**	**0.061**	**0.076**	**288ms**

degrades the far frames sharply (Far PSNR drops from 28.14 to 26.66 and LPIPS rises from 0.078 to 0.087), because the model never saw the near latent run ahead of the far latent during training and cannot denoise the far frames from an out-of-distribution noise configuration. Our model, trained with ordered noise levels (Eq. 7), suffers no such degradation: it matches the synchronous baseline on the near frames and is in fact slightly better on the far frames (28.20 vs. 28.14 PSNR, 0.076 vs. 0.078 LPIPS), while the fast schedule reduces latency to 288ms. Asynchronous denoising is therefore not a free inference-time trick but a capability that emerges only when training exposes the model to the asymmetric noise levels it will face at test time.