WORLD-AS-PROMPT: REWRITING ROBOT OBSERVATIONS FOR ROBUST POLICY EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action and world-action policies can acquire useful manipulation from a modest dataset in one visual domain, yet illumination, sensor noise, object color, and workspace texture turn the same task into an out-of-distribution (OOD) observation. Collecting action-labeled data for each such factor is slow and hard to scale, and wrapping the policy with an extra agent or corrector still leaves it reading the shifted image. We introduce **World-as-Prompt (WAP)**, in which an in-domain reference image and a structured appearance description form a prompt for the world the frozen policy should see. We instantiate that prompt as a world-to-world (W2W) translator. At each control step, the translator conditions on the depth of the current OOD observation, that reference, and the description, changing nuisance appearance while retaining geometry and interaction state. The module is attached before an existing vision-language-action model (VLA) or world-action model (WAM), and requires neither new action-labeled robot data nor policy fine-tuning at deployment. We study WAP on RoboTwin2.0 Clean2Random, LIBERO-Plus, and real-robot cloth folding. Across simulation and real-robot evaluations, WAP improves every evaluated VLA and WAM by 10.3 to 53.3 percentage points in OOD success, demonstrating that it restores much of the competence that frozen policies lose under visual shift and offers a plug-and-play way to deploy existing VLAs and WAMs in new visual conditions.

1 INTRODUCTION

After broad pretraining, vision-language-action models (VLAs) and world-action models (WAMs) have turned manipulation into a scalable learning problem. π_0 and $\pi_{0.5}$ pair a vision–language backbone with a flow-matching action expert and transfer that policy across robots and tasks (Black et al., 2024; Physical Intelligence et al., 2025); DM0 and its scaled successor DM0.5 push the same direction with an embodied-native corpus and a larger interaction dataset (DM0 Team, 2026; Dexmal Team, 2026). WAMs such as DreamZero, Fast-WAM, and X-WAM add video prediction to this stack (Ye et al., 2026; Yuan et al., 2026; Guo et al., 2026). On a familiar camera and scene, these models can acquire useful behavior from a comparatively small task-specific dataset (Kim et al., 2024).

That progress has a sharp boundary once the camera leaves the training appearance. Illumination, sensor noise, object color, and tabletop texture can turn the same task into an out-of-distribution (OOD) observation. Embodied scaling has not closed this gap. Deployment variants are difficult to demonstrate, and they compose too quickly to be covered by collecting more of the same trajectories: physical collection is slow, the visual factors multiply, and domain randomization covers only the shifts anticipated before training (Tobin et al., 2017). RoboTwin2.0 and LIBERO-Plus make the remaining drop explicit (Chen et al., 2025; Fei et al., 2026). The practical question is therefore how to present an unfamiliar view to a policy that is already competent in one visual domain.

An emerging response is to improve the *system around* a capable frozen model. Modular WAM stacks isolate design choices that were previously entangled, while agentic robot harnesses combine planners, memory, analytic primitives, and frozen VLAs instead of repeatedly fine-tuning the low-level policy (Wang et al., 2026; Zhang et al., 2026b). A related pattern appears in language and image

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.

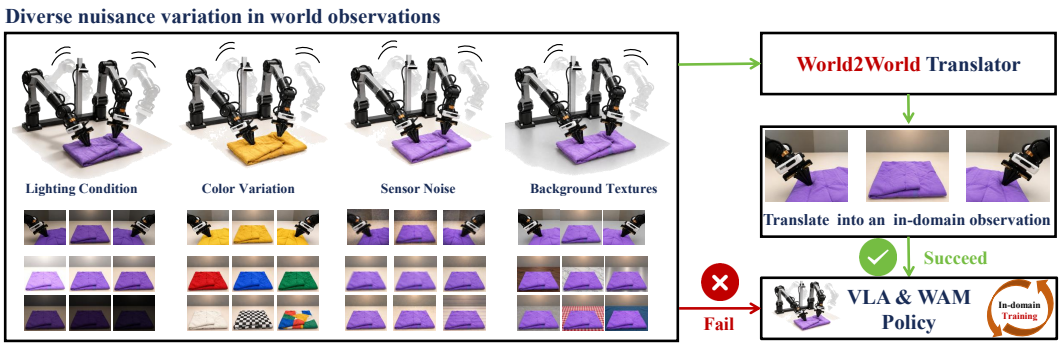

Figure 1: **World-as-Prompt overview.** Using cloth folding as an example, the left panel is the out-of-distribution scene: lighting, object color, sensor noise, and background texture. The red path sends this shifted image to an in-domain-trained VLA or WAM and fails. The green path uses a world-to-world translator to rewrite it into an in-domain observation, and the same frozen policy succeeds on that RGB.

generation: prompt optimizers rewrite a user input into a form preferred by the downstream model and improve the final task or generation output (Yang et al., 2024; Hao et al., 2023). These systems suggest a broader principle: when the core model is competent but its interface is OOD, adapting the interface can be cheaper and more reusable than adapting the model.

The same principle is now being explored by changing what serves as the policy. Code as Policies and CaP-X let a coding agent synthesize programs over perception and control primitives (Liang et al., 2023; Fu et al., 2026). HarnessVLA keeps a frozen VLA, while a memory-guided agent chooses primitives and recovery (Zhang et al., 2026b). Video generators are entering the same role: Seedance can propose a visual future that a decoder turns into action (Seedance Team, 2026; Liang et al., 2025). A stronger language model can take the role more directly. GPT-6 Astra selects manipulation actions from the camera image itself (Zhang et al., 2026a). Combining it with $\pi_{0.5}$ is more effective: $\pi_{0.5}$ proposes the joint motion, and Astra corrects only some of the executed steps (Su et al., 2026). These paradigms still share a limitation. Generated programs, harness calls, decoded video futures, and Astra corrections leave the shifted image inside the control loop, so the competent policy still acts from the unfamiliar view, or the system discards that policy and asks another controller to absorb the same visual shift.

We propose **World-as-Prompt (WAP)** for this remaining gap. Specifically, we train a world-to-world (W2W) translator (figure 3). It rewrites the current OOD observation into the visual domain the frozen policy already understands, and that policy then recovers execution on the shifted scene. The translated world keeps object pose, contact, identity, and task progress in the appearance the policy was trained to act on. Figure 1 shows this interface on cloth folding.

We design a dataset of paired robot observations for this new task. Each pair holds the task state fixed and changes appearance: the depth of the current observation, an in-domain reference image, a structured appearance description, and the matching in-domain RGB. From these pairs the translator learns to map an unfamiliar observation onto the target visual domain, while the downstream VLA or WAM is used as it stands, with no new action labels and no policy update.

We test it on RoboTwin2.0 and LIBERO-Plus with two VLAs and two WAMs, and on our real-robot cloth-folding $\pi_{0.5}$ checkpoint, where the original policies recover strongly. WAP improves OOD success for every evaluated policy under every perturbation. We also analyze its remaining failures and limitations (section 4.4). Our contributions are threefold:

- We propose World-as-Prompt, which treats a world observation as a prompt that can be rewritten to serve its downstream use.
- We define world-to-world (W2W) translation as a new task and train a translator for it on a purpose-built paired dataset.
- Our experiments show that existing VLA and WAM policies generalize across benchmarks and on a real robot without policy updates. They suggest that robustness can be supplied through what a policy sees, letting one policy trained in a familiar domain serve many complex environments.

2 Related Work

Robust training covers anticipated shifts. Visual imitation policies are sensitive to distribution shifts in viewpoint, background, and object appearance (Xie et al., 2024; Gao et al., 2026; Wang et al., 2025). Domain randomization broadens training support by varying simulator parameters (Tobin et al., 2017), and robot/view augmentation synthesizes additional training configurations (Tian et al., 2025; Chen et al., 2024b). Such methods are valuable when shifts can be enumerated before policy training, but they require retraining and grow with the desired coverage. A complementary line builds geometry-aware or view-invariant policies from point clouds, depth, and multiview features (Ze et al., 2024; Seo et al., 2023; Abouzeid et al., 2025; Wilcox et al., 2025), which alters the policy input or representation. WAP leaves both the policy and its RGB input format unchanged.

Agents reshape the control loop. Language models, coding agents, and generative models increasingly sit inside the control loop: they ground plans in robot skills (Ahn et al., 2022; Huang et al., 2022; Singh et al., 2023), compose programs or spatial constraints over primitives (Liang et al., 2023; Huang et al., 2023; 2024; Fu et al., 2026), wrap a frozen VLA with memory and recovery (Zhang et al., 2026b), or act as policies and correctors (Seedance Team, 2026; Liang et al., 2025; Zhang et al., 2026a; Su et al., 2026). WAP selects no action, so its translated stream can feed a frozen policy or a harness built around it.

Observation editing targets fixed shifts. Deployment-time preprocessing suppresses one nuisance factor at a time: photometric normalization and denoising correct lighting and noise, and segmentation masks remove distractors or the robot, with the policy trained on masked inputs (Mirjalili et al., 2026; Lepert et al., 2024). Image translators learn one domain mapping (Isola et al., 2017; Zhu et al., 2017): RCAN canonicalizes images for a policy trained on canonical renders (James et al., 2019), Mirage repaints an unseen robot as the training robot (Chen et al., 2024a), and MANGO translates simulated views to real ones for augmentation (Coholich et al., 2026). These methods correct one kind of shift or require retraining the policy on their outputs, while generic editors (Brooks et al., 2023; Zhang et al., 2023), video models (Wu et al., 2024; Bruce et al., 2024), and embodied generators such as Xiaomi-Robotics-U0 (Li et al., 2026) are used for editing, planning, or training-data synthesis. WAP instead specifies the whole target world with a reference and a structured description, so one translator handles lighting, texture, noise, color, and clutter together for different frozen VLAs and WAMs.

3 World-as-Prompt

This section develops WAP in three steps. We first define the task of rewriting an OOD observation for a frozen policy (section 3.1). We then collect state-matched observation pairs and process them into annotated training data (section 3.2). Finally, we design the World2World translator framework, covering its training, inference, and interface to the downstream policy (section 3.3).

3.1 Problem formulation

Let π_θ be a pretrained robot policy with frozen parameters θ. At time t, it maps a set of RGB observations $x_t = \{x_t^v\}_{v=1}^V$, a language instruction ℓ, and optional proprioception q_t to an action or action chunk,

$$a_t = \pi_\theta(x_t, \ell, q_t). \tag{1}$$

The policy is trained or fine-tuned on observations from an in-domain distribution $p_{\mathrm{ID}}(x \mid z, n_{\mathrm{ID}})$, where z denotes task-relevant physical state and n_{ID} denotes nuisance acquisition factors. At deployment, the same state may be observed under n_{OOD}, yielding $x_t^{\mathrm{OOD}} \sim p_{\mathrm{OOD}}(x \mid z_t, n_{\mathrm{OOD}})$. Our setting assumes that the task, embodiment, and action interface remain valid; the shift is primarily visual. This is stricter than arbitrary domain transfer but covers common changes in lighting, sensor response, object color, and workspace appearance.

World-as-Prompt treats the target world as a prompt. We train a W2W translator F_ϕ that rewrites the nuisance appearance of each observation into the in-domain appearance and preserves the task state seen by the policy:

$$d_t = D\big(x_t^{\mathrm{OOD}}\big), \qquad \hat{x}_t^{\mathrm{ID}} = F_\phi\big(d_t, r^{\mathrm{ID}}, s^{\mathrm{ID}}\big), \qquad a_t = \pi_\theta(\hat{x}_t^{\mathrm{ID}}, \ell, q_t), \tag{2}$$

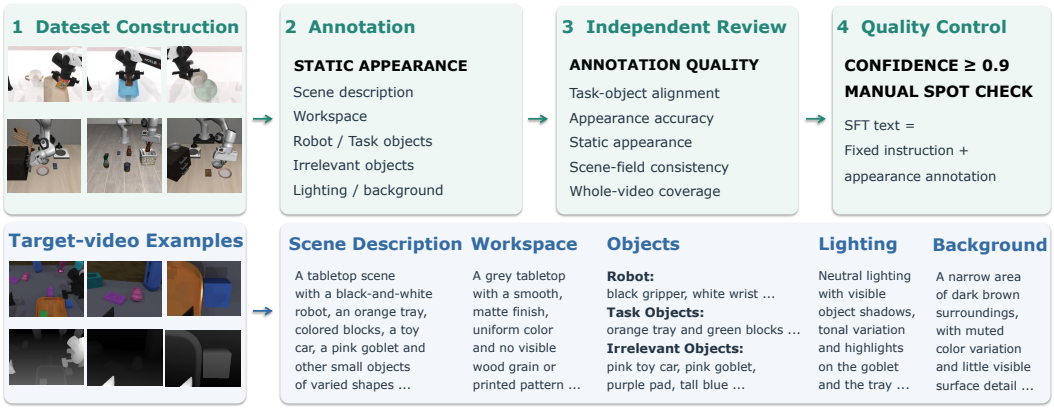

Figure 2: **Data pipeline and annotation example.** The pipeline has four parts: (1) dataset construction, (2) annotation, (3) independent review, and (4) quality control. We illustrate the structured prompt annotations using an example from the RoboTwin dataset.

where D is a frozen monocular depth estimator (Depth Anything 3 (Lin et al., 2025)), r^{ID} is a reference image from the desired domain, and s^{ID} is a structured description of that domain. The OOD image thus reaches F_ϕ only through its depth. Only ϕ is learned; D and θ stay fixed.

3.2 Structured Data Collection and Annotation

We design each training pair to separate appearance from task state. The source and target observe the same simulator state, the source under an OOD appearance and the target under the clean in-domain appearance, and the depth D estimates from the source view carries their shared geometry. The appearance reference is a clean in-domain frame taken far from the target in time or content, so it cannot serve as a shortcut to the target geometry. A structured description of the target appearance completes the pair. It uses six appearance fields: *scene description*, *workspace*, *task objects*, *irrelevant objects*, *lighting*, and *background*. The manipulated-objects field (*task_objects* in JSON) includes the robot and non-robot task entities. These fields contain appearance only; positions, poses, contacts, and action progress are excluded. Annotation warnings remain metadata, outside these six appearance fields. An attribute named by the instruction (e.g., "the red cup") is treated as task-relevant identity and kept in the task-object description rather than normalized as nuisance appearance. All pairs, drawn from RoboTwin and LIBERO scenes, share this format (section A).

Figure 2 shows the video-level annotation pipeline. Each pair uses the reviewed static appearance description of its target video. The annotation stage uses the target video, task instruction, and task-object information to produce a structured appearance annotation as JSON. An independent review checks this JSON against the same inputs and returns the issues it finds with a confidence score; only annotations with confidence of at least 0.9 are kept. We additionally conduct manual spot checks on a subset of the retained annotations to further verify annotation quality and consistency. The retained fields, combined with a fixed instruction, form the SFT text. This text is the appearance description in figure 3, used alongside the reference RGB and current depth, while the target RGB provides supervision. Review findings and model reasoning remain metadata; the full protocol is given in section B.

3.3 World2World Translator Framework

Figure 3 shows the two stages of the translator. The generator models discrete visual tokens autoregressively: a text tokenizer embeds the structured prompt, and a frozen VQ encoder maps the reference RGB, the depth, and the target RGB to token grids. Standard autoregressive decoding emits the $h \times w$ target tokens one at a time in raster order, which is too slow for a translator inside the control loop. We therefore fine-tune a FlashAR generator (Zhou et al., 2026) in the multi-condition form of Li et al. (2026): the target token at grid position (r, c) belongs to the anti-diagonal step

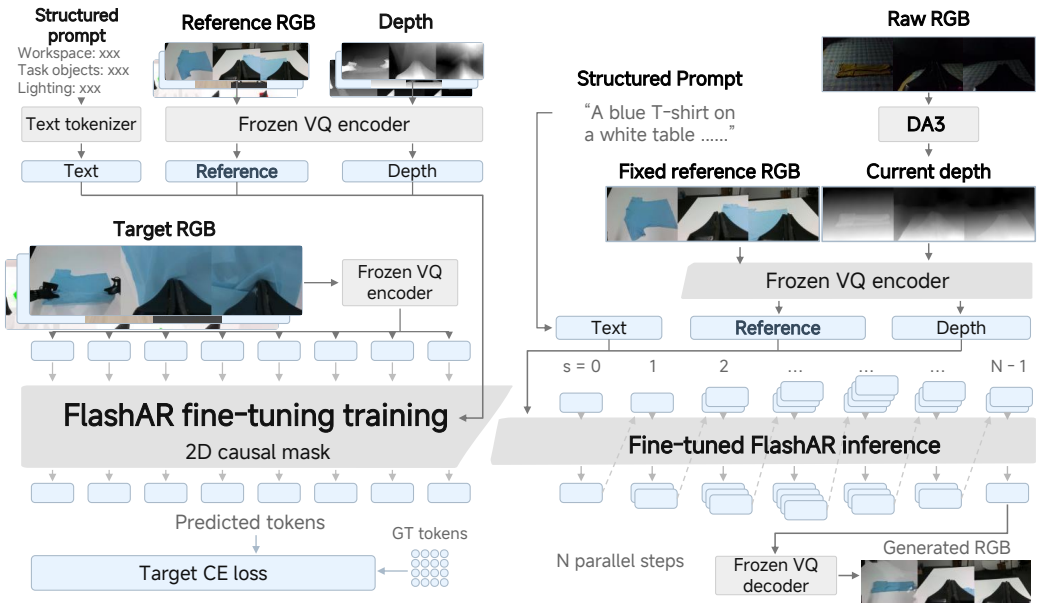

Figure 3: **World2World framework.** Left (training): the structured prompt, reference RGB, and depth condition the generator, which is fine-tuned with a cross-entropy loss on target RGB tokens. Right (inference): depth is estimated from the raw RGB and, with the prompt and a fixed reference, conditions the fine-tuned generator; a frozen VQ decoder maps the generated tokens to RGB.

$s(r, c) = r + c$, and all tokens of a step are predicted in parallel, so an image takes $N = h + w - 1$ steps instead of hw.

The three conditions in equation (2) serve different roles. Depth is the authority on the current scene: viewpoint, shape, count, pose, visibility, and contact. The reference provides high-bandwidth evidence about colors, textures, and rendering characteristics that are cumbersome to enumerate. Structured text, annotated with the six fields of section 3.2, makes the target transformation explicit and auditable and carries what depth cannot express, such as task-object identity.

Training uses the paired corpus of section 3.2,

$$\mathcal{D}_{\mathrm{tr}} = \{(d_i^{\mathrm{src}}, r_i^{\mathrm{ID}}, s_i^{\mathrm{ID}}, y_i^{\mathrm{ID}})\}_{i=1}^{K}, \qquad d_i^{\mathrm{src}} = D(x_i^{\mathrm{src}})\,. \tag{3}$$

For sample i, the condition tokens $\mathcal{C}_i$ (text, then reference, then depth; section C) form a prefix with standard causal attention, followed by the target tokens $u_{i,r,c}$ of y_i^{ID} on the grid Ω. Inside the target, a 2D causal mask lets the token at (r, c) attend to the whole prefix but only to target tokens of earlier steps,

$$M\big((r, c), (r', c')\big) = \mathbf{1}[s(r', c') < s(r, c)]\,. \tag{4}$$

Tokens of the same step cannot see each other, which matches parallel decoding and prevents leakage within a step. A horizontal and a vertical head predict each target token, fused as $z_{\mathrm{fuse}} = g_{r,c}\, z_{\mathrm{hor}} + (1 - g_{r,c})\, z_{\mathrm{ver}}$, where the scalar gate $g_{r,c} = \sigma\big(\mathrm{MLP}([\mathbf{h}^{\mathrm{hor}}_{r,c-1}; \mathbf{h}^{\mathrm{ver}}_{r-1,c}])\big)$ is computed from the hidden states of the left and upper neighbors; border positions use the single available head (Zhou et al., 2026; Li et al., 2026). Training minimizes the cross-entropy of this fused prediction on target tokens only,

$$\mathcal{L}_{\mathrm{tr}}(\phi) = -\sum_{i=1}^{K} \sum_{(r,c)\in\Omega} \log p_\phi\big(u_{i,r,c} \mid \mathcal{C}_i,\, u_{i,<s(r,c)}\big)\,, \tag{5}$$

where $p_\phi = \mathrm{softmax}(z_{\mathrm{fuse}})$ and $u_{i,<s}$ denotes the target tokens of steps before s. The prefix, including the reference, is context and never a generation target, and no actions or rewards enter the objective. The objective has no separate geometric or contact term; state preservation comes from the depth condition and from pairs that share one state. The SFT format, prompt, and configuration are given in section C.

At inference, D converts the raw OOD RGB into the depth condition d_t, and the structured prompt and a fixed reference complete the prefix. The generator decodes the target in N parallel steps, and a frozen VQ decoder returns the RGB image passed to the policy. The reference is a frame of the policy's training appearance from a disjoint training rollout, so it carries no future state of the evaluated episode. It identifies the target appearance and need not match the current state. One reference and one description serve the whole episode.

For $V > 1$, the views are tiled in a fixed order into one canvas, so the depth and the reference each form a single grid, and one request generates an RGB grid with the same layout under the shared appearance description. The generated grid is split back into views under their original names and order, so the downstream policy receives exactly the tensor layout used in training.

WAP is inserted in front of the policy's observation preprocessing. At every policy query, each camera frame is translated along the inference path of figure 3, resized by the policy's own preprocessing, and written back to its original observation key. Language, proprioception, action normalization, and the policy's internal state pass through unchanged, so VLA and WAM policies use the same wrapper. For policies with image history, each newly appended frame is translated, while caches and action chunks remain under the policy's control. Because every query waits for translation, its latency adds directly to execution time; the measured wall-clock cost is reported in section D.6, and we make no real-time claim.

Our use of *zero-shot* is correspondingly scoped: after one translator is trained, a downstream policy is deployed under a visual shift without new action labels, policy fine-tuning, or architecture changes. It is not zero-shot with respect to target-domain visual observations. WAP does not create competence for unseen tasks, embodiments, dynamics, or instructions. An empty irrelevant-object field prompts removal of irrelevant objects (section C.2).

4 Experiments

4.1 Experimental setup

Platforms and policies. We evaluate RoboTwin2.0 Clean2Random (Chen et al., 2025) with $\pi_{0.5}$ and X-WAM, LIBERO-Plus (Liu et al., 2023; Fei et al., 2026) under Light, Background, and Noise with π_0 and Fast-WAM, and real-robot cloth folding with a Piper arm and $\pi_{0.5}$ post-trained on 511 episodes (section D.3). The real-robot shifts cover cloth color, tabletop, and illumination. Its policy takes three RGB views resized to 224×224 and a 14-dimensional joint-angle state, and predicts 50-step chunks of 14-dimensional joint-angle actions. We use the original LIBERO-trained checkpoints and keep all downstream policies frozen during WAP evaluation. Each direct baseline uses the same policy checkpoint on raw observations. Published comparison models and their score sources are listed in section D.2; translator data are described in section A.

Protocol. We compare raw and translated observations while holding the policy and nonvisual inputs fixed. The translator is fixed before evaluation and trained without evaluation actions or success labels. LIBERO-Plus uses official task configurations and success predicates, with matched resets and seeds across runs. Each policy is evaluated on $1{,}142$ Light, $1{,}076$ Background, and $1{,}601$ Noise episodes per condition. The real robot uses five paired trials per condition, keeping the instruction, task geometry, initial-state distribution, and terminal-fold success criterion fixed. RoboTwin follows the clean-only protocol with 50 demonstrations per task across 50 tasks, and 100 evaluation trials per task in each of the clean and randomized conditions. The LIBERO-Plus translator uses benchmark configurations, so these experiments evaluate transductive target-domain visual adaptation. RoboTwin randomization includes physical changes, and real-robot cloth/tabletop replacements are shared across methods. Detailed protocols and evaluation scope are given in section D.1.

Metrics. We report task success rates and absolute changes with WAP, expressed in percentage points. LIBERO-Plus reports the unweighted mean over Light, Background, and Noise; real-robot Overall OOD pools trials across the three shifted conditions. Paired outcomes distinguish rescued raw failures from retained raw successes; clean-scene reruns diagnose episodes that fail in both conditions. Runtime comparisons use episodes successful in both runs and report control steps, policy calls, and episode time including translation. Failure diagnostics and runtime are reported in sections D.5 and D.6.

4.2 VISUALIZING WORLD2WORLD TRANSLATION

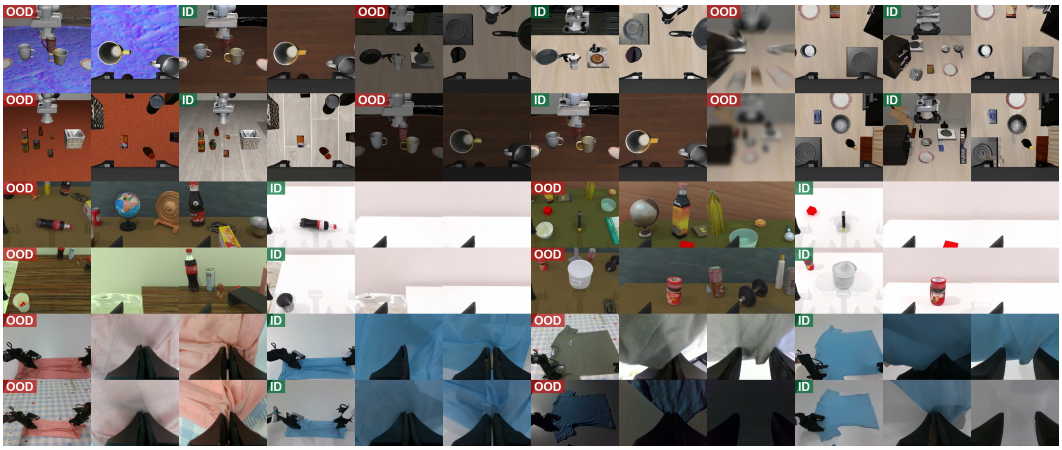

Figure 4: **World2World translation examples.** In each pair, the OOD observation (left) is translated into the in-domain observation (right). Rows 1–2: LIBERO-Plus background, lighting, and noise shifts (agent and wrist views); rows 3–4: RoboTwin2.0 randomized scenes (head and two wrist cameras); rows 5–6: real-robot cloth folding under different illumination, cloth colors, and tabletop backgrounds (top and two arm cameras).

Before the control results, figure 4 shows what the W2W translator passes to the frozen policies. On LIBERO-Plus, the purple and red-brick tabletop textures are replaced by the wooden surfaces of the training scenes, dark scenes are brightened, and zoom and Gaussian blur are removed. Because the raw and translated rollouts are executed separately, we show their shared first frame. On RoboTwin2.0, randomized textures, lighting, and clutter are mapped to the clean white workspace, while the task objects stay in place. On the real robot, the pink and olive shirts become the blue training shirt, the checkered tablecloth becomes the white table, and the dim scene is brightened. In these examples, arm and gripper poses, object layout, and cloth configuration are visually preserved. The examples are selected for illustration; the final analysis adds randomly sampled views and object-presence checks and reports failures such as thin-object deletion, contact corruption, and depth-edge artifacts.

4.3 CLOSED-LOOP RESULTS

WAP improves success across the evaluated policies and visual conditions without updating policy weights. In the result tables, red downward superscripts indicate deficits from Clean, Original, or ID; green rows mark WAP, with upward superscripts giving gains over raw success. All changes are in percentage points, and darker red indicates larger deficits. **RoboTwin2.0.** Randomized success reaches 59.1% for $\pi_{0.5}$ and 56.6% for X-WAM, exceeding their published raw scores by 13.1 and 30.8 points, respectively (table 1). Their clean reference scores are similar (70.7% versus 70.0%), but X-WAM falls further under randomization (25.8% versus 46.0%). Its larger gain with WAP is therefore consistent with a greater loss from deployment shifts, rather than weaker performance on the familiar scenes. Translation brings the two policies much closer in randomized success; score sources are given in section D.2.

LIBERO-Plus. In paired evaluations, mean success across Light, Background, and Noise rises from 82.6% to 92.9% for π_0 and from 71.4% to 92.9% for Fast-WAM (table 2). Fast-WAM has a higher published unperturbed reference (97.6% versus 94.2%), yet performs worse under Background and Noise, where WAP yields its largest gains (33.3 and 24.9 points). This suggests that strong task competence can coexist with sensitivity to appearance. A plausible explanation lies in the learned visual representations: π_0 inherits semantic features from a pretrained VLM (Black et al., 2024), whereas Fast-WAM and X-WAM use video-pretrained backbones with VAE-encoded observations (Yuan et al., 2026; Guo et al., 2026). Such reconstruction-oriented representations retain texture and illumination details that can disrupt action prediction under visual shifts, as studied in Robust-WAM (Yan et al., 2026). Moving the input toward the training appearance may therefore let these policies use their learned dynamics and control skills more reliably. For Fast-WAM, this explanation concerns its

Table 1: **RoboTwin2.0 Clean2Random evaluation.** Success rates (%) after training on clean demonstrations. Avg. averages Clean and Randomized.

Policy	Clean	Randomized	Avg.
VLA baselines			
X-VLA	68.0	20.9$^{\downarrow 47.1}$	44.5
ABot-M0	70.7	36.0$^{\downarrow 34.7}$	53.4
WAM baselines			
Fast-WAM	77.8	1.9$^{\downarrow 75.9}$	39.9
OpenWAM-α	89.4	48.7$^{\downarrow 40.7}$	69.0
Reference and WAP results			
$\pi_{0.5}$ (raw)	70.7	46.0$^{\downarrow 24.7}$	58.4
$\pi_{0.5}$ + WAP	–	**59.1**$^{\uparrow 13.1}$	**64.9**$^{\uparrow 6.5}$
X-WAM (raw)	70.0	25.8$^{\downarrow 44.2}$	47.9
X-WAM + WAP	–	**56.6**$^{\uparrow 30.8}$	**63.3**$^{\uparrow 15.4}$

Table 2: **LIBERO-Plus evaluation under visual perturbations.** Success rates (%) and their three-factor mean (Avg.). Original denotes published unperturbed performance.

Policy	Original	Light	Background	Noise	Avg.
VLA baselines					
OpenVLA-OFT	97.1	88.7$^{\downarrow 8.4}$	93.3$^{\downarrow 3.8}$	75.8$^{\downarrow 21.3}$	85.9$^{\downarrow 11.2}$
StarVLA	96.6	95.7$^{\downarrow 0.9}$	95.7$^{\downarrow 0.9}$	73.0$^{\downarrow 23.6}$	88.1$^{\downarrow 8.5}$
WAM baselines					
ABot-M0.5	99.4	94.0$^{\downarrow 5.4}$	89.7$^{\downarrow 9.7}$	75.5$^{\downarrow 23.9}$	86.4$^{\downarrow 13.0}$
OpenWAM-α	99.3	97.0$^{\downarrow 2.3}$	87.1$^{\downarrow 12.2}$	39.8$^{\downarrow 59.5}$	74.6$^{\downarrow 24.7}$
Our evaluation					
π_0 (raw)	94.2	87.0$^{\downarrow 7.2}$	80.7$^{\downarrow 13.5}$	80.0$^{\downarrow 14.2}$	82.6$^{\downarrow 11.6}$
π_0 + WAP	–	**94.8**$^{\uparrow 7.8}$	**93.6**$^{\uparrow 12.9}$	**90.3**$^{\uparrow 10.3}$	**92.9**$^{\uparrow 10.3}$
Fast-WAM (raw)	97.6	92.4$^{\downarrow 5.2}$	58.8$^{\downarrow 38.8}$	63.0$^{\downarrow 34.6}$	71.4$^{\downarrow 26.2}$
Fast-WAM + WAP	–	**98.7**$^{\uparrow 6.3}$	**92.1**$^{\uparrow 33.3}$	**87.9**$^{\uparrow 24.9}$	**92.9**$^{\uparrow 21.5}$

Table 3: **Real-robot cloth-folding success rates (%).** Each condition uses five trials; Overall OOD pools the 15 shifted trials.

Method	ID	Cloth color	Tabletop	Illumination	Overall OOD
$\pi_{0.5}$	100.0	40.0$^{\downarrow 60.0}$	0.0$^{\downarrow 100.0}$	20.0$^{\downarrow 80.0}$	20.0$^{\downarrow 80.0}$
$\pi_{0.5}$ + WAP	–	**100.0**$^{\uparrow 60.0}$	**60.0**$^{\uparrow 60.0}$	**60.0**$^{\uparrow 40.0}$	**73.3**$^{\uparrow 53.3}$

video-trained representation, since it omits future-video generation at inference (Yuan et al., 2026). Our clean-scene diagnosis supports this account: 75.9–95.3% of Fast-WAM's persistent failures succeed once perturbations are removed, compared with 26.1–48.6% for π_0 on its own failure subset (section D.5). This is evidence for a visual limitation in these checkpoints, rather than a general ranking of VLA and WAM robustness.

Real robot. Successful OOD trials increase from $3/15$ to $11/15$ (20.0% to 73.3%; table 3). Cloth-color and tabletop changes each show a 60-point gain, compared with 40 points for illumination. Both methods also succeed in all five ID trials; the WAP ID entry is omitted from the table. These results extend the benefits of familiar visual inputs from simulated perturbations to physical cloth folding. Across the three environments, WAP changes what a policy sees while preserving its learned controller, helping separate visual mismatch from a lack of task skill. Fewer mean actions on episodes successful in both runs further suggest more efficient use of those skills (section D.6), although translation can still introduce errors, including regressions under Noise.

4.4 Error analysis and limitations

After examining the failed translations on the three platforms, we identify three common error types, shown in figure 5 with a representative example of each.

- **(a) Contact collapse:** Gripper and object boundaries fuse, thin structures disappear, or contacted objects deform, misrepresenting contact even when the depth condition preserves the geometry.
- **(b) Incomplete conversion:** Despite correct conditions, the generator omits small task objects or leaves regions in their OOD appearance, losing task content or retaining visual mismatch.
- **(c) Depth blur:** Overly smooth depth near object boundaries or the gripper propagates into the output, distorting local geometry.

We manually inspected all failure cases: depth-related errors accounted for 76%, contact collapse for 13%, and incomplete conversion for 11%. These percentages describe the composition of failure cases, not their incidence across all evaluation frames or task stages. Systematic measurements of

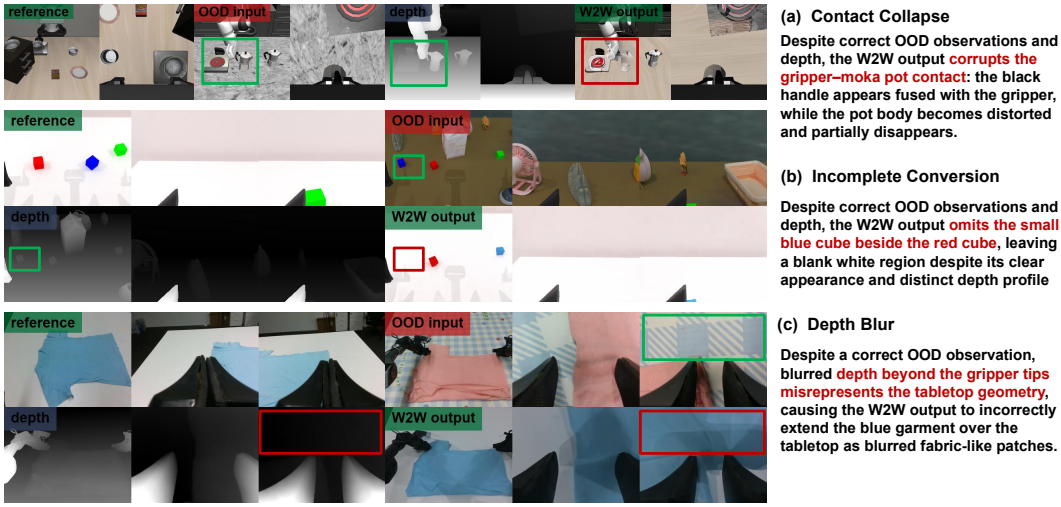

Figure 5: **Representative W2W failure cases.** Each case shows the reference, the OOD input, its estimated depth, and the W2W output: (a) contact collapse on LIBERO-Plus, (b) incomplete conversion on RoboTwin2.0, and (c) depth blur on the real robot.

object presence, object and gripper pose, contact preservation, temporal and cross-view consistency, and action agreement with clean ID observations remain future work.

Under LIBERO-Plus Noise, translation turns 51 successful π_0 episodes and 81 successful Fast-WAM episodes into failures (section D.5), underscoring the need to preserve task state while matching the target appearance.

WAP addresses visual shifts within a fixed task and embodiment, not new tasks, dynamics, or embodiments. Task-relevant layout or color changes must be preserved; LIBERO-Plus Robot, Language, and Layout remain outside the claimed gains. Evaluation covers five checkpoints and one real-robot task. The interface requires a suitable reference and adds latency (section D.6). Neither depth conditioning nor paired supervision rules out hallucinated state; safety-critical use requires independent sensing, collision monitoring, and a fail-safe controller.

Because the objective is token cross-entropy alone, future work will add explicit losses on contact, object presence, and depth boundaries to target the three error types, and will explore faster generation and temporal and cross-view coherence. Beyond appearance, *difficulty canonicalization* could map challenging inputs to simpler familiar representations while preserving obstacles, task identity, and the actual completion state.

5 Conclusion

We introduced World-as-Prompt (WAP), the idea of rewriting the robot's observations of the world to serve the downstream policy. We realize this idea with a World2World translator that renders the policy's familiar visual domain from the depth of the current observation, an in-domain reference, and a structured appearance description, replacing nuisance appearance while retaining the physical state that determines action. Across simulation benchmarks and a real robot, WAP improves every evaluated VLA and WAM without new action-labeled data or policy fine-tuning, restoring much of the competence these policies lose under visual shift. When deployment changes only how a familiar task is seen, adapting the visual interface can thus be a reusable alternative to adapting every policy. The remaining failures at contacts, on small objects, and under blurred depth, together with the current translation latency, make faster and more faithful World2World translators the main next step. Generative translation should not be applied to task-relevant state changes, and its hallucination risk must be measured and guarded before safety-critical use.

AI USE STATEMENT

We used Qwen3.8-Flash for structured appearance annotation and annotation review, with manual spot checks on a subset of retained annotations (sections B and 3.2). For manuscript preparation, generative AI assistance was limited to language polishing. The authors take responsibility for the final content.

ETHICS STATEMENT

This study uses simulated observations from RoboTwin and LIBERO and recordings collected with the authors' robot, without human-subject experiments or personal data. Third-party assets remain subject to their original licenses.

REPRODUCIBILITY STATEMENT

Section 3 describes the translator's formulation, training objective, and frozen-policy interface. Section A reports dataset sources and screening statistics, while sections B and C provide annotation, review, and SFT prompts. Sections 4.1, D.1 and D.4 describe evaluation protocols and metrics. Section D.2 documents baseline sources, section D.3 provides the real-robot policy's training configuration, and sections D.5 and D.6 report paired outcomes, failure diagnostics, and execution costs.

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.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

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.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Jake Bruce, Michael D. 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 *International Conference on Machine Learning*, 2024.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. In *Robotics: Science and Systems*, 2024a.

Lawrence Yunliang Chen et al. RoVi-Aug: Robot and viewpoint augmentation for cross-embodiment robot learning. In *Conference on Robot Learning*, 2024b.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, et al. ABot-M0.5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, 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.

Jeremiah Coholich, Justin Wit, Robert Azarcon, and Zsolt Kira. Sim2real image translation enables viewpoint-robust policies from fixed-camera datasets. *arXiv preprint arXiv:2601.09605*, 2026.

Dexmal Team. DM0.5: An open-world foundation model for general-purpose embodied intelligence. https://www.dexmal.com/blog/dm0.5/index_en.html, 2026.

DM0 Team. DM0: An embodied-native vision-language-action model towards physical AI. *arXiv preprint arXiv:2602.14974*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot manipulation policies. *IEEE Robotics and Automation Letters*, 2026.

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.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning*, 2024.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

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, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Marion Lepert, Ria Doshi, and Jeannette Bohg. SHADOW: Leveraging segmentation masks for cross-embodiment policy transfer. In *Conference on Robot Learning*, pp. 3536–3550, 2024.

Xinghang Li, Jun Guo, Qiwei Li, Long Qian, Hang Lai, Yueze Wang, Hongyu Yan, Jiahang Cao, Xi Chen, Jingen Qu, Jiaxi Song, Nan Sun, Hanye Zhao, Futeng Liu, Wanli Peng, Heyun Wang, Yunhong Wang, Caoyu Xia, Jack Zhao, Diyun Xiang, Hangjun Ye, Heng Qu, Huaping Liu, and Jason Li. Xiaomi-Robotics-U0: Unified embodied synthesis with world foundation model, 2026. URL https://arxiv.org/abs/2607.11643.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, pp. 9493–9500, 2023.

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

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

Reihaneh Mirjalili, Tobias Jülg, Florian Walter, and Wolfram Burgard. Augmented reality for RObots (ARRO): Pointing visuomotor policies towards visual robustness. *IEEE Robotics and Automation Letters*, 11(4):4785–4792, 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.

Seedance Team. Seedance 2.0: Advancing video generation for world complexity. *arXiv preprint arXiv:2604.14148*, 2026.

Younggyo Seo, Junsu Kim, Stephen James, Kimin Lee, Jinwoo Shin, and Pieter Abbeel. Multi-view masked world models for visual robotic manipulation. In *International Conference on Machine Learning*, pp. 30613–30632, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation*, 2023.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Jiayi Su, Yixin Zheng, Mi Yan, Li Yi, Zhizheng Zhang, and He Wang. GPT 6 Astra as an embodied policy. Technical report and code, 2026. URL `https://github.com/anonymous-report-421/eval-of-gpt-6-astra-as-policy`.

Stephen Tian, Blake Wulfe, Kyle Sargent, Katherine Liu, Sergey Zakharov, Vitor Campagnolo Guizilini, and Jiajun Wu. View-invariant policy learning via zero-shot novel view synthesis. In *Proceedings of the 8th Conference on Robot Learning*, volume 270, pp. 1173–1193, 2025.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 23–30, 2017.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, Yu-Kai Wang, Yuhan Rui, Shenzhe Yao, Zhen Yuan, Zhenhao Shen, Kefei Zhu, Zijie Zhu, Ning Gao, Xiaowei Chi, Guanqi He, Shanghang Zhang, Hao Dong, Lin Shao, and Hang Zhao. OpenWAM: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. VLATest: Testing and evaluating vision-language-action models for robotic manipulation. *Proceedings of the ACM on Software Engineering*, 2(FSE):1615–1638, 2025.

Albert Wilcox, Mohamed Ghanem, Masoud Moghani, Pierre Barroso, Benjamin Joffe, and Animesh Garg. Adapt3R: Adaptive 3d scene representation for domain transfer in imitation learning. *arXiv preprint arXiv:2503.04877*, 2025.

Hongtao Wu, Ya Jing, Chi-Lam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pp. 3153–3160, 2024.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, Mingming Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, Yingjie Cai, Xu Yan, Guanyi Zhao, Bingbing Liu, and Haoang Li. Robust-WAM: Bridging generative pretraining and semantic foresight in world-action models. *arXiv preprint arXiv:2608.05903*, 2026. URL https://arxiv.org/abs/2608.05903.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. ABot-M0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

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*, 2026.

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.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

Wenbo Zhang, Kaixuan Wang, Yutao Ouyang, Xiaoyu Huang, Liyang Li, Kailun Su, Weiyang Jin, Wenhao Chai, Haotian Liang, Zhiyang Dou, Yue Chen, and Tianxing Chen. An unexpected robot policy: Early evaluations of GPT-6 Astra on RoboDojo and beyond. *arXiv preprint arXiv:2609.24170*, 2026a.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Zhengru Fang, Yi Nie, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

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*, 2026.

Junkang Zhou, Yefei He, Feng Chen, Weijie Wang, and Bohan Zhuang. FlashAR: Efficient post-training acceleration for autoregressive image generation, 2026. URL https://arxiv.org/abs/2605.09430.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

A Dataset sources and composition

We collect reference-aware embodied Transfer data from RoboTwin and LIBERO. The datasets span bimanual and single-arm manipulation, diverse task objects, and varied scene appearances (Figure 6).

RoboTwin covers 70 tasks and 20 OOD tasks, with variation in object inventories, workspace appearance, and manipulation state. Three synchronized viewpoints capture the bimanual robot and its surroundings. **LIBERO** covers 40 tasks and three OOD appearance categories: lighting conditions, sensor noise, and background textures. These perturbations come from LIBERO-Plus, the robustness extension of LIBERO. External and wrist views capture the single-arm robot and task scene.

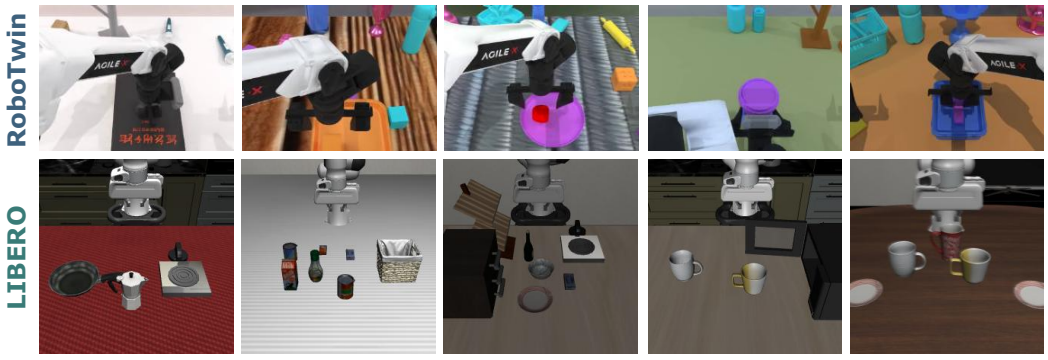

Figure 6: **Dataset sources and diversity.** Ten representative scenes illustrate the object, workspace, and appearance variation in RoboTwin (top) and LIBERO (bottom). LIBERO examples cover all three OOD categories.

Dataset size after screening. Table 4 reports the consolidated selection. Quality screening retains 32,550 RoboTwin samples and 497,018 LIBERO samples. Reserving samples close to the test distribution leaves 27,510 and 485,267 samples for SFT, respectively.

Table 4: **Dataset size before and after screening.**

Dataset	Initial samples	Quality exclusions	Test-distribution holdout	SFT training
RoboTwin	45,226	12,676	5,040	**27,510**
LIBERO	572,850	75,832	11,751	**485,267**

B Structured appearance annotation

We use Qwen3.8-Flash in a two-request annotation pipeline. The task instruction and task-object information establish relevance and physical-instance count; the complete target video, sampled at 1 FPS including its beginning and terminal state, supplies visual evidence. Generation and independent review share these inputs, while review additionally receives the generated JSON.

1 Generate →	**2 Review →**	**3 Assemble**
Video + task information	Same inputs + annotation	Retained appearance fields
Static appearance JSON	Five checks, issues, confidence	Scene text + reference + depth

Review is bound to the SHA-256 hash of the generated annotation. Only appearance fields enter the scene text; review, warnings, and thinking remain metadata.

B.1 Qwen generation prompt

Describe the persistent visual appearance of one complete TARGET video and return an English JSON annotation.

The user message supplies:

- the authoritative task instruction;
- authoritative task-object entity information;
- the complete TARGET video sampled at 1 FPS, including the beginning and the true terminal state.

Inspect the complete video before deciding object identity, physical-instance count, or appearance. Produce one static annotation that represents the entire video.

Use the task instruction and task-object information to determine which non-robot physical entities are required by the task and how many instances are involved. Use the video as the authority for visible appearance.

Place every manipulated object, destination object, receptacle, and task fixture in task_objects.objects. Preserve the correct physical-instance count and the supplied entity order. Use concise natural-English entity names. Distinguish repeated instances with stable natural descriptions or neutral instance numbers.

Place persistent visible non-robot objects that are unrelated to the task in irrelevant_objects. Place attached components within the appearance description of their parent object. Place robot components only in task_objects.robot. Describe each physical object once.

Write persistent visual attributes: identity, color, material, texture, pattern, geometry, surface finish, translucency, illumination, and other stable visible properties. Omit positions, directions, spatial relationships, poses, actions, task progress, contact states, grasp states, containment states, temporal events, and frame-specific observations.

Apply the static-appearance rule to every descriptive field, including scene_description, workspace, task_objects, irrelevant_objects, lighting, and background. Write an appearance inventory rather than a scene layout or action summary. Remove clauses such as ”on the table”, ”near the robot”, ”behind the arm”, ”visible in the background”, ”resting”, ”positioned”, ”moves”, ”holds”, ”grasps”, or ”interacts with”.

For example, write: ”Dark grey cabinetry with smooth panels and silver handles.”

Do not write: ”Dark grey cabinetry visible behind the robot.” ”The robot moves toward the cup.” ”The gripper holds the bottle.”

Use natural descriptive language throughout the annotation. Omit dataset names, variant labels, task names, copied instructions, internal entity identifiers, episode or trajectory identifiers, sample or asset identifiers, paths, seeds, hashes, timestamps, frame numbers, sampling metadata, and camera metadata.

Fields:

- **scene_description**: a concise static summary of the scene type and persistent visual inventory.
- **workspace**: the manipulation surface’s persistent color, material, texture, pattern, and finish.
- **task_objects.robot**: the robot’s persistent visible colors, materials, surface finishes, structural appearance, wrist, gripper, and end-effector appearance.
- **task_objects.objects**: task-required non-robot objects and fixtures, each with a natural-English entity name and persistent appearance.
- **irrelevant_objects**: other persistent visible non-robot objects, each with a natural-English entity name and persistent appearance. Use [] when none are present.

- **lighting**: persistent brightness, softness, color cast, highlights, and shadow characteristics.
- **background**: persistent colors, materials, textures, patterns, and finishes of the surrounding environment.
- **annotation_warnings**: concise uncertainty notes about identity, count, color, material, or inconsistent visual evidence. Use [] when there is no material uncertainty.

Use conservative descriptions supported by the complete video. Record remaining uncertainty in annotation_warnings. Return only JSON with exactly this structure:

```
{
  "scene_description": "",
  "workspace": "",
  "task_objects": {
    "robot": "",
    "objects": [{"entity": "", "appearance": ""}]
  },
  "irrelevant_objects": [{"entity": "", "appearance": ""}],
  "lighting": "",
  "background": "",
  "annotation_warnings": []
}
```

B.2 QWEN INDEPENDENT-REVIEW PROMPT

Independently review a static appearance annotation against one complete TARGET video.

The user message supplies:

- the authoritative task instruction;
- authoritative task-object entity information;
- the complete TARGET video sampled at 1 FPS, including the beginning and the true terminal state;
- the generated annotation.

Inspect the complete video and review the annotation as supplied. Return an assessment without rewriting the annotation.

Check five aspects:

1. task_object_alignment Verify that task_objects.objects contains every manipulated object, destination object, receptacle, and task fixture required by the task, with the correct identity and physical-instance count. Verify that unrelated visible objects are assigned to irrelevant_objects and that attached components are represented within their parent object.

2. appearance_accuracy Check scene, workspace, robot, task-object, irrelevant-object, lighting, and background descriptions against the complete video. Check identity, count, color, material, texture, pattern, geometry, surface finish, translucency, illumination, and other claimed persistent properties. Identify omissions, duplicates, contradictions, and unsupported detail.

3. static_appearance Verify that descriptive fields contain persistent appearance rather than positions, directions, spatial relationships, poses, actions, task progress, contact states, grasp states, containment states, temporal events, or frame-specific observations.

4. scene_fields Verify that every required field is substantive and internally consistent. Check that entity names use natural English and that annotation_warnings contains only genuine uncertainty. Flag dataset names, variant labels, task names, copied instructions, internal identifiers, episode or trajectory identifiers, sample or asset identifiers, paths, seeds, hashes, timestamps, frame numbers, sampling metadata, and camera metadata.

5. video_coverage Verify that the annotation represents the complete video rather than one isolated moment. Use evidence across the sequence to assess objects that are partially visible or occluded in individual frames.

For every concrete problem, add one issue containing a field path and a concise problem description. Use an empty issues array when there are no concrete problems.

confidence is the confidence that the annotation can be reused unchanged for the complete video:

- 0.90-1.00: identities, counts, appearance, scene fields, and static wording are accurate and complete.
- 0.60-0.89: the core annotation is correct with minor uncertainty or isolated imprecision.
- 0.30-0.59: an important identity, count, scene field, or appearance detail is missing, ambiguous, or inaccurate.
- 0.00-0.29: a required object is absent or wrong, major objects are misclassified, or substantial unsupported, dynamic, or spatial content is present.

Use the most serious finding to determine confidence. Return only JSON with exactly this structure:

```
{
  "analysis": {
    "task_object_alignment": "",
    "appearance_accuracy": "",
    "static_appearance": "",
    "scene_fields": "",
    "video_coverage": ""
  },
  "issues": [{"field": "", "problem": ""}],
  "confidence": 0.0
}
```

Scope of the review. A valid JSON structure is not sufficient evidence of a correct annotation. Object identity and appearance must remain supported across the sequence, including frames in which an object is partly occluded. Confidence refers to reuse of the supplied annotation, not to a single frame or an individual attribute. Because the review is hash-bound, its conclusions do not automatically extend to an edited version of the JSON.

Automatic screening threshold. The review rubric checks task-object alignment, appearance accuracy, static appearance, scene-field consistency, and video coverage, and instructs the reviewer to determine the score from the most serious identified issue. Even largely correct annotations with minor uncertainty or isolated imprecision belong to the 0.60–0.89 band under these rules. We therefore set the threshold to 0.9, the lower bound of the highest-quality band, to select annotations that meet the rubric's criteria for reuse without modification.

Human inspection. Following automatic screening, five members of the research team each independently inspected 1,000 sampled video–annotation pairs from the retained data, comparing the annotations with the corresponding videos and task information. The inspection focused on object identities and counts, visually supported appearance descriptions, and the exclusion of transient information such as positions, actions, and contact states.

Structured output. Generation yields the appearance inventories used to assemble SFT scene text. Scene and workspace descriptions establish the visual setting; task objects and unrelated objects separate entity roles; lighting and background describe the environment. Equal row heights make corresponding fields comparable without shortening either inventory. Blank space reflects different amounts of content, not an absent field or an additional object.

C From annotation to SFT

The SFT sequence is scene text → appearance reference → current depth → target RGB. The source RGB is not a condition; it enters only through the depth that Depth Anything 3 (`DA3-LARGE-1.1`) (Lin et al., 2025) estimates from it, stored as inverse depth so that brighter pixels are nearer. The same estimator produces the depth of the source view in training and of the raw OOD view at deployment. The scene text is tokenized by the text tokenizer; the reference and the depth are encoded by the frozen VQ encoder, each introduced by its label (`TARGET APPEARANCE REFERENCE:` or `CURRENT DEPTH:`) and a visual placeholder, and the generated target tokens are enclosed by the output markers of the runtime prompt below. Condition tokens use standard causal attention, and target tokens follow the step mask of equation (4). Multi-view observations are tiled in a fixed view order into one canvas: two 256×256 views (512×256) on LIBERO and three 320×240 views (960×240) on RoboTwin and the real robot. Figure 7 shows the scene-text fields; target RGB is supervision.

C.1 Depth estimation and normalization

For RoboTwin, LIBERO, and the real robot, we use the same frozen Depth Anything 3 model (`DA3-LARGE-1.1`) (Lin et al., 2025) to estimate depth from the source RGB in training and from the raw OOD RGB at deployment. Each camera view v is processed separately at each time step, before the resulting depth images are tiled into the multi-view canvas. Given the estimated depth $d_v(x, y)$, we first form inverse depth $u_v(x, y)$, then normalize it using the 2nd and 98th percentiles of that view:

$$
\begin{aligned}
u_v(x,y) &= \frac{1}{\max(d_v(x,y),\, 10^{-8})}, \\
a_v &= Q_{0.02}(u_v), \qquad b_v = Q_{0.98}(u_v), \\
I_v(x,y) &= \begin{cases} \left\lfloor 255\ \mathrm{clip}\left(\dfrac{u_v(x,y) - a_v}{b_v - a_v},\, 0,\, 1\right)\right\rfloor, & b_v > a_v, \\ 0, & b_v \le a_v. \end{cases}
\end{aligned}
$$

Here Q_p denotes the p-quantile over pixels in the current view. The result is an 8-bit grayscale depth condition I_v in which nearer surfaces are brighter. If the percentile range collapses, we set the image to zero. The same preprocessing is applied to training and deployment observations.

C.2 Translator configuration

Table 5 lists the translator configuration. The visual tokenizer compresses each spatial dimension by 16, so the two canvases give 32×16 and 60×15 target-token grids, decoded in $N = 47$ and $N = 74$ anti-diagonal steps. The fusion gate is one scalar per target position, predicted by an MLP with a sigmoid output from the horizontal hidden state of the left neighbor and the vertical hidden state of the upper neighbor; the first row and column use the horizontal and vertical heads alone. The gate, both heads, and the backbone receive gradients only through the fused cross-entropy of equation (5).

C.3 SFT runtime prompt

> You are an advanced controllable image generation model. You will receive a scene description, a TARGET APPEARANCE REFERENCE grid, and a CURRENT DEPTH grid. Generate a corresponding photorealistic RGB grid with the same canvas and grid layout as the depth input. Maintain consistent identity, appearance, and geometry across the complete output grid.

Table 5: **Translator configuration.**

Item	Value
Initialization	Xiaomi-Robotics-U0-FlashAR, 38B (Li et al., 2026)
Visual tokenizer	IBQ, 16×16 compression, 131,072-entry codebook
Canvas / token grid, LIBERO	512×256 / 32×16
Canvas / token grid, RoboTwin and real robot	960×240 / 60×15
Depth estimator	Depth Anything 3 (`DA3-LARGE-1.1`), inverse depth
Objective	Fused cross-entropy on target tokens (equation (5))
Fusion gate	$\sigma(\mathrm{MLP})$ of neighboring hidden states, per position
Inference backend	vLLM with anti-diagonal scheduling

The condition order is: 1. scene text;

2. TARGET APPEARANCE REFERENCE;

3. CURRENT DEPTH;

4. generated RGB output.

The TARGET APPEARANCE REFERENCE is an RGB appearance reference, not a depth map. Its primary role is to provide evidence of color, material, texture, surface finish, gloss, and reflectance. It may come from another time and may show different object poses, robot configurations, occlusions, action phases, or object inventory. These differences do not define the output geometry or object inventory.

The CURRENT DEPTH grid is the primary authority for the output viewpoint and scene geometry, including shape, size, object count, pose, position, orientation, visibility, spatial relationships, and contact state. Brighter depth pixels represent nearer surfaces and darker pixels represent farther surfaces.

When the Irrelevant Objects field contains entries, render those objects using their described appearance and the current depth geometry. When this field is empty, remove objects irrelevant to the described task or target scene, including those present in the depth input, and replace their regions with a natural continuation of the surrounding background without preserving their shapes or spatial footprints.

Transfer color, material, texture, pattern, surface finish, and reflectance from the reference only between semantically corresponding objects or surfaces. Spatial overlap alone does not establish correspondence. Ignore appearance evidence from reference-only or unrelated content.

For a current object or surface without reliable reference correspondence, use its scene-text description. If the text does not specify its appearance, generate a plausible appearance consistent with the current workspace, lighting, and background.

The Viewpoint field describes the output camera viewpoint. When Viewpoint is empty, use the default camera setting.

```
Image Style: Simulator.
Viewpoint: {viewpoint}
Scene Description: {scene_description}
Workspace: {workspace}
Task objects: robot: {robot}; {task_objects}
Irrelevant objects: {irrelevant_objects}
Lighting: {lighting}
Background: {background}

TARGET APPEARANCE REFERENCE:
<|VIS_PLH|>

CURRENT DEPTH:
<|VIS_PLH|>

<|extra_100|><|VIS_PLH|><|extra_101|>
```

(a) RoboTwin



scene_description

A simulated tabletop scene with a matte grey surface, a robotic gripper, an orange tray, colorful blocks, a pink toy car, and a metallic pink goblet.

workspace

A smooth, matte grey tabletop surface.

task_objects.robot

A black robotic arm with a visible gripper mechanism.

task_objects.objects

Orange tray: A rectangular tray with a matte orange finish and slightly raised edges.

Purple block: A small, rectangular block with a bright purple color and a textured, grid-like surface pattern.

Blue block: A small, solid blue cube with a smooth, matte finish.

Green block: A small, solid green rectangular block with a smooth, matte finish.

irrelevant_objects

Pink car: A small, pink toy car with a glossy finish.

Purple rectangular object: A flat, purple rectangular object, possibly a sponge or mat.

Blue rectangular object: A tall, blue rectangular block standing upright.

Pink goblet: A shiny, metallic pink goblet or cup.

Teal block: A small, teal-colored block with a bumpy texture.

lighting

Even, diffuse lighting that casts soft shadows, typical of an indoor or studio setting.

background

A plain, light-colored floor or wall visible at the top edge of the frame.

annotation_warnings

[]

(b) LIBERO



scene_description

A simulated indoor environment featuring a wooden desk surface with a robotic arm, a multi-compartment organizer, a book, a mug, and a potted plant.

workspace

A light brown wooden surface with a smooth finish and subtle horizontal grain patterns.

task_objects.robot

A white and grey articulated robotic arm with black joints and a parallel-jaw gripper.

task_objects.objects

black book (instance 1): A hardcover book with a solid black cover and spine, featuring white page edges.

desk caddy (instance 1): A dark brown rectangular organizer with three open compartments and a smooth, matte finish.

irrelevant_objects

mug: A cylindrical cup with a metallic silver body, a gold-colored handle, and a gold-colored interior rim.

potted plant: A green plant with long, slender leaves growing from a textured brown trunk, housed in a simple grey cylindrical pot.

lighting

Bright, evenly diffused neutral-warm illumination with soft shadows.

background

A plain light grey wall and a section of grey shelving or cabinetry on the left side.

annotation_warnings

[]

Figure 7: **Paired appearance inventories.** All displayed field values are taken directly from the annotations.

D Evaluation Protocols and Supplementary Results

D.1 Evaluation protocols and metrics

Matched evaluations. Every comparison holds the policy checkpoint, language instruction, proprioception interface, action postprocessing, initial reset, and environment seed fixed and changes only the RGB observation passed to the policy: (i) the raw OOD observation or (ii) its WAP translation. The translator is fixed before evaluation, and its training uses no evaluation actions or success labels. On RoboTwin2.0, following the co-training protocol, one policy is trained from 50 clean demonstrations for each of 50 tasks (2,500 in total) and evaluated for 100 trials per task in clean (C2C) and randomized (C2R) environments, with the three randomized camera streams translated at native resolution. LIBERO-Plus uses its official task configurations and success predicates. The real-robot trials keep the instruction, task geometry, initial-state distribution, and terminal-fold success criterion and run five paired trials per condition. On LIBERO-Plus and the real robot, each comparison reruns the same episodes with and without WAP, replaying identical reset states wherever the simulator supports it; closed-loop states may still diverge after the first different action.

Three caveats qualify these comparisons. The full RoboTwin randomized split also contains task-relevant clutter and physical changes, so overall C2R is a mixed stress test; The full RoboTwin randomized split contains appearance variation, task-relevant clutter, and physical changes; overall C2R therefore measures robustness to this combined shift. The translator corpus for LIBERO-Plus was collected from its official configurations and a disjointness audit is not yet available, so we label this experiment *transductive target-domain visual adaptation* and report a zero-shot-policy subset only for tasks, assets, variants, and seeds proven absent from translator training. On the real robot, the cloth-color and tabletop conditions include entity replacement; the same replacement is used for both methods, so the paired comparison exposes the same visual–physical shift.

Metrics and diagnostics. The primary metric is task success rate; long-horizon real-robot trials also report progress through preregistered stages. Let S_{ID} be policy success on the familiar visual domain, S_{OOD} its success on an OOD domain, and S_{F} and $S_{\mathrm{F,id}}$ its success after translation on the OOD and familiar domains, respectively. We report OOD success change $S_{\mathrm{F}} - S_{\mathrm{OOD}}$ and, where both ID evaluations are available, ID success change $S_{\mathrm{F,id}} - S_{\mathrm{ID}}$, in percentage points. Positive and negative changes indicate improved and reduced success, respectively. Confidence intervals are computed with a paired, stratified bootstrap over tasks and matched environment seeds. For the RoboTwin replay cohorts, we report success counts and success rates over valid completed rollouts.

D.2 Sources for the main benchmark tables

The RoboTwin baselines and LIBERO-Plus perturbation scores in tables 1 and 2 follow the OpenWAM technical report (Wang et al., 2026). The RoboTwin comparison retains X-VLA (Zheng et al., 2026), ABot-M0 (Yang et al., 2026), Fast-WAM (Yuan et al., 2026), and OpenWAM-α (Wang et al., 2026). The LIBERO-Plus comparison retains OpenVLA-OFT (Kim et al., 2025), StarVLA (StarVLA Community, 2026), ABot-M0.5 (Chen et al., 2026), and OpenWAM-α. Our WAP evaluations use π_0 (Black et al., 2024) and Fast-WAM on LIBERO-Plus; the RoboTwin reference rows use $\pi_{0.5}$ (Physical Intelligence et al., 2025) and X-WAM (Guo et al., 2026). For RoboTwin, we use the clean-only co-trained $\pi_{0.5}$ and X-WAM checkpoints trained, evaluated, and publicly released by the RoboTwin team. Training of the real-robot $\pi_{0.5}$ checkpoint is given in section D.3. The two benchmark tables retain their own training and evaluation protocols; a shared policy name does not imply an identical checkpoint across datasets.

D.3 Real-robot $\pi_{0.5}$ checkpoint

The real-robot policy is a $\pi_{0.5}$ checkpoint post-trained on 511 cloth-folding episodes collected in one setup. Each step provides three RGB views, resized to 224×224, and a 14-dimensional joint-angle state. The policy predicts 14-dimensional joint-angle actions over a horizon of 50, with delta-joint action targets.

Training runs for 80,000 steps on eight GPUs, with a per-GPU batch size of 16 and a global batch size of 128; the evaluated checkpoint is saved at step 80,000. The run takes about 36 hours with

fused AdamW (weight decay 1×10^{-10}), gradient clipping at 1.0, and gradient checkpointing. The learning rate warms up for 1,000 steps to 5×10^{-5}, then decays over 50,000 steps to 5×10^{-6}.

D.4 LIBERO-PLUS PROTOCOL

LIBERO-Plus treats the unperturbed task as the reference and varies one factor at progressive difficulty levels (Fei et al., 2026). The factors have different causal status:

- **Camera**: position, orientation, and field of view;
- **Robot**: initial manipulator state;
- **Language**: semantically equivalent instruction rewrites;
- **Light**: intensity, direction, color, and shadows;
- **Background**: scene and surface textures;
- **Noise**: photometric corruption and blur; and
- **Layout**: distractors and target-object displacement.

Table 2 reports Light, Background, and Noise separately; "Avg." is their unweighted mean, not the benchmark's seven-factor overall score. The published context rows follow the OpenWAM report, whereas the local comparison uses $1{,}142$ Light, $1{,}076$ Background, and $1{,}601$ Noise episodes per policy and condition. Each raw outcome is paired with its WAP outcome under the same task configuration, initial state, and seed. The total number of WAP successes equals retained raw successes plus rescued raw failures. The clean-scene diagnostic in table 6 evaluates only episodes that fail in both runs; episodes that succeed in the raw run but fail with WAP are excluded from this diagnostic. It therefore does not estimate clean success over all WAP failures or over the full test set.

Unperturbed LIBERO references. The Original column in table 2 uses 97.1% for OpenVLA-OFT (Kim et al., 2025) and 99.4% for ABot-M0.5 (Chen et al., 2026). The StarVLA reference of 96.6% and OpenWAM-α result of 99.3% follow the OpenWAM report (Wang et al., 2026); the StarVLA value corresponds to the official StarVLA-OFT benchmark snapshot. For π_0, we use the Original score of 94.2% reported by LIBERO-Plus (Fei et al., 2026); for Fast-WAM, we use its four-suite LIBERO average of 97.6% (Yuan et al., 2026). These literature aggregates provide context: checkpoints, episode sets, and evaluation protocols are not verified to match the perturbation runs. Accordingly, red downward superscripts report descriptive gaps to Original; darker red cells indicate larger gaps. Green upward superscripts instead report paired WAP gains over the corresponding local raw run. Both differences are computed from displayed success rates in percentage points. Original is excluded from Avg.; the WAP rows leave it unreported because no full-set unperturbed WAP evaluation is available.

D.5 LIBERO-PLUS PAIRED OUTCOMES AND FAILURE DIAGNOSIS

Table 6: **Paired LIBERO-Plus outcomes and failure diagnosis.** Outcomes are counts / eligible episodes (percentages); clean-scene success is evaluated on failures shared by both runs.

Perturbation	Raw success	Failure rescue	Success retention	Clean-scene success
		π_0 **(VLA)**		
Light	994/1142 (87.0%)	89/148 (60.1%)	994/994 (100.0%)	21/59 (35.6%)
Background	868/1076 (80.7%)	139/208 (66.8%)	868/868 (100.0%)	18/69 (26.1%)
Noise	1281/1601 (80.0%)	215/320 (67.2%)	1230/1281 (96.0%)	51/105 (48.6%)
		Fast-WAM (WAM)		
Light	1055/1142 (92.4%)	72/87 (82.8%)	1055/1055 (100.0%)	12/15 (80.0%)
Background	633/1076 (58.8%)	358/443 (80.8%)	633/633 (100.0%)	81/85 (95.3%)
Noise	1009/1601 (63.0%)	480/592 (81.1%)	928/1009 (92.0%)	85/112 (75.9%)

Success, rescue, and retention. Table 2 shows that WAP raises mean success from 82.6% to 92.9% for π_0 and from 71.4% to 92.9% for Fast-WAM. To separate recovered failures from lost successes, table 6 reports *failure rescue*: episodes that fail under the raw policy but succeed with WAP, divided by all raw-policy failures; and *success retention*: episodes that succeed in both runs, divided by all raw-policy successes. These rates condition on outcomes in the raw OOD run; success retention therefore differs from the ID success-rate change defined in section D.1. WAP rescues 60.1–67.2% of π_0 failures and 80.8–82.8% of Fast-WAM failures across the three factors. Both policies retain every raw success under Light and Background. Under Noise, retention falls to 96.0% and 92.0%, respectively: 51 previously successful π_0 episodes and 81 Fast-WAM episodes fail with translation. The rescued episodes still outweigh these regressions, but the losses show that aggregate gains do not imply uniformly beneficial translation.

Diagnosing persistent failures. We measure *clean-scene success on persistent failures* to diagnose errors that remain after translation. For episodes that fail in both the raw and WAP runs, we rerun the policy after removing lighting, background, or noise perturbations in the simulator, supplying clean observations directly. Each entry reports clean successes divided by the number of these persistent failures. In particular, the Noise denominators are 105 for π_0 and 112 for Fast-WAM; they exclude the 51 and 81 regressions above. A clean success suggests that WAP has not fully removed the visual mismatch or has introduced an artifact. Continued failure under clean observations is more consistent with a limitation of the policy on that task. This diagnostic provides evidence about the source of failure, rather than a definitive causal attribution.

For π_0, only 26.1–48.6% of persistent failures succeed in clean scenes, suggesting that further visual correction alone may not resolve most of this subset. For Fast-WAM, clean success reaches 75.9–95.3%, leaving more room to recover existing competence through better observation translation. This difference is relevant to the two policy families: π_0 couples a vision–language backbone to action generation, whereas Fast-WAM learns control with video-based world modeling (Black et al., 2024; Yuan et al., 2026). Both still depend on visual inputs, and both benefit from the same input-level adaptation. The higher failure-rescue and clean-scene success rates for Fast-WAM on its respective failure subsets suggest a greater contribution from visual mismatch in these evaluated episodes; they do not establish an inherent advantage or limitation of either model family.

D.6 Latency and execution efficiency across environments

Tables 7 and 8 report execution costs on episodes that succeed both with and without WAP. This same subset is used for control steps, policy calls, and all timing measurements, including LIBERO-Plus Noise. Triplets give mean raw / WAP / change (WAP minus raw), rounded to one decimal place; changes are computed before rounding. LIBERO-Plus macro averages weight the three factors equally.

Control steps count executed actions. A policy call supplies an action chunk: 50 actions on RoboTwin and 24 executed control steps on the real robot. Episode time includes translation, whose accumulated cost is reported for WAP only and is zero for the raw policy. RoboTwin episode times are estimates; real-robot timings cover the effective execution phase.

Table 7: **LIBERO-Plus execution costs.** Triplets give mean raw / WAP / change; translation is reported for WAP only.

Condition	Control steps	Policy calls	Episode time (s)	Translation time (s)
		π_0 **(VLA)**		
Light	205.8 / 167.2 / -38.6	41.4 / 33.9 / -7.6	46.3 / 307.2 / $+260.9$	253.5
Background	193.3 / 167.8 / -25.5	39.0 / 34.0 / -5.0	45.1 / 337.5 / $+292.4$	269.0
Noise	156.0 / 157.4 / $+1.4$	31.6 / 31.9 / $+0.3$	127.7 / 347.7 / $+220.0$	226.0
Macro Avg.	185.0 / 164.2 / -20.9	37.4 / 33.3 / -4.1	73.0 / 330.8 / $+257.8$	249.5
		Fast-WAM (WAM)		
Light	145.5 / 148.4 / $+2.9$	15.0 / 15.1 / $+0.1$	34.0 / 150.8 / $+116.8$	115.5
Background	146.2 / 134.0 / -12.2	15.0 / 13.9 / -1.1	34.6 / 135.9 / $+101.3$	101.6
Noise	160.5 / 148.8 / -11.7	16.5 / 15.3 / -1.1	133.3 / 234.0 / $+100.7$	98.3
Macro Avg.	150.7 / 143.7 / -7.0	15.5 / 14.8 / -0.7	67.3 / 173.6 / $+106.3$	105.1

Table 8: **RoboTwin and real-robot execution costs.** Triplets give mean raw / WAP / change; RoboTwin episode times are estimated.

Statistic	Control steps	Policy calls	Episode time (s)	Translation time (s)
		RoboTwin2.0: matched task cases		
Mean	536.5 / 425.4 / -111.0	10.8 / 8.7 / -2.1	281.0 / 305.1 / $+24.1$	78.4
		Real robot: $\pi_{0.5}$ **cloth folding**		
Mean	5881.8 / 3834.0 / -2047.8	245.1 / 159.8 / -85.3	284.2 / 1575.6 / $+1291.4$	1201.3

Decisions in a familiar visual domain. Bringing observations closer to the training appearance may help a frozen VLA or WAM use its learned manipulation skills more effectively. On LIBERO-Plus, mean steps decrease from 185.0 to 164.2 for π_0 and from 150.7 to 143.7 for Fast-WAM. The corresponding decreases are 536.5 to 425.4 on RoboTwin and 5881.8 to 3834.0 on the real robot, with fewer policy calls in each setting. Since these comparisons use episodes successful in both runs, they show that the policies often complete the same tasks with fewer actions after translation. Together with the success gains in section 4.3, this supports the interpretation that restoring a familiar visual input can improve both task completion and decision efficiency. The effect varies by perturbation: π_0 takes slightly more steps under Noise, and Fast-WAM under Light.

Translation cost and completion time. The shorter rollouts partly offset the added inference cost. On RoboTwin, translation costs 78.4 s per episode, while estimated total time increases by only 24.1 s (281.0 to 305.1 s). Excluding translation, this is 226.7 s, or 54.3 s less than the raw rollout. On LIBERO-Plus and the real robot, translation remains the larger cost: total time increases from 73.0 to 330.8 s for π_0, from 67.3 to 173.6 s for Fast-WAM, and from 284.2 to 1575.6 s on the robot. Dividing mean translation time by mean policy calls gives approximately 7.5/7.1 s per query for the two LIBERO policies, 9.0 s for RoboTwin, and 7.5 s for the real robot; these costs are incurred per action chunk, rather than per executed action. Thus, WAP already reduces mean execution length, although total time still increases in the current implementation. Faster translation could turn these action and call savings into shorter task completion times.

Table 9: **Per-call latency on A800.** Median execution times (s) for depth estimation and image translation; translation server time is measured separately.

Dataset	DA3	VQ encoding	Autoregressive generation	VQ decoding	Translation server
LIBERO-Plus	0.088	0.366	4.232	0.072	4.667
RoboTwin	0.106	0.349	6.023	0.133	6.752

Latency per translation call. Table 9 reports a separate per-call benchmark on an NVIDIA A800, using two 256×256 views for LIBERO-Plus and three 320×240 views for RoboTwin. Values are medians in seconds. For DA3, each setup uses 16 distinct image groups, three warm-up runs, and 30 timed runs. DA3 estimates depth; VQ encoding, autoregressive generation, and VQ decoding form the image translation pipeline. Server time measures processing within the translation service. These per-call medians are distinct from the episode means above; component medians need not sum to the separately measured server latency.

Autoregressive generation is the largest measured component, taking 4.232 s on LIBERO-Plus and 6.023 s on RoboTwin, whereas depth estimation takes about 0.1 s in both settings. Server-side translation takes 4.667 and 6.752 s, respectively. Accelerating autoregressive generation therefore offers more scope for reducing processing time than accelerating depth estimation alone. Combined with the shorter rollouts above, such reductions could help turn fewer policy calls into lower task completion time.