EVOSCENE-VLA: EVOLVING SCENE BELIEFS INSIDE THE ACTION DECODER FOR CHUNKED ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Chunked vision–language–action (VLA) policies generate multi-step actions from one observation and typically re-observe after executing the action chunk. Earlier actions change the scene conditions for later steps, while occlusion can compromise visual feedback. The previous scene estimate is typically not propagated to the next control call. Although spatial and temporal VLAs enhance geometric modeling and maintain temporal memory, respectively, their internal representations may become stale as robot actions alter object poses and contact states. We introduce EvoScene-VLA, which uses compact scene tokens to unify within-chunk scene prediction, cross-chunk state propagation, and observation-based correction. The action expert jointly denoises actions and future scene states in a single flow-matching process, allowing predicted scene changes to inform action generation. The final scene state serves as the next prior for subsequent correction. At the next control call, the vision–language model (VLM) fuses the incoming visual input with this prior. To preserve control-relevant geometry, two-level geometric anchoring applies local depth supervision to per-view scene features and global 3D feature supervision to the fused scene state; a shared geometric decoder imposes the same constraints on current and future states. Since demonstrations do not directly provide future scene states, we train a scene predictor using 3D teacher features from future images. Its predictions provide targets for the action expert's future scene states. At deployment, we remove both Scene Predictor and Geometric Anchor. EvoScene-VLA improves average success over the matched baseline by 2.4 and 2.2 percentage points on 31 RoboTwin tasks under fixed and randomized settings, respectively, and by 2.7 points on LIBERO. Across three real-robot tasks, average success rises from 37.3% to 42.0%. Ablations show that a recurrent scene state alone does not necessarily improve performance, whereas state propagation can improve control when combined with geometric and future scene supervision.

1 INTRODUCTION

Robots change their surroundings through actions and adjust subsequent motions based on the resulting scene. For example, to place an object into a drawer, a robot may need to adjust its trajectory to the drawer's updated position after pulling it open (Fig. 1). However, chunked vision–language–action (VLA) policies generate multi-step action sequences from a single observation. Visual feedback is typically acquired after execution of the action chunk. Within a chunk, later actions remain conditioned on the initial observation despite scene changes caused by earlier actions. This mismatch may grow as the chunk proceeds. Furthermore, between chunks, occlusion may leave task-relevant scene geometry only partially visible in new observations (Li et al., 2026a). The previous scene estimate is typically not propagated across control calls, which can lead to spatial errors or repeated execution of completed steps. Spatial and temporal VLAs improve scene representations derived from observations, but these representations are rarely updated to reflect the effects of the generated actions. Spatial VLAs (Qu et al., 2025; Li et al., 2025c;a;b; Zhou et al., 2025) model geometric relationships through depth supervision, 3D encoding, or multi-view features, but these representations describe the scene before action execution. Temporal VLAs (Shi et al., 2025; Koo et al., 2025; Lin et al., 2025; Chung et al., 2025; Zheng et al., 2024) retain past observations, which can provide information when

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.

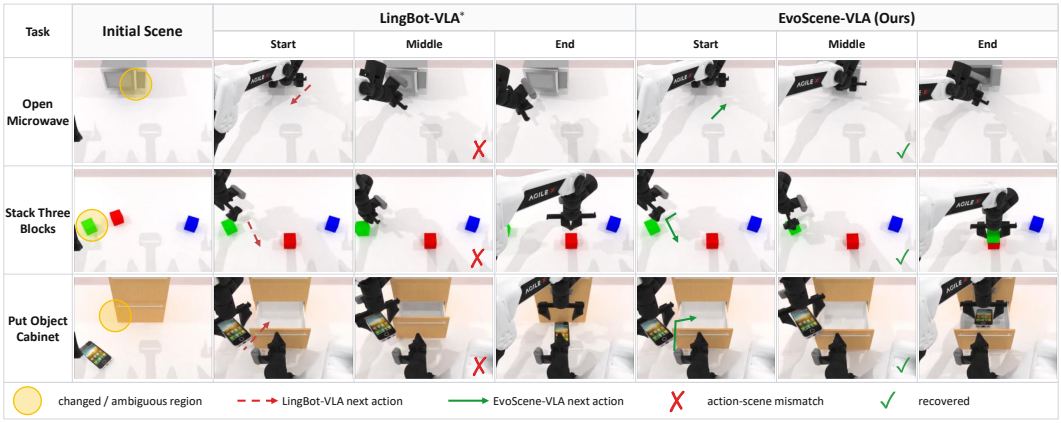


Figure 1: **Qualitative analysis of chunked-control failures.** During an action chunk, the scene can change after the planning observation is captured. In the three RoboTwin examples shown (microwave handle, block stacking, cabinet placement), the baseline stalls mid-chunk (✘), whereas EvoScene-VLA completes the rollouts (✔). EvoScene-VLA carries an action-updated scene prior into each chunk and co-denoises actions with predicted scene states at selected steps.

objects are temporarily occluded; after an object is moved, however, these observations may still reflect its previous position. Action-conditioned prediction methods (Zheng et al., 2025; Zhang et al., 2025) estimate future scene representations, but using these predictions both for action generation within a chunk and for state propagation across chunks remains underexplored.

We propose EvoScene-VLA to address the scene mismatch within a chunk and partial observations across chunks. Within a chunk, the model jointly predicts actions and future scene states, allowing later actions to account for scene changes caused by earlier actions. Specifically, we introduce a compact set of scene tokens to represent the scene state. The action expert jointly denoises the action sequence and future scene states at sparse key time steps in a single flow-matching process, with the generation of the current action chunk conditioned on the future scene representations. Across chunks, the scene state predicted at the end of an action chunk corresponds to the time of the next observation. We therefore pass this prediction to the next control call as a scene prior. The vision–language model (VLM) fuses the current visual observation with the scene prior, supplementing incomplete visual evidence under occlusion. Because prediction errors and execution deviations may cause the scene prior to differ from the actual scene, the VLM also corrects the prior with the new observation during fusion. Within-chunk prediction, cross-chunk propagation, and observation-based correction thus form a single scene-state loop, in which the scene state guides action generation in the current call and serves as a prior for correction in the next call.

However, the scene state also needs to preserve the geometric relationships needed for control. Adding scene tokens alone does not ensure that these relationships are preserved. To this end, we introduce a two-level **Geometric Anchor**. *Local* depth supervision constrains per-view scene features, while *global* feature supervision from a 3D foundation model (3DFM) constrains the fused scene state. We use a shared geometric decoder to align current and predicted future scene states with 3DFM teacher features. This design provides explicit learning objectives for state prediction and propagation.

Furthermore, we train the action expert to generate future scene states using compact state targets, which are not directly available in demonstration data. We therefore introduce a training-only **Scene Predictor**. It predicts future scene states from the current scene state and demonstration actions. We supervise these predictions with 3DFM teacher features extracted from future images. These predicted scene states serve as supervision targets for the action expert's scene flow-matching objective. At deployment, we remove both Scene Predictor and Geometric Anchor, retaining only the recurrent scene state and the joint generation of actions and scene states.

On 31 RoboTwin tasks (Mu et al., 2024), EvoScene-VLA raises the matched baseline's average success rate from 86.4% to 88.8% under fixed evaluation and from 85.7% to 87.9% under randomized initial conditions. The comparable gains across the two settings, together with the recurrence ablations (Tab. 4), suggest that the scene prior remains effective under pose and layout variation. On

LIBERO (Liu et al., 2023a), EvoScene-VLA improves average success over the matched baseline by 2.7 percentage points. On the Galaxea R1-Lite (Jiang et al., 2025) dual-arm platform, EvoScene-VLA achieves the highest observed success rate on all three cleaning tasks, raising average success from 37.3% to 42.0%. Ablations further show that a recurrent scene state alone does not necessarily improve performance, whereas state propagation can improve control when combined with geometric and future scene supervision. Prior interventions also show higher closed-loop success with matched than mismatched priors (Tab. 3), supporting the contribution of the prior's content to control.

Our main contributions are as follows:

- We introduce a recurrent scene interface bridging within-chunk scene prediction and cross-chunk state updates. Predicted future scene states condition current action generation, while the final state is carried forward as a prior and corrected using new observations.
- We formulate the action expert with future scene-state supervision and joint flow matching to generate actions and future scene states simultaneously, eliminating the need for an additional scene predictor at deployment.
- We develop local depth supervision with global 3D feature supervision and use a shared geometric decoder to align current and predicted future scene states with 3DFM teacher features, providing a common geometric basis for learning scene states.

2 Related Work

VLA policies use VLM backbones to map images and instructions to chunked motor controls. RT-2 (Brohan et al., 2023), π_0 (Black et al., 2024a), and OpenVLA (Kim et al., 2024) establish this paradigm. EvoScene-VLA adds a recurrent scene prefix to this decoder. We review geometric, temporal, and flow-based foundations.

Geometric scene representation. Spatial VLAs add geometry to the VLM prefix or visual encoder. SpatialVLA (Qu et al., 2025) projects 3D-aware features into a spatial token grid, QDepth-VLA (Li et al., 2025c) supervises quantized depth tokens, Spatial Forcing (Li et al., 2025a) enforces geometric consistency, and 3DS-VLA (Li et al., 2025b) and VLA-4D (Zhou et al., 2025) add multi-view or temporal 3D features. A parallel line uses 3D scene state: PerAct (Shridhar et al., 2022), RVT (Goyal et al., 2023; 2024), 3D Diffusion Policy (Ze et al., 2024), and ManiGaussian (Lu et al., 2024). DUSt3R (Wang et al., 2024b) and MASt3R (Leroy et al., 2024) recover 3D geometry from images. CUT3R (Wang et al., 2025) maintains a persistent scene state that is updated with new observations. EvoScene-VLA advances a compact geometric prior with the generated action chunk. The next VLM call corrects this prior using the new observation.

Temporal memory and action-conditioned prediction. Temporal VLAs retain observed history across calls. MemoryVLA (Shi et al., 2025) keeps a summary-token memory, HAMLET (Koo et al., 2025) and HiF-VLA (Lin et al., 2025) use history-aware attention, Embodied-SlotSSM (Chung et al., 2025) applies slot state-space models, TraceVLA (Zheng et al., 2024) prompts the VLM with visual traces, and AVA-VLA (Xiao et al., 2025) aligns action and visual history. Related approaches learn trajectory representations from video (Wen et al., 2024; Hu et al., 2024) or transfer across heterogeneous demonstrations (Wang et al., 2024a). Other methods predict future states from actions. FLARE (Zheng et al., 2025) inserts learnable future tokens into the VLM, UP-VLA (Zhang et al., 2025) predicts action-conditioned images or features, world models (Wu et al., 2022; Hafner et al., 2023; Wu et al., 2023; Cheang et al., 2024) roll out latent futures, and video-prediction policies (Du et al., 2023; Black et al., 2024b; Ko et al., 2024) generate or condition on pixel-level futures. EvoScene-VLA jointly denoises an action chunk and future scene states within the VLA action expert. The final scene state is carried into the next VLM prefix, where it is corrected using the new observation. This loop runs without a separate scene predictor at inference.

Flow matching and diffusion policies. Flow-matching and diffusion policies model robot control as trajectory denoising. Diffusion Policy (Chi et al., 2023) popularized diffusion-based continuous control. π_0 (Black et al., 2024a), RDT (Liu et al., 2024), Octo (Octo Model Team et al., 2024), and flow-matching VLAs (Lipman et al., 2023; Liu et al., 2023b) scale this formulation to pretrained VLAs. Consistency Policy (Prasad et al., 2024) accelerates sampling. These methods denoise actions

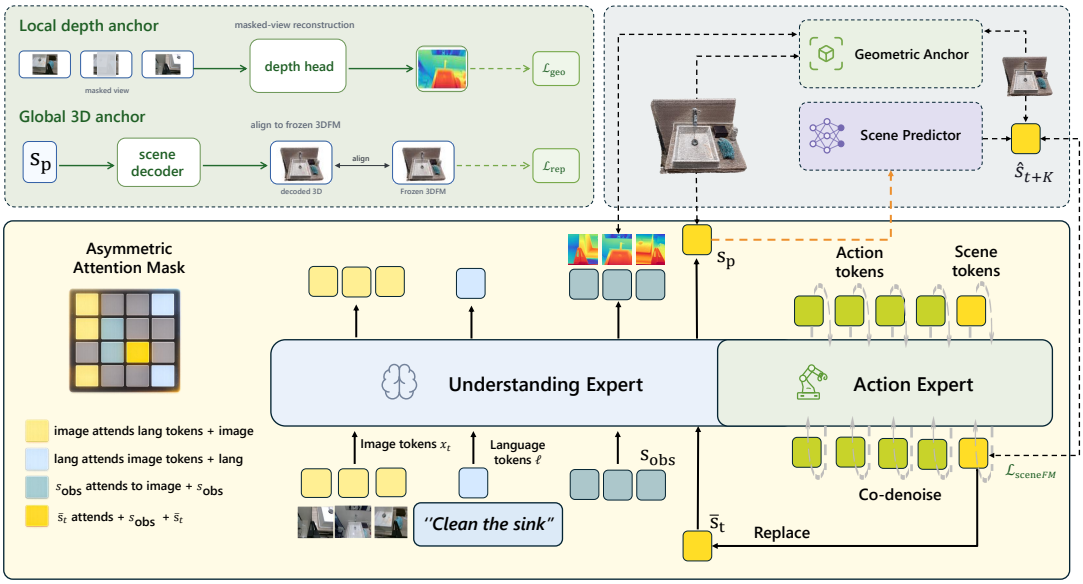


Figure 2: **Pipeline Overview.** The policy receives multi-view images, an instruction, and the robot state. The VLM prefix contains image and language tokens, per-view observation slots, and recurrent prior slots. Under asymmetric attention, observation slots attend to their corresponding image tokens, and prior slots attend to observation slots. Image and language tokens do not attend to scene slots. During training, Geometric Anchor applies local depth supervision to per-view scene features and global 3DFM feature supervision to the fused state; Scene Predictor supplies future-state targets for the action expert. At inference, the action expert jointly denoises the action chunk and future scene states at sparse offsets through flow matching. The final state becomes the next call's prior, which the VLM fuses with the new observation to obtain the corrected current scene state.

only, so their decoders do not evolve scene state. EvoScene-VLA co-denoises the action chunk and the evolved scene representation in one flow-matching pass. This joint denoising grounds each action in the recurrent scene prior without adding an inference-time predictor.

3 Method

3.1 Overview

We build EvoScene-VLA on LingBot-VLA (Wu et al., 2026), a $\pi_{0.5}$-style (Physical Intelligence et al., 2025) policy with a Qwen2.5-VL (Qwen Team, 2025) backbone. In the base policy, the VLM encodes multi-view images and a language instruction into a key–value cache, which a separate action expert uses to generate an action chunk through flow matching.

EvoScene-VLA adds compact scene tokens to this interface (Fig. 2). The action expert jointly denoises the action chunk and future scene states at sparse time steps, allowing predicted scene changes to inform action generation within the current chunk. At inference, the state predicted for the end of the chunk serves as the prior for the next control call. The VLM fuses this prior with the new observation to supplement incomplete visual evidence and correct the inherited state (Sec. 3.2).

To preserve control-relevant geometry, Geometric Anchor applies local depth supervision to per-view scene features and global 3D feature supervision to the fused scene state (Sec. 3.3). Since demonstrations do not directly provide compact future scene states, Scene Predictor supplies targets for the action expert's scene flow-matching objective. The predictor is supervised by 3DFM features extracted from future images (Sec. 3.4). A training step is summarized in Algorithm 1. At inference, we remove Geometric Anchor and Scene Predictor. Implementation details are provided in Appendix A.

3.2 Recurrent Scene-State Loop

Scene representation and observation correction. Let D denote the VLM hidden dimension and V the number of camera views. The scene interface contains N_{obs} observation slots per view, $s_{\text{obs}}^{(v)} \in \mathbb{R}^{N_{\text{obs}} \times D}$, and N_{prior} prior slots, $\bar{s}_t \in \mathbb{R}^{N_{\text{prior}} \times D}$. Together with the current multi-view

images x_t and language instruction ℓ, they form the prefix $[\, x_t,\ \ell,\ s_{\text{obs}}^{(1:V)},\ \bar{s}_t\,]$. For the first chunk, $\bar{s}_t$ is initialized from learnable scene embeddings; at subsequent inference calls, it contains the final scene state generated for the preceding chunk.

The asymmetric attention mask in Fig. 2 restricts each observation group to its own view's image tokens and observation slots, and the prior group to all observation and prior slots. Image and language tokens do not attend to scene slots, preserving the pretrained vision–language attention paths. The VLM therefore fuses current visual evidence with the prior through the per-view observation slots. Its output at the prior-slot positions is the corrected current scene state $s_p \in \mathbb{R}^{N_{\text{prior}} \times D}$:

$$s_p = \text{VLM}_{\text{scene}}\big(x_t, \ell,\ s_{\text{obs}}^{(1:V)},\ \bar{s}_t\big)_{\text{prior}} \tag{1}$$

Joint action and scene generation. Let H denote the final action offset, so a chunk contains $H+1$ actions. The action expert jointly generates this chunk and K future scene states at sparse offsets $\{k_1, \ldots, k_K\} \subseteq \{1, \ldots, H\}$, with $k_K = H$. At flow-matching time $\tau \in [0,1]$, let $a_{t:t+H}^{\tau}$ and $z^{\tau} \in \mathbb{R}^{K \times N_{\text{prior}} \times D}$ denote the noisy action and scene blocks. The expert receives the suffix $[\, r_t \mid z^{\tau} \mid a_{t:t+H}^{\tau}\,]$, where r_t is the robot state, and attends to the VLM prefix cache. Within the action expert, scene and action tokens attend to each other bidirectionally, allowing their predictions to inform each other at every denoising step. The expert predicts a velocity v_θ for every action and scene token. At inference, both blocks are initialized with independent Gaussian noise and jointly denoised using the same Euler sampling procedure as LingBot-VLA. The outputs are an action chunk $\hat{a}_{t:t+H}$ and future scene states $\hat{z}^{(k_i)}$, $i = 1, \ldots, K$.

State propagation across chunks. The final predicted state $\hat{z}^{(k_K)}$ corresponds to the time of the next observation. After executing the action chunk, we pass this state to the next control call as a prior and combine it with the new observation x_{t+1}:

$$\bar{s}_{t+1} \;=\; \hat{z}^{(k_K)}, \qquad s_p^{t+1} \;=\; \text{VLM}_{\text{scene}}\big(x_{t+1}, \ell,\ s_{\text{obs}}^{(1:V)},\ \bar{s}_{t+1}\big)_{\text{prior}} \tag{2}$$

The new observation provides visual evidence to correct discrepancies caused by prediction errors or execution deviations. Algorithm 2 gives the inference procedure for one chunk. The following sections describe the geometric supervision and future-state targets used to train this loop.

3.3 Geometric Anchor

To preserve control-relevant geometry, we apply a two-level **Geometric Anchor** during training: local depth supervision for per-view observation features $s_{\text{obs}}^{(v)}$ and global 3D feature supervision for the fused state s_p.

Local depth supervision. For each target view v_i, the **Local depth anchor** masks its dense image tokens and predicts depth features from $s_{\text{obs}}^{(v_i)}$, s_p, and the remaining views using a lightweight cross-attention head g_{depth}:

$$\hat{f}_{t,i}^{\,d} = g_{\text{depth}}\Big(q_{\text{tmpl}},\ \big[\,\tilde{h}_{t,v_1}^{\text{img},(i)}, \ldots, \tilde{h}_{t,v_V}^{\text{img},(i)},\ s_{\text{obs}}^{(v_i)},\ s_p\,\big]\Big), \quad \tilde{h}_{t,v_j}^{\text{img},(i)} = \begin{cases} m, & j = i, \\ h_{t,v_j}^{\text{img}}, & j \neq i. \end{cases} \tag{3}$$

Here q_{tmpl} reuses LingBot-VLA's query tokens and m is a learned mask embedding. Target-view image information reaches the depth head through the scene representations. A frozen Monocular Depth Teacher (MDT) (Tan et al., 2026) provides target features from the unmasked image:

$$\mathcal{L}_{\text{geo}} = \frac{1}{V} \sum_{i=1}^{V} \text{SmoothL1}\Big(\hat{f}_{t,i}^{\,d},\ f_{t,i}^{\,d}\Big), \qquad f_{t,i}^{\,d} = \text{MDT}(x_{t,v_i}). \tag{4}$$

Global 3D feature supervision. The **Global 3D anchor** supervises the fused state s_p with features from a frozen multi-view 3DFM (Wang et al., 2026). A lightweight view-conditioned decoder g_{3D} takes learnable queries q_{dec} and uses s_p as keys and values. A linear projector W_{proj} maps the decoder output to the teacher feature space:

$$H_t = g_{\text{3D}}(q_{\text{dec}};\ s_p), \qquad P_t = W_{\text{proj}}\, H_t. \tag{5}$$

We align P_t with 3DFM teacher features from the same multi-view images using an ℓ_1 loss on per-token layer-normalized features:

$$\mathcal{L}_{\text{rep}} = \frac{1}{V}\sum_{v=1}^{V}\left\|\text{LN}\big(P_t^{(v)}\big) - \text{LN}\big(Z_t^{(v)}\big)\right\|_1, \qquad Z_t = \text{3DFM}(x_t). \tag{6}$$

Here LN is parameter-free layer normalization along the feature dimension. The loss compares normalized features without directly matching their raw norms.

Shared geometric decoder. We reuse the global decoder $(g_{\text{3D}}, W_{\text{proj}})$ for the future states produced by Scene Predictor (Sec. 3.4). Current and predicted future states are decoded into the same 3DFM feature space and supervised with teacher features from their respective time steps.

3.4 Scene Predictor

Demonstrations do not directly provide compact future scene states for the action expert's scene flow-matching objective. We therefore train a **Scene Predictor** to produce these targets from the current scene state and demonstration actions.

Action-conditioned state prediction. The predictor is a causal Transformer with input sequence $[\, r_t,\ s_p,\ a_t, \ldots, a_{t+H},\ q_1, \ldots, q_K \,]$. Each query group q_i is initialized from s_p and predicts a scene state at offset k_i. Its attention mask allows access to the robot state r_t, the current scene state s_p, the demonstration action prefix $a_{t:t+k_i}$, and earlier query groups. The output $\hat{s}_{t+k_i} \in \mathbb{R}^{N_{\text{prior}} \times D}$ is thus conditioned on actions up to its target step.

Supervision from future observations. We extract teacher features $Z_{t+k_i} = \text{3DFM}(x_{t+k_i})$ from future multi-view images. The shared decoder $(g_{\text{3D}}, W_{\text{proj}})$ in Eq. (5) maps each predicted state $\hat{s}_{t+k_i}$ to features $\tilde{P}_{t+k_i}$. We train Scene Predictor with an ℓ_1 loss against the future teacher features:

$$\mathcal{L}_{\text{pred}} = \frac{1}{K \cdot V}\sum_{i=1}^{K}\sum_{v=1}^{V}\left\|\tilde{P}_{t+k_i}^{(v)} - Z_{t+k_i}^{(v)}\right\|_1, \qquad \tilde{P}_{t+k_i} = W_{\text{proj}}\big(g_{\text{3D}}(q_{\text{dec}};\, \hat{s}_{t+k_i})\big). \tag{7}$$

These predicted states provide compact targets for the action expert's scene generation (Sec. 3.5).

3.5 Training Objective

Joint action and scene flow matching. We stack the future states predicted by Scene Predictor and apply LN. This gives standardized scene targets $z_0 = \text{LN}(\hat{s}_{t+k_1:t+k_K}) \in \mathbb{R}^{K \times N_{\text{prior}} \times D}$ for training the action expert. We sample a shared flow-matching time $\tau \in [0, 1]$ for the action and scene blocks and draw independent standard Gaussian noises ϵ_a and ϵ_s. The noisy inputs to the joint generation process in Sec. 3.2 are formed by linear interpolation:

$$a_{t:t+H}^{\tau} = \tau\, \epsilon_a + (1 - \tau)\, a_{t:t+H}, \qquad z^{\tau} = \tau\, \epsilon_s + (1 - \tau)\, z_0. \tag{8}$$

We stop gradients through the action targets $a_{t:t+H}$ and scene targets z_0 for flow-matching training and optimize the velocity losses, where $\mathcal{L}_{\text{actFM}}$ is the standard $\pi_{0.5}$ action flow-matching loss:

$$\mathcal{L}_{\text{sceneFM}} = \left\|v_\theta^{(s)}(z^{\tau}, \tau) - (\epsilon_s - z_0)\right\|_2^2, \qquad \mathcal{L}_{\text{actFM}} = \left\|v_\theta^{(a)}(a^{\tau}, \tau) - (\epsilon_a - a_{t:t+H})\right\|_2^2. \tag{9}$$

Full objective. The full objective combines action generation, geometric supervision, future-state prediction, and scene generation:

$$\mathcal{L} = \mathcal{L}_{\text{actFM}} + \lambda_1 \mathcal{L}_{\text{geo}} + \lambda_2 \mathcal{L}_{\text{rep}} + \lambda_3 \mathcal{L}_{\text{pred}} + \lambda_4 \mathcal{L}_{\text{sceneFM}}. \tag{10}$$

$\mathcal{L}_{\text{geo}}$ and $\mathcal{L}_{\text{rep}}$ supervise the geometry of current scene representations. $\mathcal{L}_{\text{pred}}$ supervises the future states produced by Scene Predictor, and $\mathcal{L}_{\text{sceneFM}}$ trains the action expert using their standardized targets. We train end-to-end in a single stage. During training, Scene Predictor supplies the recurrent scene prior for chunks after the first chunk.

Table 1: **RoboTwin results** (%).

Method	Clean	Rand
X-VLA†	70.7	71.1
$\pi^{\dagger}_{0.5}$	81.2	75.9
Motus†	87.0	85.8
RotVLA†	88.6	87.3
LingBot-VLA	84.6	83.6
LingBot-VLA*	86.4	85.7
EvoScene-VLA	**88.8**	**87.9**
Δ over LingBot-VLA*	$+2.4_{\pm 0.4}$	$+2.2_{\pm 0.2}$

Table 2: **Relative-depth reconstruction.**

Readout source	AbsRel ↓	δ_1 ↑	δ_2 ↑	Pearson ↑
Recurrent prior	**0.163**	**0.732**	**0.930**	**0.831**
Cold prior	0.172	0.701	0.919	0.828
Constant baseline	0.411	0.586	0.807	N/A

Table 3: **Effect of prior content.** Success on LIBERO.

Input condition	Matched	Mismatched	Δ (pp)
unmodified-views	85.45%	83.75%	+1.7
replaced Agent-view	14.45%	1.65%	+12.8
replaced Wrist-view	12.90%	2.70%	+10.2

4 EXPERIMENTS

4.1 SETUP

Benchmarks and baselines. We evaluate EvoScene-VLA on RoboTwin (Mu et al., 2024), LIBERO (Liu et al., 2023a), and the Galaxea R1-Lite real robot (Sec. 4.4). On RoboTwin, we report main results on 31 tasks and conduct ablations on RoboTwin-5Task, which covers single-arm, dual-arm, short-horizon, and long-horizon tasks. We evaluate both RoboTwin sets under two settings. *Clean* fixes initial object positions to the training distribution, whereas *Rand* randomizes their positions and orientations within the task workspace while preserving the task goal. On LIBERO, we conduct component ablations across the Spatial, Object, Goal, and Long suites. We initialize EvoScene-VLA from the public LingBot-VLA (Wu et al., 2026) checkpoint and compare it with the released LingBot-VLA policy and LingBot-VLA*, which retains depth supervision during fine-tuning. All three models share the same backbone, chunk length, sampling schedule, and training protocol. We provide the exact task list in Appendix D.

4.2 MAIN RESULTS

Quantitative results. Tab. 1 summarizes success rates on 31 RoboTwin tasks, with three-seed means for methods trained under our protocol. EvoScene-VLA raises the matched LingBot-VLA* average from 86.4% to 88.8% under *Clean* and from 85.7% to 87.9% under *Rand*. The paired gains (Δ) are 2.4 ± 0.4 and 2.2 ± 0.2 percentage points, respectively (mean $\pm$ sample standard deviation across three seeds; Tab. 9 and Appendix E). EvoScene-VLA attains the higher mean on 22 of 31 tasks under *Clean* and 23 of 31 under *Rand*, so the gain is distributed across tasks and persists under randomized layouts. The published reference models (†) (Zheng et al., 2026; Physical Intelligence et al., 2025; Bi et al., 2026; Li et al., 2026b) use 50-task training. Our averages exceed the reported Motus averages (87.0/85.8) and are numerically close to RotVLA (88.6/87.3) after re-averaging the published results over our 31 tasks.

Qualitative results. Fig. 1 examines the failure mode at the rollout level. For each of three tasks, the figure shows the LingBot-VLA* rollout next to ours from the same initial scene at three time slices (start, middle, end). In *Open Microwave*, the baseline gripper retracts before reaching the door handle, whereas EvoScene-VLA continues the motion through to completion. We also inspect the recurrent scene representation directly. Fig. 3 passes P_t, which the trained view-conditioned decoder produces from s_p, through the frozen 3DFM point head. The resulting maps preserve coarse relative geometry without fitting an additional probe.

Fig. 4 compares end-effector trajectories for representative RoboTwin episodes. EvoScene-VLA follows smoother paths in these examples. Appendix H provides additional comparisons.

4.3 ABLATIONS

What the recurrent prior carries. On LIBERO training episodes, we evaluate relative-depth reconstruction from the corrected current scene state s_p against the frozen 3DFM reference (Tab. 2). The cold prior is reset at each frame, while the constant baseline predicts each image's median reference depth. Compared with the cold prior, propagating the prior across chunks raises δ_1 from 0.701 to 0.732, indicating more accurate relative-depth reconstruction with respect to the reference. We next test whether the prior's content affects closed-loop control (Tab. 3 and Appendix J). *Matched*

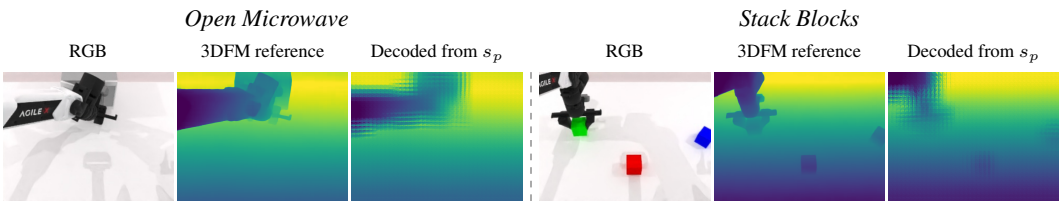


Figure 3: **Geometry decoded from the recurrent scene representation.** *3DFM reference* decodes the frozen targets Z_t; *Decoded from* s_p applies the same frozen point head to P_t produced from s_p. Each triplet shares a reference-fitted colour range. The decoded maps retain coarse relative geometry while omitting fine details.

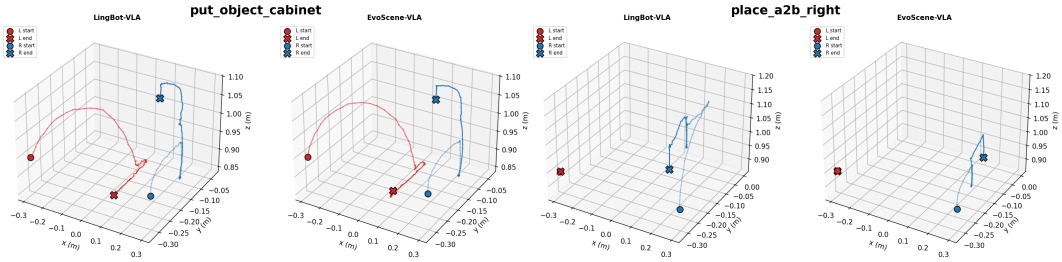


Figure 4: **End-effector trajectories in 3D.** EvoScene-VLA shows smoother paths than LingBot-VLA in these two representative RoboTwin episodes. Left/right-arm paths run from start (●) to end (✘).

uses the current episode's action-updated prior, while *mismatched* uses a prior from another episode. View replacement uses the corresponding camera image from another episode. Success is higher with matched priors in all three input conditions. The margin over mismatched priors (Δ) is 1.7 percentage points (pp) with both original views, increasing to 12.8 and 10.2 points when the agent and wrist views are replaced, respectively. These results support the contribution of prior information.

Cumulative component ablations. Tab. 4 adds joint $\mathcal{L}_{\text{pred}}/\mathcal{L}_{\text{rep}}$ supervision, local depth anchoring, and recurrence on RoboTwin-5Task. Each step improves success over the preceding configuration, starting from a baseline without these components. LingBot-VLA* is included as a reference.

Scene supervision and recurrence. Tab. 6 compares independent LIBERO configurations. The full model improves success by 2.70 points over the baseline. With full scene supervision, recurrence adds 1.10 points. Without joint $\mathcal{L}_{\text{pred}}/\mathcal{L}_{\text{rep}}$ supervision, recurrence reduces success by 2.10 points, or 2.35 points with local depth alone. Recurrence thus benefits from scene supervision.

Table 4: **Cumulative ablations.**

Variant	Clean	Rand
LingBot-VLA*	87.8	84.6
baseline	81.6	75.8
+ $\mathcal{L}_{\text{pred}}$ & $\mathcal{L}_{\text{rep}}$	89.3	86.2
+ $\mathcal{L}_{\text{geo}}$	90.1	86.5
+ Recurrence	**90.8**	**87.8**

Table 5: **Real-robot experiments.**

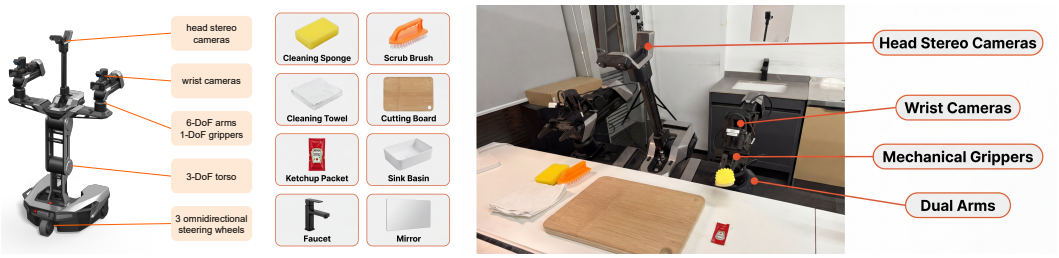
Task	$\pi_{0.5}$	LingBot VLA	LingBot VLA*	Ours
Mirror	28	27	26	29
Sink	42	44	49	51
Cutting-board	44	34	37	46
Avg.↑(%)	38.0	35.0	37.3	**42.0**

Table 6: **Full factorial LIBERO component ablations.** The ✓/× denote enabled/disabled components; Δ is the overall change from the baseline. The 3DFM($\mathcal{L}_{\text{pred}}/\mathcal{L}_{\text{rep}}$) directly supervises the recurrent state: without it, enabling recurrence lowers success below the baseline (−2.10), and $\mathcal{L}_{\text{geo}}$ alone does not recover it (−1.50).

Method	Components			LIBERO suite				Overall	Δ
	$\mathcal{L}_{\text{pred}}/\mathcal{L}_{\text{rep}}$	$\mathcal{L}_{\text{geo}}$	Recurrence	Spatial	Object	Goal	Long		
baseline	×	×	×	74.6	94.2	86.6	75.6	82.75	0.00
EvoScene-VLA	✓	×	×	74.4	92.6	90.2	76.2	83.35	+0.60
	×	✓	×	**80.4**	90.0	88.2	75.8	83.60	+0.85
	×	×	✓	72.4	89.2	86.2	74.8	80.65	-2.10
	✓	✓	×	79.4	93.4	90.2	74.4	84.35	+1.60
	✓	×	✓	79.2	91.8	90.4	76.8	84.55	+1.80
	×	✓	✓	72.6	90.2	87.2	75.0	81.25	-1.50
	✓	✓	✓	80.2	**93.8**	**90.8**	**77.0**	**85.45**	+2.70

4.4 Real-Robot Evaluation

Platform and dataset. We evaluate EvoScene-VLA on the Galaxea R1-Lite platform (Fig. 5(a)), training on the indoor-cleaning subset of the Galaxea Open-World Dataset (Jiang et al., 2025), which includes mirror, sink, and cutting-board cleaning among other indoor tasks. Following the standard VLA interface, the policy uses one head stream and two wrist streams as input. The subset contains 35 subdatasets, 2,669 episodes, and 3,667,408 frames, totaling approximately 67.9 hours of demonstrations at 15 fps. We use all demonstrations for training and evaluate on the physical robot.



(a) Robot platform. (b) Real-robot experimental setup.

Figure 5: **Real-robot evaluation. (a)** Galaxea R1-Lite with three policy cameras. **(b)** Indoor-cleaning setup for the mirror, sink, and cutting-board tasks; ketchup is the removable stain.

Training and evaluation protocol. For the real-robot comparison, we train and deploy four methods: $\pi_{0.5}$, the two LingBot-VLA baselines, and EvoScene-VLA. We run all inference on a single NVIDIA RTX 4090 GPU. We evaluate each method over 100 physical closed-loop rollouts per task and report success rates under randomized object placements, lighting, and initial robot states.

Real-robot Results. The three long-horizon tasks shown in Fig. 5(b) require the robot to track how its tool changes the surface. The surface state evolves during wiping, and subsequent controls need to target unwiped regions that can be subtle, occluded, or ambiguous in the current view. Tab. 5 reports success rates on the Galaxea cleaning tasks. EvoScene-VLA raises average success from 37.3% for LingBot-VLA* to 42.0% and achieves the highest observed success rate on each of the three tasks. Appendix F reports 95% Wilson confidence intervals for every result. Fig. 6 shows one closed-loop execution; Appendix G presents its full sampled sequence alongside a failed rollout.



Figure 6: **Representative real-robot rollout on *Clean The Sink*.** Frames progress from cloth acquisition to basin contact and wiping. The evolving interaction state motivates recurrent scene context.

5 Conclusion

EvoScene-VLA gives a chunked VLA a recurrent latent scene interface. The VLM prefix carries two slot groups. Observation slots read the current image. Prior slots inherit the evolved scene representation from the previous chunk. The action decoder jointly denoises motor commands and scene states at sparse key-frame offsets in one flow-matching pass. The scene state at the final offset becomes the next chunk's prior. Training uses two-level Geometric Anchor and a training-only Scene Predictor to supply geometric and latent supervision; inference retains only the recurrent scene prefix and the action–scene co-denoising. EvoScene-VLA improves average success on the 31-task RoboTwin benchmark and across three Galaxea R1-Lite real-robot cleaning tasks. For chunked VLAs, the action decoder should take the latent prior into account for the next visual update, rather than rebuilding it from each observation alone.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bi_Motus_A_Unified_Latent_Action_World_Model_CVPR_2026_paper.html.

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

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Chilam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

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

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. *arXiv preprint arXiv:2511.11478*, 2025. Method named "Embodied-SlotSSM" inside the paper.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning (CoRL)*, 2023.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. RVT-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025. URL https://arxiv.org/abs/2509.00576. https://github.com/OpenGalaxea/GalaxeaVLA.

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

Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations (ICLR)*, 2024.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. *arXiv preprint arXiv:2510.00695*, 2025.

Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with MASt3R. *arXiv preprint arXiv:2406.09756*, 2024.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025a.

Peiyan Li, Yuze Zhu, Yixiang Chen, Qisen Ma, Yuan Xu, Jiabing Yang, He Guan, Yan Huang, Hongtao Wu, Xiao Ma, Tao Kong, Liang Wang, and Tieniu Tan. BridgeVLA++: A data-efficient, generalizable, and memory-augmented vision-language-action framework for 3D manipulation. *arXiv preprint arXiv:2608.05042*, 2026a. URL https://arxiv.org/abs/2608.05042.

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*, 2026b. URL https://arxiv.org/abs/2605.13403.

Xiaoqi Li, Liang Heng, Jiaming Liu, Yan Shen, Chenyang Gu, Zhuoyang Liu, Hao Chen, Nuowei Han, Renrui Zhang, Hao Tang, Shanghang Zhang, and Hao Dong. 3DS-VLA: A 3d spatial-aware vision language action model for robust multi-task manipulation. In *Conference on Robot Learning (CoRL)*, 2025b.

Yixuan Li, Yuhui Chen, Mingcai Zhou, Haoran Li, Zhengtao Zhang, and Dongbin Zhao. QDepth-VLA: Quantized depth prediction as auxiliary supervision for vision-language-action models. *arXiv preprint arXiv:2510.14836*, 2025c.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. HiF-VLA: Hindsight, insight and foresight through motion representation for vision-language-action models. *arXiv preprint arXiv:2512.09928*, 2025.

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

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

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. *arXiv preprint arXiv:2410.07864*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023b.

Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. ManiGaussian: Dynamic gaussian splatting for multi-task robotic manipulation. *arXiv preprint arXiv:2403.08321*, 2024.

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

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Delin Qu et al. SpatialVLA: Exploring spatial representations for visual-language-action models. *arXiv preprint arXiv:2501.15830*, 2025.

Qwen Team. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

Bin Tan, Changjiang Sun, Xiage Qin, Hanat Adai, Zelin Fu, Tianxiang Zhou, Han Zhang, Yinghao Xu, Xing Zhu, Yujun Shen, et al. Masked depth modeling for spatial perception. *arXiv preprint arXiv:2601.17895*, 2026.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. HPT: Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *arXiv preprint arXiv:2409.20537*, 2024a.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2024.

Hongtao Wu, Ya Jing, Chilam 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. *arXiv preprint arXiv:2312.13139*, 2023.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. DayDreamer: World models for physical robot learning. In *Conference on Robot Learning (CoRL)*, 2022.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. LingBot-VLA: A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. AVA-VLA: Improving vision-language-action models with active visual attention. *arXiv preprint arXiv:2511.18960*, 2025.

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.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kt51kZH4aG.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting for generalist robot policies. *arXiv preprint arXiv:2412.10345*, 2024.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

Hanyu Zhou, Chuanhao Ma, and Gim Hee Lee. VLA-4D: Embedding 4d awareness into vision-language-action models for spatiotemporally coherent robotic manipulation. *arXiv preprint arXiv:2511.17199*, 2025.

AI USE STATEMENT

We used generative AI tools to improve the English wording of this manuscript. All suggested edits were reviewed by the authors, who take full responsibility for the final content.

OVERVIEW

This appendix provides implementation and evaluation details. Appendix A reports the optimization recipe, hyperparameters (Tab. 7), and frozen-teacher pipeline. Appendix B (Tab. 8) consolidates the symbols used throughout the method, and Appendix C provides pseudocode for one training step (Algorithm 1) and one inference chunk (Algorithm 2). Appendix D lists the 31 RoboTwin tasks, the RoboTwin-5Task ablation subset, and baseline configurations. Appendix E reports the per-task mean $\pm$ sample standard deviation across three seeds (Tab. 10), expanding Tab. 1 of the main paper, and Appendix F reports 95% Wilson binomial confidence intervals for the real-robot study (Tab. 11). Appendix G shows two closed-loop real-robot rollouts (Fig. 7). Appendix H presents additional 3D end-effector trajectory comparisons (Fig. 8). Appendix I visualizes geometry decoded from the learned scene representation on RoboTwin (Fig. 9), and Appendix J quantifies decoded depth and isolates the recurrent prior's contribution beyond the current observation (Fig. 10, Tab. 2, Tab. 3).

A IMPLEMENTATION

Optimization recipe. Following LingBot-VLA's recipe (Tab. 7), the action expert is held in fp32 while the rest of the model runs in bf16 storage with fp32 reductions. The full model optimizes Eq. (10); the w/o Geometric Anchor variant drops the local and global anchor losses.

Hyperparameters. The loss weights λ_1–λ_4 in Tab. 7 are tuned on RoboTwin-5Task and reused on the full RoboTwin and Galaxea evaluations without further sweeping.

Table 7: Hyperparameters introduced by EvoScene-VLA. Optimization values are inherited from the LingBot-VLA recipe; loss weights are EvoScene-VLA-specific.

Optimization	
Optimizer	AdamW
Learning rate (constant)	1×10^{-4}
Effective batch size	256
Total update steps	20,000
Mixed precision	bf16 storage / fp32 reductions
Architecture	
VLM backbone	Qwen2.5-VL-3B-Instruct
Hidden dimension D	2048
Camera views V	3 (head, left wrist, right wrist)
Image resolution	224×224
Template bank size	256
Observation slots per view N_{obs}	8
Prior slots N_{prior}	16
Key frames K	3
Action chunk length	50
Loss weights (Eq. (10))	
λ_1 on $\mathcal{L}_{\text{geo}}$	0.04
λ_2 on $\mathcal{L}_{\text{rep}}$	0.10
λ_3 on $\mathcal{L}_{\text{pred}}$	0.10
λ_4 on $\mathcal{L}_{\text{sceneFM}}$	0.01
Inference	
Flow-matching Euler steps	10
Re-observation period	1 chunk (50 steps)

Frozen teachers. The local depth head g_{depth} reuses the LingBot-Depth alignment block, supervised by LingBot-VLA's MoRGBD pipeline (Tan et al., 2026) (MoGe-2 ViT-B normal variant combined with the LingBot-Depth masked-depth-modeling teacher) on the unmasked target view. The global 3D supervisor is the frozen Pi^3 multi-view 3D foundation model (Wang et al., 2026), applied to current- and future-frame multi-view inputs for $\mathcal{L}_{\text{rep}}$ and $\mathcal{L}_{\text{pred}}$, respectively.

B NOTATION

Tab. 8 consolidates the symbols used throughout Secs. 3.2 to 3.4.

Table 8: Notation used in EvoScene-VLA. Shapes are written as $\mathbb{R}^{(\text{dimensions})}$ where applicable.

Symbol	Meaning
Indices and dimensions	
t	Discrete time index of the current control chunk
D	VLM hidden dimension (Qwen2.5-VL-3B: D=2048)
V	Number of camera views (V=3: head, left wrist, right wrist)
N_{obs}	Number of slots per observation group (N_{obs}=8)
N_{prior}	Number of slots in the prior group (N_{prior}=16)
K	Number of scene key-frame query groups (K=3)
H	Final action offset of the chunk; chunk length H+1=50
$\{k_1, \ldots, k_K\}$	Sparse key-frame offsets: $(1, 24, 49)$ in all experiments
Inputs	
x_t	Multi-view image input at step t
x_{t,v_i}	Image of view $i \in \{1, \ldots, V\}$
ℓ	Language instruction
r_t	Robot proprioceptive state
$a_{t:t+H}$	Ground-truth action chunk
Recurrent scene prefix	
$s_{\text{obs}}^{(v)} \in \mathbb{R}^{N_{\text{obs}} \times D}$	Observation slots for view v, gather geometric evidence per camera
$\bar{s}_t \in \mathbb{R}^{N_{\text{prior}} \times D}$	Prior slots inherited from the previous chunk
$s_p \in \mathbb{R}^{N_{\text{prior}} \times D}$	VLM output at the prior-slot positions; cross-view scene representation
$h_{t,v}^{\text{img}}$	VLM image tokens for view v
$m \in \mathbb{R}^D$	Learned mask embedding for cross-view masking
Geometric Anchor (training only)	
q_{tmpl}	256-token template query bank reused for g_{depth}
g_{depth}	Cross-attention head producing per-view depth representations
$\hat{f}_{t,i}^{d}$	Predicted depth representation for masked target view i
MDT	Frozen Monocular Depth Teacher (MoGe-2 + LingBot-Depth)
g_{3D}	View-conditioned cross-attention decoder for the global anchor
q_{dec}	Learnable view-aware queries for g_{3D}
W_{proj}	Linear projector to the 3D foundation-model feature space
$P_t,\ Z_t$	Projected output / frozen 3DFM features for the current frame
Scene Predictor (training only)	
$q_1, \ldots, q_K$	Key-frame query groups initialized from s_p, each of size N_{prior}
$\hat{s}_{t+k_i} \in \mathbb{R}^{N_{\text{prior}} \times D}$	Predicted absolute future scene latent at offset k_i
$\hat{P}_{t+k_i},\ Z_{t+k_i}$	Projected predicted feature / frozen 3DFM feature on the future frame
Joint action–scene flow matching	
$z_0 \in \mathbb{R}^{K \times N_{\text{prior}} \times D}$	Standardized future-scene targets (LayerNorm of $\hat{s}$)
$\hat{z}^{(k_i)} \in \mathbb{R}^{N_{\text{prior}} \times D}$	Denoised standardized scene state emitted by the action expert at offset k_i
$\tau \in [0, 1]$	Shared flow-matching time
$\epsilon_a,\ \epsilon_s$	Independent Gaussian noises on action / scene paths
$a_{t:t+H}^{\tau},\ z^{\tau}$	Straight-line interpolants
$v_\theta^{(a)},\ v_\theta^{(s)}$	Predicted velocities for action / scene blocks
Losses (Eq. (10)*)*	
$\mathcal{L}_{\text{geo}}$	Cross-view masked depth reconstruction loss
$\mathcal{L}_{\text{rep}}$	Global anchor ℓ_1 loss for the current frame
$\mathcal{L}_{\text{pred}}$	Future-scene ℓ_1 loss for the K key frames
$\mathcal{L}_{\text{sceneFM}},\ \mathcal{L}_{\text{actFM}}$	Scene / action flow-matching losses in the action expert
$\lambda_1, \lambda_2, \lambda_3, \lambda_4$	Loss weights $(0.04, 0.10, 0.10, 0.01)$

C ALGORITHMS

Algorithm 1 states one training step. Algorithm 2 states the deployed forward path.

Algorithm 1 EvoScene-VLA training step.

Require: Sample $\big(x_t,\, \ell,\, r_t,\, a_{t:t+H},\, x_{t+k_1}, \ldots, x_{t+k_K}\big)$ and prior carry $\bar{s}_t$ (learnable embedding for the first chunk in an episode; supplied by Scene Predictor for later chunks).

Form prefix $[\, x_t,\, \ell,\, s_{\text{obs}}^{(1:V)},\, \bar{s}_t \,]$ with the asymmetric attention mask (Sec. 3.2).
Run VLM forward; read s_p from the prior-slot positions (Eq. (1)).
for each view $i \in \{1, \ldots, V\}$ **do** ▷ Local depth anchor
    Mask $h_{t,v_i}^{\text{img}} \leftarrow m$, keep other views unchanged.
    $\hat{f}_{t,i}^{\,d} \leftarrow g_{\text{depth}}(q_{\text{tmpl}},\, [\, \tilde{h}_{t,v_1:V}^{\text{img},(i)},\, s_{\text{obs}}^{(v_i)},\, s_p \,])$
end for
$\mathcal{L}_{\text{geo}} \leftarrow \frac{1}{V} \sum_i \text{SmoothL1}\big(\hat{f}_{t,i}^{\,d},\, \text{MDT}(x_{t,v_i})\big)$ ▷ Eq. (4)
$P_t \leftarrow W_{\text{proj}}\, g_{\text{3D}}(q_{\text{dec}};\, s_p)$ ▷ Global anchor
$\mathcal{L}_{\text{rep}} \leftarrow \frac{1}{V} \sum_v \big\| \text{LN}\big(P_t^{(v)}\big) - \text{LN}\big(\text{3DFM}(x_t)^{(v)}\big) \big\|_1$
$\hat{s}_{t+k_1:t+k_K} \leftarrow \text{ScenePredictor}(r_t,\, s_p,\, a_{t:t+H},\, q_1, \ldots, q_K)$
$\tilde{P}_{t+k_i} \leftarrow W_{\text{proj}}\, g_{\text{3D}}(q_{\text{dec}};\, \hat{s}_{t+k_i})$ for each i
$\mathcal{L}_{\text{pred}} \leftarrow \frac{1}{KV} \sum_{i,v} \big\| \tilde{P}_{t+k_i}^{(v)} - \text{3DFM}(x_{t+k_i})^{(v)} \big\|_1$
$z_0 \leftarrow \text{LayerNorm}(\hat{s}_{t+k_1:t+k_K})$ ▷ Standardize future-scene targets
Sample $\tau \sim \mathcal{U}[0,1]$, $\epsilon_a, \epsilon_s \sim \mathcal{N}(0, I)$
$a^\tau \leftarrow \tau\epsilon_a + (1{-}\tau) a_{t:t+H}, \quad z^\tau \leftarrow \tau\epsilon_s + (1{-}\tau) z_0$
$v_\theta^{(a)}, v_\theta^{(s)} \leftarrow \text{ActionExpert}\big([\, r_t \,|\, z^\tau \,|\, a^\tau \,];\, \text{prefix cache}\big)$
$\mathcal{L}_{\text{actFM}} \leftarrow \| v_\theta^{(a)} - (\epsilon_a - a_{t:t+H}) \|_2^2$
$\mathcal{L}_{\text{sceneFM}} \leftarrow \| v_\theta^{(s)} - (\epsilon_s - z_0) \|_2^2$
$\mathcal{L} \leftarrow \mathcal{L}_{\text{actFM}} + \lambda_1 \mathcal{L}_{\text{geo}} + \lambda_2 \mathcal{L}_{\text{rep}} + \lambda_3 \mathcal{L}_{\text{pred}} + \lambda_4 \mathcal{L}_{\text{sceneFM}}$
Backpropagate $\mathcal{L}$; update VLM, action expert, and all training-only modules.

Algorithm 2 EvoScene-VLA inference per chunk.

Require: Observation x_t, instruction ℓ, robot state r_t, prior $\bar{s}_t$ (learnable first-chunk embedding).
Ensure: Action chunk $\hat{a}_{t:t+H}$ executed on the robot; updated prior $\bar{s}_{t+1}$.

Form prefix $[\, x_t,\, \ell,\, s_{\text{obs}}^{(1:V)},\, \bar{s}_t \,]$ and run one VLM forward.
Read s_p from prior slots; cache prefix KV. ▷ Geometric Anchor / Scene Predictor not loaded.
Initialise $a^1 \sim \mathcal{N}(0, I)$, $z^1 \sim \mathcal{N}(0, I)$.
for each Euler step $\tau \in \{1, 1{-}\Delta\tau, \ldots, \Delta\tau\}$ (10 steps total) **do**
    $v^{(a)}, v^{(s)} \leftarrow \text{ActionExpert}([\, r_t \,|\, z^\tau \,|\, a^\tau \,];\, \text{prefix cache})$
    $a^{\tau-\Delta\tau} \leftarrow a^\tau - \Delta\tau\, v^{(a)}, \quad z^{\tau-\Delta\tau} \leftarrow z^\tau - \Delta\tau\, v^{(s)}$
end for
$\hat{a}_{t:t+H} \leftarrow a^0; \quad \big(\hat{z}^{(k_1)}, \ldots, \hat{z}^{(k_K)}\big) \leftarrow z^0$ ▷ Joint denoised outputs
Execute $\hat{a}_{t:t+H}$ on the robot.
$\bar{s}_{t+1} \leftarrow \hat{z}^{(k_K)}$ ▷ Write the denoised final-offset scene state back as the next prior
Re-observe x_{t+1} and return to step 1 for the next chunk.

D DATASETS AND TASKS

RoboTwin 31 tasks. We use 31 language-conditioned manipulation tasks from RoboTwin 2.0 (Mu et al., 2024), covering single-arm and dual-arm, short-horizon and long-horizon settings. The full task list is: `place_mouse_pad`, `click_bell`, `open_microwave`, `place_shoe`, `put_object_cabinet`, `stack_blocks_three`, `beat_block_hammer`, `turn_switch`, `open_laptop`, `place_dual_shoes`, `pick_dual_bottles`, `stack_bowls_three`, `place_a2b_left`, `place_a2b_right`, `place_empty_cup`, `move_can_pot`, `place_container_plate`, `press_stapler`, `place_phone_stand`,

`place_fan`, `rotate_qrcode`, `place_object_stand`, `shake_bottle`, `scan_object`, `pick_diverse_bottles`, `place_bread_skillet`, `place_bread_basket`, `place_burger_fries`, `place_cans_plasticbox`, `put_bottles_dustbin`, and `hanging_mug`. Each task uses RoboTwin's standard episode length and the 50-step action chunk.

RoboTwin-5Task ablation subset. For ablations we use a 5-task subset that spans single-arm, dual-arm, short-horizon, and long-horizon types and keeps the ablation budget controlled: `click_bell`, `open_microwave`, `place_shoe`, `put_object_cabinet`, and `stack_blocks_three`.

Baselines. LingBot-VLA* is the depth-augmented LingBot-VLA, corresponding to the `robotwin_load20000h_depth` configuration of the LingBot-VLA repository. EvoScene-VLA shares its training data, chunk length, and compute budget with the LingBot-VLA baselines. The † rows of Tab. 1 are published results rather than our reproductions: the X-VLA and Motus values come from the per-task tables of Motus (Bi et al., 2026) and the RotVLA values from RotVLA (Li et al., 2026b). Those models train jointly on all 50 RoboTwin 2.0 tasks with 50 clean and 500 randomized demonstrations per task; we report the unweighted mean over our 31 tasks.

Tab. 9 gives the complete per-task comparison underlying the main-table averages.

Table 9: Full per-task comparison corresponding to Tab. 1. Success rates (%) on 31 RoboTwin tasks. C and R denote Clean and Rand. † marks published results re-averaged over the same tasks; other rows are three-seed means. Bold marks the column best. **Per-task mean ± sample standard deviation is in Tab. 10.**

Method	Avg.↑ (%)		Place Mouse Pad		Click Bell		Open Microwave		Place Shoe		Put Obj. Cabinet		Stack Blocks Three		Beat Block Hammer		Turn Switch		Open Laptop	
	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R
X-VLA† (Zheng et al., 2026)	70.7	71.1	70	70	**100**	**100**	79	71	96	95	46	48	6	10	92	88	40	61	93	**100**
$\pi^{\dagger}_{0.5}$ (Physical Intelligence et al., 2025)	81.2	75.9	60	39	99	66	34	77	92	93	80	79	91	76	**96**	**93**	62	54	90	96
Motus† (Bi et al., 2026)	87.0	85.8	66	68	**100**	**100**	**95**	**91**	**99**	97	**88**	71	91	**95**	95	88	**84**	**78**	95	91
RotVLA† (Li et al., 2026b)	88.6	87.3	83	87	97	92	36	12	**99**	**100**	87	82	90	86	92	85	46	46	98	99
LingBot-VLA	84.6	83.6	84	88	68	60	66	54	97	95	82	80	95	90	90	85	64	75	98	97
LingBot-VLA*	86.4	85.7	84	87	95	96	70	58	97	98	82	83	95	88	87	83	71	71	97	98
Ours	**88.8**	**87.9**	**85**	**91**	96	97	82	61	95	99	85	**87**	**96**	**95**	93	84	72	75	**99**	99
Δ	+2.4	+2.2	+1	+4	+1	+1	+12	+3	-2	+1	+3	+4	+1	+7	+6	+1	+1	+4	+2	+1

Place Dual Shoes		Pick Dual Bottles		Stack Bowls Three		Place A2B Left		Place A2B Right		Place Empty Cup		Move Can Pot		Place Cont. Plate		Press Stapler		Place Phone Stand		Place Fan	
C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R
79	88	47	36	76	86	48	49	36	36	**100**	98	89	86	97	95	92	**98**	88	87	80	75
75	75	93	63	77	71	87	82	87	84	**100**	99	51	55	99	95	87	83	81	81	87	85
93	87	**96**	**90**	79	87	88	79	91	87	99	98	34	74	98	99	**93**	**98**	87	86	91	87
93	**96**	89	88	**92**	**88**	**93**	**95**	**97**	**96**	**100**	99	**95**	**95**	99	**100**	83	81	**94**	94	**97**	**97**
90	89	88	85	72	76	76	80	77	70	**100**	**100**	75	66	96	98	90	95	84	**96**	94	89
75	87	93	88	75	82	79	77	76	76	**100**	99	82	90	95	96	92	94	91	91	93	86
91	93	87	88	83	76	80	83	81	79	**100**	**100**	75	75	**100**	98	92	95	91	**96**	**97**	91
+16	+6	-6	+0	+8	-6	+1	+6	+5	+3	+0	+1	-7	-15	+5	+2	+0	+1	+0	+5	+4	+5

Rotate QRcode		Place Obj. Stand		Shake Bottle		Scan Obj.		Pick Diverse Bottles		Place Bread Skillet		Place Bread Basket		Place Burger Fries		Place Cans Plasticbox		Put Bottles Dustbin		Hanging Mug	
C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R
34	33	86	88	99	**100**	14	36	58	36	77	67	81	71	94	94	97	98	74	77	23	27
89	87	91	85	99	97	18	17	81	71	85	66	77	64	94	87	94	84	84	79	77	71
89	73	**98**	97	**100**	97	67	66	**90**	**91**	86	83	91	**94**	98	98	98	94	81	79	38	38
95	**95**	97	**100**	**100**	**100**	**88**	**85**	88	88	89	88	**96**	93	**99**	98	**100**	**100**	91	88	44	52
78	86	96	90	**100**	99	57	47	78	69	**91**	**91**	90	90	96	96	98	98	80	83	72	76
83	87	96	85	98	99	56	38	80	74	89	**91**	91	91	95	**99**	99	96	88	87	75	**82**
88	87	**98**	96	**100**	**100**	49	57	83	80	89	87	92	91	**99**	97	99	99	**94**	**93**	**83**	76
+5	+0	+2	+11	+2	+1	-7	+19	+3	+6	+0	-4	+1	+0	+4	-2	+0	+3	+6	+6	+8	-6

E PER-TASK MULTI-SEED RESULTS

Tab. 10 expands Tab. 1 with the per-task uncertainty. For the LingBot-VLA baselines and EvoScene-VLA we trained three independent seeds and evaluated each with 100 rollouts per task and setting, and we report the mean $\pm$ sample standard deviation across those three seeds. The Δ row gives the paired per-task improvement of EvoScene-VLA over LingBot-VLA* together with the standard deviation of that difference. Aggregating over the 31 tasks, the macro averages are 86.4 ± 0.7 versus 88.8 ± 0.2 under Clean ($+2.4 \pm 0.4$) and 85.7 ± 0.3 versus 87.9 ± 0.5 under Rand ($+2.2 \pm 0.2$). EvoScene-VLA attains the higher mean on 22 of 31 tasks under Clean and 23 of 31 under Rand, so the gain is broadly distributed rather than concentrated in a few tasks.

Table 10: **Per-task success rates on the 31 RoboTwin tasks under the Clean and Rand settings** (%). Expanded version of Tab. 1. Each cell is the mean $\pm$ sample standard deviation over three independently trained seeds, with 100 rollouts per task, setting, and seed. The Δ columns give the paired improvement of EvoScene-VLA over LingBot-VLA*, with the standard deviation of that difference across the three seeds. Per-row best per setting is bolded among the three methods.

	LingBot-VLA		LingBot-VLA*		**EvoScene-VLA (Ours)**		Δ	
Task	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Place Mouse Pad	84.0 ± 2.6	88.0 ± 4.6	84.0 ± 3.6	87.0 ± 6.0	**85.0** ± 4.6	**91.0** ± 4.0	+1.0 ± 1.0	+4.0 ± 5.3
Click Bell	68.0 ± 5.3	60.0 ± 6.0	95.0 ± 5.3	96.0 ± 2.6	**96.0** ± 1.7	**97.0** ± 2.6	+1.0 ± 4.4	+1.0 ± 5.2
Open Microwave	66.0 ± 5.6	54.0 ± 5.0	70.0 ± 6.0	58.0 ± 3.0	**82.0** ± 2.6	**61.0** ± 1.7	+12.0 ± 8.2	+3.0 ± 4.6
Place Shoe	**97.0** ± 2.0	95.0 ± 5.0	**97.0** ± 1.0	98.0 ± 1.7	95.0 ± 3.0	**99.0** ± 1.0	−2.0 ± 3.6	+1.0 ± 2.6
Put Object Cabinet	82.0 ± 1.0	80.0 ± 3.6	82.0 ± 2.0	83.0 ± 3.0	**85.0** ± 5.6	**87.0** ± 4.6	+3.0 ± 3.6	+4.0 ± 6.9
Stack Blocks Three	95.0 ± 1.7	90.0 ± 4.0	95.0 ± 2.0	88.0 ± 4.4	**96.0** ± 2.6	**95.0** ± 4.6	+1.0 ± 4.4	+7.0 ± 3.6
Beat Block Hammer	90.0 ± 6.0	**85.0** ± 3.6	87.0 ± 5.0	83.0 ± 4.6	**93.0** ± 6.0	84.0 ± 6.0	+6.0 ± 9.5	+1.0 ± 9.6
Turn Switch	64.0 ± 2.6	**75.0** ± 5.3	71.0 ± 3.5	71.0 ± 5.6	**72.0** ± 3.6	**75.0** ± 2.6	+1.0 ± 2.6	+4.0 ± 3.5
Open Laptop	98.0 ± 1.7	97.0 ± 5.2	97.0 ± 1.0	98.0 ± 1.0	**99.0** ± 1.0	**99.0** ± 1.0	+2.0 ± 1.0	+1.0 ± 1.0
Place Dual Shoes	90.0 ± 4.6	89.0 ± 2.6	75.0 ± 4.6	87.0 ± 5.0	**91.0** ± 3.6	**93.0** ± 5.3	+16.0 ± 4.0	+6.0 ± 7.9
Pick Dual Bottles	88.0 ± 2.6	85.0 ± 2.0	**93.0** ± 2.6	**88.0** ± 2.6	87.0 ± 5.3	**88.0** ± 3.5	−6.0 ± 7.8	+0.0 ± 1.0
Stack Bowls Three	72.0 ± 4.0	76.0 ± 3.5	75.0 ± 5.3	**82.0** ± 1.7	**83.0** ± 1.7	76.0 ± 4.4	+8.0 ± 3.6	−6.0 ± 5.3
Place A2B Left	76.0 ± 4.6	80.0 ± 5.0	79.0 ± 3.6	77.0 ± 5.0	**80.0** ± 3.6	**83.0** ± 2.6	+1.0 ± 6.9	+6.0 ± 2.6
Place A2B Right	77.0 ± 4.4	70.0 ± 3.0	76.0 ± 5.2	76.0 ± 4.0	**81.0** ± 3.5	**79.0** ± 5.2	+5.0 ± 7.5	+3.0 ± 2.6
Place Empty Cup	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	99.0 ± 1.0	**100.0** ± 0.0	**100.0** ± 0.0	+0.0 ± 0.0	+1.0 ± 1.0
Move Can Pot	75.0 ± 4.4	66.0 ± 4.4	**82.0** ± 3.6	**90.0** ± 2.0	75.0 ± 6.0	75.0 ± 5.0	−7.0 ± 4.4	−15.0 ± 7.0
Place Container Plate	96.0 ± 2.6	**98.0** ± 1.0	95.0 ± 4.6	96.0 ± 1.7	**100.0** ± 0.0	**98.0** ± 2.0	+5.0 ± 4.6	+2.0 ± 3.6
Press Stapler	90.0 ± 3.5	**95.0** ± 5.0	**92.0** ± 3.6	94.0 ± 1.7	**92.0** ± 5.6	**95.0** ± 4.4	+0.0 ± 2.0	+1.0 ± 5.6
Place Phone Stand	84.0 ± 5.6	**96.0** ± 1.0	**91.0** ± 2.6	91.0 ± 3.0	**91.0** ± 1.7	**96.0** ± 2.6	+0.0 ± 2.6	+5.0 ± 3.6
Place Fan	94.0 ± 4.0	89.0 ± 4.6	93.0 ± 2.6	86.0 ± 5.6	**97.0** ± 3.6	**91.0** ± 3.0	+4.0 ± 1.0	+5.0 ± 5.3
Rotate QRcode	78.0 ± 5.3	86.0 ± 2.0	83.0 ± 6.0	**87.0** ± 2.6	**88.0** ± 1.0	**87.0** ± 3.6	+5.0 ± 6.6	+0.0 ± 5.3
Place Object Stand	96.0 ± 3.5	90.0 ± 3.6	96.0 ± 1.0	85.0 ± 2.0	**98.0** ± 3.5	**96.0** ± 1.7	+2.0 ± 2.6	+11.0 ± 2.6
Shake Bottle	**100.0** ± 0.0	99.0 ± 1.0	98.0 ± 1.7	99.0 ± 1.0	**100.0** ± 0.0	**100.0** ± 0.0	+2.0 ± 1.7	+1.0 ± 1.0
Scan Object	**57.0** ± 5.6	47.0 ± 3.6	56.0 ± 4.6	38.0 ± 2.6	49.0 ± 2.0	**57.0** ± 4.4	−7.0 ± 4.4	+19.0 ± 3.0
Pick Diverse Bottles	78.0 ± 5.3	69.0 ± 2.0	80.0 ± 3.6	74.0 ± 5.0	**83.0** ± 3.5	**80.0** ± 6.0	+3.0 ± 1.0	+6.0 ± 11.0
Place Bread Skillet	**91.0** ± 1.0	**91.0** ± 6.0	89.0 ± 1.0	**91.0** ± 2.6	89.0 ± 6.0	87.0 ± 5.3	+0.0 ± 6.6	−4.0 ± 4.6
Place Bread Basket	90.0 ± 4.6	90.0 ± 3.6	91.0 ± 3.6	**91.0** ± 5.3	**92.0** ± 5.0	**91.0** ± 4.6	+1.0 ± 6.9	+0.0 ± 7.0
Place Burger Fries	96.0 ± 2.0	96.0 ± 3.6	95.0 ± 2.0	**99.0** ± 1.0	**99.0** ± 1.0	97.0 ± 5.2	+4.0 ± 2.6	−2.0 ± 5.3
Place Cans Plasticbox	98.0 ± 2.6	98.0 ± 2.6	**99.0** ± 1.0	96.0 ± 4.0	**99.0** ± 1.0	**99.0** ± 1.0	+0.0 ± 1.0	+3.0 ± 5.0
Put Bottles Dustbin	80.0 ± 4.4	83.0 ± 4.0	88.0 ± 2.6	87.0 ± 3.5	**94.0** ± 1.0	**93.0** ± 5.0	+6.0 ± 3.0	+6.0 ± 6.1
Hanging Mug	72.0 ± 2.6	76.0 ± 5.3	75.0 ± 2.0	**82.0** ± 5.3	**83.0** ± 4.6	76.0 ± 2.0	+8.0 ± 4.4	−6.0 ± 6.9
Average	84.6 ± 0.6	83.6 ± 0.4	86.4 ± 0.7	85.7 ± 0.3	**88.8** ± 0.2	**87.9** ± 0.5	+2.4 ± 0.4	+2.2 ± 0.2

F Real-Robot Rollout Uncertainty

For the real-robot study in Sec. 4.4, we quantify rollout sampling uncertainty using 95% Wilson binomial confidence intervals. Tab. 11 reports the interval for every method–task cell, computed from that cell's 100 rollouts, and for every pooled Overall entry, computed from that method's 300 rollouts. The pooled intervals are $[36.6, 47.7]$ for EvoScene-VLA and $[32.1, 42.9]$ for LingBot-VLA*.

Table 11: **Real-robot rollout uncertainty.** For every method–task pair we report the SR (%) over 100 physical closed-loop rollouts and its 95% Wilson binomial confidence interval. The Overall row pools the 300 rollouts of each method (100 per task); the full study comprises 1,200 rollouts across the four methods. These intervals quantify rollout sampling uncertainty.

	$\pi_{0.5}$		LingBot-VLA		LingBot-VLA*		**EvoScene-VLA (Ours)**	
Task	SR (%)	95% CI	SR (%)	95% CI	SR (%)	95% CI	SR (%)	95% CI
Mirror	28.0	[20.1, 37.5]	27.0	[19.3, 36.4]	26.0	[18.4, 35.4]	**29.0**	[21.0, 38.5]
Sink	42.0	[32.8, 51.8]	44.0	[34.7, 53.8]	49.0	[39.4, 58.7]	**51.0**	[41.3, 60.6]
Cutting Board	44.0	[34.7, 53.8]	34.0	[25.5, 43.7]	37.0	[28.2, 46.8]	**46.0**	[36.6, 55.7]
Overall	38.0	[32.7, 43.6]	35.0	[29.8, 40.6]	37.3	[32.1, 42.9]	**42.0**	[36.6, 47.7]

G Real-Robot Closed-Loop Rollouts

Fig. 7 contrasts a failed execution (Rollout 1) with the successful execution summarized in Fig. 6 (Rollout 2). Both are closed-loop executions of EvoScene-VLA on *Clean The Sink*, the longest-horizon of the three real-robot tasks. Frames are sampled uniformly from the head camera over each execution. The task requires the robot to pick up the cloth, bring it to the basin, and wipe across the surface; because wiping changes the surface state, later controls must target regions the tool has not yet covered, which is exactly the condition the recurrent prior is meant to serve. These are physical executions on Galaxea R1-Lite under the protocol of Sec. 4.4, not dataset replay.

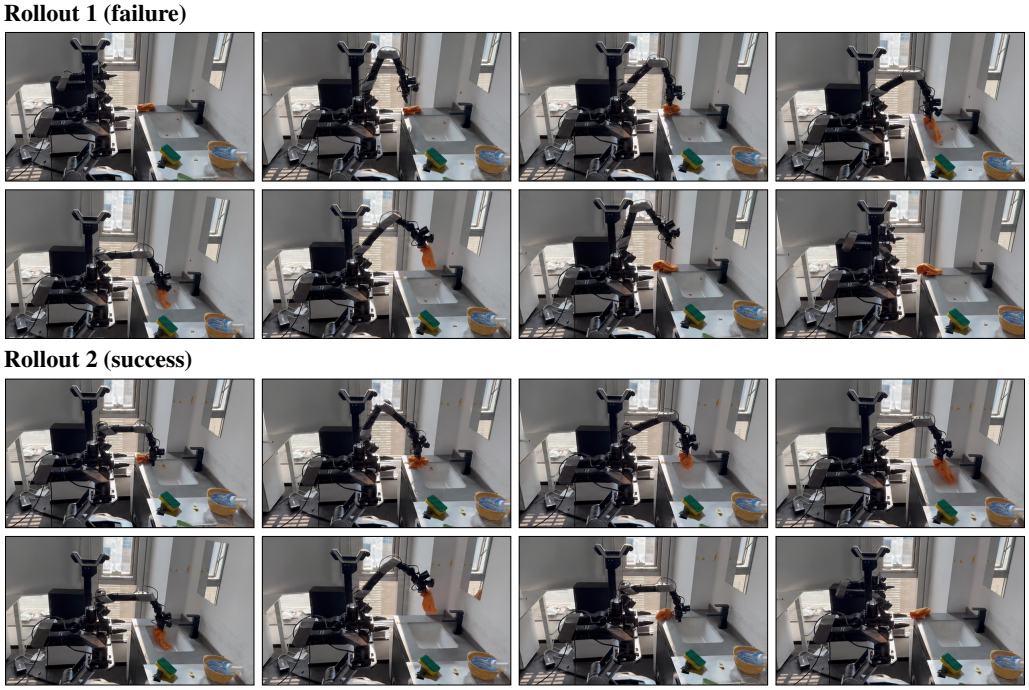
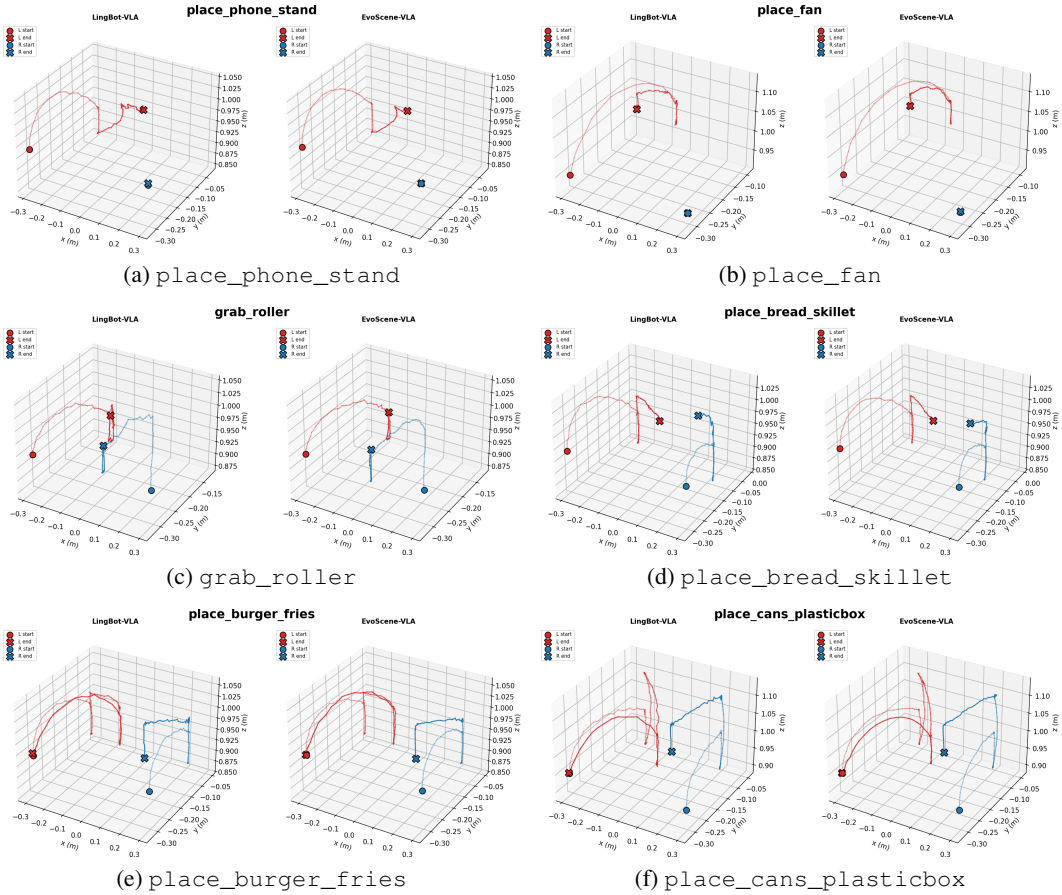


Figure 7: **Real-robot closed-loop rollouts on *Clean The Sink*.** Two executions of EvoScene-VLA on Galaxea R1-Lite, read left to right and top to bottom within each rollout. The robot picks up the cloth, transports it to the basin, and wipes across the surface. Rollout 1 fails with visible ketchup remaining in the basin at the end, whereas Rollout 2 clears the basin and succeeds.

H Additional Trajectory Visualizations

Fig. 8 shows six additional 3D end-effector trajectory comparisons complementing Fig. 4.



(a) `place_phone_stand` (b) `place_fan`

(c) `grab_roller` (d) `place_bread_skillet`

(e) `place_burger_fries` (f) `place_cans_plasticbox`

Figure 8: Additional 3D end-effector trajectory comparisons on six RoboTwin tasks, complementing Fig. 4. EvoScene-VLA's paths appear smoother than LingBot-VLA's in these six qualitative examples.

I Decoding Geometry from the Scene Representation on RoboTwin

The global anchor in Sec. 3.3 trains P_t, which the view-conditioned decoder produces from s_p, to match frozen 3DFM features Z_t. To inspect the learned representation without fitting an additional probe, we pass P_t and Z_t through the same frozen 3DFM point head. Fig. 9 renders log depth across two RoboTwin frames and all three camera views. Each pair uses a shared colour range fitted on its 3DFM reference, so independent normalization cannot make the two outputs appear more similar.

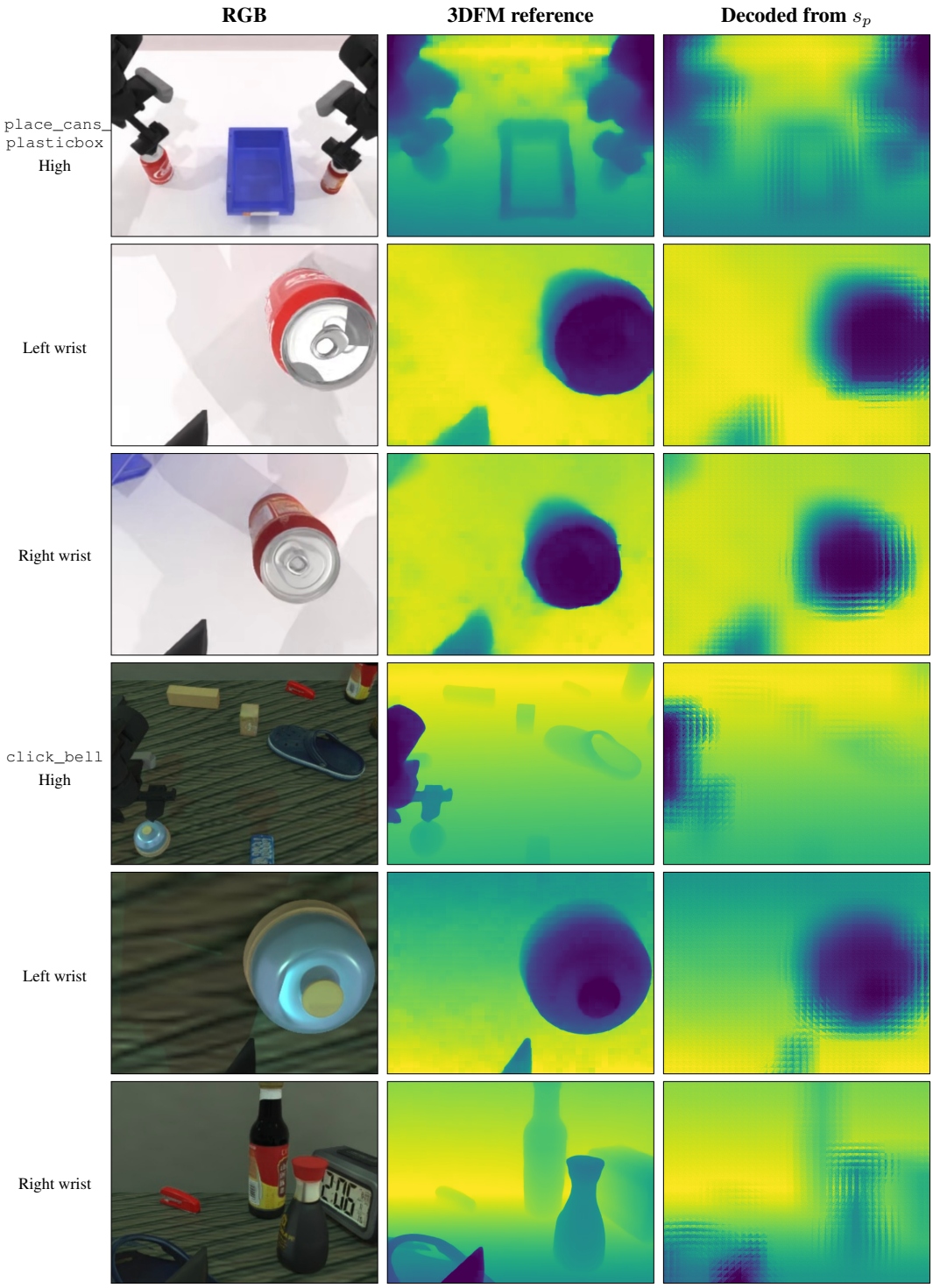


Figure 9: **Geometry decoded from the recurrent scene representation.** Two RoboTwin frames with three camera views each. *3DFM reference* decodes the frozen features Z_t, while *Decoded from* s_p applies the same frozen point head to P_t produced from s_p. Each pair shares a colour range fitted on its reference. The maps retain coarse scene layout, support surfaces, and relative depth ordering, but fine details remain incomplete.

J Depth Readouts and Prior Interventions

Appendix I shows qualitative RoboTwin decodes. This section quantifies the same question on LIBERO. Because P_t is trained to match Z_t, it lies in the frozen 3DFM's feature space, so its point head decodes it to depth without fitting to our data. A favourable score therefore cannot be attributed to a probe that learned depth on its own. The reference depth is the same head applied to Z_t.

Protocol. We evaluate all $277{,}713$ frames from $1{,}712$ episodes in the available four-suite LIBERO no-noops training corpus, giving $555{,}426$ images over two views per condition. These are training-set readout diagnostics. Each episode is evaluated using independent offset streams to cover every frame. Each stream starts with the learned initial prior and resets at the episode boundary. The cold condition resets the prior at every frame. Both P_t and Z_t are layer-normalized before their comparison in $\mathcal{L}_{\text{rep}}$ (Eq. (6)), so this loss compares standardized current-frame features rather than their raw norms. $\mathcal{L}_{\text{pred}}$ compares unnormalized future-frame features through the shared decoder (Eq. (7)), so the training objective also contains scale-sensitive supervision. For AbsRel, δ_1, and δ_2, we align each model prediction to the reference with a least-squares scale and shift in log depth before scoring. Pearson is the signed correlation of raw predicted and reference log depth, before alignment. The constant control predicts each image's median reference depth without an additional affine fit; its Pearson correlation is undefined. We average per-image scores within each episode, then average episodes equally. These readouts quantify relative geometry with respect to the frozen 3DFM reference.

The depth-readout metrics are reported in Tab. 2 in the main paper.

The representation carries geometry: δ_1 is 0.732 against 0.586 for the constant baseline, and AbsRel is 0.163 against 0.411. Fig. 10 shows the corresponding decodes. Coarse layout, support surfaces, and arm placement survive; object-level detail does not, and the 16×16 token grid leaves visible block structure. Wrist views are consistently weaker than the agent view.

What the recurrent prior contributes. Tab. 2 shows a gain from carrying the prior across chunks with both original camera images as input: δ_1 rises from 0.701 with a cold prior to 0.732 with the recurrent prior, a gain of 3.1 percentage points. δ_2 also rises from 0.919 to 0.930, while AbsRel falls from 0.172 to 0.163. This comparison tests carrying the prior against resetting it to the first-chunk initialization. We next test whether the content of the incoming prior matters to closed-loop control.

Closed-loop prior interventions. Tab. 3 compares task success under *matched* and *mismatched* incoming priors. The matched condition uses the action-updated scene state carried from the preceding chunk of the current episode. The mismatched condition replaces this state with a clean prior sampled from another episode, allowing donors from other tasks. All conditions share a clean initial action chunk; interventions apply at each subsequent replan.

We evaluate three camera-input conditions. *Both views unmodified* uses the current rollout's agent and wrist images. *Agent view replaced* and *wrist view replaced* substitute the selected camera image with the corresponding camera image from a randomly selected frame of another episode; the other camera remains unchanged. Image and prior donors are sampled independently. Matched and mismatched trials share initial conditions and the replacement-image schedule, while their actions and later observations may diverge during closed-loop execution.

We report the percentage of evaluation trials that complete the task as success rate (SR). Δ is matched minus mismatched SR in percentage points. With both views unmodified, SR is 85.45% versus 83.75%, a margin of 1.7 points. Replacing the agent view yields 14.45% versus 1.65% ($+12.8$ points); replacing the wrist view yields 12.90% versus 2.70% ($+10.2$ points). Matched priors yield higher success in all three conditions, with larger margins when one view is replaced.

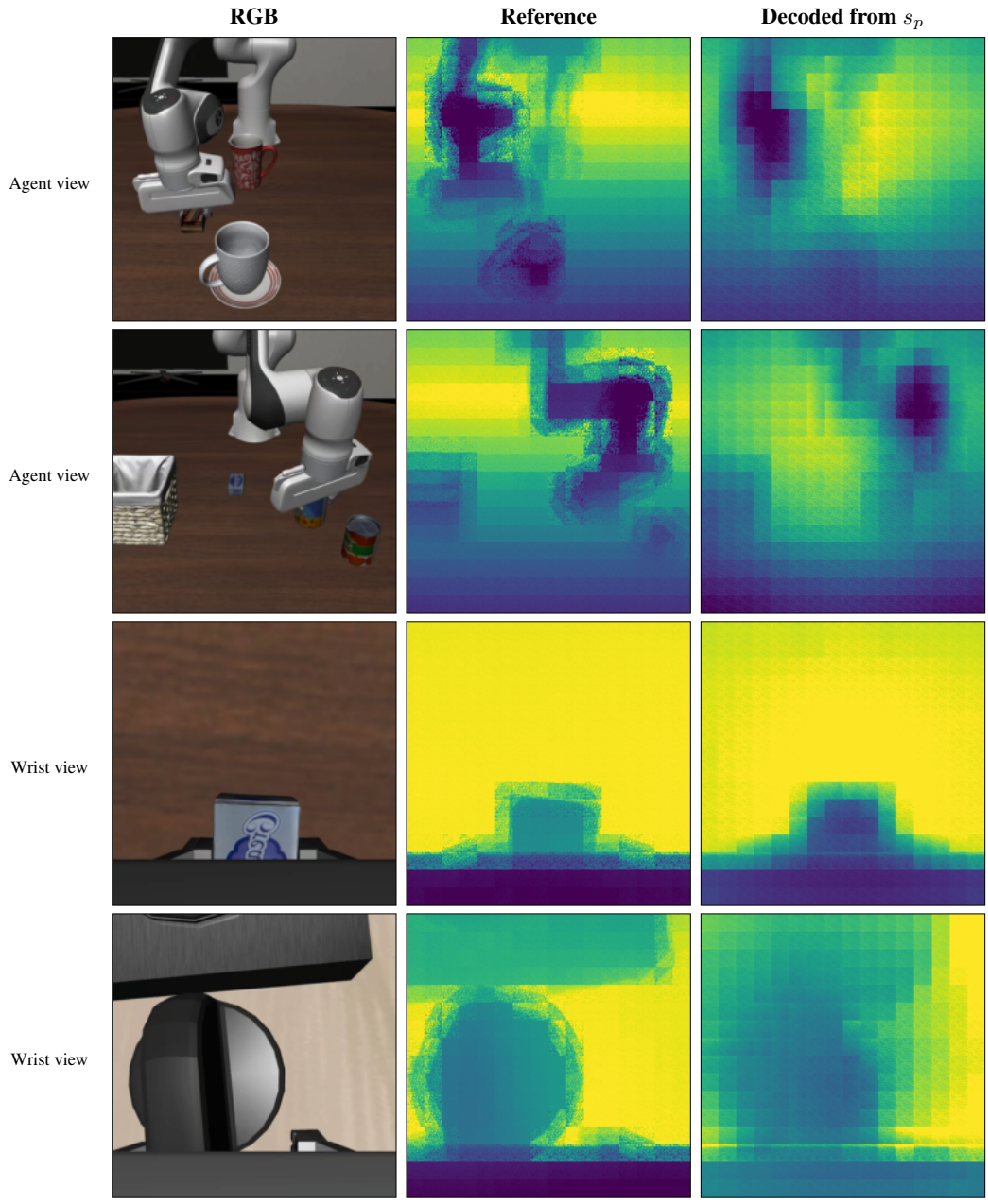


Figure 10: **Depth decoded from the scene representation.** Four LIBERO images covering two agent views and two wrist views. *Reference* decodes the frozen 3DFM target; *Decoded* decodes P_t through the same frozen head. Each row shares a colour range fitted on its reference.