arXiv:2605.21862v2 [cs.RO] 15 Aug 2026

EvoScene-VLA: Evolving Scene Beliefs Inside the Action Decoder for Chunked Robot Control

Chushan Zhang[1] Ruihan Lu[2] Jinguang Tong[1] Xuesong Li[1] Yikai Wang[3,*] Hongdong Li[1,*]

[1]Australian National University [2]The University of Queensland [3]Beijing Normal University

Abstract

Chunked vision-language-action (VLA) policies predict multi-step robot controls, conditioning each update on the current visual observation alone. Yet robot actions cause contact, occlusion, and object motion, and the geometry that later decisions depend on can change before the next visual update arrives. Spatial VLAs improve current-frame geometry. Temporal VLAs aggregate past frames. Neither maintains an action-updated scene prior across chunks. We argue for a persistent action-updated scene state across control calls, and introduce EvoScene-VLA. Its recurrent scene prefix carries a geometry-aware scene state across chunks. At each vision-language model (VLM) call, the VLM combines scene information from the current observation with the action-updated prior from the previous chunk; the action decoder outputs both the next action chunk and a compact scene update. This update becomes the next prior, which the VLM corrects against the new observation when the next call arrives. Each control call therefore starts from a scene prior that reflects both recent actions and fresh visual evidence. During training, **Scene Predictor** supplies future scene-token targets, and **Geometric Anchor** aligns scene slots with frozen depth and 3D teachers. We discard both modules at deployment. On 31 RoboTwin tasks, EvoScene-VLA raises average success from 86.4% to 88.8% in fixed evaluation and from 85.7% to 87.9% in randomized evaluation. On the Galaxea R1-Lite real robot, EvoScene-VLA outperforms all baselines.

1 Introduction

Chunked vision-language-action policies predict multi-step robot controls, but they usually condition scene context on observations rather than on the robot's own actions. Yet actions can change the scene before the next observation arrives: wiping the counter changes the surface state, closing a drawer hides its contents, and lifting a cup leaves the shelf slot below it empty. Within a chunk, the visual context may not reflect such changes at all. Across chunks, fresh observations do arrive, but the policy still lacks a compact record of how its recent actions transformed the scene, and must re-infer those changes from partial, noisy, or occluded visual evidence.

Spatial VLAs [32, 21, 18, 20, 52] improve geometric reasoning within a single image through depth supervision, 3D encoding, or multi-view features. These methods help when the full scene is visible, but they reason from current observations alone and do not update scene geometry once actions have changed it. Temporal VLAs [35, 16, 22, 7, 50] retain past observations through memory banks, trace prompts, or recurrent states, yet past observations alone do not specify how recent or planned actions should update the scene. Action-conditioned prediction methods [51, 48] anticipate future representations, but consume each prediction inside the current decision and discard it afterwards. The missing piece, we argue, is an action-updated scene representation. The policy should pass this representation to the next control call as a prior.

A useful scene representation for chunked control needs three properties. It should persist across chunks, so the policy is not forced to re-infer it from each new observation. It should update under the actions the policy generates, since those actions change the scene. And it should correct against

*Corresponding authors.

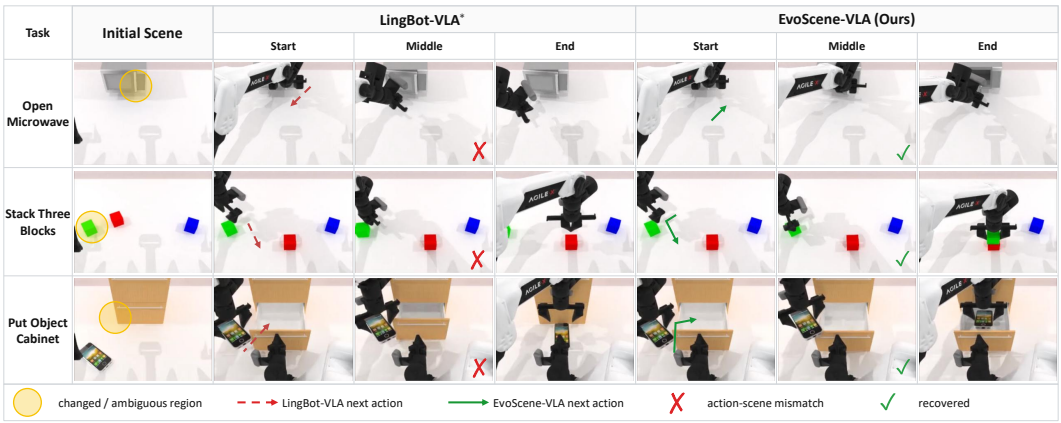


Figure 1: **Qualitative analysis of chunked-control failures.** During one action chunk, the scene can drift from the planning observation, so an action-only baseline commits to a stale target. In three RoboTwin tasks (microwave handle, block stacking, cabinet placement), the baseline follows this stale target and stalls mid-chunk (✘). EvoScene-VLA carries an action-updated scene prior into the chunk, so each predicted action uses the scene state for its own step and completes the rollout (✔).

each new observation, so prediction errors do not accumulate. Missing any one of these undermines the design: no persistence means no cross-chunk context, no action update means no post-action prior, and no correction means errors compound.

We propose EvoScene-VLA, which implements this action-updated scene prior as a recurrent scene prefix. The prefix holds two groups of scene tokens: one read from the current observation, one inherited from the previous chunk. At each vision-language model call, the model refines both groups against the new image. The action decoder then co-denoises the action chunk and a matched scene chunk in a single flow-matching pass. The resulting scene chunk becomes the inherited group in the next chunk's prefix. The deployed model therefore carries an action-updated prior into each new chunk without any separate online module for memory, prediction, or correction.

The action decoder needs supervision for the co-denoised scene chunk, but the deployed model should not depend on an auxiliary predictor. Prefix scene tokens also need geometric grounding, or they risk encoding generic appearance. We address both issues with two training-only modules: a **Geometric Anchor** that grounds scene tokens in 3D, and a **Scene Predictor** that supplies future-frame targets. Geometric Anchor operates at two levels: a *local* depth anchor [34] aligns each scene token with per-pixel depth, while a *global* 3D-foundation-model (3DFM) anchor [41] aligns it with scene-level features through a shared decoder that handles both current and future representations. Scene Predictor maps the current scene and action sequence to future scene representations, supervised by future-frame 3DFM features; flow matching then distills these targets into the action decoder's scene chunk. At inference, both modules are discarded, leaving only the recurrent scene prefix and the scene chunk co-denoised by the action decoder.

On 31 RoboTwin tasks [28], EvoScene-VLA raises the average success rate from 86.4% to 88.8% under fixed evaluation and from 85.7% to 87.9% under randomized initial conditions. The gains are comparable under fixed and randomized conditions, suggesting that the scene prior remains effective under pose and layout variation. Closed-loop trials on a Galaxea R1-Lite [13] dual-arm platform confirm similar gains under real-world deployment. Ablations further show cumulative contributions from future-scene supervision with the global anchor, local depth anchoring, and the recurrent prior.

Our contributions are threefold.

- We show that an action expert can produce an action-updated scene prior for the next call, via a recurrent scene prefix and action–scene co-denoising, with no online predictor.
- We introduce a two-level *Geometric Anchor* that combines local depth with a global 3D-foundation-model anchor, sharing one decoder to ground current and future scene latents.

- We report consistent gains on the RoboTwin benchmark and on a Galaxea R1-Lite real-robot platform, with ablations showing cumulative contributions from future-scene supervision, geometric anchoring, and the recurrent prior.

2 Related Work

VLA policies use VLM backbones to map images and instructions to chunked motor controls. RT-2 [4], π_0 [2], and OpenVLA [14] 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 [32] projects 3D-aware features into a spatial token grid, QDepth-VLA [21] supervises quantized depth tokens, Spatial Forcing [18] enforces geometric consistency, and 3DS-VLA [20] and VLA-4D [52] add multi-view or temporal 3D features. A parallel line uses 3D scene state: PerAct [36], RVT [9, 10], 3D Diffusion Policy [47], and ManiGaussian [27]. Reconstruction systems such as DUSt3R [40], MASt3R [17], and CUT3R [39] maintain state tokens for dense 3D reconstruction. These methods improve current-frame geometry. They do not advance a policy-facing scene prior with the robot's own action chunk. EvoScene-VLA updates a compact geometric prior along the generated action sequence, so the next VLM call can correct an already-advanced scene state.

Temporal memory and action-conditioned prediction. Temporal VLAs retain observed history across calls. MemoryVLA [35] keeps a summary-token memory, HAMLET [16] and HiF-VLA [22] use history-aware attention, Embodied-SlotSSM [7] applies slot state-space models, TraceVLA [50] prompts the VLM with visual traces, and AVA-VLA [46] aligns action and visual history. Related approaches learn trajectory representations from video [42, 12] or transfer across heterogeneous demonstrations [38]. Other methods predict future states from actions. FLARE [51] inserts learnable future tokens into the VLM, UP-VLA [48] predicts action-conditioned images or features, world models [44, 11, 43, 5] roll out latent futures, and video-prediction policies [8, 3, 15] generate or condition on pixel-level futures. These methods record history or predict a future for the current decision. They do not keep a self-correcting prior across chunk boundaries. EvoScene-VLA persists an action-updated scene prior in the recurrent scene prefix and corrects it with the next observation.

Flow matching and diffusion policies. Flow-matching and diffusion policies model robot control as trajectory denoising. Diffusion Policy [6] popularized diffusion-based continuous control. π_0 [2], RDT [25], Octo [29], and flow-matching VLAs [23, 26] scale this formulation to pretrained VLAs. Consistency Policy [31] accelerates sampling. These methods denoise actions 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

EvoScene-VLA is built on LingBot-VLA [45], an open-source $\pi_{0.5}$-style [30] policy with a Qwen2.5-VL [33] backbone. At each control call, a VLM prefix encodes the multi-view images x_t and the language instruction ℓ into a key–value cache; a separate action expert then denoises an action chunk from that cache with flow matching, and the chunk is executed open-loop before the next VLM call. This interface is stateless across chunks: the prefix is recomputed at every call, so the policy re-perceives the scene without recording how its own actions changed it.

3.1 Overview

As illustrated in Fig. 2, our framework processes spatial-temporal contexts via four key stages. First, we construct a Recurrent Scene Prefix and refine the inherited prior using current observations (Sec. 3.2). Next, we advance this updated state to forecast downstream actions (Sec. 3.3). To guide feature learning, we incorporate a Geometric Anchor and a Scene Predictor during training (Sec. 3.4), and finally synthesize these components into a unified objective function (Sec. 3.5).

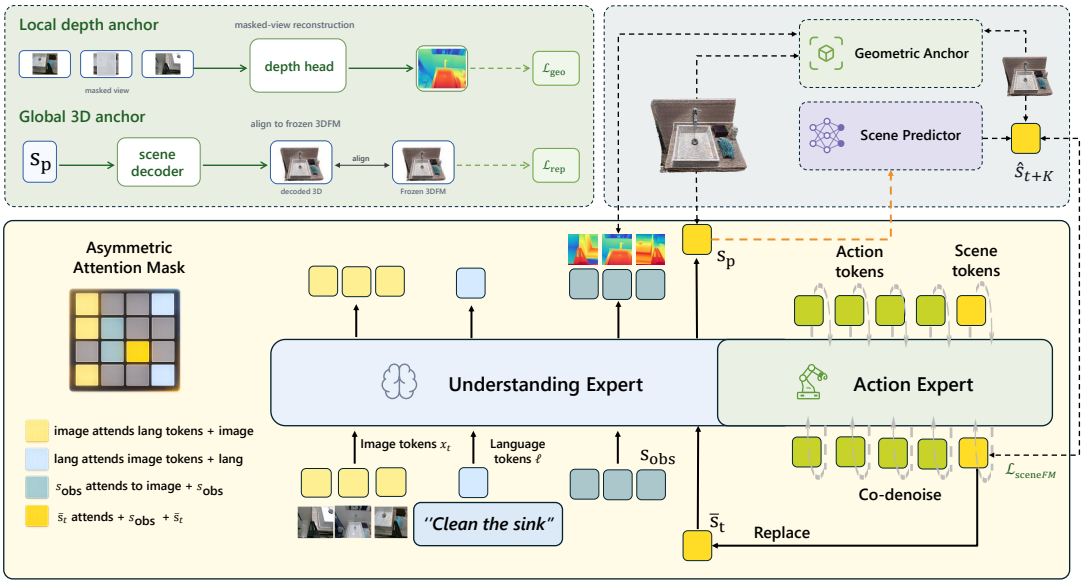


Figure 2: **Pipeline Overview.** The policy receives multi-view images, an instruction, and the robot state. The VLM prefix contains image and language tokens, per-view observation slots, and recurrent prior slots. An asymmetric attention mask lets observation slots read the current views and lets prior slots absorb this evidence while preserving the pretrained image and language pathways. During training, Geometric Anchor grounds the scene slots with Monocular Depth Teacher and 3D-foundation-model features, and Scene Predictor supplies future scene-token targets to train the action expert. At inference, the action expert denoises the actions and matched scene tokens together in the flow-matching sampling. The scene token at the executed step becomes the prior for the next VLM call, so each action chunk starts from an action-updated, observation-corrected scene prior.

3.2 Recurrent Scene Prefix

Let D denote the VLM hidden dimension, V the fixed camera views, N_{obs} the slots in each per-view observation group, and N_{prior} the slots in the prior group. The observation slots $s_{\text{obs}}^{(v)} \in \mathbb{R}^{N_{\text{obs}} \times D}$, $v \in \{1, \ldots, V\}$, gather geometric evidence from each camera, while the prior slots $\bar{s}_t \in \mathbb{R}^{N_{\text{prior}} \times D}$ inherit the action-updated scene state from the previous chunk. Together with the multi-view image input x_t and language instruction ℓ, the prefix is ordered $[\, x_t,\ \ell,\ s_{\text{obs}}^{(1:V)},\ \bar{s}_t \,]$. At the first chunk, $\bar{s}_t$ is initialized from learnable scene embeddings; for later chunks, it carries the recurrent state denoised by the action expert at the previous step. The VLM thus consumes a prior when one exists and falls back to learned embeddings otherwise.

An asymmetric attention mask routes information through this prefix as illustrated in Fig. 2. Image and language tokens ignore scene slots, preserving pretrained visual and linguistic paths. Each observation slot attends to its own view's image tokens and to other slots in its own observation group. Prior slots attend to all observation slots and to themselves, but not to image or language tokens, and no other token attends back to them. Current image evidence therefore reaches the prior slots only through the observation slots. This isolation keeps scene slots out of the pretrained vision–language pathway while giving the scene state a dedicated bottleneck. Let $s_p \in \mathbb{R}^{N_{\text{prior}} \times D}$ denote the VLM output at the prior-slot, which we take as the corrected scene representation for the current chunk:

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

The representation s_p is the recurrent scene state. The action expert advances and writes it back, while training supervision specifies its content.

3.3 Action-Conditioned Scene Update

The action expert carries the recurrence. At deployment, each chunk runs one VLM forward followed by the same Euler denoising loop as LingBot-VLA, which emits the action chunk together with K scene states at sparse key-frame offsets $\{k_1, \ldots, k_K\} \subseteq \{1, \ldots, H\}$. The robot executes the action chunk, and the scene state at the final offset k_K becomes the prior for the next chunk:

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

where $\hat{z}^{(k_K)}$ is the denoised scene state at offset k_K. The next VLM call corrects this scene prior against the new observation, closing the loop. This recurrence defines deployment. The remaining method explains how training grounds these state updates and distills them into the action expert.

3.4 Training-Only Supervision

The recurrent prefix specifies where the scene state lives but not what it should encode; without supervision, the slots are free to absorb generic image features rather than 3D structure. The two-level Geometric Anchor and a Scene Predictor close this gap. Both modules are discarded after training.

Cross-view 3D-feature supervision. The **Global 3D anchor** grounds the cross-view representation s_p in 3D structure by aligning it with frozen multi-view 3DFM features [41]. A lightweight view-conditioned decoder $g_{3\mathrm{D}}$ takes a set of learnable queries q_{dec} and uses s_p from Eq. (1) as keys and values; a linear projector W_{proj} then maps the decoder output to the foundation-model feature space:

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

We supervise P_t to match the frozen foundation model's features on the same multi-view input via an ℓ_1 loss on per-token layer-normalized features:

$$\mathcal{L}_{\mathrm{rep}} = \frac{1}{V}\sum_{v=1}^{V} \big\| \mathrm{LN}\big(P_t^{(v)}\big) - \mathrm{LN}\big(Z_t^{(v)}\big) \big\|_1, \qquad Z_t = \mathrm{3DFM}(x_t). \tag{4}$$

Here LN is a parameter-free normalization along the feature dimension of each token, so the objective constrains the direction and relative shape of each token's feature rather than absolute magnitude.

Per-view depth supervision. The global anchor directly supervises the aggregated state s_p, but does not provide an explicit geometric target for each per-view observation slot $s_{\mathrm{obs}}^{(v)}$. We therefore add a **Local depth anchor** that grounds each observation slot with target-view depth supervision. For each view v_i, we mask its dense image tokens and use a lightweight cross-attention head g_{depth} to predict its depth features from $s_{\mathrm{obs}}^{(v_i)}$, s_p, and the remaining views:

$$\hat{f}_{t,i}^{\,d} = g_{\mathrm{depth}}\Big(q_{\mathrm{tmpl}},\; \big[\, \tilde{h}_{t,v_1}^{\mathrm{img},(i)}, \ldots, \tilde{h}_{t,v_V}^{\mathrm{img},(i)},\; s_{\mathrm{obs}}^{(v_i)},\; s_p \,\big]\Big), \quad \tilde{h}_{t,v_j}^{\mathrm{img},(i)} = \begin{cases} m, & j = i, \\ h_{t,v_j}^{\mathrm{img}}, & j \neq i. \end{cases} \tag{5}$$

Here q_{tmpl} inherits from LingBot-VLA's query tokens and m is a learned mask embedding. Masking removes the dense target-image shortcut, forcing target-view geometry through the scene-slot bottleneck while the remaining views provide cross-view context. We supervise each prediction with a frozen Monocular Depth Teacher (MDT) [37] applied to the unmasked target image:

$$\mathcal{L}_{\mathrm{geo}} = \frac{1}{V}\sum_{i=1}^{V} \mathrm{SmoothL1}\Big(\hat{f}_{t,i}^{\,d},\, f_{t,i}^{\,d}\Big)\,, \qquad f_{t,i}^{\,d} = \mathrm{MDT}(x_{t,v_i}). \tag{6}$$

Future scene targets. The **Scene Predictor** produces future targets $\hat{s}$ for joint action and scene flow matching. Let H denote the final action offset of the chunk. The module is a causal Transformer that takes as input the robot state r_t, the current scene representation s_p, the action sequence $a_{t:t+H}$, and K key-frame query groups $q_1, \ldots, q_K$ initialized from s_p, arranged as $[\, r_t,\; s_p,\; a_t, \ldots, a_{t+H},\; q_1, \ldots, q_K \,]$. Under a causal mask, each query group q_i attends to r_t, s_p, the action prefix $a_{t:t+k_i}$ up to its target step, and earlier query groups, so that the prediction at step $t + k_i$ is conditioned only on actions executed up to that step. The output is a sequence of future scene latents $\hat{s}_{t+k_1:t+k_K}$, with each $\hat{s}_{t+k_i} \in \mathbb{R}^{N_{\mathrm{prior}} \times D}$.

This module reuses the view-conditioned decoder (g_{3D}, W_{proj}) from Eq. (3): the same operator that grounds s_p also decodes each predicted future latent and matches it to foundation-model features on the corresponding future multi-view frame. Let $Z_{t+k_i} = 3DFM(x_{t+k_i})$ denote the future-frame teacher features:

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

Joint action and scene flow matching. With the future targets now defined, training couples the action and scene paths under a single flow matching vector field, so the same denoising loop learns to emit both outputs. We stack the future scene targets $\hat{s}_{t+k_1:t+k_K}$ and apply LayerNorm without affine parameters over the feature dimension to obtain standardized targets $z_0 = \mathrm{LayerNorm}(\hat{s}_{t+k_1:t+k_K}) \in \mathbb{R}^{K \times N_{prior} \times D}$. LayerNorm gives each scene token zero feature mean and unit feature variance, so the scene path can use $\epsilon_s \sim \mathcal{N}(0, I)$ directly, without additional rescaling. We then sample flow matching time $\tau \in [0, 1]$ shared between the action and scene paths, draw independent standard Gaussian noises ϵ_a and ϵ_s, and form linear interpolants

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

The action expert receives the suffix $[\, r_t \mid z^{\tau} \mid a^{\tau}_{t:t+H} \,]$ under a causal suffix mask and attends to the VLM prefix cache, which holds image, language, observation, and prior slots. It predicts a velocity v_θ for every token in the action and scene blocks. We stop gradients through both targets, so velocity matching uses

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

$\mathcal{L}_{actFM}$ is the standard $\pi_{0.5}$ action FM loss. $\mathcal{L}_{sceneFM}$ distills future scene representations into the action expert, which lets the same denoising pass replace the predictor at inference.

3.5 Training Objective

The full training objective combines the action flow-matching loss with the four scene-grounding and scene-transfer terms introduced above:

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

The four scene-side losses play three roles: $\mathcal{L}_{geo}$ and $\mathcal{L}_{rep}$ ground current scene representations in geometry, $\mathcal{L}_{pred}$ trains future representations in the same coordinate, and $\mathcal{L}_{sceneFM}$ transfers those representations into the action expert. We train end-to-end in a single stage. All numerical settings, including the loss weights, the slot counts N_{obs} and N_{prior}, the camera views V, the number of key frames K, the final action offset H, and the number of Euler steps, are reported in Appendix A.

4 Experiments

4.1 Setup

Benchmarks and baselines. We evaluate EvoScene-VLA on RoboTwin [28], LIBERO [24], 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 [45] 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 reports success rates on the 31 RoboTwin tasks. EvoScene-VLA raises the LingBot-VLA* average from 86.4 to 88.8 under *Clean* (+2.4) and from 85.7 to 87.9 under

Table 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. 7.**

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† [49]	70.7	71.1	70	70	**100**	**100**	79	71	96	95	46	48	6	10	92	88	40	61	93	**100**
$\pi^{\dagger}_{0.5}$ [30]	81.2	75.9	60	39	99	66	34	77	92	93	80	79	91	76	**96**	**93**	62	54	90	96
Motus† [1]	87.0	85.8	66	68	**100**	**100**	**95**	**91**	**99**	97	**88**	71	91	**95**	95	88	**84**	**78**	95	91
RotVLA† [19]	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

Rand (+2.2), and it attains the higher mean on 22 of 31 tasks under *Clean* and 23 of 31 under *Rand*. Against the † reference rows, it exceeds Motus (87.0/85.8) and is on par with the strongest, RotVLA (88.6/87.3). The *Rand* setting introduces additional layout variation. Such variation can make the scene harder to perceive from a single observation and allow perception errors to accumulate across chunks. Nevertheless, the gain remains broadly distributed across tasks and does not materially shrink under *Rand*. This result is consistent with the recurrent scene prefix mitigating these effects.

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). Because chunked baselines plan the entire chunk from the start-state observation, they lack any view of the intermediate state that the chunk itself produces: in *Open Microwave*, the gripper retracts before reaching the door handle. EvoScene-VLA, whose action expert co-denoises a scene update alongside the action chunk, conditions on this intermediate scene state and 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.

The benefit also shows in motion quality. Fig. 4 plots end-effector trajectories in 3D for two representative episodes; EvoScene-VLA produces noticeably smoother paths than LingBot-VLA. Co-denoising ties each predicted action to a consistent scene context, so the action expert avoids the abrupt corrections of action-only chunked baselines. Additional tasks are shown in Appendix H.

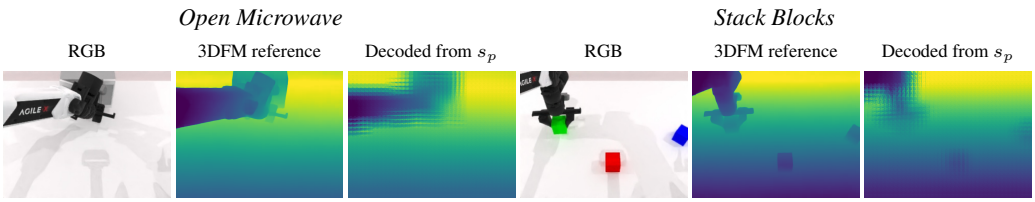


Figure 3: **Geometry decoded from the recurrent scene representation.** *3DFM reference* decodes the frozen targets Z_t; *Decoded from* s_p applies the same frozen point head to P_t produced from s_p. Each triplet shares a reference-fitted colour range. The decoded maps retain coarse relative geometry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

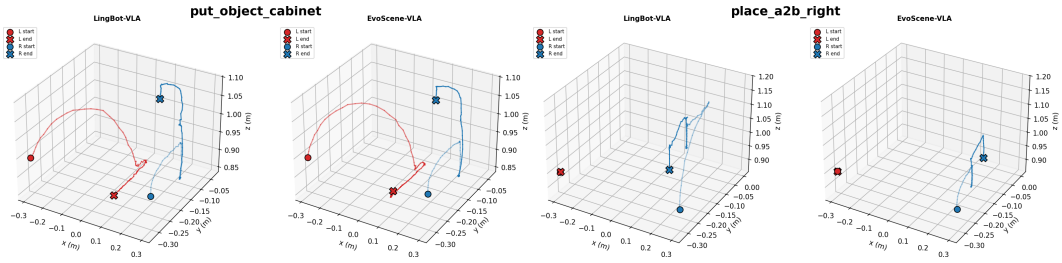


Figure 4: **End-effector trajectories in 3D.** Two representative RoboTwin episodes comparing LingBot-VLA with EvoScene-VLA. Left- and right-arm paths run from start (●) to end (✘); ours are smoother.

4.3 Ablations

Tab. 2 reports additive ablations on RoboTwin-5Task. The first two rows show the LingBot-VLA baselines. Both lack our Geometric Anchor, Scene Predictor, and recurrent prior. The first step adds the global anchor ($\mathcal{L}_{\text{rep}}$) and Scene Predictor ($\mathcal{L}_{\text{pred}}$) jointly, since the two share the view-conditioned decoder and only become useful together. The local depth anchor ($\mathcal{L}_{\text{geo}}$) further improves performance by grounding each observation slot in per-view geometry, complementing the cross-view supervision from the global anchor. Finally, propagating the recurrent prior across chunks at inference, rather than reinitializing $\bar{s}_t$ from learnable embeddings at every chunk, further improves performance and isolates the contribution of cross-chunk recurrence. By contrast, Tab. 4 organizes the same three components on LIBERO as independent configurations rather than cumulative additions. With the recurrent prior disabled, scene-only and local-only improve overall success from the matched baseline of 82.75% to 83.35% (+0.60 percentage points) and 83.60% (+0.85 percentage points), respectively.

Table 2: **Ablation study.** Success rates averaged over the RoboTwin-5Task subset.

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
+ recurrent prior-on	**90.8**	**87.8**

Table 3: **Real-robot experiments.** Success rates (%) on Galaxea indoor-cleaning tasks.

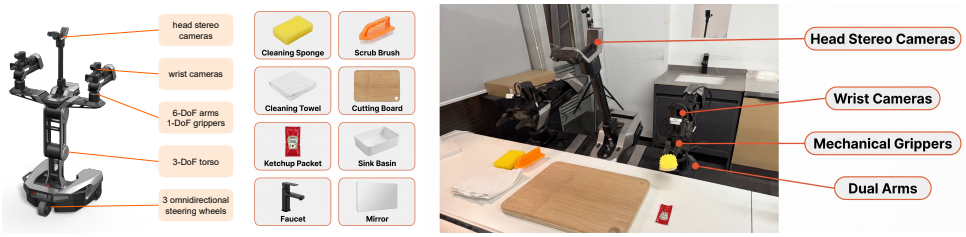
Task	$\pi_{0.5}$	LingBot VLA	LingBot VLA*	Ours
Mirror	28	27	26	29
Sink	42	44	49	51
Cutting-board	44	34	37	46
Avg.↑(%)	38.0	35.0	37.3	42.0

Table 4: **Full factorial LIBERO component ablations.** Success rates across four suites, each with 10 tasks and 500 rollouts (2,000 total). ✓/× denote enabled/disabled components; Δ is the overall change from the baseline. The global 3DFM anchor ($\mathcal{L}_{\text{pred}}$/$\mathcal{L}_{\text{rep}}$) directly supervises the recurrent state: without it, enabling the recurrent prior falls below the baseline (-2.15), and $\mathcal{L}_{\text{geo}}$ alone does not recover it (-1.47).

Method	**Components**			**LIBERO suite**				**Overall**	**Δ**
	$\mathcal{L}_{\text{pred}}$ $\mathcal{L}_{\text{rep}}$	$\mathcal{L}_{\text{geo}}$	Recurrent prior-on	**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
	×	×	✓	72.3	89.2	86.1	74.8	80.60	-2.15
	✓	✓	×	79.4	93.4	90.2	74.4	84.35	+1.60
	✓	×	✓	79.2	91.9	90.4	76.8	84.58	+1.83
	×	✓	✓	72.5	90.2	87.3	75.1	81.28	-1.47
	✓	✓	✓	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 dual-arm platform (Fig. 5(a)), training on the indoor-cleaning subset of the Galaxea Open-World Dataset [13] (mirror, sink, and cutting-board cleaning). Following the standard VLA interface, the policy uses one head stream and two wrist streams as its three-view input. The subset spans 7 recording sessions, 439 episodes, 48,419 frames, and 1,756 video clips, totaling approximately 9 hours of bimanual demonstrations at 15 fps. We use all demonstrations for training and evaluate on the physical robot.

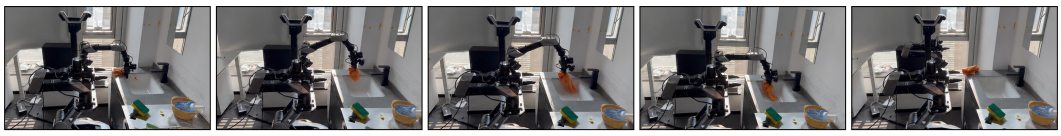


(a) Robot platform. (b) Real-robot experimental setup.

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. Unlike in simulation, we train and deploy four methods for the real-robot comparison under the same single-stage schedule: $\pi_{0.5}$, the two LingBot-VLA baselines, and EvoScene-VLA. We run all inference on a single NVIDIA RTX 4090 GPU. We evaluate one fixed checkpoint per 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 must target unwiped regions that can be subtle, occluded, or ambiguous in the current view. Tab. 3 reports success rates on the Galaxea cleaning tasks. EvoScene-VLA improves average success from 37.3% to 42.0%. These gains show the same pattern as simulation: the method helps when the robot changes the scene as it acts. Each method uses one fixed checkpoint, and 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.



Initial Grasp cloth Enter basin Wipe surface Return

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 tokens at different key frames in one flow-matching pass. The scene token at the executed step 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. On 31 RoboTwin tasks, EvoScene-VLA lifts average success from 86.4 to 88.8 under clean evaluation and from 85.7 to 87.9 under randomized initial conditions. Real-robot cleaning trials on Galaxea R1-Lite show the same pattern: the robot's own actions evolve the scene state. 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

[1] 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.

[2] 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*, 2024.

[3] 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)*, 2024.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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 *International Conference on Machine Learning (ICML)*, 2023.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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*, 2025.

[19] 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*, 2026. URL `https://arxiv.org/abs/2605.13403`.

[20] 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)*, 2025.

[21] 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*, 2025.

[22] 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.

[23] 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.

[24] 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)*, 2023.

[25] 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.

[26] 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)*, 2023.

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

[28] 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.

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

[30] 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.

[31] 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.

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

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

[34] Robbyant Team. LingBot-Depth: Masked depth modeling for spatial perception. `https://huggingface.co/robbyant/lingbot-depth`, 2026. Open-source RGB-D representation model used as the frozen depth teacher.

[35] 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.

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

[37] 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.

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

[39] 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.

[40] 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)*, 2024.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

Overview

This appendix supplements the main paper with implementation, evaluation, and discussion details. Appendix A reports the optimization recipe, hyperparameters (Tab. 5), and frozen-teacher pipeline. Appendix B (Tab. 6) 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. 7), expanding Tab. 1 of the main paper, and Appendix F reports 95% Wilson binomial confidence intervals for the real-robot study (Tab. 8). 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 the decoded depth and isolates what the recurrent prior contributes beyond the current observation (Fig. 10, Tab. 9, Tab. 10). We close with limitations (Appendix K) and broader impact (Appendix L).

A Implementation

Optimization recipe. Following LingBot-VLA's recipe (Tab. 5), 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. 5 are tuned on RoboTwin-5Task and reused on the full RoboTwin and Galaxea evaluations without further sweeping.

Table 5: 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 [37] (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 [41], applied to the current-frame multi-view input for $\mathcal{L}_{\text{rep}}$ and to the future-frame multi-view input for $\mathcal{L}_{\text{pred}}$.

B Notation

Tab. 6 consolidates the symbols used throughout Secs. 3.2 to 3.4.

Table 6: 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, $k_i \in \{1, \ldots, H\}$
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
$\tilde{P}_{t+k_i},\ Z_{t+k_i}$	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).

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. (6)
$P_t \leftarrow W_{\text{proj}}\, g_{\text{3D}}(q_{\text{dec}};\, s_p)$ ▷ 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: Current observation x_t, instruction ℓ, robot state r_t, prior buffer $\bar{s}_t$ (learnable embedding for the first chunk).
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 at the prior-slot positions; cache the 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 [28], 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 all 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 [1] and the RotVLA values from RotVLA [19]. Those models are jointly trained on all 50 RoboTwin 2.0 tasks with 50 clean and 500 randomized demonstrations per task, so we select the 31 tasks used here from the source tables and report their unweighted mean.

E Per-Task Multi-Seed Results

Tab. 7 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.

F Real-Robot Rollout Uncertainty

Each real-robot method is deployed from a single fixed checkpoint, so cross-seed or cross-checkpoint standard deviation is not defined for the study in Sec. 4.4. We therefore quantify the remaining source of variability, namely the finite number of physical rollouts, with 95% Wilson binomial confidence intervals. Tab. 8 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*.

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.

H Additional Trajectory Visualizations

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

Table 7: **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 ± 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. Tasks appear in the same order as in Tab. 1.

	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

Table 8: **Real-robot rollout uncertainty.** For every method–task pair we report the success rate (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 for one fixed checkpoint and do not represent cross-seed variation.

	$\pi_{0.5}$		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]

I Decoding Geometry from the Scene Representation on RoboTwin

The global anchor in Sec. 3.4 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.

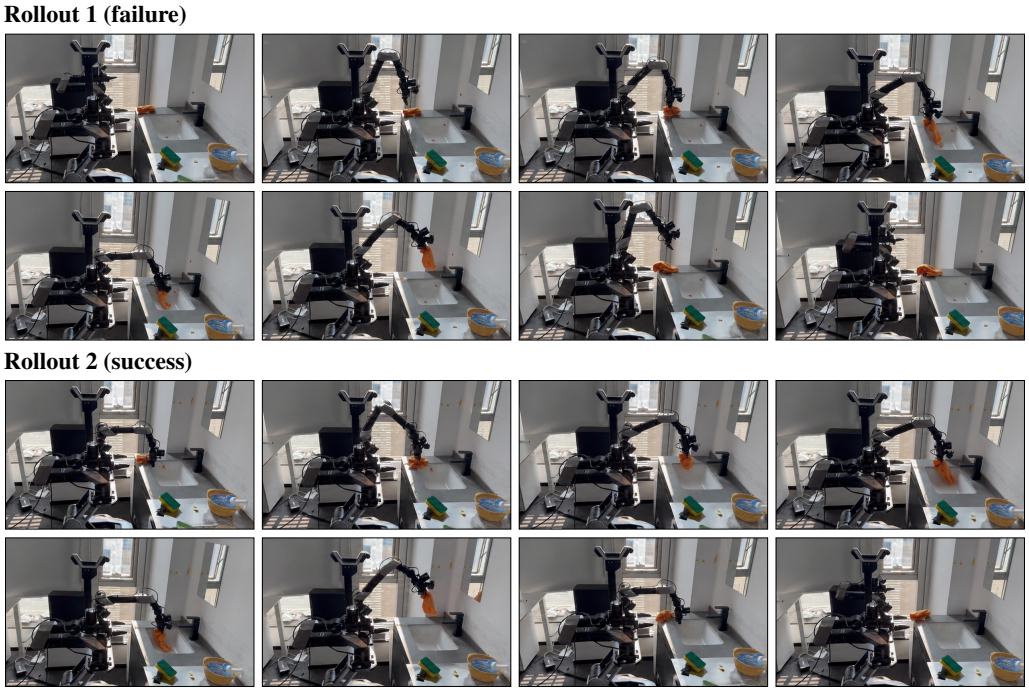


Figure 7: **Real-robot closed-loop rollouts on *Clean The Sink*.** Two executions of EvoScene-VLA on Galaxea R1-Lite, read left to right and top to bottom within each rollout. The robot picks up the cloth, transports it to the basin, and wipes across the surface. Rollout 1 fails with visible ketchup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

J Decoding Depth from the Scene Representation

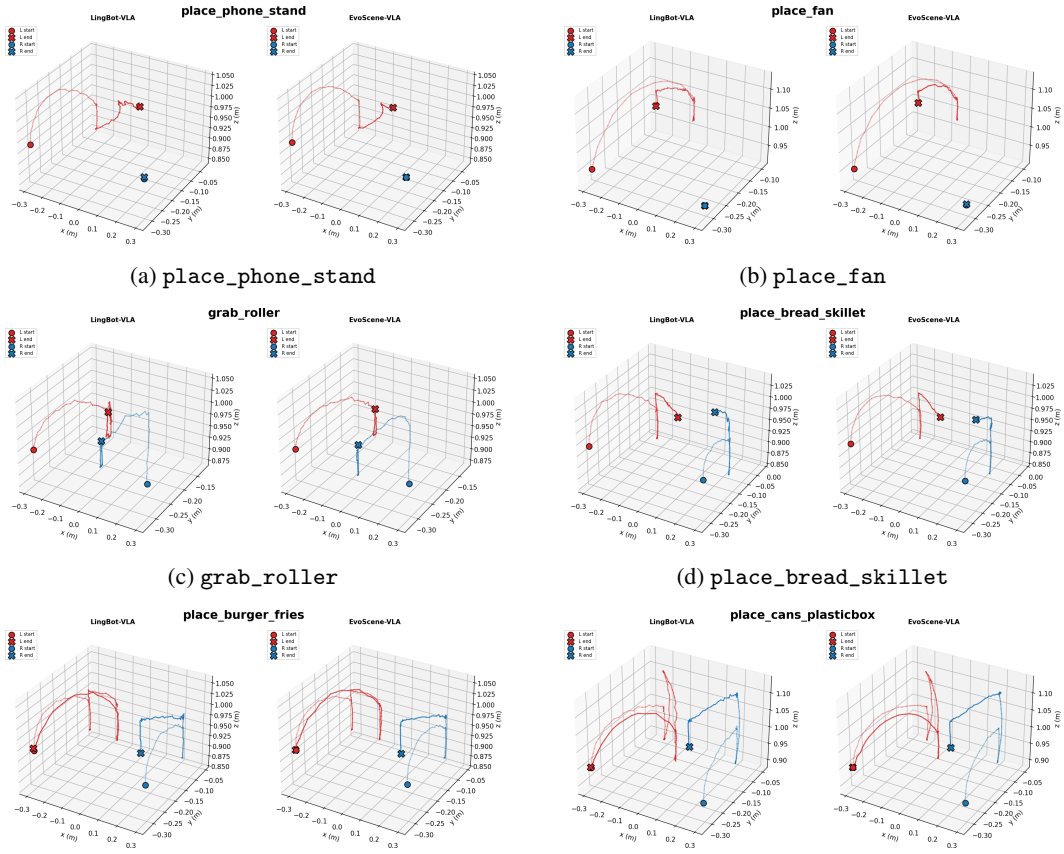
Appendix I provides qualitative RoboTwin decodes. This section quantifies the same question on LIBERO. Because P_t is trained to match Z_t, it already lives in the frozen 3DFM's feature space, so that model's own point head decodes it to depth without fitting anything on our data. A favourable score 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 501 warm-history LIBERO-10 samples drawn from 167 episodes (1002 images over two views), disjoint by seed from any diagnostic used during development. $\mathcal{L}_{\text{rep}}$ compares *layer-normalized* features (Eq. (4)), so training never constrains the absolute feature scale: the readout's feature norm is $22.1\times$ the teacher's. Raw decoded depth is consequently on the wrong scale, and we align each prediction to the reference with a least-squares scale and shift in log depth before scoring. All numbers below are therefore claims about *relative* geometry, not metric depth.

Table 9: Depth decoded from the scene representation, scored against the frozen 3DFM reference over 1002 images after log-depth scale-shift alignment. *Constant* predicts each image's median reference depth and bounds what is achievable with no geometric content.

Readout source	AbsRel ↓	δ_1 ↑	δ_2 ↑	Pearson ↑
Recurrent prior	0.130	**0.828**	**0.968**	**0.879**
Cold prior	0.130	0.824	0.963	0.872
Constant baseline	0.360	0.389	0.816	0.000

The representation carries geometry: δ_1 is 0.825 against 0.389 for the constant baseline, and AbsRel is 0.130 against 0.360. 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.



(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 produces noticeably smoother paths than LingBot-VLA across all six cases.

What the recurrent prior contributes. Tab. 9 also shows that carrying the prior across chunks buys almost nothing when every view is visible: query-warm beats the cold prior by $+0.0013$ δ_1 (episode-balanced paired margin over 167 episodes, 108W/59L, $p < 10^{-5}$) — statistically detectable but negligible in size. This is expected, because the current frame already determines the current frame's geometry.

We therefore isolate the prior's *content* by replacing each sample's incoming prior with one captured from a different episode, and repeat the comparison with one camera view replaced by another episode's frame so that view's geometry is absent from the observation. Depth is then scored on the replaced view only.

Table 10: Own versus cross-episode donor prior, episode-balanced paired δ_1 margins over 167 episodes. The prior's content is irrelevant while the scene is fully observed and becomes measurable once a view is removed.

Observation	$\Delta\delta_1$ (own − donor)	Episodes won	p
All views visible	$+0.0003$	$90/167$	0.218
Agent view replaced	$+0.0027$	$110/167$	$< 10^{-5}$
Wrist view replaced	$+0.0022$	$104/167$	$< 10^{-5}$

The dissociation is clean: swapping in another episode's prior changes nothing under full observation ($p = 0.218$) and costs measurable accuracy once a view is missing ($p < 10^{-5}$ for both views, comfortably past a Bonferroni threshold over the twelve paired tests we ran). The magnitude is nonetheless small — removing the agent view costs 0.206 δ_1, of which the prior's content restores

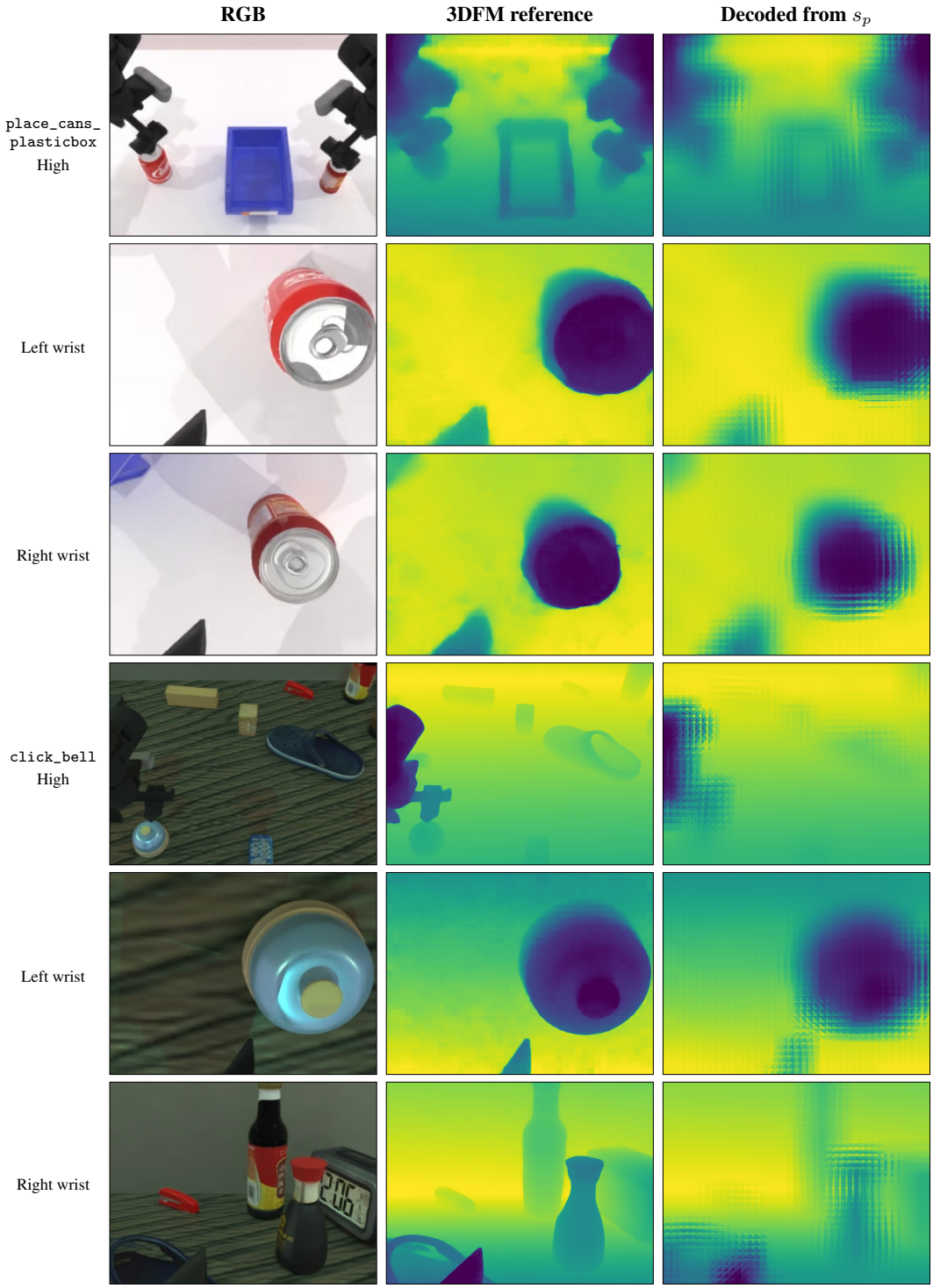


Figure 9: **Geometry decoded from the recurrent scene representation.** Two RoboTwin frames with three camera views each. *3DFM reference* decodes the frozen features Z_t, while *Decoded from* s_p applies the same frozen point head to P_t produced from s_p. Each pair shares a colour range fitted on its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

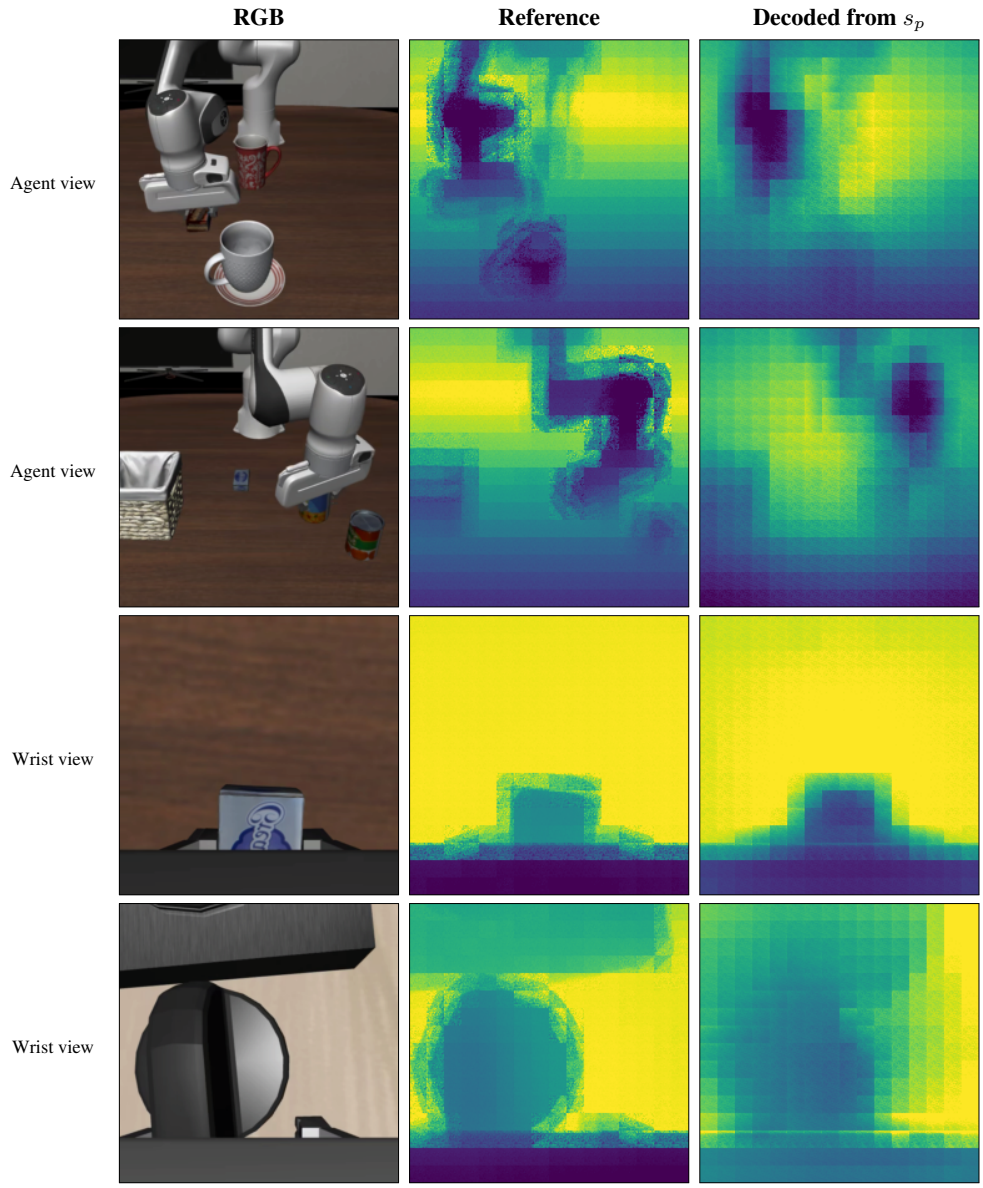


Figure 10: **Depth decoded from the scene representation.** Four LIBERO images covering two agent views and two wrist views. *Reference* decodes the frozen 3DFM target; *Decoded* decodes P_t through the same frozen head. Each row shares a colour range fitted on its reference.

0.0027, or 1.3%; the wrist view restores 2.2%. We report this as directional evidence that the recurrent state carries episode-specific geometry, not as a claim that it substitutes for observation. Replaying each sample's own prior through the explicit intervention path reproduces the query-warm readout exactly (maximum absolute error 0.0), so the comparison is not confounded by the intervention itself.

K Limitations

EvoScene-VLA carries a latent recurrent state across chunk boundaries, and its geometric content is only partially legible. Decoding it through the frozen 3DFM head recovers relative structure but not absolute scale, and only at the resolution of a 16×16 token grid (Appendix J); the prior's own content contributes measurably only when a camera view is missing, and even then restores a small fraction of what the missing view provided. The value of the prior must therefore still be judged primarily

through downstream behavior and through the controlled ablation in Tab. 2. Future-scene supervision flows through a frozen 3D foundation model on future frames, so target quality upper-bounds the prior the action expert can learn. We train Scene Predictor and the action expert's scene branch only at a fixed set of key-frame offsets $\{k_1, \ldots, k_K\}$. Deployments that re-observe at intervals diverging from these offsets therefore receive a temporally misaligned prior at the next chunk. Real-robot evaluation is currently limited to three indoor-cleaning tasks on a single dual-arm platform; behavior under category shift, novel scenes, and multi-step task chaining remains future work. The effect of chunk length on prior quality is likewise unresolved: longer chunks may raise the value of recurrence by creating larger scene changes, but they also push Scene Predictor targets farther into the future, and we have not separated these two effects under a fixed training budget. A useful next step is to treat the mismatch between observation slots and prior slots as an uncertainty signal for replanning or adaptive chunk execution.

L Broader Impact

EvoScene-VLA is a manipulation policy aimed at indoor household assistance. Robots that run such policies inside homes raise standard considerations around physical safety, user privacy from on-robot camera streams, and potential displacement of routine paid labor. These considerations apply to vision–language–action policies broadly and are not specific to the recurrent scene prefix introduced here. All training data in this work come from public datasets or platform-provided demonstrations under their respective licenses; real-robot evaluation is conducted in a controlled lab setting only.