JOINT MULTI-VIEW LATENT ACTIONS FOR ROBUST VISION-LANGUAGE-ACTION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A latent action model (LAM) maps a pair of video frames to a discrete latent action that a vision-language-action (VLA) policy learns to predict, so that the policy can learn from videos without action labels. Existing LAMs infer the latent action, mostly, from one camera, whose frames record the physical change only as that camera projects it, together with camera-specific content; the latent action can therefore lose part of the change and carry that content instead. JMV-LA infers the latent action from all cameras a recording provides: a shared transition encoder summarizes every available camera's frame pair, learned queries read all summaries before quantization, and one shared decoder must predict every camera's next features from the single latent action and that camera's first frame, so that the latent action has to supply the change in a form that serves every projection. The VLA predicts it from the primary camera alone and needs no additional camera at deployment. Compared with LAMs learned from the primary camera only, JMV-LA yields lower action-decoding error and a readout that is more stable under image perturbations. The single-camera VLA supervised by JMV-LA achieves the highest average success among the compared policies on the four LIBERO suites, LIBERO-10 camera-pose perturbations, and LIBERO-Pro, as well as the highest observed success in six of seven ALOHA real-robot evaluation settings.

1 INTRODUCTION

A vision-language-action (VLA) policy (Kim et al., 2024; Black et al., 2025) can learn from videos that carry no action labels if a latent action model (LAM) first maps each pair of frames to a latent action for the policy to predict (Ye et al., 2025; Chen et al., 2025; Bu et al., 2025). The quality of that target shapes what the videos can teach, and representative LAMs infer it from one camera (Ye et al., 2025; Bu et al., 2025). A single-camera transition offers only one projection of the physical change: motion toward the camera may produce little image-plane displacement, and occlusion may hide an object's movement. It also contains background and appearance information specific to the viewpoint. Future-observation prediction alone does not ensure that the latent action retains action-relevant change rather than incidental visual information (Zhang et al., 2025; Nikulin et al., 2025). We seek a latent action that captures the robot's underlying physical change while reducing dependence on camera-specific appearance.

Open robot datasets provide synchronized views of the same physical transition (O'Neill et al., 2023; Walke et al., 2023). Their complementary motion cues can reveal changes that are ambiguous or occluded in a single view. We jointly encode all available camera transitions before quantization, forming the latent action from their combined evidence. A shared decoder predicts each camera's next-frame features from this latent action and that camera's first-frame features. The first-frame features supply appearance and scene context, while reconstructing all views through one bottleneck encourages the latent action to capture the underlying physical change. This design aims to retain more action-relevant information in the latent representation and thereby provide better supervision for downstream VLA learning.

We build this into JMV-LA (Fig. 1). A shared transition encoder, told whether a camera is external or wrist-mounted, summarizes every available frame pair; four learned queries read all summaries at once before quantization, so that a sample with one, two or three cameras yields the same four-token latent action; and then one shared decoder must predict every camera's next features from that

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.

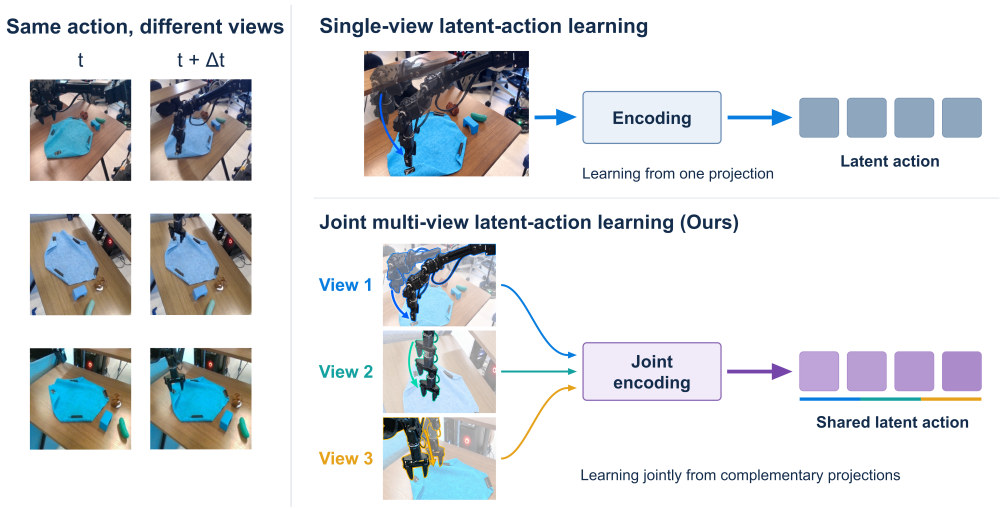

Figure 1: Single-camera and joint multi-camera latent action learning. Synchronized frame pairs record the same physical change from different cameras. JMV-LA reads all of them before quantization into one shared four-token latent action, from which every camera's next features are predicted.

latent action and the camera's own first frame. The policy predicts the latent action from the primary camera alone and needs no other camera at deployment. Compared with a primary-only LAM matched in data, LAM training settings, and bottleneck capacity, JMV-LA reduces action-decoding error by 11.56% and preserves a more stable readout under image perturbations. For downstream control, JMV-LA supervised policy achieves 96.35% average success on the four LIBERO suites, 54.76% mean success under LIBERO-10 camera-pose perturbations, and 41.94% on LIBERO-Pro, the highest averages among the compared policies. It also attains the highest observed success in six of seven ALOHA real-robot settings. Section 4 evaluates both the quality of the learned latent actions and the performance of policies trained with this supervision.

2 Related Work

2.1 Latent Action Learning from Videos

Latent action models infer compact transition variables from video, enabling policy learning without action labels for every transition. LAPA (Ye et al., 2025) and Moto (Chen et al., 2025) use discrete transition codes for policy pretraining and subsequently ground the learned representations in robot actions. UniVLA (Bu et al., 2025) uses language and pretrained visual features to learn task-centric latent actions for cross-embodiment policy learning. These methods establish video-derived targets as an interface between representation learning and control.

The quality of such targets depends on which visual changes they encode. Linear-model analysis shows how latent-action learning can capture exogenous variation (e.g., visual shortcuts) alongside controllable change (Zhang et al., 2025). LAOM (Nikulin et al., 2025) finds that, by adding robot action readout from the learned latent representation, one can improve the downstream control performance beyond the frame reconstruction objective. These results show that the learned latent action representation should encode more of the underlying action/change applied to the physical scene, rather than focusing on visual quality.

Other approaches improve latent actions through alignment or structure. JALA (Luo et al., 2026) replaces full visual reconstruction with alignment between predictive embeddings and inverse-dynamics latents, adding masked motion-token prediction when hand-motion annotations are available. CLAP (Zhang et al., 2026) learns an executable action-token vocabulary with Act-VAE and contrastively aligns visual transition latents with representations of robot action trajectories. Moreover, ALAM (Tang et al., 2026) regularizes latent transitions through composition and reversal con-

sistency. While these techniques improve the utility of the limited information used to form the action latent, we focus on a framework that can consume multiple sources of training signals recorded of the same interaction to derive one shared predictive target for policy learning.

2.2 Viewpoint Robustness and Multi-View Learning

Viewpoint robustness for a policy has been obtained by giving the policy geometric information or by changing its observations. Camera conditioning (Jiang et al., 2025) supplies the camera extrinsics as per-pixel ray embeddings, and Adapt3R (Wilcox et al., 2025) builds a 3D scene representation from calibrated RGB-D observations; VISTA (Tian et al., 2025) augments the training data with synthesized viewpoints. These approaches use camera geometry, depth or view synthesis at training or deployment. Our LAM instead uses synchronized RGB transitions without calibration, depth or synthesized views, while the policy retains a single-camera input.

Multiple cameras have entered robot learning in three roles. As policy inputs, Look Closer (Jangir et al., 2022) fuses an egocentric and a third-person camera with cross-view attention, which gives the policy more evidence at deployment but requires both cameras on the deployed robot. As supervision for a representation, Time-Contrastive Networks (Sermanet et al., 2018) align simultaneous observations across viewpoints using metric learning; the Multi-View Masked World Models framework (Seo et al., 2023) reconstructs masked multi-view observations and transfers the representation to single-view control. These methods show that multi-view observations can help improve policy performance, yet our work investigates the value of multi-view observations in learning effective and aligned action representations.

As supervision for a latent action, VILA (Jeong et al., 2026) aligns latents across viewpoints with ground-truth action sequences and transfers the resulting policy as a visual encoder, and MVP-LAM (Lee et al., 2026) infers each latent from one camera's frame pair and reconstructs another camera's next-frame features. Our latent action differs in where the cameras enter: all available cameras are read jointly before quantization, so one latent action is inferred with every projection in view, and that one latent action must explain every camera's next features. This gives the bottleneck access to complementary evidence that may be missing from a single projection. LAM training requires no action labels, and the deployed policy requires no additional camera.

3 Method

We learn latent-action targets from any-view robot videos without action labels for training a single-camera VLA. First, a LAM maps frame pairs a short interval apart to discrete latent actions. The VLA then predicts these targets from the primary camera, available in every video. This prediction task shapes the VLA's features, which an action decoder uses, to produce quality robot actions. Section 3.1 formulates the problem, Sections 3.2–3.4 describe the LAM (Fig. 2a; Appendix B), and Section 3.5 covers policy learning and control (Fig. 2b).

3.1 Problem Formulation

Let $t \in \mathbb{N}$ index the frames of a video and $v \in \{1, \dots, V\}$ its cameras, where V is the largest number of cameras that any video in the training data provides ($V = 3$, App. D); the set of cameras available at frame t is $\mathcal{V}_t \subseteq \{1, \dots, V\}$, which is never empty. Let $O_t^v \in \mathbb{R}^{h \times w \times 3}$ be the RGB frame of camera v at t, with $h \times w$ the image resolution, and $O_{t+\Delta}^v$ the frame of the same camera at $t + \Delta$, where the interval $\Delta \in \mathbb{N}$ is selected according to a dataset-specific sampling rule (App. D); the frame pair $X_t^v = (O_t^v, O_{t+\Delta}^v) \in (\mathbb{R}^{h \times w \times 3})^2$ is the transition observed by camera v. Between the two frames the scene changes physically: the robot moves, together with any object it manipulates. The two frames record this change only as their camera projects it, and they record, in addition, content that is not part of the change: the background and the appearance of the robot, both specific to the camera. A latent action learned from one camera can therefore lose part of the change of the underlying scene, and nothing in a single-camera objective prevents it from spending its capacity on camera-specific content instead (Nikulin et al., 2025; Zhang et al., 2025). Our goal is one latent action per transition, $Z_t^q \in \mathbb{R}^{4 \times d_z}$, whose four tokens are each drawn from a codebook of $K = 16$ learned vectors in $\mathbb{R}^{d_z}$, so that it is equivalently the index sequence $Y_t^{\text{lat}} \in \{1, \dots, K\}^4$. The latent

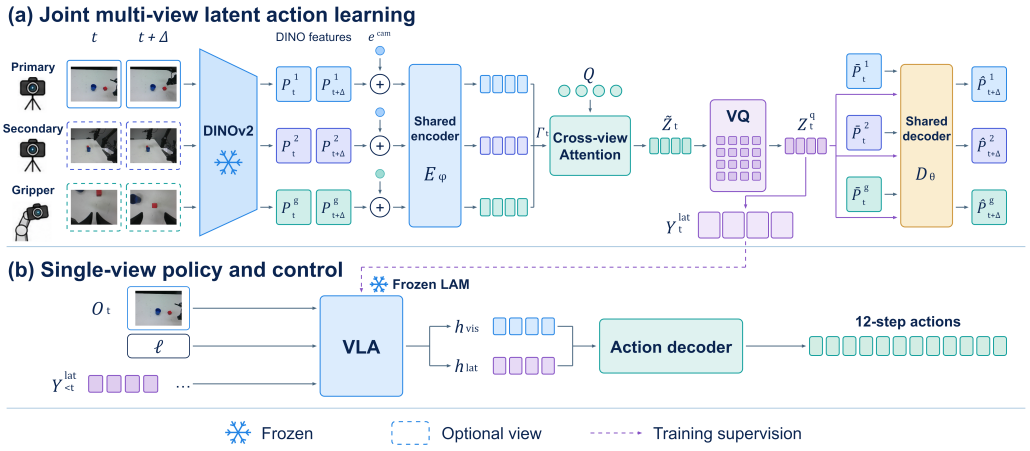

Figure 2: Method overview. (a) Latent action learning. A frozen image encoder maps the two frames of every available camera to patch features, a camera-type embedding marks each camera as third-view or wrist-mounted, and one shared transition encoder E_ϕ produces four transition tokens per camera. Four learned queries read the tokens of all available cameras at once, with missing cameras masked by Γ_t, and the fused tokens $\widetilde{Z}_t$ are quantized into the latent action Z_t^q with code indices Y_t^{lat}. One shared decoder D_θ predicts the future patch features of every available camera from Z_t^q and that camera's own first-frame features. (b) Policy learning and control. The frozen latent action model supplies the latent-action targets; the VLA receives the primary frame, the instruction and the latent actions of the preceding steps, and the action decoder maps its hidden states at the image tokens (h_{vis}) and at the four latent-action tokens (h_{lat}) to a chunk of continuous robot actions.

action should carry the underlying physical change regardless of the view, and the VLA should be able to predict it from the primary camera before action decoding.

Consider, for example, a gripper that reaches forward to a cube. In the front camera the gripper recedes and shrinks by a few pixels; in a camera at the side it sweeps across the image; in the wrist camera the cube grows while the table passes beneath. Synchronized cameras thus provide what a single camera lacks: a part of the change that one camera loses can be evident in another. These views offer complementary evidence of the same physical change, although they also share scene content. We therefore design the LAM to read all available cameras before it infers the latent action, so that the inference has every projection in view (Secs. 3.2–3.3), and we require a single shared decoder to predict what every camera records next from that latent action and the camera's own first frame, so that the latent action is trained to explain the change for every camera while the camera-specific content is available from the first frame (Sec. 3.4; App. B.1 gives the information-theoretic motivation). The additional cameras are required only while the latent actions are learned and the training videos are labeled with them; the VLA (Sec. 3.5) never observes them.

3.2 View-Aware Transition Encoding

A frozen pretrained DINOv2 (Oquab et al., 2024) encoder Φ maps each frame O_τ^v, $\tau \in \{t, t+\Delta\}$, to N_p patch features, providing fixed reconstruction targets (Sec. 3.4). Following OpenVLA (Kim et al., 2024), cameras occupy three slots: primary and secondary are fixed third-view cameras; the gripper camera is wrist-mounted and observes the end effector's surroundings at close range. Hand motion shifts the entire scene in its frames. Without camera type, the encoder must infer whether a shifting table reflects table or hand motion. We therefore add a learned embedding $e^{\text{cam}}_{\rho(v)}$ to every patch, where $\rho(v) \in \{\text{third}, \text{wrist}\}$ identifies the camera type. Embeddings are shared by type, so no encoder parameter is specific to an individual training camera. With camera type supplied, a shared transition encoder E_ϕ maps both feature maps to $L = 4$ tokens $H_t^v \in \mathbb{R}^{4\times d_z}$, matching the latent-action length predicted by the VLA:

$$\overline{P}_\tau^v = \Phi(O_\tau^v) + e^{\text{cam}}_{\rho(v)}, \qquad H_t^v = E_\phi(\overline{P}_t^v, \overline{P}_{t+\Delta}^v). \tag{1}$$

Each H_t^v is a continuous summary of the change observed by one camera; no camera is quantized on its own (the architecture is given in App. B.2).

3.3 Joint Cross-View Latent Fusion

One to three cameras must yield a target of fixed length, and they must be read jointly rather than quantized one at a time; averaging their tokens would weight a camera that occludes the motion as heavily as one that reveals it. Four learned queries $Q \in \mathbb{R}^{L \times d_z}$ therefore read the tokens of all cameras at once through a fixed-size attention bottleneck. The tokens of the V slots are concatenated into $H_t^{\mathrm{mv}} \in \mathbb{R}^{VL \times d_z}$, and the queries attend to them:

$$\widetilde{Z}_t = \mathrm{LN}[Q + \mathrm{Attn}(QW_Q, H_t^{\mathrm{mv}} W_K, H_t^{\mathrm{mv}} W_V; \Gamma_t)]\,. \tag{2}$$

Here Attn is softmax attention, W_Q, W_K and W_V are learned projections, LN is layer normalization, and the additive mask Γ_t is $-\infty$ on the tokens of a missing camera and 0 elsewhere, so that a missing camera contributes nothing. Queries attend to all available cameras to extract, as much as possible, the underlying scene-change information, before quantization. Even with a single camera, the module can still process it (mask and attention details in App. B.3) to produce an aggregated representation $\widetilde{Z}_t$ consisting of four tokens regardless of the number of cameras.

3.4 Shared Quantized Bottleneck and Multi-View Reconstruction

The VLA will predict the latent action as tokens, and a finite vocabulary also fixes the capacity that forces compression. A vector quantizer (van den Oord et al., 2017) therefore replaces each of the four fused tokens of $\widetilde{Z}_t$ with the nearest of the K codebook vectors, which are shared across the four positions: the quantized tokens form the latent action Z_t^q, their indices $k_{t,\ell}$ form $Y_t^{\mathrm{lat}} = (k_{t,1}, \ldots, k_{t,4})$, and the capacity is $4 \log_2 K = 16$ bits. A decoder D_θ, shared by all cameras so that a single latent action must serve every one of them, by predicting the future feature map $\widehat{P}_{t+\Delta}^v$ of every available camera from Z_t^q and that camera's own first-frame features $\overline{P}_t^v$. The reconstruction loss is the squared error averaged over every patch of available cameras in a mini-batch $\mathcal{B}$:

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{N_{\mathrm{valid}} N_p d} \sum_{t \in \mathcal{B}} \sum_{v \in \mathcal{V}_t} \|\widehat{P}_{t+\Delta}^v - \overline{P}_{t+\Delta}^v\|_F^2, \tag{3}$$

where $N_{\mathrm{valid}} = \sum_{t \in \mathcal{B}} |\mathcal{V}_t|$ is the number of available cameras in the batch, each feature map consists of N_p patches of width d, and $\|\cdot\|_F^2$ denotes the sum of squared entries. Because D_θ already receives the first frame of camera v, everything that persists in that camera is available to it without the latent action, so the latent action only has to supply the change; and because a single latent action serves every camera, it must ensure that the prediction is faithful in the front, side and wrist views. Therefore, the objective does not need to explicitly penalize memorizing camera-specific content. Section 4.2 tests whether the resulting representation reduces the impact of image-space perturbations in specific camera views on action decoding. The codebook and commitment losses and the complete training procedure are given in Apps. B.4 and C.

3.5 Single-View Policy Learning and Control

After latent action learning, we freeze the LAM and label every transition of the training videos with the latent action computed from all of its available cameras:

$$Y_t^{\mathrm{lat}} = \mathrm{LAM}(\{X_t^v\}_{v \in \mathcal{V}_t}). \tag{4}$$

Policy learning then uses the primary camera alone for deployment flexibility. For example, a robot is not required to carry the synchronized cameras of the training data, whereas a single third-view camera is sufficient for deployment on most platforms. More explicitly, the multi-view information is used where it is already available, in the training videos, and it reaches the VLA through its target rather than through its inputs, so the VLA retains the single-camera interface of existing VLA policies with no added inference cost. Specifically, during policy learning with robot-action-labeled demonstrations (App. C), a vision-language model receives the primary camera frames and the language instruction, and a prediction head on its features is trained with a cross-entropy loss to

output the four indices. This prediction task is what allows the VLA to learn from unlabeled video. To obtain executable actions, an decoder reads the VLA's shaped features and outputs a chunk of continuous robot actions with an action-regression loss; and the latent actions of the preceding steps are fed back to the VLA as history. For action prediction, neither the LAM nor D_θ is needed. The losses and the decoding equations are given in App. B.5. Section 4 evaluates action information, readout stability under image perturbations, and single-camera policy success, including robustness to camera changes.

4 EXPERIMENTS

This section evaluates JMV-LA at both the representation and policy levels. At the representation level, Section 4.2 examines whether jointly encoding the available camera transitions produces latent actions that retain more action-relevant information than a primary-view-only counterpart, and whether their action readout remains more stable under image perturbations. We assess these properties using controlled action-decoding probes. At the policy level, Sections 4.3 and 4.4 evaluate JMV-LA supervised policy against existing latent action policies in closed loop across the four LIBERO suites (Liu et al., 2023) and under LIBERO-10 camera-pose perturbations. Sections 4.5 and 4.6 then compare the JMV-LA policy with existing methods under broader distribution shifts, on LIBERO-Pro and on a real robot.

4.1 EXPERIMENTAL SETUP

Data and training. We train the latent action model on a video mixture comprising 26 Open X-Embodiment manipulation datasets (O'Neill et al., 2023) and three GNM navigation datasets. Camera availability varies across datasets: in the effective training distribution, 49.90% of the samples contain only the primary view, while the remainder contain two or three views. After training the LAM for 200k steps, we freeze it and infer discrete latent-action tokens for each transition in the same video mixture. These tokens serve as supervision for pretraining a Prismatic-7B VLA (Karamcheti et al., 2024). For each downstream benchmark, an action decoder is subsequently trained on the benchmark demonstrations to predict continuous 12-step action chunks. Data, architectures and optimization settings are given in Appendices D and C.

Baselines. We compare direct-action policies (OpenVLA, OpenVLA-OFT-SV, SpatialVLA, and NORA-1.5) with latent-action-based policies (LAPA, JALA-SV, OpenCLAP-SV, MVP-LAM, and UniVLA). All receive a single third-person RGB observation at deployment. We use published or official LIBERO results, except for LAPA, whose reproduction is reported by UniVLA, and OpenCLAP-SV, which we reproduce. For representation analysis, a primary-only LAM shares JMV-LA's data, encoder, bottleneck, codebook, and LAM training settings, but removes auxiliary views and cross-view fusion and reconstructs only primary-view features. Details can be found in Appendices E.1 and A.

4.2 ACTION DECODABILITY AND READOUT STABILITY OF THE LATENT ACTIONS

Action decodability. This experiment tests whether joint multi-view latent actions retain more decodable action-related information. We freeze our LAM and the primary-only LAM and train identical MLP probes to decode normalized 12-step action chunks from their four code indices. Evaluation uses a trajectory-held-out split of LIBERO-Spatial and five probe seeds (Appendix E.1). Table 1 shows that the JMV-LA latent action reduces overall MAE by 11.56%, with translation and gripper MAE decreasing by 19.89% and 25.11%, only with a modest rise in rotation by 4.18%. These results suggests that joint multi-view latent learning makes action-relevant information more accessible, particularly for translation and gripper state.

Image-space perturbations. We next examine whether the action information encoded by JMV-LA remains accessible under image perturbations. Probes trained only on clean (unperturbed) data are evaluated on the same held-out transitions after five transformations: horizontal and vertical shifts in either direction and a 10° counterclockwise rotation. Each transformation is applied consistently to both frames of a transition and either to the primary view alone or to all views read by the LAM (Appendix E.1). When all input views are perturbed, JMV-LA's MAE increases by only

Table 1: Action decodability on LIBERO-Spatial using probing networks trained and tested with trajectory-held-out splits. Mean ± sample standard deviation over five MLP-probe seeds. Translation, rotation and gripper MAE summarize normalized 12-step action chunks; R^2_{global} uses the global target mean. **Bold** denotes the best result in each column.

LAM	MAE ↓	Trans. MAE ↓	Rot. MAE ↓	R^2_{global} ↑	Gripper MAE ↓
Primary-only	0.2695 ± 0.0004	0.2683 ± 0.0005	$\mathbf{0.2429 \pm 0.0006}$	0.4441 ± 0.0024	0.3525 ± 0.0022
JMV-LA	$\mathbf{0.2383 \pm 0.0017}$	$\mathbf{0.2149 \pm 0.0032}$	0.2531 ± 0.0007	$\mathbf{0.5342 \pm 0.0062}$	$\mathbf{0.2640 \pm 0.0017}$

Table 2: Action-readout stability under image-space perturbations on LIBERO-Spatial. MAEs use normalized 12-step action chunks; values are mean ± sample standard deviation over five clean-trained probe seeds. ΔMAE is computed per seed relative to clean inputs. Values are rounded to four decimal places; relative changes use unrounded values. **Bold** denotes the best result for each metric under all-camera perturbations.

Perturbed cameras	LAM	Clean MAE ↓	Perturbed MAE ↓	ΔMAE ↓
Primary only	JMV-LA	0.2383 ± 0.0017	0.2371 ± 0.0009	-0.0012 ± 0.0009
All	Primary-only	0.2695 ± 0.0004	0.2758 ± 0.0006	$+0.0064 \pm 0.0004$
	JMV-LA	$\mathbf{0.2383 \pm 0.0017}$	$\mathbf{0.2405 \pm 0.0014}$	$\mathbf{+0.0022 \pm 0.0008}$

0.0022, whereas the primary-only LAM's MAE increases by 0.0064. Thus, the absolute MAE increase is 65.32% smaller for JMV-LA. Normalized by each model's own clean MAE, these changes correspond to relative increases of 0.93% for JMV-LA and 2.37% for the primary-only LAM. When only the primary view is perturbed, JMV-LA's readout remains close to its clean performance (Table 2). This suggests that JMV-LA preserves a more stable action readout and carries more action-relevant information even when its input frames are perturbed.

4.3 Main Results on LIBERO

We next evaluate the policy trained with JMV-LA supervision on the standard LIBERO Spatial, Object, Goal and Long (LIBERO-10) suites. We train one policy per suite and perform 50 rollouts per task, reporting success averaged over the 10 tasks in each suite (Appendix A.1). As shown in Table 3, JMV-LA achieves the highest average success among the compared single-view policies, reaching 96.35%, and the best LIBERO-Long result of 94.2%. These exceed the strongest baseline average by 1.05 percentage points and the next-best Long result by 2.2 points. JMV-LA also leads the latent-action group on Spatial and Object. These results support the effectiveness of JMV-LA supervision for downstream control in the standard LIBERO setting. We next introduce controlled camera-pose perturbations to assess robustness under viewpoint changes.

4.4 Robustness to Camera Viewpoint Perturbations

Camera-pose perturbations directly test the viewpoint dependence motivating our design by changing the visual projection while preserving the task and physical scene. We perturb the LIBERO-10 third-person `agentview` camera during inference, using translations of up to ± 5 or ± 10 cm along each axis, rotations of up to $\pm 5^\circ$ or $\pm 10^\circ$ about each axis, and a combined ± 5 cm/$\pm 5^\circ$ condition. Each sampled pose remains fixed throughout an episode, and we run 500 rollouts per condition, reporting mean perturbed success and retention relative to clean performance (Appendix A.2). Pose sampling and rollout settings are given in Appendix E.2. JMV-LA achieves the highest success under all five perturbations, with 54.76% mean perturbed success and 58.13% retention (Table 4). Compared with MVP-LAM, the strongest baseline under perturbations, its mean perturbed success is 8.56 percentage points higher, exceeding the 3.4-point gap under the clean camera in the same evaluation. These results demonstrate stronger camera-shift performance for the policy trained with JMV-LA supervision. Together with the representation analyses in Section 4.2, they support its

Table 3: Success rates (%) on the four LIBERO suites. † denotes previously reported results, ‡ the LAPA result reproduced by UniVLA, and § our single-view OpenCLAP reproduction. **Bold** denotes the best result and underline the second best.

Method	Spatial	Object	Goal	Long	Avg.
Direct-action VLA policies					
OpenVLA†	84.7	88.4	79.2	53.7	76.5
OpenVLA-OFT-SV† (Kim et al., 2025)	96.2	**98.3**	96.2	90.7	95.3
SpatialVLA† (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1
NORA-1.5† (Hung et al., 2025)	**97.3**	96.4	94.5	89.6	94.5
Latent-action VLA policies					
LAPA‡	73.8	74.6	58.8	55.4	65.7
JALA-SV†	90.4	96.4	95.2	87.2	92.3
OpenCLAP-SV§ (Zhang et al., 2026)	95.4	97.2	**97.4**	87.0	94.3
MVP-LAM†	96.0	94.6	94.8	90.8	94.1
UniVLA†	96.5	96.8	95.6	92.0	95.2
JMV-LA	97.2	98.0	96.0	**94.2**	**96.35**

Table 4: Success rates (%) under third-person camera-extrinsic perturbations on LIBERO-10. All compared policies observe the single `agentview` camera. Pert. Avg. is the mean over the five perturbation conditions, and Retention is its ratio to clean success. † denotes the official MVP-LAM policy checkpoint and § our single-view OpenCLAP reproduction. **Bold** denotes the best result and underline the second best.

		Translation		Rotation				
Method	Clean	±5 cm	±10 cm	±5°	±10°	Combo	Pert. Avg.	Retention
UniVLA	92.6	59.4	19.0	46.8	10.4	31.6	33.44	36.11
OpenCLAP-SV§	87.0	48.2	21.0	43.8	15.6	36.6	33.04	37.98
MVP-LAM†	90.8	66.6	28.6	66.2	23.6	46.0	46.20	50.88
JMV-LA	**94.2**	**74.8**	**38.8**	**72.0**	**30.8**	**57.4**	**54.76**	**58.13**

utility for action readout and robust closed-loop control. We next examine generalization beyond camera-pose changes.

4.5 Generalization under Broader LIBERO-Pro Shifts

To test whether the benefits extend beyond camera-pose changes, we evaluate on LIBERO-Pro (Zhou et al., 2025), which introduces object, position, semantic, task, and environment shifts. Each category is averaged equally over the four LIBERO suites, and the overall score is the unweighted mean across the five categories. We compare JMV-LA with existing latent-action methods; per-suite results are provided in Appendix E.3. JMV-LA reaches the highest overall score, 41.94% (Table 5) and leads under object, position, and task shifts. Its largest gains over UniVLA are 8.05 points under task shifts and 6.75 under object shifts. OpenCLAP-SV leads under semantic and environment shifts. Position shifts remain difficult for every method, with JMV-LA highest at 5.45%. These results suggest that the advantage of JMV-LA extends beyond viewpoint changes, particularly to object, position and task shifts.

4.6 Real-Robot Experiments

Finally, we examine whether these benefits transfer to physical robot manipulation. We evaluate three ALOHA (Zhao et al., 2023) manipulation tasks: cube-to-cup insertion under an in-distribution layout and a spatial out-of-distribution (OOD) layout, multi-stage drawer manipulation under three visibility conditions, and tape-to-stand hanging under a support-layout shift. All policies receive the same high-camera image and instruction at deployment. We report strict end-to-end success with trial counts. Hardware, training and evaluation protocols, task configurations, and qualitative failure

Table 5: Closed-loop success rates (%) on LIBERO-Pro. Each perturbation result is the unweighted mean over LIBERO-Spatial, Object, Goal and LIBERO-10; Avg. is the unweighted mean over the five perturbation categories. All policies observe a single third-person RGB camera and receive the language instruction at deployment. § denotes our single-view OpenCLAP reproduction. **Bold** denotes the best result and underline the second best.

Method	Object	Position	Semantic	Task	Environment	Avg.
UniVLA	69.50	2.70	89.05	11.75	15.15	37.63
OpenCLAP-SV§	69.75	1.55	**94.75**	4.35	**23.90**	38.86
MVP-LAM	69.25	1.10	84.45	11.05	16.20	36.41
JMV-LA	**76.25**	**5.45**	91.30	**19.80**	16.90	**41.94**

Table 6: End-to-end success on the ALOHA real-robot tasks. Counts and percentages are both shown because trial counts differ by setting. All policies receive only `cam_high` and the language instruction at deployment. The corresponding 95% Wilson score intervals are reported in Table 11. **Bold** denotes the best result.

Task	Evaluation setting	UniVLA	MVP-LAM	JMV-LA
Cube-to-cup	In-distribution	$11/16$ (68.8%)	$11/16$ (68.8%)	$\mathbf{14/16}$ (**87.5%**)
	Spatial OOD	$0/10$ (0.0%)	$3/10$ (30.0%)	$\mathbf{9/10}$ (**90.0%**)
Drawer	No occlusion	$11/24$ (45.8%)	$13/24$ (54.2%)	$\mathbf{17/24}$ (**70.8%**)
	Behind basket	$5/12$ (41.7%)	$5/12$ (41.7%)	$\mathbf{9/12}$ (**75.0%**)
	Inside basket	$10/27$ (37.0%)	$9/27$ (33.3%)	$\mathbf{15/27}$ (**55.6%**)
Tape-to-stand	In-distribution	$\mathbf{19/20}$ (**95.0%**)	$16/20$ (80.0%)	$18/20$ (90.0%)
	Support-layout OOD	$7/20$ (35.0%)	$10/20$ (50.0%)	$\mathbf{14/20}$ (**70.0%**)

observations are given in Appendix F. JMV-LA achieves the highest observed success in six of the seven settings: both cube-to-cup layouts, all three drawer conditions and the tape-to-stand support-layout shift (Table 6). It completes $9/10$ spatial-OOD insertions, against $3/10$ for MVP-LAM and $0/10$ for UniVLA, and reaches 70.0% under the support-layout shift, where its success drops by 20 percentage points against 30 for MVP-LAM and 60 for UniVLA, the policy with the highest in-distribution success on that task. JMV-LA achieves the highest success among the compared methods under the spatial and support-layout shifts and across all three drawer visibility conditions, including settings that require locating displaced or partially occluded targets from a single camera.

5 Discussions

We show that latent actions inferred jointly from all available cameras come with lower action probing errors than those inferred from the primary camera alone, suggesting that they retain more action-relevant information. Action readout is also more stable under image perturbations. At the policy level, the single-camera VLA trained with JMV-LA supervision achieves the highest average success among the compared policies on LIBERO, LIBERO-10 camera-pose perturbations, and LIBERO-Pro. It also achieves the highest success rates in six of seven real-robot evaluation settings.

The results are consistent with the design motivation: each camera's first frame provides appearance and scene context, while the latent action inferred jointly from all available views, whether one or several, is trained to explain the scene changes from observed views. These findings support using multi-camera recordings to improve latent-action supervision for VLA learning while retaining the policy's observational input interface, thus, no additional cameras or LAM execution are required at deployment. However, due to missing views in existing datasets, our framework may still under-utilize the meaningful interaction information in single-view demonstrations compared to when full views are available. In the future, we plan to leverage view synthesis modules to further promote the efficiency of latent action learning, as well as quantifying how the cross-view synthesis quality relates to the effectiveness of learned latent action representations.

AI USE STATEMENT

AI USE STATEMENT

In this work, we used generative AI tools to assist with code generation for implementing our methods, translation, and qualitative and thematic data analysis. We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, write proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat datasets, or interpret results. Generating synthetic datasets is not applicable to this work. Additionally, we used generative AI tools to help edit images, suggest experimental parameters for tuning, and search for information.

We have reviewed all AI-assisted work. The authors verified and tested the AI-generated code for correctness. We also reviewed the translations, qualitative and thematic analyses, image edits, and parameter suggestions, and manually verified the authenticity of the cited literature. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read the ICLR Code of Ethics, and this research was conducted in accordance with its principles. All datasets were used in compliance with their respective licenses and usage conditions. Demonstration collection and real-robot evaluations were conducted with safety precautions, including emergency-stop mechanisms during robot operation. We conduct appropriate safety assessment and safeguards during all experiments. The authors declare no conflicts of interest. Planned releases of code, model weights, and data will respect applicable licenses and privacy considerations.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 and Appendix B describe the proposed method and its detailed formulation. Appendix C documents the model architecture, training procedures, and hyperparameters, while Appendix D specifies the training data mixture and processing steps. Section 4 and Appendices A and E describe the benchmark evaluations, baseline provenance, diagnostic experiments, and camera-perturbation protocols. Appendix F provides the real-robot setup, demonstration counts, training and evaluation protocols, task configurations, and success criteria. We plan to publicly release the code, model weights, and collected demonstration data to facilitate independent reproduction and further research, subject to applicable licensing and privacy requirements.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19752–19763, 2025.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck. In *International Conference on Learning Representations*, 2020.

Noriaki Hirose, Dhruv Shah, Ajay Sridhar, and Sergey Levine. SACSoN: Scalable autonomous control for social navigation. *IEEE Robotics and Automation Letters*, 9(1):49–56, 2024. doi: 10.1109/LRA.2023.3329626.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Chia-Yu Hung, Navonil Majumder, Haoyuan Deng, Liu Renhang, Yankang Ang, Amir Zadeh, Chuan Li, Dorien Herremans, Ziwei Wang, and Soujanya Poria. NORA-1.5: A vision-language-action model trained using world model- and action-based preference rewards. *arXiv preprint arXiv:2511.14659*, 2025.

Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *arXiv preprint arXiv:2201.07779*, 2022.

Youngjoon Jeong, Junha Chun, and Taesup Kim. Learning to act robustly with view-invariant latent actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6781–6790, 2026.

Tianchong Jiang, Jingtian Ji, Xiangshan Tan, Jiading Fang, Anand Bhattad, Vitor Guizilini, and Matthew R. Walter. Do you know where your camera is? view-invariant policy learning with camera conditioning. *arXiv preprint arXiv:2510.02268*, 2025.

Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *IEEE International Conference on Robotics and Automation*, pp. 5129–5136, 2018.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the design space of visually-conditioned language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23123–23144, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Jung Min Lee, Dohyeok Lee, Seokhun Ju, Taehyun Cho, Jin Woo Koo, Li Zhao, Sangwoo Hong, and Jungwoo Lee. MVP-LAM: Learning action-centric latent action via cross-viewpoint reconstruction. *arXiv preprint arXiv:2602.03668*, 2026.

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*, volume 36, 2023.

Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable VLA pretraining in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35047–35058, 2026.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46427–46447, 2025.

Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, et al. Open X-embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Younggyo Seo, Junsu Kim, Stephen James, Kimin Lee, Jinwoo Shin, and Pieter Abbeel. Multi-view masked world models for visual robotic manipulation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30613–30632, 2023.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, 2018.

Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 674–684, 2022.

Zuojin Tang, Haoyun Liu, Xinyuan Chang, Changjie Wu, Dongjie Huo, Yandan Yang, Bin Liu, Zhejia Cai, Feng Xiong, Mu Xu, Jiachen Luo, De Ma, Zhiheng Ma, and Gang Pan. ALAM: Algebraically consistent latent action model for vision-language-action models. *arXiv preprint arXiv:2605.10819*, 2026.

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

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint arXiv:physics/0004057*, 2000.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736, 2023.

Albert Wilcox, Mohamed Ghanem, Masoud Moghani, Pierre Barroso, Benjamin Joffe, and Animesh Garg. Adapt3R: Adaptive 3D scene representation for domain transfer in imitation learning. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1486–1514, 2025.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. CLAP: Contrastive latent action pretraining for learning vision-language-action models from human videos. *arXiv preprint arXiv:2601.04061*, 2026.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A Additional Experimental Details and Diagnostics

A.1 LIBERO Protocol and Baseline Provenance

Benchmark and evaluation protocol. We evaluate on the four suites of the LIBERO benchmark (Liu et al., 2023), which probe distinct forms of manipulation generalization. LIBERO-Spatial varies spatial relationships while retaining similar task semantics; LIBERO-Object varies the manipulated objects; and LIBERO-Goal assigns different language-conditioned goals in related scenes. LIBERO-Long, also known as LIBERO-10, contains long-horizon tasks composed of multiple sequential manipulation stages.

Each suite contains 10 tasks with 50 successful human-teleoperated demonstrations per task. Following the OpenVLA and UniVLA evaluation protocols, we process the demonstrations with the OpenVLA pipeline and train a separate policy for each suite on its demonstrations. Spatial, Object, and Goal policies are trained for 30k steps, while the Long policy is trained for 40k steps, all with a global batch size of 128. The action decoder predicts a 12-step continuous action chunk. We evaluate 50 rollouts per task, yielding 500 trials per suite, and report the mean task success rate.

The policy receives only the current primary-camera RGB frame and the language instruction; the latent actions of the preceding steps are supplied as action-history context during training on the demonstrations. The LAM reads the synchronized third-person and wrist cameras to compute the latent-action targets, but the wrist image is never a policy input, so the deployed policy retains the single-camera observation interface.

Baselines. We compare with direct-action and latent-action-based VLA policies. The direct-action group includes OpenVLA, which autoregressively represents discretized robot actions as tokens, and OpenVLA-OFT, which adds parallel decoding, action chunking, and continuous regression. For an input-aligned comparison, we use the single-view OpenVLA-OFT variant rather than the full variant augmented with a wrist image and proprioceptive state. We also include the reported SpatialVLA results and the NORA-1.5 results without DPO post-training (Qu et al., 2025; Hung et al., 2025).

The latent-action group includes LAPA, JALA, MVP-LAM, and UniVLA. LAPA learns discrete latent actions from unlabeled frame transitions; we use the result reproduced in the UniVLA paper. JALA aligns predictive embeddings with inverse-dynamics latents and additionally predicts motion tokens when hand-motion annotations are available; we report its single-view variant. MVP-LAM performs cross-viewpoint reconstruction using a latent inferred independently from one view to predict another. UniVLA extracts four task-centric tokens from a single visual transition. We additionally reproduce OpenCLAP with its downstream policy restricted to a single third-person view, denoted OpenCLAP-SV. Baseline values are taken from the corresponding papers or official project releases unless noted otherwise. These policy comparisons evaluate complete systems with their respective pretraining and downstream training pipelines.

A.2 Camera-Perturbation Metrics and Evaluation Scale

Camera perturbation protocol. For each episode, we independently perturb translation along all three axes by up to ± 5 or ± 10 cm, or local XYZ rotation by up to $\pm 5^\circ$ or $\pm 10^\circ$. The combined condition jointly applies the ± 5 cm and $\pm 5^\circ$ perturbations. We modify the MuJoCo extrinsics of `agentview`; the sampled pose remains fixed throughout the rollout, and the wrist camera is not perturbed.

We run 50 trials for each of the 10 LIBERO-10 tasks under every condition, yielding 500 rollouts per condition and 3,000 per model. All evaluated policies receive only the perturbed third-person image and language. Exact pose sampling, deterministic seeding, environment reset, and rollout settings are provided in Appendix E.2.

Metrics. Besides success under each condition, we report the mean perturbed success

$$\text{Pert. Avg.} = \frac{1}{5} \sum_{c \in \mathcal{C}_{\text{pert}}} \text{SR}_c, \tag{5}$$

Table 7: Token diagnostics under image-space perturbations on LIBERO-Spatial. D_{view} is token disagreement between clean and perturbed versions of the same transition, whereas D_{motion} compares clean tokens from distinct same-task transitions as a non-collapse diagnostic. Values are deterministic statistics of the fixed LAM checkpoints.

Frozen LAM	Perturbed views	D_{view}	D_{motion}
Primary-only	All	0.30685	0.68375
JMV-LA	Primary only	0.14405	0.89231
JMV-LA	All	0.31770	0.89231

and retention relative to clean performance,

$$\text{Retention} = 100 \times \frac{\text{Pert. Avg.}}{\text{SR}_{\text{clean}}}. \tag{6}$$

Retention measures the fraction of nominal success preserved after changing the camera extrinsics.

A.3 Token Diagnostics under Image-Space Perturbations

We complement the action-readout results in Table 2 with token-level diagnostics on the same frozen LAMs and transformed LIBERO-Spatial transitions. The primary-only LAM has one input view, so its result is shared across both perturbation conditions. Full transformation and metric definitions are provided in Appendix E.1.

In Table 7, D_{view} measures the fraction of the four quantized-token positions whose code indices change between clean and perturbed versions of the same transition. Under all-view perturbations, D_{view} is 0.31770 for JMV-LA and 0.30685 for the primary-only LAM, indicating similar token-identity sensitivity. We also train an MLP to predict which of the five image-space transformations was applied from the quantized latent tokens. Perturbation-category accuracies are similarly close (25.24% and 25.18%, respectively, versus a 20% chance level). Thus, the multi-view representation does not exhibit lower token disagreement or reduced perturbation decodability; instead, its advantage lies in preserving a more stable action readout, as reflected by the smaller increase in action MAE. Closed-loop robustness to physical camera-extrinsic changes is evaluated separately in Section 4.4.

B Detailed Method Formulation

This appendix expands the formulation of Section 3: the information-theoretic motivation, the construction of the transition tokens, the attention mask, the quantization losses, the auxiliary latent stream that the main text omits, and the policy-learning objectives. The mutual-information formulation motivates the architecture; training uses the reconstruction and vector-quantization losses.

B.1 Problem Formulation

Physical transitions and multi-view observations. Let $S_t \in \mathcal{S}$ denote the underlying physical state of the robot and its environment at time t. We use $M_t \in \mathcal{M}$ to represent the physical change described in words in Sec. 3.1, the motion of the robot and of whatever it moves, which induces a state transition over a temporal interval Δ:

$$S_{t+\Delta} = \mathcal{T}(S_t, M_t), \tag{7}$$

where $\mathcal{T}$ denotes the underlying system dynamics. During latent action learning, S_t and M_t are not observed; only the frames of the transition are available.

Suppose the same transition is observed by V cameras. The observation from camera v is generated by

$$\begin{aligned} O_t^v &= \mathcal{G}(S_t, C_t^v), \\ O_{t+\Delta}^v &= \mathcal{G}\big(\mathcal{T}(S_t, M_t), C_{t+\Delta}^v\big), \end{aligned} \tag{8}$$

where C_t^v summarizes view-dependent factors, such as camera pose, projection, background, and embodiment-specific appearance. For wrist-mounted cameras, $C_{t+\Delta}^v$ may also vary with the robot motion. We therefore define the visual transition observed from camera v as

$$X_t^v = \left(O_t^v, O_{t+\Delta}^v\right), \tag{9}$$

which generally entangles information about the underlying motion M_t with view-specific factors $C_{t:t+\Delta}^v$.

The joint transition contains at least as much information about M_t as any individual view:

$$\begin{aligned} I(M_t; X_t^1) &\le I(M_t; X_t^{1:V}) \\ &= I(M_t; X_t^1) + \sum_{v=2}^{V} I\left(M_t; X_t^v \mid X_t^{1:v-1}\right). \end{aligned} \tag{10}$$

The conditional terms quantify the motion information contributed by each additional view. They can be strictly positive when a motion component is ambiguous or occluded in one view but observable from another. Thus, although multiple views may contain redundant information, their joint transition cannot contain less information about M_t than any individual view.

The information-bottleneck principle motivates retaining relevant information under a compression constraint (Tishby et al., 2000); multi-view extensions study how shared observations can help identify useful information (Federici et al., 2020). In our setting, cues visible in only one camera can still describe the motion, so we do not equate all unshared information with nuisance. Ideally, a capacity-limited discrete latent Z_t^q should preserve information about the physical motion while discarding information that distinguishes camera views after the motion is known. We express this objective as

$$\begin{aligned} \max_{p(Z_t^q \mid X_t^{1:V})} \quad & I(Z_t^q; M_t) - \lambda_{\text{view}} I\left(Z_t^q; C_{t:t+\Delta}^{1:V} \mid M_t\right), \\ \text{s.t.} \quad & \mathbb{H}(Z_t^q) \le R, \end{aligned} \tag{11}$$

where $C_{t:t+\Delta}^{1:V}$ denotes the collection of view-specific factors, $\mathbb{H}$ is the entropy, and R is the capacity of the discrete bottleneck. The first term encourages Z_t^q to retain action-relevant transition information. The second penalizes information that distinguishes camera views after the underlying motion is already known. Conditioning on M_t expresses the goal of penalizing residual view-specific information rather than motion-related variation. This is a conceptual objective, not a directly optimized training loss: M_t and the view factors are unobserved, and training uses the reconstruction and vector-quantization objectives below.

If Z_t^q is inferred from the joint visual transition, the data-processing inequality gives

$$I(M_t; Z_t^q) \le I(M_t; X_t^{1:V}). \tag{12}$$

Together with Eq. (10), this indicates that jointly encoding synchronized views provides a potentially richer information source for recovering M_t than encoding a single projected transition. This motivates fusing complementary transition evidence from all available views before quantization. This information-theoretic motivation does not imply that the resulting representation explicitly reconstructs or recovers true 3D motion.

The second term in Eq. (11) also motivates how the latent should be decoded. Rather than assigning an independent latent to each camera, we require one shared Z_t^q to predict the future patch features of every available camera from that camera's own first-frame features:

$$\widehat{P}_{t+\Delta}^v = D_\theta\left(\overline{P}_t^v, Z_t^q\right), \qquad v \in \mathcal{V}_t, \tag{13}$$

where $\overline{P}_t^v$ are the first-frame patch features of camera v with its camera-type embedding added (Appendix B.2) and $\mathcal{V}_t$ is the set of cameras available for the sample. Because the decoder receives the first frame of the camera it predicts for, that camera's appearance and scene context are available to it without the latent action, while the capacity-limited Z_t^q is shared across all reconstruction targets. The shared bottleneck is therefore under pressure to encode the change in a form that is useful for every camera rather than to spend its capacity on content specific to one camera. The objective does not explicitly penalize camera-specific content. Section 4.2 tests whether the resulting representation reduces the impact of image-space perturbations in specific camera views on action decoding.

B.2 VIEW-AWARE TRANSITION ENCODING

Open X-Embodiment (OXE) contains heterogeneous robot datasets collected with different camera configurations. Following the data mixture and preprocessing protocol of OpenVLA (Kim et al., 2024), we organize the available observations into three camera slots: a primary camera, a secondary camera, and a gripper camera. The set of available cameras varies across the constituent datasets. Moreover, the primary and secondary cameras are typically static third-person cameras, whereas the gripper camera is a wrist camera that moves together with the robot end effector.

These camera types exhibit physical motion in substantially different visual forms. In a static third-person view, frame transitions primarily reveal the displacement of the robot and manipulated objects with respect to a relatively stable background. In a gripper view, embodiment motion is often manifested as an apparent displacement of the surrounding scene and background. Directly processing these observations without camera identity requires the transition encoder to infer both the camera configuration and the underlying motion from visual content. We therefore introduce a learnable camera-type embedding to explicitly distinguish external and wrist cameras.

For each available camera $v \in \mathcal{V}_t$, we use the synchronized transition $X_t^v = (O_t^v, O_{t+\Delta}^v)$ from Eq. (9). A frozen DINOv2 encoder Φ first converts each frame into spatial patch features:

$$P_\tau^v = \Phi(O_\tau^v) \in \mathbb{R}^{N_p \times d}, \qquad \tau \in \{t, t+\Delta\}, \tag{14}$$

where N_p is the number of image patches and d is the feature dimension. Let $\rho(v) \in \{\text{ext}, \text{wrist}\}$ denote the camera type of view v. We augment every patch feature with its camera-type embedding:

$$\overline{P}_\tau^v = P_\tau^v + \mathbf{1}_{N_p}(e_{\rho(v)}^{\text{cam}})^\top, \tag{15}$$

where $e_{\rho(v)}^{\text{cam}} \in \mathbb{R}^d$ is learned jointly with the latent-action model. Primary and secondary cameras share the external-camera type, while the gripper camera uses the wrist type. This encoding communicates how image-plane changes should be interpreted without tying the model to a particular dataset or physical camera instance.

We prepend $L = 4$ learnable transition tokens to the patch sequence at each time step and process the resulting two-frame sequence with a causal spatio-temporal Transformer:

$$U_t^v = E_\phi\Big([A, \overline{P}_t^v], [A, \overline{P}_{t+\Delta}^v]\Big), \tag{16}$$

where $A \in \mathbb{R}^{L \times d}$ denotes the learnable transition tokens. Because temporal attention is causal, the transition tokens at $t + \Delta$ can aggregate information from both the initial and future frames. We extract and project these tokens to obtain the view-specific transition representation:

$$H_t^v = W_z U_t^v[t + \Delta, 1{:}L] \in \mathbb{R}^{L \times d_z}. \tag{17}$$

Importantly, H_t^v is an intermediate representation rather than the final latent action. Each individual view captures only a particular projection of the common physical transition. The representations from all available cameras are subsequently integrated before vector quantization.

B.3 JOINT CROSS-VIEW LATENT FUSION

A practical challenge of learning from OXE is that different datasets provide different subsets of the camera slots. Treating a missing observation as a zero-valued image can introduce dataset-dependent artifacts and allow the model to infer dataset identity from padding patterns. We instead associate every sample with a binary availability mask

$$\begin{aligned} m_t^v &= \begin{cases} 1, & \text{if camera } v \text{ is available,} \\ 0, & \text{otherwise,} \end{cases} \\ \mathcal{V}_t &= \{v \mid m_t^v = 1\}. \end{aligned} \tag{18}$$

Only available cameras are passed through DINOv2 and the transition encoder, avoiding unnecessary computation on padded inputs. The resulting features are then scattered back into the camera slots for batched cross-view fusion.

Given the per-view tokens $\{H_t^v\}_{v\in\mathcal{V}_t}$, we flatten their camera and token dimensions:

$$H_t^{\mathrm{mv}} = \mathrm{Concat}(H_t^1, \ldots, H_t^V) \in \mathbb{R}^{VL\times d_z}. \tag{19}$$

We introduce $L = 4$ learnable fusion queries $Q \in \mathbb{R}^{L\times d_z}$. They aggregate complementary evidence through cross-attention:

$$\alpha_t = \mathrm{softmax}\left(\frac{(QW_Q)(H_t^{\mathrm{mv}}W_K)^\top}{\sqrt{d_z}} + \Gamma_t\right),$$
$$\widetilde{Z}_t = \mathrm{LN}(Q + \alpha_t H_t^{\mathrm{mv}} W_V), \tag{20}$$

where $\Gamma_t \in \mathbb{R}^{L\times VL}$ is the attention mask. For every missing camera v, entries corresponding to its L transition tokens are set to $-\infty$; entries for available cameras are zero:

$$\Gamma_t[:, (v-1)L+1 : vL] = \begin{cases} 0, & m_t^v = 1, \\ -\infty, & m_t^v = 0. \end{cases} \tag{21}$$

This asymmetric cross-attention has two useful properties. First, view-specific transition features serve only as keys and values, while learned queries define a common output space. The model therefore produces one sample-level representation rather than one independently quantized action for each camera. Second, its output always contains four tokens, irrespective of whether a sample contains one, two, or three available cameras. It can exploit additional observations when available while retaining a fixed latent-action interface for heterogeneous data and downstream VLA training. With one camera, the same module naturally reduces to masked single-view aggregation.

B.4 Shared Quantized Bottleneck and Multi-View Reconstruction

Two latent streams. The main text describes the stream whose code indices supervise the VLA. The full LAM has a second, auxiliary stream, following the decomposition of UniVLA (Bu et al., 2025) into a controllable latent, which is the latent action Z_t^q of the main text, and an uncontrollable latent, which models transition variation not attributable to the agent. Both streams use the view-aware encoding and the masked cross-view fusion before quantization, with separate fusion queries and separate standard VQ codebooks; their quantized outputs jointly condition the reconstruction decoder, while only the four controllable code indices supervise the VLA. Latent action learning proceeds in two phases: the first trains the uncontrollable stream alone, and the second adds the controllable stream while the uncontrollable codebook is kept frozen. Appendix C specifies the initialization and the schedule of the two phases. The formulation below first defines the representation transferred to the VLA, then gives the reconstruction decoder and the training objective of the full model.

Shared quantized representation. The fused representation $\widetilde{Z}_t = (\widetilde{z}_{t,1}, \ldots, \widetilde{z}_{t,L})$ is compressed using a standard vector quantizer. We maintain one shared codebook

$$\mathcal{C} = \{c_k\}_{k=1}^K, \qquad c_k \in \mathbb{R}^{d_z}, \tag{22}$$

and independently assign each of the $L = 4$ fused token positions to its nearest entry:

$$k_{t,\ell} = \arg\min_{k\in\{1,\ldots,K\}} \|\widetilde{z}_{t,\ell} - c_k\|_2^2,$$
$$z_{t,\ell}^q = c_{k_{t,\ell}}, \qquad \ell = 1, \ldots, L. \tag{23}$$

Thus, the latent action is the four-index sequence

$$Y_t^{\mathrm{lat}} = (k_{t,1}, k_{t,2}, k_{t,3}, k_{t,4}). \tag{24}$$

These indices represent four token positions in a shared latent sequence; they are not successive residual quantization stages. The bottleneck capacity obeys $\mathbb{H}(Z_t^q) \le L\log K = 4\log K$.

For one latent stream, the vector-quantization objective over a mini-batch of transitions $\mathcal{B}$ contains codebook and commitment terms:

$$\mathcal{L}_{\mathrm{VQ}} = \frac{1}{|\mathcal{B}|Ld_z}\sum_{t\in\mathcal{B}}\sum_{\ell=1}^{L}\big[\|\,\mathrm{sg}[\widetilde{z}_{t,\ell}] - c_{k_{t,\ell}}\|_2^2 + \beta\|\widetilde{z}_{t,\ell} - \mathrm{sg}[c_{k_{t,\ell}}]\|_2^2\big], \tag{25}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator and $\beta = 0.25$. The normalization averages over samples, latent positions, and feature dimensions.

Two-stream reconstruction and training objective. In the second phase, both quantized streams are shared across cameras. For each available camera $v \in \mathcal{V}_t$, their separately projected tokens are concatenated with the camera's first-frame patch features:

$$\widehat{P}^v_{t+\Delta} = D_\theta\Big([W_a Z_t^{q,\mathrm{ctrl}}, W_u Z_t^{q,\mathrm{unctrl}}, W_p \overline{P}^v_t]\Big). \tag{26}$$

Here $Z_t^{q,\mathrm{ctrl}} = Z_t^q$ is the latent action that supervises the VLA, $Z_t^{q,\mathrm{unctrl}}$ contains the four auxiliary tokens, and W_a and W_u are separate projections into the decoder dimension. The spatial Transformer decoder is shared across cameras. Let $N_{\mathrm{valid}} = \sum_{t\in\mathcal{B}} |\mathcal{V}_t|$ be the number of available cameras in the mini-batch. Reconstruction is an elementwise mean squared error in feature space:

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{N_{\mathrm{valid}} N_p d} \sum_{t\in\mathcal{B}} \sum_{v\in\mathcal{V}_t} \|\widehat{P}^v_{t+\Delta} - \overline{P}^v_{t+\Delta}\|_F^2. \tag{27}$$

Here $N_p = 256$ and $d = 768$. Each available camera contributes equally, so a sample with three cameras contributes three times as many feature elements as a sample with one. Missing cameras are excluded. The complete latent action learning objective sums the VQ penalties of the active streams:

$$\mathcal{L}_{\mathrm{LAM}} = \mathcal{L}_{\mathrm{rec}} + \sum_{s\in\mathcal{S}} \mathcal{L}^s_{\mathrm{VQ}}. \tag{28}$$

The first phase uses $\mathcal{S} = \{\mathrm{unctrl}\}$; the second uses $\mathcal{S} = \{\mathrm{ctrl}, \mathrm{unctrl}\}$.

The first-frame features provide spatial layout, appearance, and camera context. Sharing a capacity-limited latent across the reconstruction targets encourages it to encode changes useful for predicting all available views. This objective does not guarantee the exclusion of scene or camera information; the action-readout and token diagnostics assess the learned representation.

B.5 Single-View Policy Learning and Control

The available cameras determine the latent-action targets but do not enter the policy's inputs. After latent action learning, we freeze the LAM and label each transition using all synchronized cameras available for that sample:

$$Y_t^{\mathrm{lat}} = \mathrm{LAM}\big(\{(O_t^v, O_{t+\Delta}^v)\}_{v\in\mathcal{V}_t}\big) = (k_{t,1}, \ldots, k_{t,4}). \tag{29}$$

The VLA receives only the current primary-camera frame O_t^{pri} and the language instruction $\mathbf{w}_t$, and it is trained to predict the four indices. The sequence is modeled autoregressively, with the cross-entropy averaged over the four latent-action positions and the samples of a mini-batch $\mathcal{B}$:

$$\mathcal{L}_{\mathrm{lat}} = -\frac{1}{4|\mathcal{B}|} \sum_{t\in\mathcal{B}} \sum_{j=1}^{4} \log p_\omega\Big(k_{t,j} \mid O_t^{\mathrm{pri}}, \mathbf{w}_t, k_{t,<j}\Big). \tag{30}$$

The LAM thus defines the target from all available cameras, and the VLA learns to infer that target from one camera.

For the action decoder and the action history we follow the policy-learning framework of UniVLA (Bu et al., 2025). Hidden states at the image and latent-action positions condition a continuous action decoder, and the latent actions of the preceding steps provide action-history context. Let h_t^{vis} denote the VLA hidden states at the image tokens and $h_t^{\mathrm{lat}} \in \mathbb{R}^{4\times d_h}$ those at the four latent-action positions. The action decoder predicts a chunk of n_a robot actions:

$$\widehat{a}_{t:t+n_a-1} = G_\eta\big(h_t^{\mathrm{vis}}, h_t^{\mathrm{lat}}\big), \tag{31}$$

where n_a is the action horizon. On the demonstrations, the VLA and the decoder are optimized with the latent-action and action-regression objectives:

$$\mathcal{L}_{\mathrm{VLA}} = \mathcal{L}_{\mathrm{act}} + \lambda_{\mathrm{lat}} \mathcal{L}_{\mathrm{lat}}, \tag{32}$$

where $\lambda_{\mathrm{lat}} = 1$ and $\mathcal{L}_{\mathrm{act}}$ is the mean absolute error over the action chunk. The latent-action loss is averaged only over the four latent-action positions and the batch. Appendix C gives the action normalization, the loss reduction, and the settings of both parts of policy learning.

At inference, the VLA requires the primary-camera RGB frame and the language instruction as its inputs.

C Implementation and Training Details

LAM architecture. Each sample contains a first and a future frame from every available camera, together with camera-type indices and an availability mask. Only available cameras are passed through the visual backbone. Images are resized to 224×224, normalized with ImageNet statistics, and encoded by a frozen DINOv2 ViT-B/14 with register tokens (Oquab et al., 2024; Darcet et al., 2024). We use its normalized patch-token outputs, yielding 256 patch features of dimension 768 per image, and add a learned camera-type embedding to every patch. The encoder and decoder each contain 12 blocks with hidden dimension 768, 12 self-attention heads, and zero dropout. The latent dimension is 128.

The shared spatio-temporal encoder processes each view independently before cross-view fusion. Learnable transition tokens are prepended to the patch sequence at both time steps. The input is projected to the model dimension with layer normalization and augmented with sinusoidal spatial positional encodings. Each encoder block applies spatial self-attention within a frame, causal temporal self-attention at each token position, and a feed-forward network with GELU activation and a fourfold hidden expansion. All three operations use pre-normalization and residual connections. The transition-token outputs at the future time step are linearly projected to the latent dimension.

Multi-view fusion, quantization, and reconstruction. For each latent stream, four learned queries attend to the concatenated transition tokens from all available cameras using four-head cross-attention with zero attention dropout. A key-padding mask excludes missing cameras, and the attention output is added to the queries and layer-normalized before quantization. Each of the four fused positions is assigned to its nearest codebook entry in Euclidean distance, using a straight-through estimator for backpropagation. The controllable stream uses one shared 16-entry codebook, giving a 16^4 discrete bottleneck regardless of the number of input cameras.

The quantized tokens are projected to the decoder dimension and shared across all available cameras. For each camera, they are concatenated with the projected first-frame patch features. A shared spatial Transformer, with sinusoidal positional encodings, pre-normalized self-attention, and GELU feed-forward blocks, predicts the future patch features. Its output projection restores the 768-dimensional feature space; only patch positions contribute to reconstruction. The targets include the same camera-type embeddings as the encoder inputs. Reconstruction is evaluated only on available cameras.

Two-phase latent action learning. The first phase trains the auxiliary uncontrollable stream for 120k gradient steps. It uses four transition tokens per frame, four fusion queries, and a 16-entry codebook. A frozen T5-base encoder (Raffel et al., 2020) supplies instruction embeddings, which are projected to the model dimension and appended as language tokens in both the spatio-temporal encoder and the spatial decoder. Language padding is masked, and the instruction embeddings are shared across the available cameras of a sample.

The second phase trains for 80k gradient steps and introduces the controllable stream while retaining the first-phase uncontrollable codebook with frozen weights. The uncontrollable transition tokens are also loaded from the first-phase checkpoint, and the learned fusion queries initialize both streams with identical values that can subsequently diverge. The remaining trainable modules are initialized independently for the second phase. Two groups of four transition tokens are prepended to each frame and processed by the shared encoder. The two streams have separate latent projections, fusion queries, cross-attention modules, and standard VQ codebooks. Their quantized outputs are separately projected and concatenated with each view's initial-frame patch features for reconstruction. No language embeddings are supplied to either the encoder or decoder in this phase. DINOv2 remains frozen throughout both phases. We do not restart codebook entries during training. The same 120k/80k schedule is used for the multi-view model and the primary-only LAM ablation, giving each a total LAM budget of 200k steps. For downstream policy supervision, we retain JMV-LA's four controllable code indices as latent-action targets.

LAM optimization and loss reduction. Both phases use AdamW with a constant learning rate of 10^{-4} and weight decay 10^{-2}, without warmup or a learning-rate scheduler. Training runs on eight GPUs with 32 samples per GPU and two gradient-accumulation steps, giving an effective global batch size of 512. We use FP16 mixed precision and gradient clipping at 0.1.

Reconstruction is performed in DINOv2 patch-feature space. The future targets and predictions each have shape $N_{\text{valid}} \times 1 \times N_p \times 768$, with the camera embeddings included in the targets as described above. We take the mean squared error over every element of this tensor in each local mini-batch. Missing camera slots are filtered out before feature extraction and do not contribute. Each sample is consequently weighted in proportion to its number of available cameras. In the primary-only LAM, N_{valid} equals the batch size. For each stream s, let $\widetilde{Z}^s$ be the fused tokens before quantization and $Z^{q,s}$ the selected codebook vectors. We use

$$\mathcal{L}^s_{\text{VQ}} = \text{mean}\Big[(\text{sg}[\widetilde{Z}^s] - Z^{q,s})^2\Big] + 0.25\ \text{mean}\Big[(\widetilde{Z}^s - \text{sg}[Z^{q,s}])^2\Big], \tag{33}$$

where the squares and means are elementwise and include the batch, token, and latent-feature dimensions. The first phase minimizes $\mathcal{L}_{\text{rec}} + \mathcal{L}^{\text{unctrl}}_{\text{VQ}}$; the second minimizes $\mathcal{L}_{\text{rec}} + \mathcal{L}^{\text{ctrl}}_{\text{VQ}} + \mathcal{L}^{\text{unctrl}}_{\text{VQ}}$. There is no additional VQ loss multiplier. In the second phase, the uncontrollable codebook term contributes no parameter gradients because the codebook is frozen, while its commitment term still trains the corresponding features.

Policy learning on the labeled videos. The Prismatic-7B backbone (Karamcheti et al., 2024) is initialized from the OpenVLA-7B checkpoint (Kim et al., 2024). We train the VLA to predict the latent-action targets for 25k steps with a global batch size of 1024 and a learning rate of 2×10^{-5}, updating all backbone parameters. The trained LAM is frozen while it labels the video transitions. Its four codebook indices are represented by 16 special action tokens in the VLA vocabulary and predicted autoregressively. Only the primary camera's image and the language instruction are passed to the VLA. The LAM reads all available synchronized cameras to compute the target, but the secondary and gripper cameras are never policy inputs.

Policy learning on the demonstrations and the action loss. Training on the demonstrations uses LoRA (Hu et al., 2022) with rank 32, scaling parameter $\alpha = 16$, dropout 0, and a learning rate of 3.5×10^{-4}. Hidden states at the image-token and latent-action positions condition the action decoder, which predicts a 12-step continuous action chunk; the latent actions of the preceding steps provide action-history context. For LIBERO, we train a separate policy for each suite: Spatial, Object, and Goal use 30k gradient steps, Long uses 40k, and all use a global batch size of 128. Each suite is evaluated with 50 rollouts per task. For the real-robot tasks, all compared models use batch size 64 and the same task-specific training budgets: 4k steps for cube-to-cup, 4k for drawer manipulation, and 3.2k for tape-to-stand.

For LIBERO, the decoder predicts seven action dimensions at each of the $n_a = 12$ time steps. The action loss is

$$\mathcal{L}_{\text{act}} = \frac{1}{|\mathcal{B}|\, n_a \cdot 7} \sum_{b \in \mathcal{B}} \sum_{h=1}^{n_a} \sum_{j=1}^{7} \left|\widehat{a}_{b,h,j} - a'_{b,h,j}\right|, \tag{34}$$

where $\mathcal{B}$ is the mini-batch and a' denotes the normalized training target. All time steps contribute, including padded positions. We add this loss directly to the latent-action cross-entropy, averaged over the batch and the four action-token positions, with no additional weighting.

LIBERO action normalization. We normalize the first six action dimensions, representing relative end-effector translation and rotation, using the training-data 1st and 99th percentiles:

$$a'_d = \text{clip}\left(2\frac{a_d - q_{0.01,d}}{q_{0.99,d} - q_{0.01,d} + \epsilon} - 1,\ -1,\ 1\right). \tag{35}$$

Dimensions that are constant in the training data are set to zero. The gripper target is transformed separately as $g' = 1 - \text{clip}(g, 0, 1)$, with 1 denoting open and 0 denoting closed, and is excluded from percentile normalization. A final Tanh activation bounds the decoder outputs to $[-1, 1]$. We save the training-data normalization statistics with the policy and use them at inference to restore the first six dimensions via $a_d = (a'_d + 1)(q_{0.99,d} - q_{0.01,d})/2 + q_{0.01,d}$.

D Latent Action Learning Data and Processing

Data mixture and camera availability. Our mixture contains 26 manipulation datasets from Open X-Embodiment (O'Neill et al., 2023) and three navigation datasets: Berkeley GNM Recon (Shah et al., 2022), Cory Hall (Kahn et al., 2018), and SACSoN (Hirose et al., 2024).

Table 8: Effective camera configurations in latent action learning. Probabilities include dataset weights and Bridge secondary-view subsampling.

Available cameras	Sampling probability
Primary only	49.90%
Primary + gripper	37.00%
Primary + secondary	4.94%
Primary + secondary + gripper	8.17%

For BridgeData V2 (Walke et al., 2023), we use the original multi-camera TFDS release, `bridge_dataset/1.0.0`,[1] identified as `bridge_orig` in our loader, rather than its single-view conversion, preserving synchronized secondary-camera observations. We do not include Ego4D or other human-video sources. Before weighted resampling, the mixture contains 1,269,431 trajectories and 37,521,017 visual transitions.

We map observations to primary, secondary, and gripper slots. Primary and secondary denote external third-person cameras, while gripper denotes an wrist camera. An availability mask records which slots exist for each sample. After dataset weighting and Bridge secondary-view subsampling, the effective camera distribution is shown in Table 8.

Mixture weights and sampling probabilities. Table 9 lists all constituent datasets, their mixture weights, sizes, and camera slots. For dataset i with weight w_i and N_i transitions, the training sampling probability is

$$p_i = \frac{w_i N_i}{\sum_j w_j N_j}. \tag{36}$$

The weights sum to 36.2. Normalizing the weights alone, $w_i / \sum_j w_j$, does not give the training sampling probability because dataset sizes differ. The camera combinations in this table precede Bridge secondary-view subsampling; Table 8 reports the effective camera distribution after that subsampling.

Temporal sampling and visual augmentation. We sample the interval between initial and future frames according to the dataset. The low-frequency group comprises Fractal, Berkeley Autolab UR5, NYU Franka Play, UCSD Kitchen, DLR EDAN Shared Control, and DObbE; its offsets are sampled uniformly from $\{3, 4, 5\}$ frames. The high-frequency group comprises UT Austin Mutex, IAMLab CMU Pickup Insert, Austin Sailor, TOTO, VIOLA, and DROID; its offsets are sampled uniformly from $\{15, 16, 17, 18, 19, 20\}$ frames. All other datasets use a fixed offset of 10 frames. TOTO is assigned to the high-frequency group. The same frame pair is used across the synchronized cameras of a sample. Images are resized to 224×224. Training uses a random resized crop with scale 0.9 and fixed aspect ratio, followed by random brightness (0.2), contrast ($[0.8, 1.2]$), saturation ($[0.8, 1.2]$), and hue (0.05) augmentation.

E Additional Evaluation Protocols

E.1 Action Decodability Probe

The primary-only LAM. The full model and the primary-only LAM use the same OXE dataset composition, trajectories, and sampling weights. We retain the same transition encoder architecture, four-token bottleneck, 16-entry codebook, and LAM training settings, including 120k first-phase and 80k second-phase steps (200k in total). In particular, the primary-only LAM retains the training trajectories that contain multiple cameras but uses only their primary-camera frame pairs; it does not discard these trajectories. Secondary and gripper cameras are excluded from both encoding and reconstruction. We remove the cross-view fusion module and feed the primary camera's four transition tokens, after the required projection to the quantizer input dimension, directly to VQ. The full model instead quantizes four jointly fused tokens and reconstructs the future features of all

[1] `https://rail.eecs.berkeley.edu/datasets/bridge_release/data/tfds/bridge_dataset/1.0.0/`

Table 9: Latent action learning data mixture: 26 OXE manipulation datasets and three navigation datasets. Dataset identifiers are shortened where needed. w_i is the mixture weight and p_i is the training sampling probability. P, S, and G denote the primary, secondary, and gripper slots, respectively. Camera combinations are listed before Bridge secondary-view subsampling. Displayed probabilities are rounded and sum to 99.99%.

Dataset	w_i	p_i (%)	Trajectories	Transitions	Views
OXE manipulation datasets					
`fractal20220817_data`	0.5	13.14	87,212	3,786,400	P
`kuka`	0.1	5.96	580,351	8,583,467	P
`bridge_orig`	1.0	14.82	60,064	2,135,463	P+S
`taco_play`	2.0	3.30	3,603	237,798	P+G
`jaco_play`	1.0	0.54	1,085	77,965	P+G
`berkeley_cable_routing`	1.0	0.29	1,647	42,328	P+S+G
`roboturk`	2.0	2.60	1,993	187,301	P
`viola`	2.0	1.06	150	76,324	P+G
`berkeley_autolab_ur5`	2.0	1.36	1,000	97,939	P+G
`toto`	1.0	2.26	1,003	325,699	P
`language_table`	0.1	4.89	442,226	7,045,476	P
`stanford_hydra`	2.0	4.97	570	358,234	P+G
`austin_buds`	1.0	0.24	50	34,112	P+G
`nyu_franka_play`	3.0	0.93	456	44,875	P+S
`furniture_bench`	0.1	2.74	5,100	3,948,057	P+G
`ucsd_kitchen`	2.0	0.06	150	3,970	P
`austin_sailor`	1.0	2.45	240	353,094	P+G
`austin_sirius`	1.0	1.94	559	279,939	P+G
`dlr_edan_shared_control`	1.0	0.06	104	8,928	P
`iamlab_cmu_pickup_insert`	1.0	1.02	631	146,241	P+G
`utaustin_mutex`	1.0	2.51	1,500	361,883	P+G
`berkeley_fanuc_manipulation`	2.0	0.87	415	62,613	P+G
`cmu_stretch`	1.0	0.17	135	25,016	P
`bc_z`	0.2	8.35	43,264	6,015,535	P
`fmb`	1.0	7.88	8,595	1,134,471	P+S+G
`dobbe`	0.2	1.58	5,208	1,139,911	P
Navigation datasets					
`berkeley_gnm_recon`	2.0	8.48	11,834	610,907	P+G
`berkeley_gnm_cory_hall`	2.0	2.17	7,331	156,012	P+G
`berkeley_gnm_sac_son`	2.0	3.35	2,955	241,059	P+G
Total	36.2	100.00	1,269,431	37,521,017	—

available cameras; the primary-only LAM reconstructs only the primary camera's future features. The comparison thus removes the multi-camera design as a whole while holding the data mixture, training settings, and bottleneck capacity fixed.

Shared probe protocol. Both frozen models are evaluated with the same probe architecture, data split, training settings, and evaluation procedure. We split complete trajectories before sampling transitions from `libero_spatial_no_noops`. With split seed 0, 382 trajectories yield 7,640 training transitions and 50 held-out trajectories yield 1,000 test transitions. Each trajectory contributes up to 20 evenly spaced valid starting points. Trajectories are held out for probe fitting; overlap with the LAM's training videos has not been verified. The primary-only LAM receives the primary camera's frame pair; JMV-LA receives the primary pair and any available synchronized secondary and wrist-camera pairs.

The frozen LAM produces four 16-way code indices. Position-wise one-hot encoding yields a 64-dimensional probe input. The probe has a 512-unit ReLU hidden layer and an 84-dimensional output, predicting 12 seven-dimensional actions from frames t and $t + 11$. Training uses the implementation defaults, 40 epochs of AdamW, learning rate 10^{-2}, batch size 1,024, zero weight decay, and a mean-squared-error loss; the saved metrics do not independently record these runtime settings. Reported MAEs use normalized action coordinates and are averaged over samples and all 12 steps. Translation, rotation, and gripper MAEs use dimensions 0–2, 3–5, and 6, respectively. The reported R^2_{global} uses the grand mean of all test target entries as its baseline, rather than separate means per output coordinate. Means and sample standard deviations are computed across five probe initialization seeds with fixed LAM checkpoints.

For the image-perturbation robustness probe in Table 2, the clean MAEs and split are the same as in Table 1. The five test-time transformations are horizontal shifts in both directions, vertical shifts in both directions, and a 10° counterclockwise image rotation; the same transformation is applied to both frames of a transition. Perturbed MAE is averaged over the five transformations. The probe is fit only to clean training tokens and then evaluated on paired clean and transformed test tokens.

For four-token sequences z and z', token disagreement is the fraction of positions with different code indices. D_{view} averages this quantity between clean and perturbed versions of the same transition. D_{motion} instead compares clean tokens from randomly selected, distinct transitions belonging to the same task and is reported only as a diagnostic against a constant representation. Action MAE and ΔMAE report the mean and sample standard deviation over five probe initializations; token distances are deterministic statistics of the fixed checkpoints. These tests use image-space transformations and should not be interpreted as camera-pose perturbations or errors in physical units.

E.2 Camera-Perturbation Implementation

Let $(\mathbf{p}_0, \mathbf{q}_0)$ denote the clean position and orientation of `agentview`. Translation is sampled independently along all three axes:

$$\boldsymbol{\epsilon}_t \sim \mathcal{U}([-\delta, \delta]^3), \qquad \mathbf{p} = \mathbf{p}_0 + \boldsymbol{\epsilon}_t, \tag{37}$$

where $\delta \in \{5, 10\}$ cm. For rotation, local XYZ Euler offsets are sampled from $\mathcal{U}([-\theta, \theta]^3)$ for $\theta \in \{5^\circ, 10^\circ\}$, converted to a quaternion $\mathbf{q}_\epsilon$, and composed as

$$\mathbf{q} = \mathbf{q}_0 \otimes \mathbf{q}_\epsilon. \tag{38}$$

The combined condition independently samples $\delta = 5$ cm and $\theta = 5^\circ$.

At the beginning of every episode, we restore the clean camera pose, reset the environment, sample a perturbation relative to the clean extrinsics, and load the official LIBERO initial state. We then execute 10 no-op simulation steps. The sampled camera remains fixed for the rollout, and image-space corruptions are disabled. Only `agentview` is modified. Deterministic seeds are

$$s_{i,j} = 7 \times 1000003 + i \times 1009 + j, \tag{39}$$

where i and j index task and episode, ensuring that every policy receives the same transformations.

The maximum horizon is 520 control steps. UniVLA, MVP-LAM, and JMV-LA receive only the perturbed third-person image and language and use a 12-step decoder window.

E.3 PER-SUITE LIBERO-PRO AGGREGATION

Table 10 reports the per-suite results for all four methods in Table 5. Each perturbation-category mean gives equal weight to the four suites; the overall mean gives equal weight to the five categories. Each suite's Avg. is the mean of its five category scores. All means are computed from the per-suite entries before rounding to two decimal places. For OpenCLAP-SV, the evaluation fields `swap` and `lan` correspond to Position and Semantic, respectively.

Table 10: Per-suite success rates (%) on LIBERO-Pro. For each method, the four-suite mean row reproduces its aggregate results in Table 5. OpenCLAP-SV is our single-view reproduction.

Suite	Object	Position	Semantic	Task	Environment	Avg.
UniVLA						
LIBERO-Spatial	87.20	0.20	79.80	31.00	9.40	41.52
LIBERO-Object	78.80	5.40	91.40	0.40	0.00	35.20
LIBERO-Goal	63.00	4.20	94.00	5.60	6.20	34.60
LIBERO-10	49.00	1.00	91.00	10.00	45.00	39.20
Four-suite mean	69.50	2.70	89.05	11.75	15.15	37.63
OpenCLAP-SV						
LIBERO-Spatial	97.00	0.00	94.80	0.00	20.60	42.48
LIBERO-Object	87.40	0.00	98.20	0.00	25.80	42.28
LIBERO-Goal	55.20	5.80	97.40	9.40	4.40	34.44
LIBERO-10	39.40	0.40	88.60	8.00	44.80	36.24
Four-suite mean	69.75	1.55	94.75	4.35	23.90	38.86
MVP-LAM						
LIBERO-Spatial	92.40	3.80	82.20	14.80	11.00	40.84
LIBERO-Object	82.60	0.00	97.60	10.00	0.00	38.04
LIBERO-Goal	58.60	0.60	71.80	10.40	8.00	29.88
LIBERO-10	43.40	0.00	86.20	9.00	45.80	36.88
Four-suite mean	69.25	1.10	84.45	11.05	16.20	36.41
JMV-LA						
LIBERO-Spatial	87.20	19.00	85.20	41.00	13.20	49.12
LIBERO-Object	87.00	0.00	94.20	9.00	0.00	38.04
LIBERO-Goal	73.80	0.80	92.80	10.20	4.40	36.40
LIBERO-10	57.00	2.00	93.00	19.00	50.00	44.20
Four-suite mean	76.25	5.45	91.30	19.80	16.90	41.94

F REAL-ROBOT EXPERIMENTAL DETAILS

F.1 HARDWARE AND EVALUATION PROTOCOL

Hardware, cameras, and control. All tasks use the right arm of an ALOHA platform (Zhao et al., 2023) and three synchronized RGB cameras: a fixed front camera (`cam_high`), a fixed left camera (`cam_left`), and a right-wrist camera (`cam_right_wrist`). All policies receive only `cam_high` and language at deployment. Figure 3 shows the platform, camera locations, and example frames from the three cameras.

The controller runs at 40 Hz. Each query predicts a 12-step continuous action chunk, executes three steps, and then replans; overlapping predictions for the same timestep are temporally aggregated. All policies use the same chunk length, replanning interval, aggregation rule, and control frequency. A rollout lasts at most 2,000 published control steps (50 s). For applicable grasp subgoals—the cube, drawer handle, or tape—the policy may make at most three attempts; failure to secure the target on the third attempt counts as rollout failure. Success is strict and end-to-end, with partial completion counted as failure.

Training and evaluation protocol. We collect 64 demonstrations for cube-to-cup, 64 for drawer manipulation, and 72 for tape-to-stand. Every model uses the same collected trajectories for each task. All compared models use batch size 64 and the same predetermined checkpoint budgets: 4k

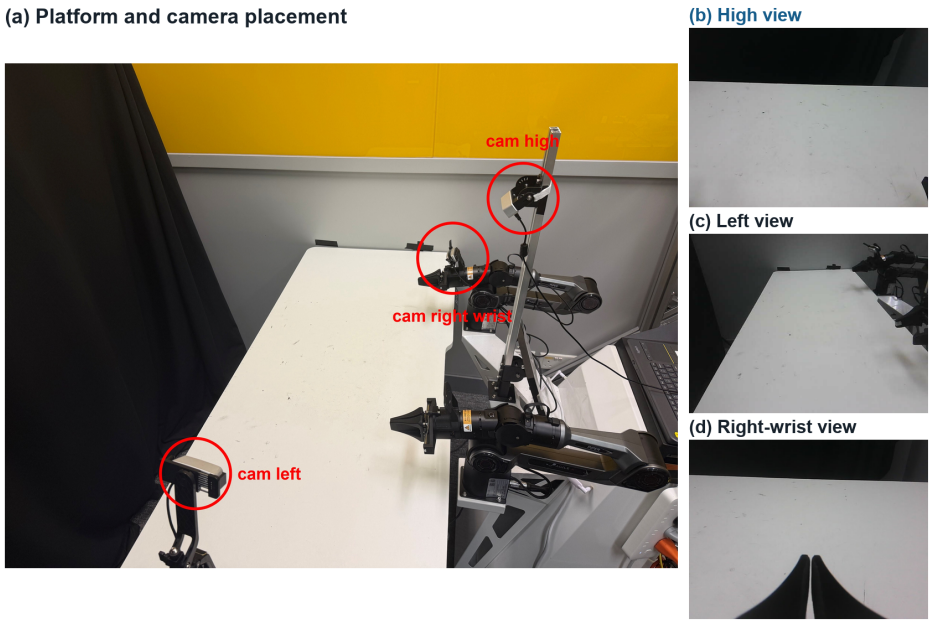

Figure 3: Real-robot platform and camera observations. (a) Physical locations of the high, left, and right-wrist cameras. (b–d) Example images from `cam_high`, `cam_left`, and `cam_right_wrist`, respectively. The LAM reads all three cameras to compute the latent-action targets. At deployment, all compared policies receive only the high-camera image and language instruction.

steps for cube-to-cup, 4k for drawer manipulation, and 3.2k for tape-to-stand. Objects are manually reset before every rollout. In-distribution layouts are randomly initialized within their specified regions, whereas OOD layouts use predetermined positions. Within each task and setting, the three models are evaluated sequentially rather than with interleaved rollouts.

F.2 Task Configurations and Success Criteria

Figure 4 summarizes the seven evaluation conditions in Table 6.

Cube-to-cup layouts. The red cube is approximately 4 cm wide and the blue cup 6 cm in diameter. Demonstrations place the cup near the table center and sample the cube from eight surrounding positions, 15–20 cm away. In-distribution evaluation repeats each position twice (16 trials). Spatial OOD moves the cup right and cube left, increasing their separation to 40–50 cm, and uses 10 predetermined layouts. Success requires the released cube to come to rest inside the cup; contact with the rim followed by the cube sliding into the cup is counted as success.

Drawer layouts and occlusion. The long-horizon instruction requires opening a drawer, placing a blue cube inside, and closing it. The fixed drawer has a green basket and cube to its left. Demonstrations vary their locations; roughly half show the cube, while the rest place it inside or behind the basket, partially occluding it from `cam_high`. We test these three visibility conditions separately, all of which occur during training. Success requires completing the full sequence of opening the drawer, grasping and placing the cube, and closing the drawer, with the cube remaining inside and a final drawer gap of at most 1 cm.

Tape-to-stand hanging. The instruction is "Pick up the tape and hang it on the stand." The robot must grasp a white tape roll from the red mat and place it over the branch facing `cam_high`. Success requires releasing the tape and leaving it hanging from that branch; another branch, a dropped tape, or no release is a failure.

All training demonstrations align the stand base with the mat's upper-left corner; the support-layout OOD arrangement is absent from training. The tape placement region to the right of the stand is divided into a 3×2 grid, and two positions are sampled within each cell during data collection. For in-distribution evaluation, the aligned stand–mat assembly is jointly translated by up to 3 cm along both the front–back and left–right directions. In support-layout OOD, the base extends 7 cm beyond the mat's left boundary and 5 cm beyond its upper boundary. The stand orientation is fixed in both settings, and tape positions follow the same 3×2 grid sampling scheme. We evaluate 20 rollouts per condition.

F.3 Uncertainty Intervals

Each rollout is treated as a binary success or failure. For x successes in n rollouts, we compute the 95% Wilson score interval with $z = 1.96$:

$$\frac{\widehat{p} + z^2/(2n) \;\pm\; z\sqrt{\widehat{p}(1-\widehat{p})/n + z^2/(4n^2)}}{1 + z^2/n}, \qquad \widehat{p} = \frac{x}{n}. \tag{40}$$

Table 11 reports these intervals for all conditions. Wilson intervals quantify within-condition binomial uncertainty and exclude temporal hardware variation from sequential evaluation.

Table 11: Real-robot success rates with 95% Wilson score intervals. Each entry reports the observed success percentage followed by [lower, upper]; all values are percentages. Raw success counts appear in Table 6.

Task	Evaluation setting	UniVLA	MVP-LAM	JMV-LA
Cube-to-cup	In-distribution	68.8 [44.4, 85.8]	68.8 [44.4, 85.8]	87.5 [64.0, 96.5]
	Spatial OOD	0.0 [0.0, 27.8]	30.0 [10.8, 60.3]	90.0 [59.6, 98.2]
Drawer	No occlusion	45.8 [27.9, 64.9]	54.2 [35.1, 72.1]	70.8 [50.8, 85.1]
	Behind basket	41.7 [19.3, 68.0]	41.7 [19.3, 68.0]	75.0 [46.8, 91.1]
	Inside basket	37.0 [21.5, 55.8]	33.3 [18.6, 52.2]	55.6 [37.3, 72.4]
Tape-to-stand	In-distribution	95.0 [76.4, 99.1]	80.0 [58.4, 91.9]	90.0 [69.9, 97.2]
	Support-layout OOD	35.0 [18.1, 56.7]	50.0 [29.9, 70.1]	70.0 [48.1, 85.5]

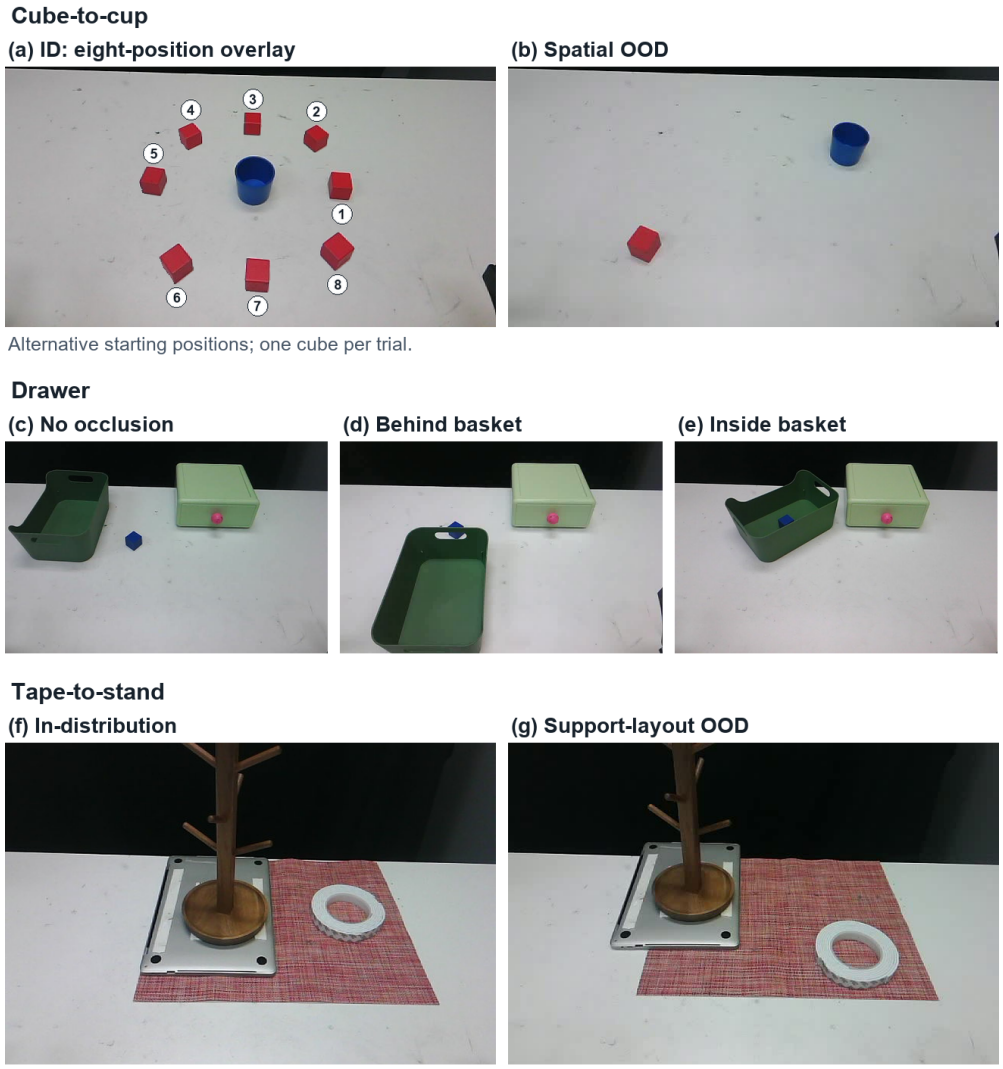

Figure 4: Real-robot task configurations viewed from `cam_high`. (a) The eight cube-to-cup ID starting positions are overlaid from separate photographs; each actual trial contains only one cube. (b) A spatial-OOD layout. (c–e) The three drawer visibility conditions, all represented in training. (f–g) Tape-to-stand ID and support-layout OOD, showing the changed position of the stand relative to the red mat. The seven conditions correspond to the rows of Table 6.

F.4 Qualitative Executions and Failure Observations

Figure 5 shows selected successful executions for cube-to-cup spatial OOD, drawer manipulation behind the basket, and tape-to-stand support-layout OOD. The frames preserve the source video timestamps and show the release and final task states.

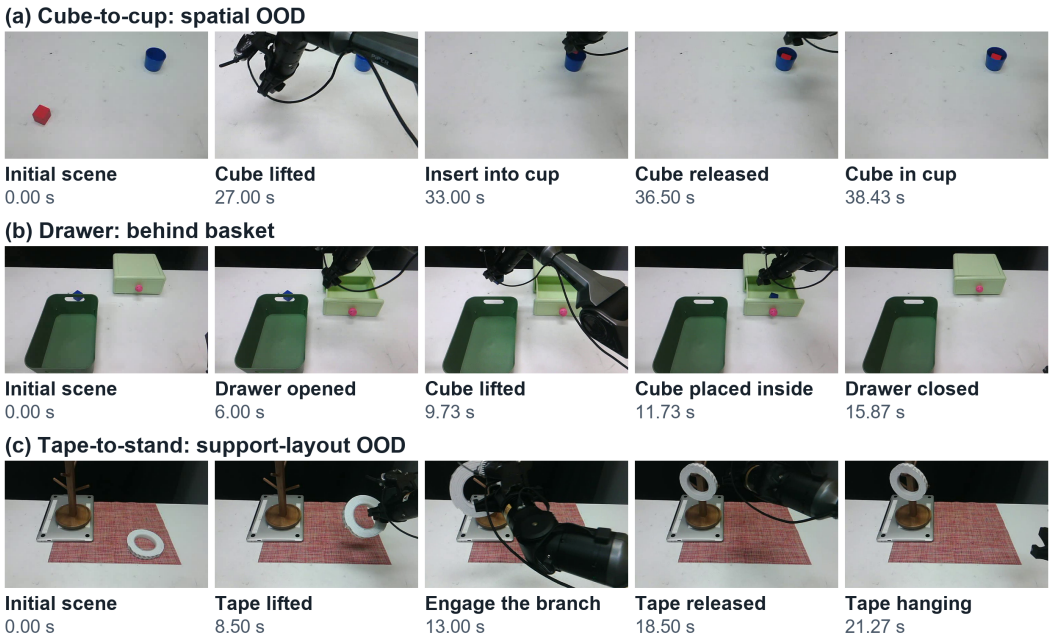

Figure 5: Selected keyframes of successful real-robot executions. (a) Cube-to-cup under spatial OOD, (b) drawer manipulation with the cube behind the basket, and (c) tape-to-stand under support-layout OOD. Each row comes from one video, with a consistent spatial crop and original video timestamps. Frames show selected action stages rather than uniformly sampled intervals. The release frames show the cube inside its target container or the tape remaining on the branch after the gripper lets go.

Qualitative failure observations. With binarized gripper commands, we observe no failures in which an object is securely grasped and subsequently dropped during transport. Cube-to-cup failures instead arise from missed cube grasps or unsuccessful placement into the cup. We observe placement failures only in the spatial-OOD condition. In that condition, the cube is farther across the table; a characteristic failure is that the gripper reaches toward a nearer location and stops short of the cube.

Drawer failures occur when the policy misses the handle, fails to grasp the cube, or does not place the cube inside the drawer. Under partial occlusion, the most common observed failure is that the arm reaches toward the cube but collides with the wall of the basket that partially occludes it. This contact can block the arm's motion or cause the cube to be knocked farther away.

Tape-to-stand failures divide into failure to pick up the tape and failure to engage the target branch. All in-distribution failures are pickup failures; in some cases, the descending gripper contacts the tape's upper surface and becomes obstructed. Under the support-layout shift, we observe hanging failures in which the policy moves the tape toward the stand location associated with the training layout, causing it to fall during the attempted placement.

These observations summarize qualitative failure patterns.