ACTION-MANIFOLD VLA: GEOMETRY-AWARE FLOW MATCHING IN JOINT AND TASK SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based and flow-based action experts have become a dominant paradigm for vision-language-action (VLA) models. However, they require two important design decisions. (1) **What geometry should the action representation obey?** Actions are usually treated as vectors in Euclidean space, but in reality a bimanual action chunk lives on a product manifold. (2) **In which space should the generative dynamics evolve?** Joint space and task space each have their trade-offs. To study these two questions, we propose **Action-Manifold VLA (AM-VLA)**, a framework with two variants sharing the same backbone and a unified action space but differing in where the flow lives. AM-VLA models the action chunk on a product manifold and instantiates the flow in either space. **AM-VLA-J** (joint-space flow) builds the flow on joint residual and gripper channels while introducing manifold-aware task-space auxiliary supervision. **AM-VLA-E** (task-space flow) builds the flow on the per-arm EE residual manifold $\mathcal{M}_E = (\mathbb{R}^1_g \times \mathbb{R}^3_{\Delta p} \times SO(3)_{\Delta R})^{2H}$: gripper, position residual, and rotation residual. Rotations use shortest-geodesic conditional paths and Log-map target fields with time-constant norm. On the 50-task RoboTwin 2.0 benchmark, AM-VLA-J and AM-VLA-E achieve average success rates of 92.5% and 93.1%, respectively, compared with 92.1% for the backbone. Experiments show that manifold-aware task-space supervision improves joint-space generation, while manifold-native $SO(3)$ dynamics provide substantially larger gains when the generative process evolves directly in task space.

1 INTRODUCTION

Vision–language–action (VLA) models have emerged as a general paradigm for robot manipulation: a pretrained vision–language backbone encodes observations and instructions, and an action expert decodes them into executable action chunks (Zhao et al., 2023; Black et al., 2024; Zheng et al., 2025). Recent systems increasingly employ diffusion- or flow-based generative action experts to model continuous action distributions (Chi et al., 2023; Liu et al., 2024; Black et al., 2024; Lin et al., 2026). We argue that continuous VLA policies involve two fundamental and coupled design choices.

(1) **What geometry should the action representation obey?** Robot actions are typically concatenated and treated as vectors in a flat Euclidean space, although their components have heterogeneous geometric structure. End-effector orientations are elements of $SO(3)$, commonly represented computationally by unit quaternions on the double cover S^3 (Zhou et al., 2019); translational variables are Euclidean; joint variables may be periodic or bounded depending on the mechanism; and gripper apertures are bounded scalars. (2) **In which space should the generative dynamics evolve?** Joint-space generation produces commands that are directly compatible with low-level control and avoids solving inverse kinematics (IK) during action decoding, but joint coordinates are only indirectly related to the Cartesian quantities that determine manipulation success. End-effector (EE) generation instead models positions, orientations, and gripper commands that are more directly aligned with task-space objectives, but requires a downstream kinematic controller and introduces non-Euclidean rotational variables into the generative process.

To systematically study this interaction, we propose **Action-Manifold VLA (AM-VLA)**, a unified framework built upon HoloBrain-0 (Lin et al., 2026), an x-prediction diffusion-based VLA trained

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.

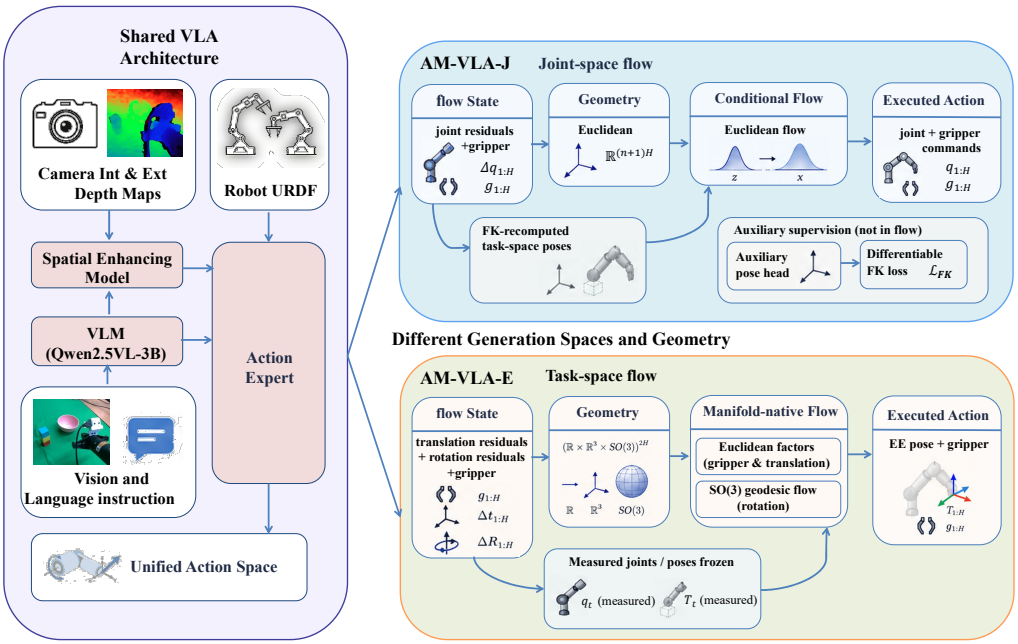

Figure 1: Overview of AM-VLA. Both variants share the VLA backbone and clean-action prediction interface. AM-VLA-J evolves joint residuals and gripper commands in Euclidean space and applies auxiliary manifold-aware supervision. AM-VLA-E instead evolves gripper, translation, and rotation residuals directly in task space, using Euclidean flows for flat factors and a manifold-native $SO(3)$ flow for rotations.

with DDPM (Ho et al., 2020). AM-VLA preserves the visual–language backbone, action-decoder interface, and residual action representation while changing the geometry and domain of the generative dynamics. We instantiate two variants that share the same backbone and data pipeline but place the flow in different spaces.

AM-VLA-J evolves joint residuals and gripper commands through a Euclidean conditional flow. AM-VLA-J retains directly executable joint-space generation while injecting task-space geometric supervision outside the flow. **AM-VLA-E** instead evolves the generative state directly on the EE product manifold defined in Equation 2. Translation and gripper factors follow Euclidean conditional paths, whereas rotational states follow shortest geodesics on $SO(3)$.

Experiments on the 50-task RoboTwin 2.0 benchmark (Chen et al., 2025) show that both variants improve over the HoloBrain-0 backbone, with AM-VLA-J increasing average success from 92.1% to 92.5% and AM-VLA-E further reaching 93.1%. Controlled ablations reveal that generation space and action geometry should be designed jointly: task-space generation becomes most effective when its rotational structure is respected by the generative dynamics themselves.

Our main contributions are:

- We formulate continuous VLA action generation from a geometric perspective and separate two design axes that are often conflated: the *space in which the generative dynamics evolve* and the *geometry imposed on the action variables*.
- We introduce AM-VLA, a unified framework with a joint-space variant using manifold-aware task-space supervision and a task-space variant performing manifold-native flow matching on $\mathbb{R}^3 \times SO(3)$ EE residuals.

2 Related Work

VLA policies and generative action experts. VLA policies connect pretrained vision–language representations to robot control, with early systems encoding actions as tokens and later systems emphasizing open-source and cross-embodiment scaling (Brohan et al., 2023; Kim et al., 2024; Zheng et al., 2025). In parallel, action chunking and diffusion policies established expressive sequence prediction for imitation learning (Zhao et al., 2023; Chi et al., 2023); recent VLA action experts adopt diffusion (RDT-1B and HoloBrain-0) or flow matching (π_0 and X-VLA) for continuous chunks (Liu et al., 2024; Lin et al., 2026; Black et al., 2024; Zheng et al., 2025). Despite their architectural differences, these methods generally learn in a Euclidean action-coordinate space. AM-VLA instead asks how the heterogeneous geometry of continuous actions should constrain the generator itself.

Joint-space and task-space actions. Joint coordinates and EE poses offer complementary policy interfaces. Joint-space chunks are directly executable and avoid online IK, as in ACT, whereas task-space chunks align predictions with manipulation-relevant Cartesian motion but require kinematic realization, as commonly used by diffusion policies (Zhao et al., 2023; Chi et al., 2023). Cross-embodiment models further expose action representation as a bottleneck: RDT-1B defines a physically interpretable unified action space, whereas X-VLA uses embodiment-specific soft prompts to absorb platform heterogeneity (Liu et al., 2024; Zheng et al., 2025). Prior work mainly chooses an output interface; we instead hold the backbone and training pipeline fixed and compare the space in which the generative dynamics evolve.

Rotation geometry and manifold generative modeling. Three-dimensional rotations lie on $SO(3)$, and common low-dimensional coordinates, including quaternions, cannot provide a globally continuous Euclidean representation; 5D and 6D alternatives improve continuity for neural rotation regression (Zhou et al., 2019). Representation validity alone, however, does not make a Euclidean diffusion or flow intrinsic: normalized quaternion paths need not follow rotational geodesics. Riemannian score models and flow matching instead define stochastic or ODE dynamics directly on manifolds through tangent vector fields and geometric maps (De Bortoli et al., 2022; Chen & Lipman, 2024). AM-VLA brings this distinction to VLA action generation by separating manifold-aware supervision from manifold-native generative dynamics.

3 Method

3.1 Overview: One Framework, Two Generation Spaces

AM-VLA separates the geometry of an action representation from the space in which its generative dynamics evolve. A variant $\nu \in \{\mathrm{J}, \mathrm{E}\}$ is specified by its flow state $\mathcal{F}_\nu$, geometry $\mathcal{M}_\nu$, conditioning assembly Φ_ν, and execution mapping $\mathcal{E}_\nu$:

$$\mathcal{C}_\nu = (\mathcal{F}_\nu, \mathcal{M}_\nu, \Phi_\nu, \mathcal{E}_\nu). \tag{1}$$

Both variants share the Qwen2.5-VL-3B backbone (Bai et al., 2025), decoder, horizon, and clean-action interface of HoloBrain-0 (Lin et al., 2026). AM-VLA-J evolves joint residuals and gripper commands in Euclidean space; FK supplies state-coupled pose conditioning and task-space supervision, but rotations are not integrated. AM-VLA-E instead evolves EE residuals directly on

$$\mathcal{M}_\mathrm{E} = (\mathbb{R}_g \times \mathbb{R}^3_{\Delta p} \times SO(3)_{\Delta R})^{2H}, \tag{2}$$

using Euclidean paths for gripper/translation and geodesics for rotation; measured joints remain fixed conditioning. Figure 1 shows the shared architecture, and Table 1 summarizes the controlled differences.

3.2 Residual Action Representation

At each control cycle, the measured state supplies joint angles θ^a_{cur} and EE pose $(p^a_{\mathrm{cur}}, q^a_{\mathrm{cur}})$ for arm a. For action step τ, we represent the targets by

$$\Delta\theta^a_\tau = \mathrm{wrap}(\theta^a_{\mathrm{tar},\tau} - \theta^a_{\mathrm{cur}}), \quad \Delta p^a_\tau = p^a_{\mathrm{tar},\tau} - p^a_{\mathrm{cur}}, \quad \Delta q^a_\tau = (q^a_{\mathrm{cur}})^{-1} \otimes q^a_{\mathrm{tar},\tau}, \tag{3}$$

while $g^a_\tau \in [0, 1]$ is an absolute gripper command. Translation therefore composes additively and rotation multiplicatively.

Table 1: The two instantiations of AM-VLA. All unlisted components are shared.

	AM-VLA-J	AM-VLA-E
Flow state	joint residuals + gripper	gripper + translation + rotation residuals
Geometry	Euclidean	$(\mathbb{R} \times \mathbb{R}^3 \times SO(3))^{2H}$
Conditioning	FK from evolving joints	measured joints/poses frozen
Task-space loss	auxiliary pose + differentiable FK	direct vector-field loss
Executed action	joint + gripper	EE pose + gripper
Integration	additive Euler	additive + Lie-group Euler

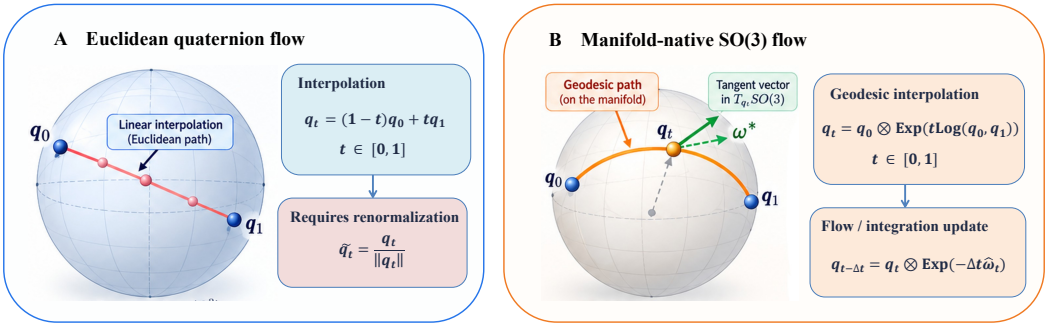

Figure 2: Euclidean quaternion flow versus manifold-native $SO(3)$ flow. Left: linear interpolation treats quaternion components as ordinary Euclidean coordinates, cuts through the interior of the unit sphere, and therefore requires renormalization. Right: the proposed construction follows the shortest rotational geodesic, represents the velocity in the tangent space, and applies Lie-group updates that preserve unit norm while respecting physical rotation geometry.

We model rotations intrinsically on $SO(3)$ and use unit quaternions only as its double-cover representation (Shoemake, 1985; Zhou et al., 2019). Endpoint signs are aligned before constructing the short arc, and body-frame tangent vectors are applied by right multiplication. Thus, the geodesic from q_0 to q_1 is

$$q_t = q_0 \otimes \mathrm{Exp}(t\,\mathrm{Log}(q_0, q_1))\,, \tag{4}$$

where $\|\,\mathrm{Log}(q_0, q_1)\|$ is the physical rotation angle. Full wrapping, reconstruction, and quaternion conventions are given in Appendix B. Figure 2 contrasts this construction with an ambient Euclidean quaternion flow: normalization recovers a valid endpoint but does not make the probability path geodesic on $SO(3)$.

3.3 Masked Conditional Flow Matching with Clean-Action Prediction

Both variants use conditional flow matching (Lipman et al., 2023; Tong et al., 2024); $SO(3)$ factors follow its Riemannian extension (Chen & Lipman, 2024). Flow time runs from clean data at $t = 0$ to the prior at $t = 1$. Given endpoints (z_0^ν, z_1^ν), flat and rotational factors use

$$\begin{aligned} x_t &= (1-t)x_0 + tx_1, & u_t^x &= x_1 - x_0, \\ q_t &= q_0 \otimes \mathrm{Exp}(t\,\mathrm{Log}(q_0, q_1)), & \omega_t^\star &= \mathrm{Log}(q_0, q_1). \end{aligned} \tag{5}$$

The action expert retains its clean-action interface. With $X_t^\nu = \Phi_\nu(z_t^\nu; s_{\mathrm{cur}})$, its predictions are bridged to velocities by

$$v_\phi^x = \frac{x_t - \hat{x}_0}{\max(t, \epsilon)}, \qquad \hat{\omega}_t = -\frac{\mathrm{Log}(q_t, \hat{q}_0)}{\max(t, \epsilon)}. \tag{6}$$

Only channels in $\mathcal{F}_\nu$ are sampled, integrated, and written back; all others are rebuilt by Φ_ν from measured state. The shared objective is

$$\mathcal{L}_{\mathrm{vf}}^\nu = \mathbb{E}_{z_0, z_1, t}\left[\sum_{c \in \mathcal{F}_\nu} \lambda_c^{\mathrm{vf}} \left\| v_\phi^c - u_t^c \right\|_2^2\right], \tag{7}$$

where rotational errors are measured in 3D body-frame coordinates, never as additive four-dimensional quaternion velocities. The path identity, prediction bridge, and masking construction are detailed in Appendix B.

3.4 AM-VLA-J: Joint-Space Flow with Manifold-Aware Supervision

AM-VLA-J evolves standardized joint residuals and gripper commands from a Gaussian prior. At each flow time, it recomposes the residual joints and rebuilds pose conditioning through FK:

$$\begin{aligned} z_0^{\mathrm{J}} &= \{(\Delta\theta_\tau^a, g_\tau^a)\}_{a,\tau}, & z_1^{\mathrm{J}} &\sim \mathcal{N}(0, I), \\ \theta_{t,\tau} &= \mathrm{wrap}(\theta_{\mathrm{cur}} + \mathrm{denorm}(\Delta\theta_{t,\tau}))\,, & P_{t,\tau}^{\mathrm{FK}} &= \mathrm{FK}(\theta_{t,\tau}), \\ X_t^{\mathrm{J}} &= \Phi_{\mathrm{J}}(z_t^{\mathrm{J}}; \theta_{\mathrm{cur}}, P_t^{\mathrm{FK}}). \end{aligned} \tag{8}$$

Thus, joint and pose input channels describe the same kinematic state instead of being perturbed independently. The decoder additionally predicts off-flow pose residuals for tracked frames. These outputs receive an auxiliary pose loss but are neither integrated nor executed. To supervise the executable action itself, the clean joint prediction passes through a differentiable FK chain,

$$(\hat{p}_\tau^{\mathrm{FK}}, \hat{q}_\tau^{\mathrm{FK}}) = \mathrm{FK}\Big(\mathrm{wrap}\Big[\theta_{\mathrm{cur}} + \mathrm{denorm}(\widehat{\Delta\theta}_\tau)\Big]\Big)\,, \tag{9}$$

and is compared with the demonstrated EE pose. The auxiliary head therefore teaches representation, whereas differentiable FK teaches execution; neither adds rotation to the integrated state.

3.5 AM-VLA-E: Manifold-Native End-Effector Flow

AM-VLA-E evolves the EE residual state in Equation 2. Flat factors use standard Gaussian priors, while rotation uses an identity-centered tangent Gaussian:

$$z_{0,\tau}^{\mathrm{E},a} = (g_\tau^a, \Delta p_\tau^a, \Delta q_\tau^a), \qquad (g_1, \Delta p_1) \sim \mathcal{N}(0, I_4), \qquad q_1 = \mathrm{Exp}(\xi),\ \xi \sim \mathcal{N}(0, \sigma_q^2 I_3). \tag{10}$$

Here σ_q is the training-set geodesic standard deviation of EE rotation residuals. Measured joints and their FK poses remain fixed throughout integration. Although the solver evolves residuals, every network evaluation preserves the pretrained absolute-pose interface:

$$g_{t,\mathrm{in}} = g_t, \qquad p_{t,\mathrm{in}} = p_{\mathrm{cur}} + \mathrm{denorm}(\Delta p_t), \qquad q_{t,\mathrm{in}} = q_{\mathrm{cur}} \otimes \Delta q_t. \tag{11}$$

The final state is composed in the same way and dispatched to the task-space controller. Predictions on joint-token channels are masked and never enter the solver state.

Figure 3 makes the conditioning distinction explicit. AM-VLA-J recomputes pose channels from the evolving joint state at every solver step, whereas AM-VLA-E keeps measured joint context fixed and updates only the EE residual factors.

3.6 Training and Sampling

For normalized quaternions, we use the smooth sign-invariant loss $\ell_{\mathrm{rot}}(\hat{q}, q) = 4(1 - \langle \hat{q}, q \rangle^2)$, a second-order surrogate for squared geodesic error at the small residual angles induced by closed-loop control. The variant objectives are

$$\begin{aligned} \mathcal{L}^{\mathrm{J}} &= \lambda_{\mathrm{vf}}\mathcal{L}_{\mathrm{vf}}^{\mathrm{J}} + \lambda_{\mathrm{pose}}\mathcal{L}_{\mathrm{pose}}^{\mathrm{J}} + \lambda_{\mathrm{FK}}\mathcal{L}_{\mathrm{FK}} + \lambda_{\mathrm{depth}}\mathcal{L}_{\mathrm{depth}}, \\ \mathcal{L}^{\mathrm{E}} &= \lambda_{\mathrm{vf}}\mathcal{L}_{\mathrm{vf}}^{\mathrm{E}} + \lambda_{\mathrm{depth}}\mathcal{L}_{\mathrm{depth}}. \end{aligned} \tag{12}$$

AM-VLA-E needs no FK loss because it directly generates task-space targets. At inference, both variants integrate from $t = 1$ to 0 with deterministic Euler steps. Flat and rotational factors update as

$$x_{t-\Delta t} = x_t - \Delta t\, v_\phi^x, \qquad q_{t-\Delta t} = q_t \otimes \mathrm{Exp}(-\Delta t\, \hat{\omega}_t), \tag{13}$$

respectively; the Lie-group step remains on $SO(3)$ without post-hoc normalization. Detailed objectives and Algorithms 1–3 are in Appendices B and C.

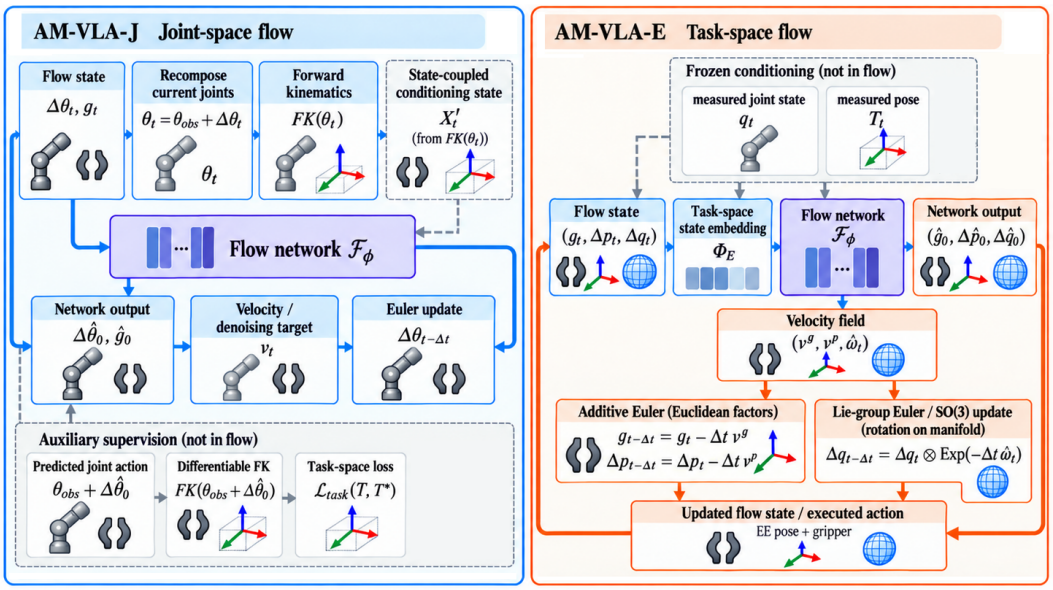

Figure 3: Generation and conditioning in AM-VLA-J and AM-VLA-E. Solid arrows denote evolved variables; dashed arrows denote conditioning-only paths. AM-VLA-J recomputes FK from evolving joints, whereas AM-VLA-E freezes measured joints and evolves EE residuals with additive and Lie-group updates.

4 Experiments

We conduct experiments on the RoboTwin 2.0 simulation benchmark (Chen et al., 2025) and real-robot tasks, comparing against strong baselines.

4.1 Evaluation in Simulation Benchmarks

Experiments setup. We train a 50-task multi-task model. We utilize a dataset mix of 50 clean and 500 randomized demonstrations per task, where randomization includes background, lighting, table height, and cluttered table. The model undergoes 200k steps of post-training. For comprehensive evaluation, we execute 100 trials per task in both clean and randomized settings aggregating to 10,000 total rollouts.

Main results. Table 2 reports success rates on RoboTwin 2.0 under clean and randomized evaluation. We compare against π_0 (Black et al., 2024), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and HoloBrain-0 (Lin et al., 2026). Both AM-VLA variants outperform strong baselines on average, and AM-VLA-E attains the best result in every column: 93.1% Avg., improving over the base model HoloBrain-0 (92.1%) by +1.0 points and over the baseline $\pi_{0.5}$ (79.8%) by +13.3 points (+30.9 over π_0). AM-VLA-J (92.5%) also surpasses HoloBrain-0 by +0.4. Generating in task space is consistently stronger than generating in joint space: AM-VLA-E improves over AM-VLA-J by +0.6 points on average (+0.9 clean, +0.3 randomized).

4.2 Evaluation in Real-world Tasks

Experimental platform and tasks. We further evaluate the proposed methods on a real-world manipulation platform to assess their practical effectiveness beyond simulation. As shown in Fig. 4, the platform uses a 6-DoF AgileX Piper equipped with both a wrist-view camera and a third-view camera. We consider two representative tabletop manipulation tasks:

- **Task 1: Pick and Place.** The robot is required to grasp a block placed at different positions on the table and place it into a bowl.

Table 2: Results on RoboTwin 2.0. We report the average success rate of 50 tasks under clean and randomized evaluation settings.

Method	Clean	Rand.	Avg.
π_0	65.9	58.4	62.2
$\pi_{0.5}$	82.7	76.8	79.8
HoloBrain-0	91.9	92.3	92.1
AM-VLA-J (Ours)	92.6	92.3	92.5
AM-VLA-E (Ours)	**93.5**	**92.6**	**93.1**

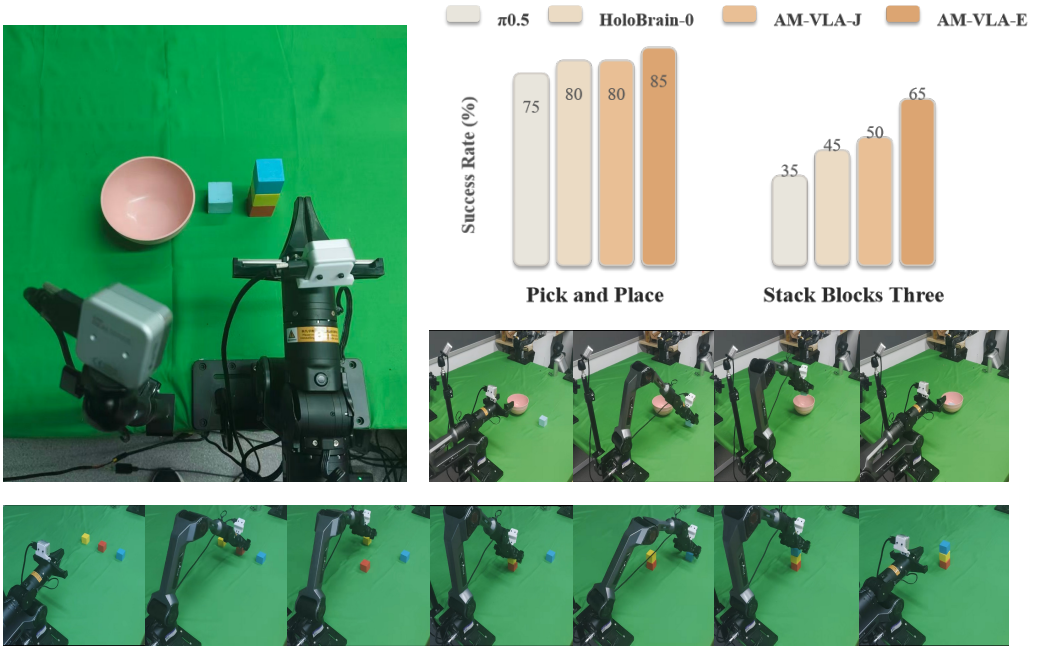

Figure 4: Real-robot evaluation. Left: the experimental facility and scene settings. Upper right: success rates of $\pi_{0.5}$, HoloBrain-0, AM-VLA-J, and AM-VLA-E on Pick and Place and Stack Blocks Three; AM-VLA-E achieves the highest success rate on both tasks. Lower right: representative execution sequences for the two tasks.

- **Task 2: Stack Blocks Three.** The robot is required to stack three blocks of different colors in a specified order, from bottom to top: red, yellow, and blue.

These two tasks test different aspects of real-world manipulation. Task 1 mainly evaluates object localization, grasping robustness, and target-directed placement, whereas Task 2 requires not only repeated pick-and-place operations but also order-sensitive sequential reasoning and more precise motion execution.

Data collection and evaluation protocol. For each real-world task, we collect 100 demonstration trajectories for training. During evaluation, each task is repeated 20 times, and we report the number of successful executions and the corresponding success rate.

Results. We compare $\pi_{0.5}$ (Physical Intelligence et al., 2025), HoloBrain-0 (Lin et al., 2026), AM-VLA-J, and AM-VLA-E. Figure 4 summarizes the real-world results alongside representative task executions. Both variants of AM-VLA remain competitive or superior to the baselines

Table 3: Ablation of manifold-aware task-space supervision in AM-VLA-J.

Loss Setting	Hanging Mug	Move Stapler Pad	Turn Switch	Place Mouse Pad	Place A2B Right	Avg.
No Pose & FK	53	68	83	77	74	71.0
No Pose	64	75	79	73	85	75.2
No FK	56	68	85	75	79	72.6
Full Loss	65	74	78	80	87	76.8

Table 4: Ablation of generation space and rotational geometry. Euclidean rot. denotes task-space generation with quaternion channels treated as Euclidean coordinates rather than an $SO(3)$ factor.

Method	Hanging Mug	Move Stapler Pad	Turn Switch	Place Mouse Pad	Place A2B Right	Avg.
HoloBrain-0	53	73	83	82	74	73.0
AM-VLA-J	65	74	78	80	87	76.8
AM-VLA-E (Euclidean rot.)	52	67	80	73	75	69.4
AM-VLA-E ($SO(3)$ rot.)	68	82	85	84	89	81.6

in real deployment. AM-VLA-E achieves the best performance on both tasks, which suggests that manifold-native task-space generation is particularly beneficial for more challenging long-horizon manipulation tasks requiring accurate placement and sequential control.

4.3 Ablation Study

Due to the computational cost of retraining and evaluating every ablation over the full 50-task RoboTwin 2.0 benchmark (Chen et al., 2025), we conduct controlled ablations on a fixed five-task *stress-test subset*: *Hanging Mug*, *Move Stapler Pad*, *Turn Switch*, *Place Mouse Pad*, and *Place A2B Right*. These are the five tasks on which the HoloBrain-0 base model (Lin et al., 2026) obtains the lowest success rates among the 50 RoboTwin 2.0 tasks.

Task-Space Supervision in AM-VLA-J. Table 3 separates representation-level supervision from execution-level supervision. Removing both the auxiliary pose loss and the differentiable FK loss reduces the average success rate from 76.8% to 71.0%, showing that task-space supervision contributes substantially even when the generative state remains in joint space. Using only the auxiliary pose objective (*No FK*) improves the average from 71.0% to 72.6% ($+1.6$ points), whereas using only the differentiable FK objective (*No Pose*) reaches 75.2% ($+4.2$ points). Overall, the results indicate that task-space information is useful for a joint-space generator.

Generation Space and Rotational Geometry. We next study the interaction between the space in which actions are generated and the geometry used for rotational variables.

The results show that replacing the Euclidean treatment of quaternion channels with the proposed $SO(3)$ flow increases average success from 69.4% to 81.6%, an absolute improvement of 12.2 points while keeping the task-space generation state and execution interface unchanged. The gain is consistent across all five tasks: $+16$, $+15$, $+5$, $+11$, and $+14$ points on *Hanging Mug*, *Move Stapler Pad*, *Turn Switch*, *Place Mouse Pad*, and *Place A2B Right*, respectively. This controlled comparison provides direct evidence that the geometry of the rotational generative process is critical in this task-space configuration.

Summary. The ablations support two complementary conclusions. First, task-space geometric information is useful even for a joint-space generator. Second, manifold-native $SO(3)$ dynamics provide substantially larger gains when the generative process evolves directly in task space.

5 Conclusion

We presented **Action-Manifold VLA (AM-VLA)**, a unified framework for studying two coupled design choices in continuous VLA action generation: the space in which the generative dynamics evolve and the geometry imposed on the action variables. AM-VLA-J retains directly executable joint-space generation while introducing manifold-aware task-space supervision through auxiliary pose prediction and differentiable forward kinematics. AM-VLA-E instead evolves end-effector residuals directly on a structured product manifold, using Euclidean flows for translation and gripper variables and a geodesic flow on $SO(3)$ for rotations.

Experiments show that both designs improve upon the HoloBrain-0 backbone on the 50-task RoboTwin 2.0 benchmark, increasing average success from 92.1% to 92.5% with AM-VLA-J and 93.1% with AM-VLA-E. The ablation studies indicate that the generation space and the geometry of its variables should be designed jointly rather than treated as independent choices. Future work should test this principle across additional embodiments, action parameterizations, and task-space controllers, and extend manifold-aware generation to other constrained components of robot action spaces.

AI use statement

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for polishing the language of our paper, making the expression more fluent. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Shuai Bai et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations*, 2024.

Tianxing Chen et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

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

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

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.

Xuewu Lin, Tianwei Lin, Yun Du, Hongyu Xie, Yiwei Jin, Jiawei Li, Shijie Wu, Qingze Wang, Mengdi Li, Mengao Zhao, Ziang Li, Chaodong Huang, Hongzhe Bi, Lichao Huang, and Zhizhong Su. HoloBrain-0 technical report. *arXiv preprint arXiv:2602.12062*, 2026.

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

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Physical Intelligence et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 245–254, 1985. doi: 10.1145/325334.325242.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

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.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

A Preliminaries

A.1 Flow Matching

Flow matching (FM) (Lipman et al., 2023) learns a time-dependent vector field $v_\psi(a, t, o)$ whose ODE

$$\dot{a}_t = v_\psi(a_t, t, o) \tag{14}$$

transports a prior distribution p_1 to the data distribution p_0 when integrated backward in time, conditioned on observation o. Conditional flow matching (CFM) (Tong et al., 2024) enables simulation-free training by regressing the vector field toward the velocity of a prescribed conditional path connecting a data sample $a_0 \sim p_0$ and a prior sample $a_1 \sim p_1$. For Euclidean variables, we use the linear conditional path

$$a_t = (1-t)a_0 + ta_1, \qquad u_t = \dot{a}_t = a_1 - a_0. \tag{15}$$

The corresponding CFM objective is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\substack{t\sim\mathcal{U}[0,1],\\ a_0\sim p_0,\, a_1\sim p_1}} \left[\left\| v_\psi(a_t, t, o) - u_t \right\|_2^2 \right]. \tag{16}$$

Along each conditional path, u_t is independent of t, yielding a time-independent target magnitude, while the interpolation itself follows a straight Euclidean path. Note that this property pertains to the conditional paths; the marginal vector field learned by CFM need not induce globally straight generation trajectories.

x-prediction and velocity parameterization. For the linear conditional path,

$$a_t = a_0 + t u_t, \tag{17}$$

so clean-data and velocity predictions are related by

$$\hat{a}_0 = a_t - t\, v_\psi(a_t, t, o), \qquad v_\psi(a_t, t, o) = \frac{a_t - \hat{a}_0}{t}, \tag{18}$$

for $t > 0$. In practice, t is truncated from below at a small ϵ when evaluating the velocity parameterization. The corresponding prediction errors satisfy

$$\|\hat{a}_0 - a_0\|_2^2 = t^2 \|v_\psi(a_t, t, o) - u_t\|_2^2, \tag{19}$$

showing that an unweighted clean-data objective induces a t^2 weighting of the equivalent velocity error.

A.2 Riemannian Flow Matching

Riemannian flow matching (RFM) (Chen & Lipman, 2024) extends flow matching to a Riemannian manifold $\mathcal{M}$ by replacing Euclidean interpolation with geodesic conditional paths. Given endpoints $a_0, a_1 \in \mathcal{M}$, a minimizing geodesic can be written locally as

$$a_t = \mathrm{Exp}_{a_0}\left(t\ \mathrm{Log}_{a_0}(a_1)\right), \tag{20}$$

with conditional velocity

$$u_t = \frac{\mathrm{Log}_{a_t}(a_1)}{1-t} \in T_{a_t}\mathcal{M}. \tag{21}$$

Along a constant-speed minimizing geodesic,

$$\|u_t\|_{a_t} = d_{\mathcal{M}}(a_0, a_1), \tag{22}$$

so the norm of the conditional target field is constant in time, analogous to the constant target magnitude of linear Euclidean interpolation. Under our time convention, generation integrates the learned field backward from the prior to the data endpoint. A first-order manifold update can be written as

$$a_{t-\Delta t} = \mathrm{Exp}_{a_t}\left(-\Delta t\, v_\psi(a_t, t, o)\right). \tag{23}$$

For a product manifold

$$\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_K \tag{24}$$

equipped with a product metric, exponential and logarithmic maps, geodesics, and tangent-space norms decompose across factors. This decomposition provides a natural construction for heterogeneous action variables containing both Euclidean and rotational components.

Rotation geometry and quaternion representation. We model physical rotations on $SO(3)$ and use unit quaternions $q \in S^3 \subset \mathbb{R}^4$ only as a computational representation of its double cover. Thus, q and $-q$ represent the same element of $SO(3)$. Before constructing a relative rotation or interpolation path, we sign-align a pair of quaternion representatives so that $\langle q_0, q_1 \rangle \geq 0$, selecting the short-arc representative. The physical geodesic distance is

$$d_{SO(3)}(q_0, q_1) = 2 \arccos(|\langle q_0, q_1 \rangle|) \in [0, \pi]. \tag{25}$$

We express tangent vectors in body-frame Lie-algebra coordinates $\omega \in \mathbb{R}^3 \cong \mathfrak{so}(3)$, with $\|\omega\|$ equal to the physical rotation angle. The quaternion realization of the exponential map is

$$\mathrm{Exp}(\omega) = \left[\cos\frac{\|\omega\|}{2},\ \frac{\sin(\|\omega\|/2)}{\|\omega\|}\,\omega\right], \tag{26}$$

with its continuous limit at $\|\omega\| = 0$. For two sign-aligned unit-quaternion representatives, the corresponding logarithmic map is

$$\mathrm{Log}(q_a, q_b) = \frac{2\ \mathrm{atan2}(\|\mathbf{u}\|, w)}{\|\mathbf{u}\|}\,\mathbf{u}, \qquad [w, \mathbf{u}] = q_a^{-1} \otimes q_b. \tag{27}$$

Hence $\| \operatorname{Log}(q_a, q_b)\| = d_{SO(3)}(q_a, q_b)$. A constant-speed shortest geodesic from q_0 to q_1 is represented by

$$q_t = q_0 \otimes \operatorname{Exp}(t \operatorname{Log}(q_0, q_1)), \tag{28}$$

which is equivalent to Slerp between the sign-aligned quaternion representatives (Shoemake, 1985). For the rotational prior used in the main text, we sample an isotropic Gaussian in the Lie algebra,

$$\xi \sim \mathcal{N}(0, \sigma^2 I_3), \qquad \xi \in \mathfrak{so}(3) \cong \mathbb{R}^3, \tag{29}$$

and map it to a unit-quaternion representative of an $SO(3)$ rotation via $q_1 = \operatorname{Exp}(\xi)$. Throughout the paper, references to rotational geodesics, tangent norms, and Riemannian flow use the $SO(3)$ physical-angle metric; S^3 denotes only the unit-quaternion representation space.

B Additional Method Details

B.1 Residual Labels and Reconstruction

Let $\tau \in \{1, \dots, H\}$ denote the action-chunk index. Joint wrapping in Equation 3 is performed in physical radians before standardization and only on channels selected by the revolute-joint wrap mask; limited joints are not wrapped. The gripper occupies an additional joint slot in the shared token layout but remains an absolute aperture rather than a residual. Absolute EE targets are reconstructed by

$$p^a_{\mathrm{tar},\tau} = p^a_{\mathrm{cur}} + \Delta p^a_\tau, \qquad q^a_{\mathrm{tar},\tau} = q^a_{\mathrm{cur}} \otimes \Delta q^a_\tau. \tag{30}$$

The identity quaternion $e = (1, 0, 0, 0)$ therefore represents zero rotational displacement.

B.2 Numerical Rotation Conventions

Quaternions are stored in $wxyz$ order, normalized with $q/\sqrt{\|q\|^2 + \epsilon^2}$, and sign-aligned before interpolation to select the short-arc representative (Shoemake, 1985). Rotational velocities use body-frame Lie-algebra coordinates and right-multiplicative updates,

$$q^+ = q \otimes \operatorname{Exp}(\delta\omega). \tag{31}$$

The explicit exponential and logarithmic maps, physical-angle metric, and prior construction are given in Appendix A. We use the product $\mathbb{R}^3 \times SO(3)$ rather than coupled $SE(3)$ geodesics so that each factor has a closed-form path matching the pretrained channel semantics.

B.3 Conditional Paths, Assembly, and Prediction Bridge

For a clean state z^ν_0 and prior sample z^ν_1, Equation 5 defines componentwise constant-speed paths. Along the rotational path,

$$\operatorname{Log}(q_t, q_1) = (1 - t) \operatorname{Log}(q_0, q_1), \tag{32}$$

so $\omega^\star_t = \operatorname{Log}(q_t, q_1)/(1 - t) = \operatorname{Log}(q_0, q_1)$. This constancy applies to each conditional path; the learned marginal field is generally not constant.

The assembly operator $X^\nu_t = \Phi_\nu(z^\nu_t; s_{\mathrm{cur}})$ inserts evolving variables into the shared token representation and reconstructs every non-flow channel from measured state. Only $\mathcal{F}_\nu$ persists across integration steps, so masked predictions cannot contaminate subsequent conditioning. The decoder predicts

$$\hat{z}^\nu_0 = \operatorname{Read}_\nu[F_\phi(X^\nu_t, t, o)], \tag{33}$$

where Read_ν selects active channels. Equation 6 is equivalent to $\hat{x}_0 = x_t - t v^x_\phi$ and $\hat{q}_0 = q_t \otimes \operatorname{Exp}(-t\hat{\omega}_t)$, preserving the pretrained quaternion-valued clean prediction while training and sampling in tangent velocity coordinates.

B.4 Variant Interfaces

For AM-VLA-J, $z^{\mathrm{J}}_0 = \{(\Delta\theta^a_\tau, g^a_\tau)\}_{a,\tau}$ and $z^{\mathrm{J}}_1 \sim \mathcal{N}(0, I)$. At each flow time,

$$\theta_{t,\tau} = \operatorname{wrap}(\theta_{\mathrm{cur}} + \operatorname{denorm}(\Delta\theta_{t,\tau})), \qquad P^{\mathrm{FK}}_{t,\tau} = \mathrm{FK}(\theta_{t,\tau}), \tag{34}$$

and $X_t^{\mathrm{J}} = \Phi_{\mathrm{J}}(z_t^{\mathrm{J}}; \theta_{\mathrm{cur}}, P_t^{\mathrm{FK}})$. This input-side FK need not be differentiable. Separately, auxiliary predictions $(\widehat{\Delta p}_{i,\tau}, \widehat{\Delta q}_{i,\tau})$ supervise tracked frames $i \in \mathcal{I}$. The clean joint prediction is reconstructed and passed through differentiable FK,

$$(\hat{p}_\tau^{\mathrm{FK}}, \hat{q}_\tau^{\mathrm{FK}}) = \mathrm{FK}(\mathrm{wrap}(\theta_{\mathrm{cur}} + \mathrm{denorm}(\widehat{\Delta\theta}_\tau))), \tag{35}$$

so task-space error reaches the executed joint command. The auxiliary pose head is not integrated or executed.

For AM-VLA-E, $z_{0,\tau}^{\mathrm{E},a} = (g_\tau^a, \Delta p_\tau^a, \Delta q_\tau^a)$. Its flat priors are standard Gaussian, while $q_1 = \mathrm{Exp}(\xi)$ for $\xi \sim \mathcal{N}(0, \sigma_q^2 I_3)$, with σ_q set to the training-set geodesic standard deviation. Measured joints and FK poses are replicated over the horizon and never written back. Although the solver state is residual, network inputs retain absolute pose semantics:

$$g_{t,\mathrm{in}} = g_t, \qquad p_{t,\mathrm{in}} = p_{\mathrm{cur}} + \mathrm{denorm}(\Delta p_t), \qquad q_{t,\mathrm{in}} = q_{\mathrm{cur}} \otimes \Delta q_t. \tag{36}$$

The final state is composed identically and sent to the task-space controller, with the gripper destandardized and clamped to $[0, 1]$.

B.5 Auxiliary Objectives and Sampling

For $\Delta\phi(\hat{q}, q) = 2\arccos|\langle \hat{q}, q\rangle|$, the rotational surrogate satisfies

$$\ell_{\mathrm{rot}}(\hat{q}, q) = 4(1 - \langle \hat{q}, q\rangle^2) = 4\sin^2(\Delta\phi/2) = \Delta\phi^2 + O(\Delta\phi^4). \tag{37}$$

It is sign invariant and locally equivalent to squared geodesic error. The AM-VLA-J auxiliary objectives are

$$\mathcal{L}_{\mathrm{pose}}^{\mathrm{J}} = \sum_{i\in\mathcal{I},\tau} \left[\lambda_p \|\widehat{\Delta p}_{i,\tau} - \Delta p_{i,\tau}\|_2^2 + \lambda_q \ell_{\mathrm{rot}}(\widehat{\Delta q}_{i,\tau}, \Delta q_{i,\tau})\right], \tag{38}$$

$$\mathcal{L}_{\mathrm{FK}} = \sum_{\tau} \left[\lambda_p^{\mathrm{FK}} \|\hat{p}_\tau^{\mathrm{FK}} - p_{\mathrm{tar},\tau}\|_2^2 + \lambda_q^{\mathrm{FK}} \ell_{\mathrm{rot}}(\hat{q}_\tau^{\mathrm{FK}}, q_{\mathrm{tar},\tau})\right]. \tag{39}$$

Together with the vector-field and inherited HoloBrain-0 depth objectives (Lin et al., 2026), these yield Equation 12.

Deterministic sampling uses $x_{t-\Delta t} = x_t - \Delta t\, v_\phi^x$ for flat factors and

$$q_{t-\Delta t} = q_t \otimes \mathrm{Exp}(-\Delta t\, \hat{\omega}_t) \tag{40}$$

for rotation, preserving unit norm without post-hoc normalization.

C Training and Sampling Algorithms

Algorithm 1: Unified training step (both variants)

1. Build residual labels from the measured base state; standardize flat channels; sign-align quaternion endpoint pairs;
2. Sample prior z_1^ν componentwise: $\mathcal{N}(0, I)$ on flat factors, $\mathrm{Exp}(\xi)$ with $\xi \sim \mathcal{N}(0, \sigma_q^2 I_3)$ on $SO(3)$; sample $t \sim \mathcal{U}[\epsilon, 1-\epsilon]$;
3. Form the path z_t^ν (linear / short-arc geodesic) and the target field u_t^ν ($z_1 - z_0$ / $\mathrm{Log}(q_0, q_1)$);
4. Assemble the network input $X_t^\nu = \Phi_\nu(z_t^\nu; s_{\mathrm{cur}})$ per variant (Table 1);
5. Forward $F_\phi(X_t^\nu, t, o)$; read flow-channel (and, for J, auxiliary-head) outputs under the output mask; bridge clean predictions to velocities;
6. Compute $\mathcal{L}^\nu$ and backpropagate;

Algorithm 2: AM-VLA-J sampling (additive Euler)

Sample $z \sim \mathcal{N}(0, I)$ over standardized joint-residual and gripper channels;
for $k = 0$ **to** $K - 1$ **do**
  $t \leftarrow 1 - k/K, \quad \Delta t \leftarrow 1/K$;
  Recompute $\theta_{t,\tau}$ and $P^{\mathrm{FK}}_{t,\tau}$ from z; assemble $X^{\mathrm{J}}_t = \Phi_{\mathrm{J}}(z; \theta_{\mathrm{cur}}, P^{\mathrm{FK}}_t)$;
  $\hat{z}_0 \leftarrow \mathrm{Read}_{\mathrm{J}}[F_\phi(X^{\mathrm{J}}_t, t, o)]; \quad v \leftarrow (z - \hat{z}_0)/\max(t, \epsilon)$;
  $z \leftarrow z - \Delta t\, v$;
end
return executed action: $\mathrm{wrap}(\theta_{\mathrm{cur}} + \mathrm{denorm}(\widehat{\Delta\theta}))$ and $\mathrm{clamp}(\mathrm{denorm}(\hat{g}))$;

Algorithm 3: AM-VLA-E sampling (additive + Lie-group Euler)

Sample $g_1, \Delta p_1 \sim \mathcal{N}(0, I); \quad \Delta q_1 \leftarrow \mathrm{Exp}(\xi)$ with $\xi \sim \mathcal{N}(0, \sigma_q^2 I_3)$;
for $k = 0$ **to** $K - 1$ **do**
  $t \leftarrow 1 - k/K, \quad \Delta t \leftarrow 1/K$;
  Rebuild conditioning slots from the measured state; assemble $X^{\mathrm{E}}_t = \Phi_{\mathrm{E}}(z^{\mathrm{E}}_t; s_{\mathrm{cur}})$;
  Read EE-slot outputs under the mask; bridge to velocities $(v^g, v^p, \hat{\omega}_t)$;
  $g \leftarrow g - \Delta t\, v^g; \quad \Delta p \leftarrow \Delta p - \Delta t\, v^p; \quad \Delta q \leftarrow \Delta q \otimes \mathrm{Exp}(-\Delta t\, \hat{\omega}_t)$;
end
return EE targets $(p_{\mathrm{tar}}, q_{\mathrm{tar}}, g_{\mathrm{tar}})$ composed with the measured pose; dispatch to the task-space controller;