GeomVLA: Unifying Scene, Motion, and Action in 3D

Anonymous Author(s)
Affiliation
Address
email

Abstract: We present **GeomVLA**, a Vision-Language-Action (VLA) model that unifies perception, latent scene motion prediction, and action generation within a shared robot-centric 3D coordinate frame. Our approach lifts VLM features into spatially grounded 3D scene tokens by adjusting their positional encodings while preserving pretrained visual-language priors. We further introduce a *3D Scene Trajectory Denoiser*, a task-conditioned latent motion representation that captures how scene points are expected to move in 3D space. Rather than serving as an explicit open-loop plan, the predicted scene trajectory acts as a latent geometric reasoning process that conditions a 3D flow-based action denoiser through 3D-aware attention. GeomVLA achieves state-of-the-art performance on CALVIN and LIBERO, competitive performance on RoboTwin2.0, and consistently outperforms strong baselines in real-world manipulation settings without access to large scale pretraining. Extensive ablations demonstrate that future-motion reasoning alone is insufficient: the primary gains arise from maintaining geometric consistency between scene representation, motion prediction, and robot actions throughout the entire perception-to-action pipeline.

Keywords: Vision-Language-Action Models, 3D Motion Prediction

1 Introduction

Vision-language-action (VLA) models [1, 2, 3] have recently emerged as a promising paradigm for language-conditioned robot manipulation by leveraging the broad semantic knowledge of pretrained vision-language models (VLMs) [4, 5, 6]. However, most existing VLAs operate primarily on 2D images while robot manipulation fundamentally requires reasoning about geometry, contact, and motion in 3D space [7, 8, 3]. Consequently, policies must implicitly recover complex spatial interactions from semantic image features and sparse action supervision, creating a fundamental mismatch between representation and control.

One natural solution is to represent both the scene and robot actions directly in metric 3D coordinates. Recent 3D manipulation policies demonstrate that explicit geometric representations substantially improve spatial reasoning and data efficiency [9, 10, 11, 12]. More recent VLA systems further attempt to combine such geometric representations with pretrained VLMs through lifted 3D features, spatial positional encodings, or dual-stream architectures [13, 14, 15, 16, 17]. While these approaches improve geometric grounding, they primarily focus on static scene 3D representation and action generation, without explicitly addressing how future reasoning should be represented and integrated within the control pipeline.

A complementary line of work introduces future prediction as an intermediate reasoning process for robot manipulation [18, 19, 20, 21]. Rather than directly mapping observations to actions, these methods encourage policies to reason about how the scene is expected to evolve through predicted videos, optical flow, visual traces, or trajectories. Such future-oriented representations expose latent task dynamics and improve long-horizon planning. However, these methods typically perform future reasoning in representation spaces that are disconnected from robot control, such as pixel space for video prediction and optical flow, or image space for trajectories. We argue that the limitation

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

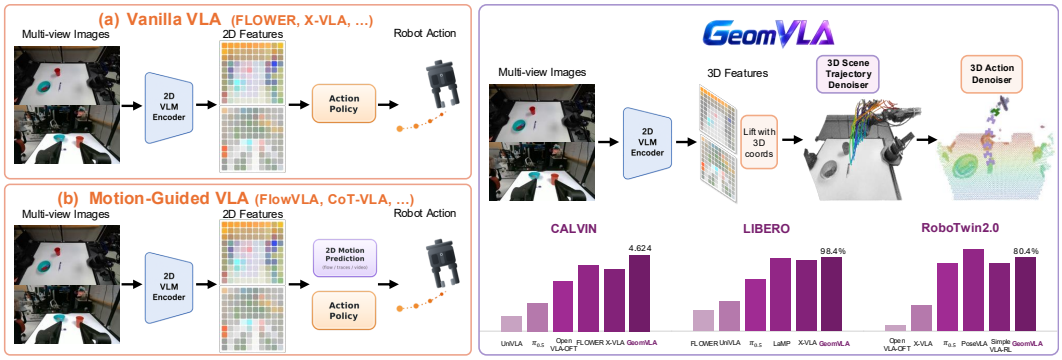

Figure 1: **GeomVLA enables motion-guided 3D visual-language-action learning by bridging perception, latent future motion, and robot control in a shared 3D space.** Conventional VLAs either map visual observations directly to actions or reason about motion in feature spaces misaligned with robot actions. GeomVLA lifts pretrained 2D VLM features into a geometrically grounded 3D representation, predicts latent 3D scene trajectories as an intermediate reasoning signal, and conditions a 3D action denoiser on the resulting scene-motion representation to generate robot actions. This design improves performance both in simulation and real-world.

is not that current VLAs lack explicit 3D representations or future-motion reasoning individually, but that these components operate in different representation spaces (Figure 1). Consequently, scene understanding, future reasoning, and robot control cannot fully interact through a common geometric representation.

In this work, we introduce **GeomVLA**, a 3D motion-aware VLA that unifies perception, latent future-motion reasoning, and action generation within a shared robot-centric 3D coordinate frame. We hypothesize that the effectiveness of future-motion reasoning depends not only on predicting future scene evolution, but also on maintaining geometric consistency between perception, reasoning, and control. Our key insight is that **future-motion reasoning is most effective when scene representation, motion reasoning, and robot actions are expressed within the same geometric coordinate frame.** Concretely, GeomVLA first lifts pretrained VLM features into spatially grounded 3D scene tokens while preserving pretrained visual-language priors. We then introduce an *3D Scene Trajectory Denoiser*, a task-conditioned latent motion representation that predicts how scene points move in 3D space. Rather than serving as an explicit open-loop planner, the predicted trajectory field acts as a latent geometric reasoning process that conditions a 3D flow-based action policy through geometry-aware spatial attention. Consequently, scene features, motion representations, and robot trajectories interact within a unified metric coordinate frame throughout action generation.

We evaluate GeomVLA across CALVIN [22], LIBERO [23], RoboTwin2.0 [24], and real-world manipulation benchmarks. Our approach achieves state-of-the-art performance on CALVIN and LIBERO, competitive performance on RoboTwin2.0, and consistently outperforms strong baselines in real-world settings without requiring large-scale robot pretraining. Extensive ablations further demonstrate that future-motion reasoning alone is insufficient: the primary gains arise from maintaining geometric consistency between scene representation, motion prediction, and robot actions throughout the entire perception-to-action pipeline.

In summary, our contributions are as follows:

- We introduce GeomVLA, a Vision-Language-Action framework that unifies scene representation, latent future-motion reasoning, and action generation within a shared robot-centric 3D coordinate frame.
- We propose an *3D Scene Trajectory Denoiser*, a task-conditioned latent motion representation that predicts future scene motion directly in 3D space and conditions downstream action generation through geometry-aware attention.

- We achieve state-of-the-art performance on CALVIN and LIBERO, competitive performance on RoboTwin2.0, and strong real-world manipulation results without large-scale robot pretraining.
- We demonstrate through extensive ablations that maintaining geometric consistency between scene representation, motion prediction, and robot actions is critical for effective long-horizon manipulation.

2 Related Work

3D Vision-Language-Action Models. Recent 3D manipulation policies [10, 9, 25, 11, 26, 27, 28, 29] improve spatial reasoning and data efficiency by representing scene observations and robot actions within a shared geometric space. More recent 3D VLA models [13, 14, 17, 16, 30, 31, 32, 33, 34] further incorporate pretrained VLM semantics into such frameworks. Existing approaches generally either inject 3D representations directly into pretrained VLM backbones through 3D positional encodings or couple VLM features with separate 3D encoders for action prediction. While these designs improve geometric grounding, they primarily focus on aligning scene representation and action generation, without explicitly addressing how intermediate reasoning processes should be represented within the control pipeline. In contrast, our method unifies scene representation, future-motion reasoning, and action generation within the same robot-centric 3D coordinate frame.

Motion Priors for Visuomotor Policy. Recent visuomotor policies increasingly incorporate future prediction as an auxiliary structure for policy learning. Prior work explores video prediction [35, 36, 18, 37, 38], optical flow or visual traces [39, 19, 40, 20, 41, 42, 43, 44], object/gripper trajectories [21, 45, 46, 47, 48], and depth [49, 50] to provide temporal cues beyond direct action supervision. These methods differ primarily in how future motion is represented and how it interacts with downstream control. Some approaches use explicit plan-then-act formulations [46, 20], where predicted trajectories or flow fields are generated before policy execution, while others incorporate motion prediction as a latent supervisory signal within the policy itself [43]. These approaches thus predominantly reason in pixel or image-space representations, even when actions are ultimately executed in 3D. In contrast, our method represents future scene motion directly within the same robot-centric 3D coordinate frame used for scene representation and action generation.

3 Method

The architecture of GeomVLA is depicted in Figure 2. Given RGB-D observations $\{(I_v, D_v)\}_{v=1}^{V}$, a language instruction ℓ and proprioception $\mathbf{p}$, the model first lifts VLM contextualized features to 3D by assigning 3D positional encodings (§3.1). It then predicts a task-conditioned latent 3D motion representation through an *3D Scene Trajectory Denoiser* (§3.2), which conditions a flow-based action denoiser through 3D-aware attentions to output an action chunk $A = (a_1, \ldots, a_T)$ (§3.3). Scene tokens, 3D motion trajectories and action tokens are all predicted in the frame.

3.1 3D Scene Features

A pretrained VLM [6] encodes RGB observations and the language instruction ℓ into patch-level visual tokens f_v and language embeddings f_ℓ. Using depth and camera calibration, visual tokens f_v are lifted into 3D scene tokens f_v^{lift} with explicit positions in the robot base frame. Proprioceptive features $\mathbf{p}$ are represented as learnable tokens anchored at the current end-effector pose.

3.2 3D Scene Trajectory Denoiser

We introduce a *3D Scene Trajectory Denoiser* that predicts future scene motion directly in the robot base frame, serving as a latent geometric reasoning representation for the downstream action prediction. Given a front-view RGB-D observation (I, D), language instruction ℓ, and camera calibration, we sample a regular grid of image points and lift them into 3D point anchors $\{q_i^{(0)}\}_{i=1}^{N}$ in the robot

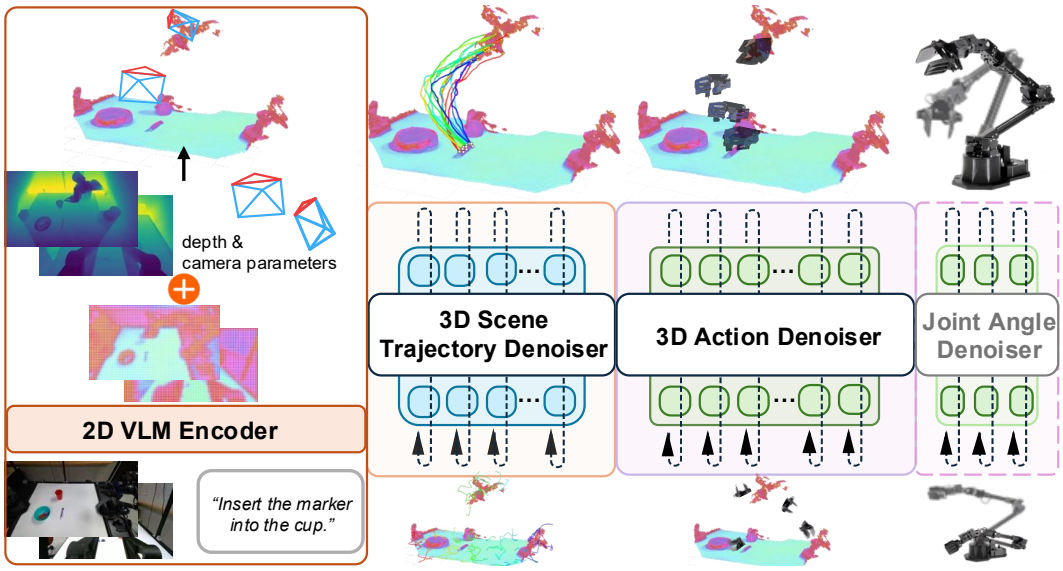

Figure 2: **Detailed model architecture.** Left: A shared 2D VLM encodes multi-view RGB observations and language instructions. Depth and camera geometry lift image features into 3D scene tokens with positions in the robot base frame. Middle: The 3D Scene Trajectory Denoiser predicts future motion trajectories for query anchors, whose latent motion tokens are fused with the 3D scene tokens. Right: The 3D Action Denoiser predicts end-effector trajectory chunks conditioned on the fused scene-motion representation, language features, and robot proprioception. Optionally, the Joint Angle Denoiser refines the predicted action latents into joint-angle commands.

base frame. Our motion decoder predicts future 3D point trajectories in terms of per-frame 3D displacement sequences:

$$\mathcal{F} = \{\Delta q_i^{(t)}\}_{i=1,t=1}^{N,H} \in \mathbb{R}^{N\times H\times 3}, \tag{1}$$

where $\Delta q_i^{(t)}$ denotes the predicted displacement increment of point anchor i at timestep t. Future point 3D positions are recovered through cumulative integration $q_i^{(t)} = q_i^{(0)} + \sum_{s=1}^{t} \Delta q_i^{(s)}$.

Take VLM-encoded f_v, f_ℓ from §3.1. Visual features f_v are bilinearly sampled at the anchor locations and summed with a projection of their 3D coordinates, producing task-conditioned 3D scene tokens $\{c_i\}_{i=1}^{N}$ using local geometry-aware attention.

Conditioned on these tokens, a lightweight flow-matching motion decoder predicts future displacement trajectories jointly across all point anchors and timesteps. The decoder factorizes spatial and temporal interactions through alternating spatial attention over 3D anchors and temporal attention across future trajectory steps. Spatial attention operates over serialized local 3D neighborhoods to preserve geometric locality efficiently.

The *3D Scene Trajectory Denoiser* is trained using 3D point tracks extracted from demonstration videos. We supervise future displacement trajectories in the robot base frame using a motion-weighted flow-matching objective:

$$\mathcal{L}_{\text{traj}} = \mathbb{E}_{\tau,\mathcal{F}_0}\left[\sum_{i=1}^{N}\sum_{t=1}^{H} w_{i,t}\left\|u_\theta(\mathcal{F}_\tau,\tau,\mathcal{C})_{i,t} - u^\star_{i,t}\right\|_2^2\right], \tag{2}$$

where $\mathcal{C}$ denotes the visual-language conditioning context and $w_{i,t}$ emphasizes motion-relevant points while preserving static geometric structure (see Appendix for details).

3.3 3D Action Denoiser

Our action denoiser generates end-effector trajectories through rectified-flow action denoising conditioned on scene f_v^{lift}, language f_ℓ and proprioception $\mathbf{p}$ from §3.1, along with latent motion representations from §3.2.

To incorporate future-motion reasoning, we initialize the *3D Scene Trajectory Denoiser* from Gaussian noise and run a velocity prediction step. The resulting hidden states are pooled across the temporal dimension to obtain motion-aware 3D tokens $\{z_i^{\mathrm{m}}\}_{i=1}^{N}$, one per 3D anchor point. Scene tokens are then refined through geometry-aware cross-attention to these motion tokens, allowing scene tokens to interact with predicted future motion before action generation.

The 3D action denoiser represents noisy end-effector trajectories as spatial action tokens with explicit 3D positions. A stack of geometry-aware transformer blocks alternates cross-attention to scene and language tokens with self-attention over action trajectories:

$$f^{\mathrm{a}} \leftarrow \mathrm{CrossAttn}_{\mathrm{3D}}\big(f^{\mathrm{a}}, [f^{\mathrm{s}}; f^{\mathrm{t}}]\big), \tag{3}$$

$$f^{\mathrm{a}} \leftarrow \mathrm{SelfAttn}_{\mathrm{3D}}(f^{\mathrm{a}}), \tag{4}$$

$$f^{\mathrm{a}} \leftarrow \mathrm{adaLN\text{-}FFN}(f^{\mathrm{a}}). \tag{5}$$

All attentions operate directly over 3D positions using rotary 3D positional embeddings.

The 3D action denoiser predicts rectified-flow velocities for end-effector translation and rotation jointly with gripper actions. We optimize the standard rectified-flow objective:

$$\mathcal{L}_\pi = \big\|(\tau^0 - \epsilon) - v_\theta(o, \ell, \tau^i, i, \mathcal{F})\big\|_2^2 + \mathrm{BCE}(f_\theta^{\mathrm{open}}, a^{\mathrm{open}}), \tag{6}$$

where τ^0 denotes the clean action trajectory and $\epsilon \sim \mathcal{N}(0, I)$ is Gaussian noise.

3.4 Training

Training proceeds in two stages. We first train the *3D Scene Trajectory Denoiser* using frozen VLM features and 3D point-track supervision extracted from demonstrations. We then jointly train the 3D action denoiser together with the pretrained *3D Scene Trajectory Denoiser* and VLM backbone. GeomVLA is trained solely on benchmark demonstrations without prior large-scale pretraining.

Implementation details To obtain supervision for the trajectory predictor, we use SpatialTrackerV2 [51] to estimate 3D point trajectories from demonstration videos. The predictor uses a 20×20 query grid of 3D anchors. Videos are temporally downsampled to one-third of their original frame rate and the predictor estimates a horizon of 15 future timesteps in the downsampled sequence. The trajectories are represented as per-step 3D displacement increments in the robot base frame.

4 Experiments

We evaluate GeomVLA through experiments designed to isolate the role of explicit 3D future-motion reasoning in vision-language-action control: (i) whether a latent 3D trajectory field improves over an otherwise identical 3D action-only policy, (ii) whether shared robot-centric 3D representation is beneficial, (iii) comparison against recent VLA, 3D-VLA, and motion-guided policies, and (iv) real-world robustness under noisy sensing and partial observability.

4.1 Simulation Experiments

We evaluate GeomVLA on CALVIN [22], LIBERO [23], and RoboTwin2.0 [24], covering long-horizon manipulation, compositional generalization, and single-/dual-arm control. The policy predicts end-effector deltas (translation, rotation, gripper state). RoboTwin2.0 and real-world tasks use current end-effector deltas, while CALVIN and LIBERO use per-step end-effector deltas. Training uses 4–8 L40S GPUs without robot-scale pretraining beyond benchmark data.

Results. Full results are in Table 1. GeomVLA achieves state-of-the-art performance on CALVIN and LIBERO and competitive performance on RoboTwin2.0 against (i) 2D VLAs [3, 53, 7, 52, 8], (ii) 3D policies and VLAs [13, 14, 15, 54] and (iii) motion-guided VLAs [20, 55, 18, 36, 56, 19, 43, 44]. On CALVIN, it improves over FLOWER (4.480 $\rightarrow$ 4.624) and X-VLA (4.430 $\rightarrow$ 4.624)

Table 1: **Comparison with prior methods on simulation benchmarks.** GeomVLA w/o motion denotes a variant of GeomVLA that removes the 3D trajectory predictor and only trains the 3D action policy. *Pretrained* denotes whether the method employs large-scale pretraining. Tasks (a)–(e) correspond to five representative RoboTwin2.0 tasks selected following SimpleVLA-RL [52].

Method	Pretrained	Params	CALVIN	LIBERO					RoboTwin2.0					
			ABC_D	Spatial	Object	Goal	Long	Avg.	a	b	c	d	e	Avg.
3D VLA														
3D-VLA	Yes	–	0.707	–	–	–	–	–	–	–	–	–	–	–
SpatialVLA	Yes	4B	–	88.2	89.9	78.6	55.5	78.1	–	–	–	–	–	–
4D-VLA	Yes	4B+	–	88.9	95.2	90.0	79.1	88.6	–	–	–	–	–	–
PoseVLA	Yes	3B+	–	96.5	98.0	97.1	92.4	96.0	**93.5**	87.0	58.0	93.0	**96.5**	**85.6**
Video/Motion-Guided														
TraceVLA	Yes	7B	–	84.6	85.2	75.1	54.1	74.8	–	–	–	–	–	–
mimic-video	Yes	2B+	–	94.2	96.8	90.6	–	93.9	–	–	–	–	–	–
WorldVLA	No	7B	–	87.6	96.2	83.4	60.0	81.8	–	–	–	–	–	–
UniVLA	Yes	7B	3.801	96.5	96.8	95.6	92.0	95.2	–	–	–	–	–	–
CoT-VLA	Yes	7B	–	87.5	91.6	87.6	69.0	81.1	–	–	–	–	–	–
FlowVLA	Yes	8.5B	–	93.2	95.0	91.6	72.6	88.1	–	–	–	–	–	–
LaMP	Yes	5B	–	**99.4**	**99.8**	97.4	96.7	98.3	–	–	–	–	–	–
RoboFlow4D	Yes	0.76B+	–	90.2	97.0	88.4	75.2	87.7	–	–	–	–	–	–
Others														
$\pi_{0.5}$	Yes	3B+	3.964	98.8	98.2	98.0	92.4	96.8	85.5	42.0	63.5	**94.5**	94.5	76.0
OpenVLA-OFT	Yes	7B	4.279	97.6	98.4	97.9	94.5	97.1	28.1	29.7	28.1	45.3	40.6	34.4
FLOWER	Yes	0.95B	4.480	97.2	99.3	96.9	94.5	97.0	–	–	–	–	–	–
SimpleVLA-RL	Yes	7B	–	**99.4**	99.1	**99.2**	**98.5**	**99.1**	87.5	68.3	61.2	89.2	75.8	76.4
X-VLA	Yes	0.9B	4.430	98.2	98.6	97.8	97.6	98.1	48.0	28.5	47.0	69.0	46.5	47.8
GeomVLA w/o motion	No	1.2B	4.508	94.8	99.2	88.2	91.8	93.5	45.0	96.0	83.0	83.0	43.0	70.0
GeomVLA	No	1.2B	**4.624**	98.6	**99.8**	97.6	97.6	98.4	62.0	**98.0**	**90.0**	90.0	62.0	80.4

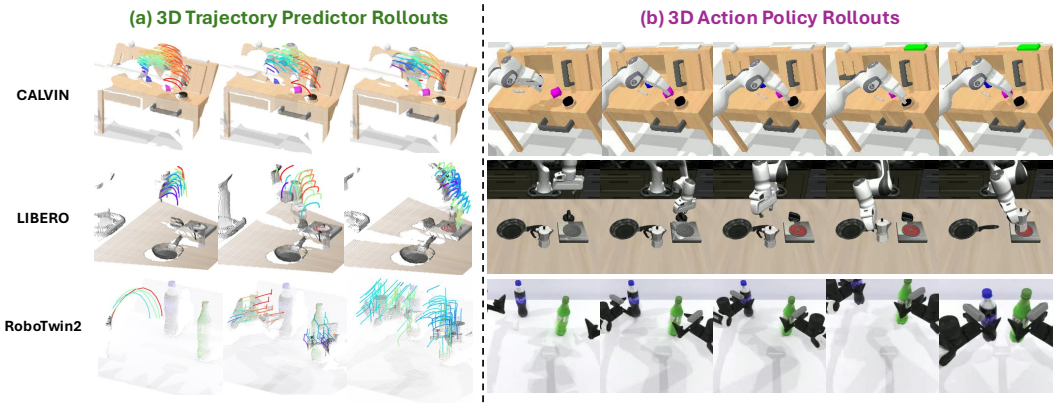

Figure 3: **Simulation rollouts across CALVIN, LIBERO, and RoboTwin2.0.** GeomVLA grounds future-motion reasoning in 3D and converts it into robot actions across diverse manipulation tasks. Left: 3D Trajectory Predictor's output; Right: closed-loop rollouts from the 3D Action Policy.

without additional robot-scale pretraining. On LIBERO, it attains the best average performance among methods without reinforcement-learning fine-tuning. Improvements are most pronounced in long-horizon and contact-rich tasks requiring precise spatial reasoning and delayed interaction effects. On RoboTwin2.0, GeomVLA outperforms recent VLA baselines and remains competitive with RL-enhanced approaches, despite training only on demonstration data. These results highlight the effectiveness of maintaining geometric consistency between scene representation, future reasoning, and action generation.

4.2 Ablation Studies

We analyze two questions: (i) the effect of future-motion reasoning, and (ii) the role of shared 3D representation. We consider the following ablative versions of our model:

- **GeomVLA w/o motion**: removes the 3D Scene Trajectory Denoiser and performs direct action prediction using only the 3D action denoiser.
- **2DGeomVLA w/o motion**: replaces the 3D action denoiser with a standard 2D image-based action denoiser and removes future-motion reasoning entirely.

- **2DGeomVLA**: uses image-space future-motion reasoning together with a 2D action denoiser, without explicit geometric grounding in a shared 3D coordinate frame.
- **GeomVLA w/ 2D motion**: retains the 3D action denoiser but replaces the 3D Scene Trajectory Denoiser with image-space future-motion reasoning, resulting in mismatched motion and action representation spaces.
- **2DGeomVLA LaMP-style**: uses standard 2D image-based action denoiser and replaces the 3D Scene Trajectory Denoiser's geometric representation with camera-frame pixel uv coordinates plus metric depth (2.5D scene flow) rather than metric 3D points.
- **GeomVLA w/ denoised trace**: uses a fully-denoised 3D trajectory as the motion variable for conditioning the 3D action denoiser.
- **GeomVLA w/ denoised latent**: uses a fully-denoised latent feature as the motion variable for conditioning the 3D action denoiser.
- **GeomVLA w/ partial denoising**: reads the velocity latent at a $1/10$-denoised trajectory (τ=0.1, one Euler step out of ten) instead of at pure noise (τ=0) as in the full model.
- **GeomVLA w/ encoder only**: uses only the encoder of the 3D Scene Trajectory Denoiser, discarding the decoder and its velocity prediction.
- **GeomVLA**: the full model, which unifies scene representation, future-motion reasoning, and action generation within a shared robot-centric 3D coordinate frame.

We show our ablation results on CALVIN in Figure 4. We draw the following conclusions:

Future-motion reasoning helps both 2D (4.411 $\rightarrow$ 4.437) and 3D denoisers (4.508 $\rightarrow$ 4.624), suggesting that predicting future scene evolution provides useful temporal structure beyond direct action imitation alone.

Explicit 3D geometry in VLAs helps. Replacing the 2D action denoiser with the corresponding 3D action denoiser improves performance from 4.411 to 4.508, demonstrating the importance of representing scene features and robot actions directly in metric 3D coordinates.

Scene motion representation matters. The full model conditions the action denoiser on the motion produced by the scene trajectory denoiser's decoder, paired with pure noise. Alternatives that weaken this scene motion representation all underperform: conditioning on final denoised predictions, whether one-step denoised trajectories (4.047) or latents (4.334), is the least effective; conditioning on the velocity latent together with a $1/10$-denoised trajectory (4.484) rather than pure noise still trails the full model; and using only the denoiser's encoder without decoder-based velocity prediction (4.443) discards useful motion structure. Together, these results show that the action denoiser benefits most from the scene motion latent paired with pure noise rather than from partially- or fully-denoised trajectory estimates or the encoder representation alone.

Variant	Perf.
GeomVLA w/o motion	4.508
2DGeomVLA w/o motion	4.411
2DGeomVLA	4.437
GeomVLA w/ 2D motion	4.085
2DGeomVLA LaMP-style	4.462
GeomVLA w/ denoised trace	4.047
GeomVLA w/ denoised latent	4.334
GeomVLA w/ partial denoising	4.484
GeomVLA w/ encoder only	4.443
GeomVLA	**4.624**

Figure 4: **CALVIN ABC-D ablations.**

Geometric consistency is critical for effective future reasoning. Conditioning the 3D action policy on image-space motion reasoning significantly degrades performance to 4.085, performing worse than both the fully 2D model and the unified 3D model. 2DGeomVLA LaMP-style changes the space in which the trajectory denoiser reasons to camera-frame uv plus depth and removes all 3D RoPE in scene features and action denoiser, and degrades to 4.452. The loss indicates that mixing image coordinates with metric depth yields a space whose geometric structure is neither a valid image-plane nor a valid Euclidean metric, so the denoiser's spatial reasoning is built on incoherent geometry. In contrast, the full GeomVLA model, which unifies scene representation, future-motion reasoning, and action generation within a shared robot-centric 3D coordinate frame, achieves the strongest overall performance. Together, these re-

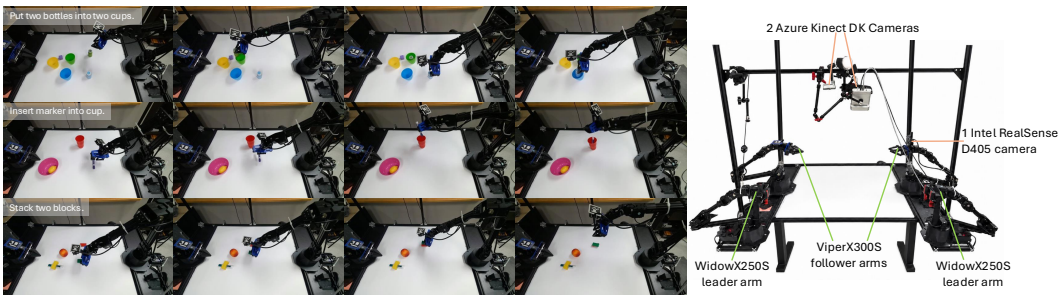

Figure 5: **Real-world evaluation setup and tasks.** We evaluate GeomVLA on an ALOHA2 platform on single-arm tabletop tasks. The left panel shows representative rollouts on three real-world tasks: placing two bottles into two cups, inserting a marker into a cup, and stacking two blocks.

sults suggest that future-motion reasoning alone is insufficient: the primary gains arise when perception, motion prediction, and robot actions interact within the same geometric representation space throughout the entire perception-to-action pipeline.

4.3 Real-World Experiments

Setup. We evaluate GeomVLA on an ALOHA2 robot platform equipped with ViperX300S follower arms (Figure 5). Demonstrations are collected through teleoperation using the corresponding WidowX250S leader arms. The visual observation setup consists of two elevated Microsoft Azure Kinect DK cameras mounted at the front and back of the workspace, together with a wrist-mounted Intel RealSense D405 camera attached to the right follower arm. Although the platform supports bimanual manipulation, our experiments focus on single-arm tabletop interaction tasks.

Tasks. We evaluate three representative tasks requiring different forms of spatial reasoning and contact interaction: inserting a marker into a cup, stacking two blocks, and placing two bottles into two cups. These tasks involve varying levels of precision, object motion prediction, and multi-stage interaction structure. For each task, we collect 50 demonstrations. We train one multi-task policy across all three tasks for each method. GeomVLA follows the same training protocol as in our simulation benchmarks and is trained only on these task demonstrations. For $\pi_{0.5}$, we start from the $\pi_{0.5}$-base checkpoint and apply LoRA finetuning on the same data. Both methods use a binary gripper openness command. For arm control, GeomVLA predicts end-effector poses, whereas $\pi_{0.5}$ predicts joint-angle actions.

Results. Results are summarized in Table 2. GeomVLA substantially outperforms $\pi_{0.5}$ on two of the three tasks, with strong gains on block stacking and the multi-stage bottle placement task. As highlighted in Figure 1, these tasks evaluate key real-world capabilities including spatial understanding, multi-stage execution, and robustness to unseen distractors and layouts. The results suggest that GeomVLA benefits from aligning scene perception, future-motion reasoning, and action generation in a shared 3D space, enabling the policy to anticipate object motion while maintaining coherent actions under perception noise and contact-induced deviations.

We further observe that GeomVLA can often recover from transient execution errors, partial occlusions, and small object pose inaccuracies without explicit replanning, reducing compounding errors during rollout. $\pi_{0.5}$ performs better on marker insertion, a high-precision task with tight geometric tolerances; we hypothesize that this task is more sensitive to calibration noise and depth reconstruction errors, which can disproportionately affect 3D geometric reasoning. Overall, the results indicate that geometric consistency between scene representation, future-motion reasoning, and action generation improves robustness in real-world long-horizon manipulation.

Table 2: **Real-world success rates on ALOHA2.** Each task 20 trails.

Task	$\pi_{0.5}$	GeomVLA
Insert marker	9/20	**5/20**
Stack blocks	2/20	**19/20**
Bottle placement	10/20	**17/20**

Table 3: **3D Chain-of-Thought with a joint-angle action denoiser.** JA denotes joint-angle.

Benchmark	$\pi_{0.5}$	GeomVLA	2D JA	JA w/ 3D CoT
CALVIN	3.964	**4.624**	4.195	4.508
RoboTwin2.0	76.0	80.4	66.2	**81.6**
Insert marker	9/20	5/20	3/20	**13/20**

4.4 3D Chain-of-Thought

GeomVLA grounds scene representation, future-motion reasoning, and end-effector action generation in a single robot-centric 3D frame. We view this whole chain as a *3D chain-of-thought* (3D CoT): the predicted end-effector action is itself an intermediate step reasoned about in metric 3D, and it need not be the executed command. Decoupling reasoning from execution matters when depth or camera calibration is unreliable, since end-effector pose execution is directly sensitive to such errors while joint-angle execution is not.

Setup. We attach a **joint-angle action denoiser** conditioned on the full 3D CoT—the fused scene-motion feature together with the output latent of the end-effector action denoiser—which emits joint commands directly. A 2D joint-angle variant driven by a 2D image-based policy ablates the contribution of the 3D geometric reasoning. We evaluate on CALVIN ABC_D, five RoboTwin2.0 tasks, and a real-world *insert pen* task which is especially sensitive to calibration and depth noise.

Results. As shown in Table 3, the joint-angle denoiser with 3D CoT beats the 2D joint-angle denoiser, and matches or exceeds GeomVLA on all three benchmarks. This demonstrates that the gain from 3D scene-motion-action reasoning is not tied to end-effector execution. It restores robustness under poor calibration. On *insert pen*, routing the 3D CoT through the joint-angle denoiser reaches the best performance. The 3D CoT thus improves action prediction regardless of the execution space, and helps most when calibration noise makes direct 3D end-effector execution unreliable.

5 Limitations

GeomVLA demonstrates strong performance across simulation and real-world manipulation benchmarks, but several limitations remain. First, it relies on accurate camera calibration; errors in depth estimation or camera extrinsics can degrade the quality of geometric representations, particularly in real-world settings. Second, unlike recent large-scale VLA systems, GeomVLA is trained only on benchmark-scale robot demonstrations without large-scale robot pretraining or cross-embodiment data. Scaling to larger and more diverse datasets could further improve semantic generalization, robustness, and long-horizon reasoning. Finally, GeomVLA performs reasoning through latent geometric trajectory representations rather than explicit language-based reasoning. Combining geometric reasoning with language-conditioned planning or hierarchical task decomposition is a promising direction for long-horizon manipulation and failure recovery.

6 Conclusion

We presented GeomVLA, a 3D motion-aware vision-language-action framework that unifies semantic perception, latent future-motion reasoning, and action generation within a shared robot-centric 3D coordinate frame. Our approach introduces an *3D Scene Trajectory Denoiser* that predicts future scene motion directly in 3D space and integrates this motion representation into a flow-based 3D action denoiser through geometry-aware attention. Extensive experiments across CALVIN, LIBERO, RoboTwin2.0, and real-world manipulation benchmarks demonstrate that maintaining geometric consistency between scene representation, future reasoning, and robot actions substantially improves manipulation performance. We hope this work encourages future research toward more tightly inte-

grated geometric reasoning architectures for embodied intelligence, combining large-scale semantic knowledge with physically grounded spatial and temporal reasoning.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

[3] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[4] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

[5] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[6] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks, 2023. URL https://arxiv.org/abs/2311.06242.

[7] M. Reuss, H. Zhou, M. Rühle, Ömer Erdinç Yağmurlu, F. Otto, and R. Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies, 2025. URL https://arxiv.org/abs/2509.04996.

[8] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, Y.-Q. Zhang, J. Pang, J. Liu, T. Wang, and X. Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

[9] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. *CoRL*, 2023.

[10] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[11] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[12] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. *arXiv preprint arXiv:2306.14896*, 2023.

[13] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: A 3d vision-language-action generative world model, 2024. URL https://arxiv.org/abs/2403.09631.

[14] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, and X. Li. Spatialvla: Exploring spatial representations for visual-language-action model, 2025. URL https://arxiv.org/abs/2501.15830.

[15] J. Zhang, Y. Chen, Y. Xu, Z. Huang, Y. Zhou, Y.-J. Yuan, X. Cai, G. Huang, X. Quan, H. Xu, and L. Zhang. 4d-vla: Spatiotemporal vision-language-action pretraining with cross-scene calibration, 2025. URL https://arxiv.org/abs/2506.22242.

[16] Y. Jia, J. Liu, S. Chen, C. Gu, Z. Wang, L. Luo, L. Lee, P. Wang, Z. Wang, R. Zhang, and S. Zhang. Lift3d foundation policy: Lifting 2d large-scale pretrained models for robust 3d robotic manipulation, 2024. URL https://arxiv.org/abs/2411.18623.

[17] C. Li, J. Wen, Y. Peng, Y. Peng, F. Feng, and Y. Zhu. Pointvla: Injecting the 3d world into vision-language-action models, 2025. URL https://arxiv.org/abs/2503.07511.

[18] J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang, D. Zhao, and H. Chen. Worldvla: Towards autoregressive action world model, 2025. URL https://arxiv.org/abs/2506.21539.

[19] Z. Zhong, H. Yan, J. Li, X. Liu, X. Gong, T. Zhang, W. Song, J. Chen, X. Zheng, H. Wang, and H. Li. Flowvla: Visual chain of thought-based motion reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2508.18269.

[20] R. Zheng, Y. Liang, S. Huang, J. Gao, H. D. III, A. Kolobov, F. Huang, and J. Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2025. URL https://arxiv.org/abs/2412.10345.

[21] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu, P. Sundaresan, P. Xu, H. Su, K. Hausman, C. Finn, Q. Vuong, and T. Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches, 2023. URL https://arxiv.org/abs/2311.01977.

[22] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[23] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li, Q. Liang, X. Lin, Y. Ge, Z. Gu, W. Deng, Y. Guo, T. Nian, X. Xie, Q. Chen, K. Su, T. Xu, G. Liu, M. Hu, H. ang Gao, K. Wang, Z. Liang, Y. Qin, X. Yang, P. Luo, and Y. Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

[25] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Conference on Robot Learning*, pages 2323–2339. PMLR, 2023.

[26] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *arXiv preprint arXiv:2409.07343*, 2024.

[27] I.-C. A. Liu, S. He, D. Seita, and G. Sukhatme. Voxact-b: Voxel-based acting and stabilizing policy for bimanual manipulation. *CoRL*, 2024.

[28] M. Grotz, M. Shridhar, T. Asfour, and D. Fox. Peract2: A perceiver actor framework for bimanual manipulation tasks. *arXiv preprint arXiv:2407.00278*, 2024.

[29] N. Gkanatsios, J. Xu, M. Bronars, A. Mousavian, T.-W. Ke, and K. Fragkiadaki. 3d flowmatch actor: Unified 3d policy for single- and dual-arm manipulation, 2025. URL https://arxiv.org/abs/2508.11002.

[30] P. Li, Y. Chen, H. Wu, X. Ma, X. Wu, Y. Huang, L. Wang, T. Kong, and T. Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models, 2025. URL https://arxiv.org/abs/2506.07961.

[31] T. Lin, G. Li, Y. Zhong, Y. Zou, Y. Du, J. Liu, E. Gu, and B. Zhao. Evo-0: Vision-language-action model with implicit spatial understanding, 2025. URL https://arxiv.org/abs/2507.00416.

[32] S. Deng, M. Yan, Y. Zheng, J. Su, W. Zhang, X. Zhao, H. Cui, Z. Zhang, and H. Wang. Stereovla: Enhancing vision-language-action models with stereo vision, 2025. URL https://arxiv.org/abs/2512.21970.

[33] X. Fan, S. Deng, X. Wu, Y. Lu, Z. Li, M. Yan, Y. Zhang, Z. Zhang, H. Wang, and H. Zhao. Any3d-vla: Enhancing vla robustness via diverse point clouds, 2026. URL https://arxiv.org/abs/2602.00807.

[34] W. Li, J. Liu, L. Yixing, J. Tong, R. Shao, and L. Nie. Consisvla-4d: Advancing spatiotemporal consistency in efficient 3d-perception and 4d-reasoning for robotic manipulation, 2026. URL https://arxiv.org/abs/2605.05126.

[35] Y. Du, M. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation, 2023. URL https://arxiv.org/abs/2302.00111.

[36] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified video action model, 2025. URL https://arxiv.org/abs/2503.00200.

[37] H. Bi, H. Tan, S. Xie, Z. Wang, S. Huang, H. Liu, R. Zhao, Y. Feng, C. Xiang, Y. Rong, H. Zhao, H. Liu, Z. Su, L. Ma, H. Su, and J. Zhu. Motus: A unified latent action world model, 2025. URL https://arxiv.org/abs/2512.13030.

[38] G. Team, B. Wang, B. Li, C. Ni, G. Huang, G. Zhao, H. Li, J. Li, J. Lv, J. Liu, L. Feng, M. Yu, P. Li, Q. Deng, T. Liu, X. Zhou, X. Chen, X. Wang, Y. Wang, Y. Li, Y. Nie, Y. Li, Y. Zhou, Y. Ye, Z. Liu, and Z. Zhu. Gigabrain-0.5m*: a vla that learns from world model-based reinforcement learning, 2026. URL https://arxiv.org/abs/2602.12099.

[39] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface, 2024. URL https://arxiv.org/abs/2407.15208.

[40] H. Zhi, P. Chen, S. Zhou, Y. Dong, Q. Wu, L. Han, and M. Tan. 3dflowaction: Learning cross-embodiment manipulation from 3d flow world model, 2025. URL https://arxiv.org/abs/2506.06199.

[41] S. Lee, Y. Jung, I. Chun, Y.-C. Lee, Z. Cai, H. Huang, A. Talreja, T. D. Dao, Y. Liang, J.-B. Huang, and F. Huang. Tracegen: World modeling in 3d trace space enables learning from cross-embodiment videos, 2025. URL https://arxiv.org/abs/2511.21690.

[42] S. Noh, D. Nam, K. Kim, G. Lee, Y. Yu, R. Kang, and K. Lee. 3d flow diffusion policy: Visuomotor policy learning via generating flow in 3d space, 2025. URL https://arxiv.org/abs/2509.18676.

[43] X. Wang, C. Wang, Y. Xu, M. Ye, F.-C. Zhang, J. Tian, X. Zhan, L. Zhu, C. Lu, and L. Yang. Lamp: Learning vision-language-action policies with 3d scene flow as latent motion prior, 2026. URL https://arxiv.org/abs/2603.25399.

[44] S. Lin, J. Chen, H. Xu, Z. Li, G. Wang, Y. Jing, S. Xu, R. Zhao, B. Sheil, L.-P. Chau, and G. Liu. Roboflow4d: A lightweight flow world model toward real-time flow-guided robotic manipulation, 2026. URL https://arxiv.org/abs/2605.17522.

[45] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation, 2024. URL https://arxiv.org/abs/2405.01527.

[46] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel. Any-point trajectory modeling for policy learning, 2024. URL https://arxiv.org/abs/2401.00025.

[47] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2407.08693.

[48] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025. URL https://arxiv.org/abs/2502.05485.

[49] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, W. Han, W. Pumacay, A. Wu, R. Hendrix, K. Farley, E. VanderBilt, A. Farhadi, D. Fox, and R. Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

[50] Y. Li, Y. Chen, M. Zhou, H. Li, Z. Zhang, and D. Zhao. Qdepth-vla: Quantized depth prediction as auxiliary supervision for vision-language-action models, 2025. URL https://arxiv.org/abs/2510.14836.

[51] Y. Xiao, J. Wang, N. Xue, N. Karaev, Y. Makarov, B. Kang, X. Zhu, H. Bao, Y. Shen, and X. Zhou. Spatialtrackerv2: 3d point tracking made easy, 2025. URL https://arxiv.org/abs/2507.12462.

[52] H. Li, Y. Zuo, J. Yu, Y. Zhang, Z. Yang, K. Zhang, X. Zhu, Y. Zhang, T. Chen, G. Cui, D. Wang, D. Luo, Y. Fan, Y. Sun, J. Zeng, J. Pang, S. Zhang, Y. Wang, Y. Mu, B. Zhou, and N. Ding. Simplevla-rl: Scaling vla training via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.09674.

[53] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

[54] H. Lin, H. Yu, J. Huang, H. Zhang, Y. Ling, P. Tan, X. Xue, and Y. Fu. Universal pose pretraining for generalizable vision-language-action policies, 2026. URL https://arxiv.org/abs/2602.19710.

[55] J. Pai, L. Achenbach, V. Montesinos, B. Forrai, O. Mees, and E. Nava. mimic-video: Video-action models for generalizable robot control beyond vlas, 2025. URL https://arxiv.org/abs/2512.15692.

[56] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, A. Handa, M.-Y. Liu, D. Xiang, G. Wetzstein, and T.-Y. Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

Appendix

Model Architecture Overview. Figure 2 summarizes the full GeomVLA architecture. It first lifts pretrained VLM features from RGB-D observations into robot-centric 3D scene tokens, then predicts an Actionable 3D Trajectory Field as a latent future-motion representation, which conditions a geometry-aware flow-based 3D action policy for end-effector trajectory generation. We report the per-benchmark model configurations and training hyperparameters in Table 4.

Actionable 3D Trajectory Field Encoder. The encoder binds VLM-derived semantics to local 3D geometry through a point-wise design. Each query pixel $u_i = (u_i, v_i)$ is lifted with depth and camera calibration into a robot-base anchor, and a per-sample centering removes absolute translation while leaving q_i in the base frame for downstream use:

$$q_i \;=\; T_{\mathrm{cam}\to\mathrm{base}}\, K^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} D(u_i), \qquad \tilde{q}_i \;=\; q_i - \tfrac{1}{N}\sum_{j=1}^{N} q_j. \tag{7}$$

A frozen Florence-2 VLM encodes the image into a patch feature map $V \in \mathbb{R}^{H_v \times W_v \times d_v}$ and the instruction into language tokens $E \in \mathbb{R}^{L \times d_t}$. For each query we bilinearly sample V to obtain a visual descriptor v_i and form an initial point token that fuses geometry, image location, and semantics:

$$h_i^{(0)} \;=\; \mathrm{MLP}\big([\,\tilde{q}_i;\; u_i;\; v_i\,]\big) \;\in\; \mathbb{R}^D. \tag{8}$$

We build a k-nearest-neighbor graph $\mathcal{N}_k(i)$ once over $\{\tilde{q}_i\}$ and refine tokens through L_{enc} local geometry-aware attention layers, where each query attends to its neighbors with attention logits augmented by a learned relative-position bias $b(\tilde{q}_j - \tilde{q}_i)$:

$$h_i^{(\ell+1)} \;=\; h_i^{(\ell)} + \sum_{j\in\mathcal{N}_k(i)} \alpha_{ij}^{(\ell)}\, W_v^{(\ell)} h_j^{(\ell)}, \qquad \alpha_{ij}^{(\ell)} \propto \exp\!\left(\frac{\big(W_q^{(\ell)} h_i^{(\ell)}\big)^{\top}\big(W_k^{(\ell)} h_j^{(\ell)}\big)}{\sqrt{D}} + b(\tilde{q}_j - \tilde{q}_i)\right). \tag{9}$$

The encoder outputs per-point context features $\{c_i\}_{i=1}^N$ that couple task-relevant semantics with local 3D structure, and retains pooled visual summary tokens V_g together with the projected text tokens E as global multimodal conditioning $\mathcal{C} = (\{c_i\}, V_g, E)$ for the decoder.

Actionable 3D Trajectory Field Decoder. The decoder predicts $\mathcal{F}$ by conditional flow matching. Let $F \in \mathbb{R}^{N\times H\times 3}$ denote the normalized ground-truth Δq-tensor. We sample Gaussian noise $F_0 \sim \mathcal{N}(0, I)$ and a flow time $\tau \sim \mathcal{U}(0,1)$, and define the linear interpolant and the velocity target

$$F_\tau \;=\; (1-\tau)\, F_0 + \tau\, F, \qquad u^\star \;=\; F - F_0. \tag{10}$$

A velocity network $u_\theta(F_\tau, \tau, \mathcal{C})$ is trained to regress $u^\star$, where $\mathcal{C} = (\{c_i\}, V_g, E)$ denotes the multimodal context.

The decoder input at point i and step t fuses the noisy displacement, the per-point context, a learned step embedding e_t and a sinusoidal flow-time embedding $\phi(\tau)$,

$$Z_{i,t}^{(0)} \;=\; W_F\, F_{\tau,i,t} \;+\; c_i \;+\; e_t \;+\; \phi(\tau). \tag{11}$$

Each decoder block factorizes attention along the spatial and temporal axes of the trajectory field. Spatial attention operates over 3D anchors at each future step. Rather than re-using the encoder's kNN graph, we adopt a serialized window attention along a 3D space-filling curve, which scales to the full $N\times H$ token set with hardware-friendly fixed-size windows. Concretely, for each future step t we normalize the anchors to the unit cube, $\hat{q}_i = (\tilde{q}_i - \min_j \tilde{q}_j)/(\max_j \tilde{q}_j - \min_j \tilde{q}_j)$, quantize them to an integer grid at bit-depth d_q, and assign each anchor a 3D Morton (Z-order) code $\mathrm{m}(\hat{q}_i) \in \mathbb{N}$ that yields a permutation π sorting the N points along the curve. Self-attention is then applied within non-overlapping windows of P consecutive points in the order π, and the result is restored to the original point order by the inverse permutation π^{-1}:

$$\mathrm{SerSpatial}(Z)_{\pi(i),t} \;=\; \mathrm{Attn}\big(Z_{\pi(i),t};\; \{Z_{\pi(j),t} : j \in \mathcal{W}_P(i)\}\big), \qquad \mathcal{W}_P(i) = \big\{j : \lfloor j/P \rfloor = \lfloor i/P \rfloor\big\}. \tag{12}$$

Table 4: Model configurations across simulation benchmarks. The trajectory predictor is the motion expert for future 3D point trajectories; its hidden dimension, encoder layers, decoder layers, and attention heads refer to the sparse motion decoder. The 3D action policy shared attention layers are the action–scene fusion blocks that are shared by translation and rotation prediction. Rotation format is the policy rotation representation, with RoboTwin2.0 rotations converted to 6D internally; action horizon is the number of future actions predicted per policy query, and embodiment specifies the controlled arm setup.

Module		CALVIN	LIBERO	RoboTwin2.0
Trajectory Predictor	Hidden dim	512	384	320
	Encoder layers	6	6	6
	Decoder layers	8	8	8
	Attention heads	16	8	8
	Batch size	128	128	128
	Optimizer	AdamW	AdamW	AdamW
	Learning rate	1e-4	1e-4	1e-4
3D Action Policy	Hidden dim	960	912	912
	Shared attention layers	16	16	16
	Attention heads	16	16	16
	Batch size	32	128	64
	Optimizer	AdamW	AdamW	AdamW
	Learning rate	2e-5	5e-5	4e-5
	Rotation format	Euler	Axis-angle	6D rotation
	Action horizon	10	10	16
	Embodiment	Single-arm	Single-arm	Bimanual

To enlarge the effective receptive field while keeping per-window cost $\mathcal{O}(P^2)$, we alternate between two axis orderings of the Morton code (canonical and xy-transposed) and apply a Swin-style half-window shift on alternating layers; a learned linear projection of the normalized coordinates is added to each token before the QKV map so that serialized attention remains explicitly geometry-aware.

For every point i, a temporal self-attention $\mathrm{Temporal}(\cdot)$ couples its H future steps. The resulting tokens then attend to the conditioning sequence $[V_g;\, E]$ through a cross-attention block $\mathrm{Cross}(\cdot, [V_g; E])$, followed by a position-wise feed-forward $\mathrm{FFN}(\cdot)$. With pre-LayerNorm and residual connections, one block computes

$$Z \xrightarrow{\text{SerSpatial}} Z \xrightarrow{\text{Temporal}} Z \xrightarrow{\text{Cross}([V_g;E])} Z \xrightarrow{\text{FFN}} Z. \tag{13}$$

A linear head reads off the predicted velocity $u_\theta(F_\tau, \tau, \mathcal{C}) \in \mathbb{R}^{N\times H\times 3}$. We optimize the motion-weighted flow-matching objective

$$\mathcal{L} \;=\; \mathbb{E}_{\tau, F_0}\left[\sum_{i,t} w_{i,t}\left\|u_\theta(F_\tau, \tau, \mathcal{C})_{i,t} - u^\star_{i,t}\right\|_2^2\right], \tag{14}$$

where $w_{i,t} = m_{i,t}\cdot \max\!\big(\sigma\big((\|\Delta q_{i,t}\| - \tau_w)\,\kappa/\tau_w\big)^\gamma,\, w_{\min}\big)$ combines the trajectory validity mask $m_{i,t}$ with a soft motion-saliency term that emphasizes moving points while still supervising static ones through the floor $w_{\min}$. At inference, $\mathcal{F}$ is recovered from $F_0 \sim \mathcal{N}(0, I)$ by an explicit Euler integration of u_θ over a small number of flow steps and then unnormalized to metric displacements.

Ablation on 3D Trajectory Field Horizon. We ablate the prediction length of the Actionable 3D Trajectory Field, i.e., the number of future steps H over which the decoder forecasts per-point displacements. Keeping all other components fixed, we train on the same CALVIN demonstration dataset with a longer horizon of $H = 32$ and compare it with our default horizon of $H = 15$. This isolates the effect of the temporal extent of the predicted motion field on downstream policy performance. The results are summarized in Table 5.

The longer horizon does not improve performance on CALVIN. While the 15-step horizon captures local future motion, a dense 32-step trajectory field covers more than half of an average CALVIN

Table 5: **Ablation on the prediction horizon of the 3D Trajectory Field.** We compare the default $H = 15$ trajectory-field horizon with a longer dense horizon of $H = 32$ on CALVIN, while keeping all other model components fixed. In this CALVIN setting, increasing the horizon does not improve performance, suggesting that a longer motion target may be less aligned with the local future motion needed for downstream action generation.

Variant	Perf.
GeomVLA w/ motion horizon=32	4.272
GeomVLA w/ motion horizon=15	**4.624**

instruction window and nearly the entire remaining future for the shortest windows. Consequently, the motion target may extend beyond the immediately actionable phase and include later-stage or terminal motion that is difficult to infer from the current observation. This makes the learned motion tokens more susceptible to noisy distant-step supervision and can misguide the downstream policy.

Inference Latency. We measure single-action-chunk inference latency on CALVIN inside live online evaluation, where one `model.step()` call predicts a chunk of 10 future actions. The benchmark is run during PyBullet rollouts with EGL rendering on a single NVIDIA L40S GPU, using CUDA-synchronized timers around `model.step()` and the main forward stages. We evaluate four instruction sequences per model and discard the first 10 chunks as warmup. The baseline GeomVLA w/o motion, which consists of the Florence-2 VLM encoder and a 5-step rectified-flow 3D action head, takes 219.0 ± 11.3 ms per chunk over 165 chunks, corresponding to 4.57 chunks/s or 45.7 actions/s. Its latency decomposes into 35.5 ms for the VLM encoder and 182.4 ms for the 3D action policy. GeomVLA takes 280.0 ± 14.0 ms per chunk over 155 chunks, corresponding to 3.57 chunks/s or 35.7 actions/s. Its latency decomposes into 37.2 ms for the VLM encoder, 37.5 ms for the 3D trajectory field, and 202.1 ms for the 3D action policy. Thus, GeomVLA adds 61.0 ms per chunk, or about 28% overhead. Two-thirds of this overhead comes from the 3D trajectory field. In both models, the 3D action policy, rather than the VLM encoder, is the dominant time cost.

Supplementary Real-World Tasks. We provide detailed descriptions of all real-world manipulation tasks, grouped by the primary capability each task is designed to evaluate. The three tasks marked with † are reported in the main paper and evaluated with 20 rollouts each. The remaining five tasks are included as supplementary results under a reduced experimental setting, with fewer demonstrations, 10 evaluation rollouts per task, and lower train/test variation in object layouts and distractors. Together, these tasks broaden the evaluation along the same axes of precision, fine-grained manipulation, and multi-stage spatial reasoning. The number of demonstrations is reported separately for each task.

Precision and rotation control.

- **Insert marker**† (50 demos). Pick up a marker lying flat on the table and insert it upright into a cup. This task requires fine-grained geometric reasoning and precise end-effector rotation under tight tolerances.
- **Stand bottle upright** (20 demos). Pick up a bottle lying flat on the table and place it in an upright standing pose. This task isolates rotation understanding and stable placement.
- **Transfer marker** (20 demos). Pick up a marker standing upright inside one cup and re-insert it into another cup. Since the marker pose varies across trials, the policy must adapt its gripper orientation to grasp and re-insert the marker reliably.

Fine small-object manipulation.

- **Stack blocks**† (50 demos). Pick up one block and stack it precisely on top of another block with identical size and shape. This task requires accurate grasping and fine placement of small objects.

Table 6: **Success rates on the five supplementary real-world tasks.** We compare GeomVLA with $\pi_{0.5}$ on additional tasks evaluated with 10 rollouts each. These tasks are conducted under a reduced experimental setting, with fewer demonstrations and reduced train/test variation in object layouts and distractors.

Method	Stand bottle upright	Transfer marker	Uncap marker	Rank by color	Place in triangle
$\pi_{0.5}$	4/10	0/10	7/10	0/10	2/10
GeomVLA	**9/10**	**6/10**	**10/10**	**2/10**	**8/10**

- **Uncap marker** (20 demos). Grasp the cap of a marker lying flat on the table and pull it off to uncap the marker. This task requires fine-grained contact control and small-object manipulation.

Color recognition and multi-stage spatial reasoning.

- **Bottle placement**† (50 demos). Pick up the green bottle and place it into the green cup, then pick up the blue bottle and place it into the blue cup. This two-stage task evaluates color recognition and sequential object placement.
- **Rank by color** (20 demos). Sequentially pick up three bottles and arrange them from left to right, as viewed from the wrist camera, in pink–green–blue order. This three-stage long-horizon task combines color recognition with ordered spatial placement.
- **Place in triangle** (20 demos). Using the middle bottle as the anchor vertex, sequentially pick up the left and right bottles, as viewed from the wrist camera, and place them so that the three bottles form a triangle. This two-stage task evaluates spatial-relationship reasoning and relative placement.

Table 6 reports success rates on the five supplementary real-world tasks, comparing two multi-task policies trained under the same reduced experimental setting. Each task is evaluated with 10 rollouts. Across these tasks, GeomVLA achieves higher success rates than $\pi_{0.5}$, with particularly large gains on Stand bottle upright ($9/10$ vs. $4/10$), Transfer marker ($6/10$ vs. $0/10$), and Place in triangle ($8/10$ vs. $2/10$). These tasks require precise 3D rotation, object reorientation, and spatial placement, and the results are consistent with the benefit of conditioning action generation on future motion represented in a shared 3D coordinate frame. On Uncap marker, both methods perform relatively well, while GeomVLA reaches $10/10$ success.

Rank by color remains challenging for both methods ($2/10$ for GeomVLA and $0/10$ for $\pi_{0.5}$). Unlike the other supplementary tasks, it requires three sequential placements and is visually similar to Place in triangle: both tasks start from nearly identical three-bottle scenes but require different target configurations. This can make intermediate observations ambiguous during rollout, causing the policy to select an incorrect target order or target configuration and accumulate errors over the long horizon.

Figure 6 visualizes representative GeomVLA rollouts on the five supplementary real-world tasks. Each row shows a temporal sequence from one successful execution, illustrating the policy's ability to handle the precision, fine manipulation, and multi-stage spatial reasoning required by these tasks.

Supplementary RoboTwin2.0 Results. Table 7 reports per-task success rates on fifteen additional RoboTwin2.0 tasks under both `Easy` and `Hard` domain randomization settings. We compare $\pi_{0.5}$, Pose-VLA, X-VLA, and GeomVLA.

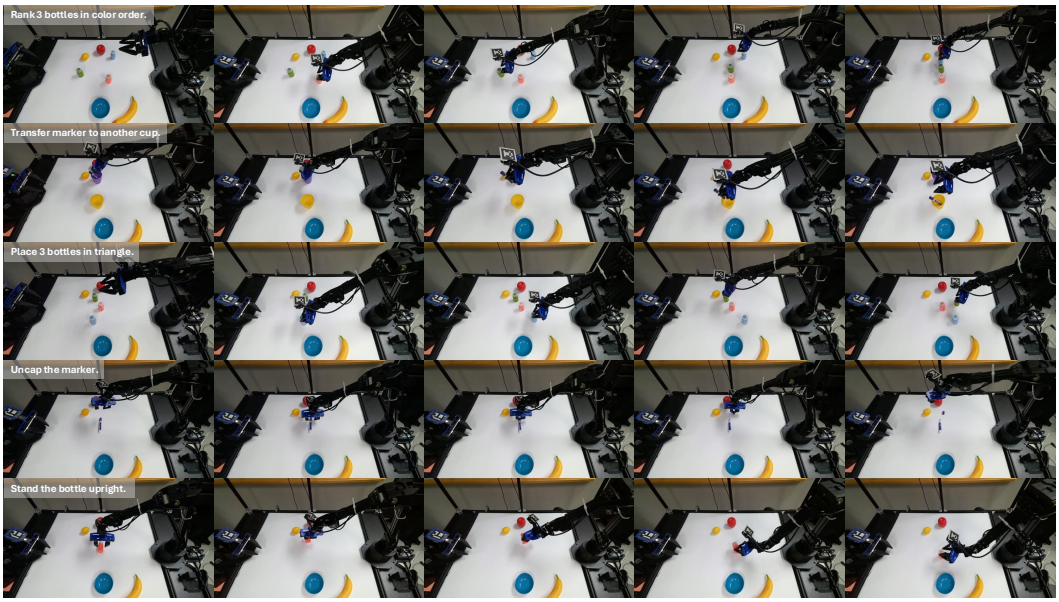

Figure 6: **Qualitative GeomVLA rollouts on the five supplementary real-world tasks.** Each row shows a temporal sequence from one successful rollout, with tasks shown from top to bottom: Rank by color, Transfer marker, Place in triangle, Uncap marker, and Stand bottle upright.

Table 7: **Per-task success rates on 15 RoboTwin2.0 tasks.** Each task is evaluated under `Easy` and `Hard` domain randomization. Success rates are reported as percentages.

Task	$\pi_{0.5}$		**Pose-VLA**		**X-VLA**		**GeomVLA**	
	Easy	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**
beat_block_hammer	78	93	100	87	78	18	82	62
pick_dual_bottles	21	63	87	87	30	27	100	100
move_can_pot	72	55	63	53	50	44	98	100
handover_mic	92	97	93	93	100	38	98	96
stack_bowls_two	93	96	93	100	83	10	62	54
place_phone_stand	83	81	87	87	80	9	82	84
click_alarmclock	92	89	83	93	96	69	82	74
grab_roller	100	100	97	100	99	66	100	98
place_a2b_left	84	82	90	80	62	21	66	64
place_bread_skillet	86	66	93	77	82	17	86	54
scan_object	80	65	87	90	60	44	88	78
handover_block	84	57	73	80	27	30	54	60
open_laptop	88	96	93	93	85	73	84	96
blocks_ranking_rgb	86	85	83	77	79	26	62	58
hanging_mug	22	17	27	23	34	15	42	50
Average (%)	77.4	76.1	83.3	81.3	69.7	33.8	79.1	75.2

Table 8: **Per-task GeomVLA success rates on 50 RoboTwin2.0 tasks.** Each task is evaluated under both `Easy` and `Hard` domain randomization, with 50 rollouts per setting. Success rates are reported as percentages.

Task	Easy	Hard	Task	Easy	Hard	Task	Easy	Hard
Adjust Bottle	100	100	Open Microwave	94	82	Place Object Stand	96/88	98/98
Beat Block Hammer	88/96	90/90	Pick Diverse Bottles	86/86	80/88	Place Phone Stand	80/84	82/78
Blocks Ranking RGB	46/52	44/32	Pick Dual Bottles	100/100	96/100	Place Shoe	96	100
Blocks Ranking Size	24	32	Place A2B Left	90/80	92/90	Press Stapler	78/70	60/78
Click Alarmclock	98/94	96/90	Place A2B Right	96	90	Put Bottles Dustbin	18/34	12/18
Click Bell	100/100	100/100	Place Bread Basket	88/84	80/74	Put Object Cabinet	62	48
Dump Bin Bigbin	72	68	Place Bread Skillet	92	80	Rotate QRcode	80	80
Grab Roller	100/100	98/100	Place Burger Fries	94/86	90/88	Scan Object	36/62	60/50
Handover Block	14	4	Place Can Basket	84	76	Shake Horizontally	100/98	100/94
Handover Mic	72	32	Place Cans Plasticbox	96	92	Shake Bottle	100/98	100/96
Hanging Mug	10/0	4/10	Place Container Plate	100/98	100/100	Stack Blocks Three	28	28
Lift Pot	96/96	92/94	Place Dual Shoes	84	86	Stack Blocks Two	68/66	92/80
Move Can Pot	100	98	Place Empty Cup	100/100	96/96	Stack Bowls Three	46	36
Move Pillbottle Pad	96/98	100/92	Place Fan	88	86	Stack Bowls Two	88/74	72/74
Move Playingcard Away	98/96	96/88	Place Mouse Pad	88	90	Stamp Seal	94/94	96/94
Move Stapler Pad	72/68	76/76	Place Object Basket	50/48	44/56	Turn Switch	72/84	66/78
Open Laptop	94/82	92/92	Place Object Scale	96	98	**Average**	**79.0**	**76.2**