GF-VLA: DUAL GEOMETRY FORCING FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models generate robot actions from visual observations and language instructions, but understanding a manipulation goal and procedural steps does not determine how the end effector should move through the surrounding scene. We introduce **Geometry Forcing for VLA (GF-VLA)**, which brings both motion and scene geometry together on a shared action-conditioning pathway. During training, motion geometry forcing supervises time-indexed motion latents with positions and successive displacements from a sparse, temporally ordered 3D end-effector (EEF) trace within an action chunk. Scene geometry forcing reconstructs future depth at the chunk endpoint, guiding the scene latents read by motion queries. These motion latents, together with native vision-language features, condition a flow-matching Action Expert, supporting embodied reasoning that jointly considers 3D motion and future scene geometry during action generation. At inference, scene and motion queries remain, but neither geometric decoder is used. GF-VLA achieves 98.40% mean success on LIBERO and 83.8% overall zero-shot success on LIBERO-Plus; on RoboCasa-GR1, it improves success by 6.3 pp over a strong VLA baseline. Across three real-world tasks, GF-VLA achieves 73.3% average success under standard conditions, exceeding the baseline by 12.2 pp. These results support the benefit of coupling 3D motion trajectories with scene geometry at the action-chunk endpoint for action generation.

1 INTRODUCTION

Vision-language-action (VLA) models leverage pretrained vision-language models (VLMs) to map visual observations and language instructions to robot actions, achieving strong performance across manipulation tasks (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). Yet complex long-horizon tasks and fine-grained manipulation expose a *seeing-to-doing gap* (Yuan et al., 2026a): understanding the task goal and procedural steps does not determine how the end effector should move in a specific scene. Successful manipulation further requires grounding task-level guidance in motion that is spatially compatible with the target location and surrounding scene structure. Conventional action supervision provides target control commands, but only indirectly constrains whether the policy's internal representations capture this motion and its relationship to scene geometry. We therefore focus on *execution-oriented geometric reasoning*: jointly representing the end-effector motion path and the relevant scene structure over the temporal horizon of the actions being predicted, providing a geometric basis for action generation.

Embodied-reasoning VLAs (Team et al., 2025) often introduce intermediate representations between task understanding and action execution, focusing on task planning, future scene, or robot motion. Task-level methods use textual steps or subgoals to organize the process (Zawalski et al., 2024; Sun et al., 2026b), but these task-level descriptions typically do not explicitly establish a correspondence between procedural steps and execution. Scene-level methods predict future keyframes, or depth maps to describe future visual states and spatial structure (Zhao et al., 2025; Shen et al., 2025; Zhang et al., 2025), but do not explicitly specify the 3D end-effector path for reaching states. Motion-level methods use 2D image-plane points or traces to describe end-effector movement (Li et al., 2025a; Lee et al., 2025; Yuan et al., 2026a), preserving correspondence with visual observations but lacking the depth needed to determine 3D positions. As illustrated in fig. 1, these representations provide task guidance, future scenes, or motion cues, yet concrete motion still needs to be connected to its scene geometry. Adding depth to a trace specifies 3D coordinates at sampled end-effector

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.

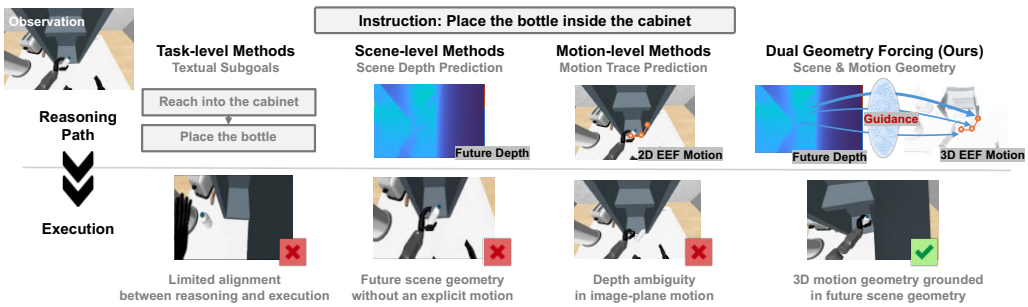

Figure 1: **Motivation for dual geometry forcing.**

positions; however, trace supervision covers only these sampled positions and does not explicitly supervise the surrounding spatial structure. We therefore propose *dual geometry forcing*, pairing a 3D end-effector trace within an action chunk with future depth at the same chunk's endpoint: the former supervises how the end effector moves, while the latter supervises scene geometry after that movement. Both supervision signals thus correspond temporally to the same action chunk and jointly shape the motion representation used for action generation.

We introduce **Geometry Forcing for VLA (GF-VLA)**, which realizes dual geometry forcing within a shared VLM. Given the current images and instruction, the VLM jointly processes native vision-language tokens, scene geometry queries, and time-indexed motion queries. Motion geometry forcing supervises motion latents with a sparse, temporally ordered UVD end-effector trace over the action chunk, where UVD combines image-plane coordinates with camera depth. Scene geometry forcing supervises scene latents with future depth at the same chunk's endpoint. Motion queries read these depth-supervised scene latents, bringing spatial context into the trace-supervised motion latents that, together with native vision-language features, condition a flow-matching Action Expert. The geometric decoders impose supervision only during training; at inference, the queries produce these latents without decoding either geometric target. We evaluate GF-VLA on three simulation benchmarks and three real-world tasks, with ablations examining the contributions of motion and scene geometry forcing.

Our contributions are threefold: (1) We introduce dual geometry forcing, pairing 3D end-effector motion within an action chunk with future scene geometry at its endpoint. (2) We couple these targets by letting motion queries read future-depth-supervised scene latents; the resulting UVD trace-supervised motion latents condition the Action Expert without geometric decoding at inference. (3) We evaluate GF-VLA on three simulation benchmarks and three real-world tasks. GF-VLA achieves 98.4% mean success on LIBERO, 83.8% overall zero-shot success on LIBERO-Plus, and 71.9% mean success on RoboCasa-GR1. In real-world experiments, it outperforms the baseline across long-horizon execution, bimanual fine manipulation, and precise spatial placement, with average gains of 12.2 pp under Easy conditions and 13.3 pp under Hard conditions.

2 Related Work

Embodied reasoning for VLA spans task-, motion-, and scene-level representations (Yuan et al., 2026b;c). We focus on motion and scene reasoning, which relate to GF-VLA's geometric targets.

Motion Reasoning. HAMSTER (Li et al., 2025a) and LoHo-Manip (Liu et al., 2026) use 2D traces to guide policies, while MolmoAct (Lee et al., 2025) combines depth-aware perception with image-space traces. A 2D trace alone leaves the metric 3D path ambiguous. 3D HAMSTER (Hwang et al., 2026) and OASIS (Chen et al., 2026a) address this with metric end-effector trajectories. MolmoMotion (Zhang et al., 2026b) and μ_0 (Lee et al., 2026) forecast 3D object or interaction-point motion for downstream policies. GF-VLA instead supervises time-indexed motion latents with positions and successive displacements along a sparse UVD end-effector trace within each action chunk. Motion queries read scene latents supervised by future-depth reconstruction at the same chunk's endpoint, providing geometric context for predicting that trace.

Scene Reasoning. CoT-VLA (Zhao et al., 2025) predicts future image goals, while LaRA-VLA (Bai et al., 2026) and DreamVLA (Zhang et al., 2025) use latent predictions to guide ac-

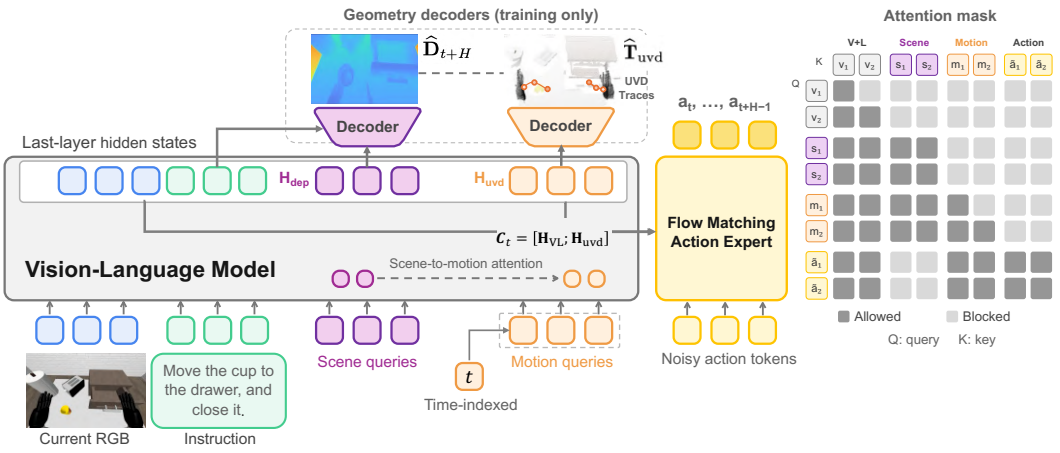

Figure 2: **GF-VLA framework.** The VLM processes the current RGB observation and instruction with scene and time-indexed motion queries. Motion queries read scene latents; the resulting motion latents and native vision-language features condition the Action Expert. Training-only decoders predict endpoint depth and the within-chunk UVD end-effector trace.

tions. DepthVLA (Yuan et al., 2025) and GeoVLA (Sun et al., 2025) incorporate depth-based features, whereas QDepth-VLA (Li et al., 2025b) uses quantized depth prediction as auxiliary supervision. LaMP (Wang et al., 2026b) conditions an Action Expert on latent 3D scene flow, while Track4Action (Wang et al., 2026a) distills future 3D tracking features into a policy. GF-VLA uses the depth map at the action-chunk endpoint as a supervision target for future scene geometry. The resulting scene latents are read by motion queries, providing spatial context for the UVD motion latents that condition action generation.

3 Dual Geometry Forcing for VLA

The core of GF-VLA is dual geometry forcing: end effector (EEF) motion geometry within an action chunk and scene geometry at its endpoint are brought together for action generation (fig. 2). We first describe motion geometry forcing, which supervises motion latents with a sparse 3D EEF trace over the action chunk (sec. 3.1). We then introduce scene geometry forcing through future depth at the same chunk's endpoint (sec. 3.2). Finally, we show how dual geometry latents that condition the Action Expert (sec. 3.3). Both geometric decoders are needed only during training.

At time t, the policy receives a visual observation o_t and a language instruction ℓ and predicts an H-step action chunk $\mathbf{A}_t = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1})$. We consider a VLA policy comprising a VLM backbone F_ϕ and a flow-matching Action Expert F_θ (Black et al., 2024; Bjorck et al., 2025):

$$\mathbf{H}_{\mathrm{VL}} = F_\phi(o_t, \ell), \qquad \mathbf{A}_t = F_\theta(\boldsymbol{\epsilon}; \mathbf{H}_{\mathrm{VL}}, \boldsymbol{\tau}), \tag{1}$$

Here, $\mathbf{H}_{\mathrm{VL}}$ denotes vision-language features, $\boldsymbol{\epsilon}$ is the initial noise, and $\boldsymbol{\tau}$ is the flow-time sequence.

3.1 Motion Geometry Forcing

Action labels specify the control commands to predict, but do not explicitly constrain whether the policy's internal representation captures the EEF's motion path through an action chunk. A temporally ordered EEF trace makes this execution-relevant path explicit, specifying both the positions traversed and their order. A 2D trace records only the image-plane projection, leaving the EEF's 3D position ambiguous. We use a sparse 3D EEF trace as the target for motion-geometry forcing.

UVD Motion Representation. We express demonstrated EEF positions in the main camera's UVD coordinates. Image coordinates (u, v) preserve correspondence between the trace and the visible scene, while metric camera depth d specifies position along the camera axis. Given the camera intrinsics, (u, v, d) uniquely determines a 3D point in the main camera frame. Over the interval from

the current state through the next H actions, we sample K temporally spaced positions, including the start at t and the endpoint at $t + H$, to form the ordered trace

$$\mathbf{T}_{\text{uvd}} = (\mathbf{t}_1, \ldots, \mathbf{t}_K), \qquad \mathbf{t}_k = (u_k, v_k, d_k). \tag{2}$$

During training, we divide u_k and v_k by the image width minus one and height minus one, respectively. The UV targets therefore lie in $[0, 1]$, while d_k remains in meters. For bimanual data, we form one trace per hand and assign K motion latents to each trace.

Time-Indexed Motion Latents. We assign one motion query to each sampled trace point. The k-th query combines a shared learnable embedding with its normalized time $\xi_k \in [0, 1]$:

$$\mathbf{q}_{\text{uvd},k} = \mathbf{e}_{\text{uvd}} + f_{\text{time}}(\xi_k). \tag{3}$$

Here, f_{time} maps time to an embedding. Equations in this subsection use single-hand notation; for bimanual data, queries for hand $b \in \{\text{L}, \text{R}\}$ also receive a distinct learned embedding $\mathbf{e}^{(b)}_{\text{hand}}$. The VLM transforms the motion queries $\mathbf{Q}_{\text{uvd}}$ into latents $\mathbf{H}_{\text{uvd}} = (\mathbf{h}_{\text{uvd},1}, \ldots, \mathbf{h}_{\text{uvd},K})$, with $\mathbf{h}_{\text{uvd},k}$ corresponding to trace point $\mathbf{t}_k$.

Position and Displacement Supervision. A shared two-layer MLP maps each motion latent to its corresponding UVD point, yielding the trace $\widehat{\mathbf{T}}_{\text{uvd}} = \text{Decoder}_{\text{uvd}}(\mathbf{H}_{\text{uvd}})$. We supervise the predicted trace at two levels. The absolute loss compares each predicted UVD point $\widehat{\mathbf{t}}_k$ with its demonstrated point $\mathbf{t}_k$, anchoring the path at each sampled time. The relative loss compares predicted and demonstrated UVD displacements between adjacent points, directly emphasizing local changes in image position and depth:

$$\mathcal{L}_{\text{uvd}} = \underbrace{\text{SmoothL1}(\widehat{\mathbf{T}}_{\text{uvd}}, \mathbf{T}_{\text{uvd}})}_{\mathcal{L}_{\text{abs}}} + \lambda_{\text{rel}} \underbrace{\text{SmoothL1}(\Delta\widehat{\mathbf{T}}_{\text{uvd}}, \Delta\mathbf{T}_{\text{uvd}})}_{\mathcal{L}_{\text{rel}}}. \tag{4}$$

Here, $(\Delta\mathbf{T}_{\text{uvd}})_k = \mathbf{t}_{k+1} - \mathbf{t}_k$. Both losses use only trace points whose EEF projections lie within the main-camera image; the relative term includes an adjacent pair only when both points are valid. $\mathcal{L}_{\text{rel}}$ additionally emphasizes adjacent UVD changes, with weight λ_{rel}. The decoded coordinates are used only to compute training losses; at inference, action generation uses the motion latents without running the UVD decoder.

3.2 Scene Geometry Forcing

Although a UVD trace specifies the EEF's future path, predicting that path from the current RGB observation still requires estimating spatial relationships among the EEF, objects, and surrounding workspace. Trace supervision alone does not directly provide this scene-level geometric context. We therefore use depth at the endpoint of the same action chunk to supervise scene latents, encouraging them to carry spatial information for motion queries predicting the 3D trace. Because the motion latents condition the Action Expert, this architecture gives future-depth supervision a path to action generation. We choose geometry at the chunk endpoint as the target to align scene supervision with the actions being predicted. Future depth is used only as a training target, not as a policy input.

Geometry Latent Construction. The VLM processes a set of learnable scene-geometry queries $\mathbf{Q}_{\text{dep}}$ with the current image and language instruction, producing scene latents $\mathbf{H}_{\text{dep}}$. Unlike time-indexed motion queries, scene queries do not correspond to individual trace times; their latents are jointly supervised by depth at the chunk endpoint. Motion queries read these depth-supervised scene latents when forming UVD motion latents.

Depth Decoding and Supervision. During training, attention pooling summarizes the scene latents into a conditioning vector. The depth decoder combines this vector with native vision-language features to predict the depth map at the chunk endpoint:

$$\mathbf{z}_{\text{dep}} = \text{AttnPool}(\mathbf{H}_{\text{dep}}), \qquad \widehat{\mathbf{D}}_{t+H} = \text{Decoder}_{\text{dep}}(\mathbf{H}_{\text{VL}}, \mathbf{z}_{\text{dep}}). \tag{5}$$

The image tokens for the corresponding camera view in $\mathbf{H}_{\text{VL}}$ are reshaped into a spatial feature grid. The pooled scene vector $\mathbf{z}_{\text{dep}}$ generates FiLM scale and shift parameters that modulate the grid across three residual-convolution and upsampling stages. The image features provide spatial detail while the scene latents condition the depth prediction. Through attention pooling and FiLM

modulation, the depth loss backpropagates to the scene latents read by motion queries. We apply Smooth-L1 over valid pixels of the target depth map $\mathbf{D}_{t+H}$:

$$\mathcal{L}_{\text{dep}} = \text{SmoothL1}(\widehat{\mathbf{D}}_{t+H}, \mathbf{D}_{t+H}). \tag{6}$$

When multiple camera views are available, we predict depth at the chunk endpoint for each view and sum the view-specific losses. These targets supervise scene geometry visible from each camera, whereas the EEF trace remains in the main camera's UVD frame. This keeps motion supervision in one consistent coordinate system while allowing geometric information from other views to guide motion queries through the scene latents.

3.3 Joint Geometry Forcing and Action Generation

Motion geometry forcing supervises the EEF path, while scene geometry forcing supervises the spatial context of that path. GF-VLA allows motion queries to read scene latents when forming UVD motion latents. The resulting motion latents then condition the Action Expert.

Coupling Scene and Motion Geometry. The VLM jointly processes native image and text tokens, scene-geometry queries, and motion queries to obtain $\mathbf{H}_{\text{VL}}$, $\mathbf{H}_{\text{dep}}$, and $\mathbf{H}_{\text{uvd}}$, respectively:

$$[\mathbf{H}_{\text{VL}}; \mathbf{H}_{\text{dep}}; \mathbf{H}_{\text{uvd}}] = F_\phi(o_t, \ell; \mathbf{Q}_{\text{dep}}, \mathbf{Q}_{\text{uvd}}). \tag{7}$$

As shown by the attention mask in fig. 2, scene queries read native vision-language tokens; the motion query for trace point k reads native tokens, all scene latents, and motion queries up to k. This directed interaction gives motion latents access to scene geometry while preserving their temporal order and leaving native vision-language features unchanged.

The two forcing objectives are also coupled during optimization. The future-depth loss trains the scene latents through the depth decoder, while the trace loss directly supervises the motion latents. Because motion queries attend to the scene latents, gradients from both the trace and action losses can also propagate into the scene-query pathway through these attention connections. Consequently, the scene latents are learned not only from future-depth supervision, but also in relation to the motion representation used for action generation.

Action Generation and Joint Training. The Action Expert cross-attends to native vision-language features and motion latents, but not directly to scene latents. Noisy action tokens attend bidirectionally within the chunk, so each flow-matching update can use the full motion-latent sequence. At a sampled flow time $\tau \in [0, 1]$, we define the conditioning sequence and interpolate between noise $\boldsymbol{\epsilon}$ and the demonstrated action chunk $\mathbf{A}_t$:

$$\mathbf{C}_t = [\mathbf{H}_{\text{VL}}; \mathbf{H}_{\text{uvd}}], \qquad \mathbf{A}_t^\tau = (1-\tau)\boldsymbol{\epsilon} + \tau \mathbf{A}_t. \tag{8}$$

We train the Action Expert's velocity field v_θ against the target velocity $\mathbf{A}_t - \boldsymbol{\epsilon}$ with the flow-matching loss (Lipman et al., 2022):

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\mathbf{A}_t, \boldsymbol{\epsilon}, \tau} \left\| v_\theta(\mathbf{A}_t^\tau, \tau; \mathbf{C}_t) - (\mathbf{A}_t - \boldsymbol{\epsilon}) \right\|_2^2. \tag{9}$$

Integrating v_θ over the flow-time sequence $\boldsymbol{\tau}$ yields the action generator F_θ in eq. (1). We jointly optimize the action, trace, and depth losses:

$$\mathcal{L} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{uvd}}\mathcal{L}_{\text{uvd}} + \lambda_{\text{dep}}\mathcal{L}_{\text{dep}}. \tag{10}$$

At inference, scene and motion queries remain in the VLM forward pass, but the UVD and depth decoders are omitted. The Action Expert generates the action chunk from $\mathbf{C}_t$ without explicitly predicting either geometric target.

4 Experiments

Our experiments examine whether jointly learning 3D motion geometry within each action chunk and the scene geometry at its endpoint improves action generation. We evaluate GF-VLA across simulation and real-world tasks, and further analyze the two forms of geometric forcing through alternative supervision targets, component ablations, and visualizations of predicted motion and scene geometry.

Table 1: Success rates (%) on the four LIBERO evaluation suites. Bold indicates the best performing model, and underline indicates the runner-up model; ties receive the same formatting.

Category	Method	Spatial	Object	Goal	Long	Avg.
Motion-related VLA	MolmoAct-7B-D (Lee et al., 2025)	87.0	95.4	87.6	77.2	86.6
	LoHo-Manip (Liu et al., 2026)	98.0	98.6	98.0	95.2	97.5
	OASIS (Chen et al., 2026a)	**99.0**	98.8	97.4	95.2	97.6
	ELAN4D ($\pi_{0.5}$) (He et al., 2026)	98.2	98.8	96.8	94.2	97.0
	Track4Action (Wang et al., 2026a)	95.0	99.6	97.6	95.8	97.0
Scene-related VLA	4D-VLA (Zhang et al., 2026a)	88.9	95.2	90.9	79.1	88.6
	DepthVLA (Yuan et al., 2025)	96.4	98.0	95.8	89.2	94.9
	DreamVLA (Zhang et al., 2025)	97.5	94.0	89.5	89.5	92.6
	VLA-JEPA (Sun et al., 2026a)	96.2	99.6	97.2	95.8	97.2
General VLA	π_0 (Black et al., 2024)	96.8	98.8	95.8	85.2	94.2
	$\pi_{0.5}$ (Physical Intelligence et al., 2025)	98.8	98.2	98.0	92.4	96.9
	GR00T-N1.6 (NVIDIA GEAR Team, 2025)	97.7	98.5	97.5	94.4	97.0
	ZR-0 (Li et al., 2026)	97.4	99.4	98.0	**96.4**	97.8
Ours	**GF-VLA**	98.8	**99.8**	**99.2**	95.8	**98.4**

Table 2: Zero-shot success rates (%) on LIBERO-Plus. Bold indicates the best performing model, and underline indicates the runner-up model.

Method	Cam.	Robot	Lang.	Light	Back.	Noise	Layout	Overall
DreamVLA (Zhang et al., 2025)	65.0	40.8	63.5	85.7	82.6	84.9	74.0	69.9
ELAN4D ($\pi_{0.5}$) (He et al., 2026)	63.7	70.7	77.8	89.8	91.4	79.9	81.4	78.2
Track4Action (Wang et al., 2026a)	**76.6**	60.7	87.1	96.3	96.1	88.4	78.7	82.3
VLA-JEPA (Sun et al., 2026a)	63.3	67.1	85.4	95.6	93.6	66.3	**85.1**	79.5
Cosmos-Policy (Kim et al., 2026)	75.8	63.3	81.7	**96.5**	88.9	**92.7**	82.2	82.2
ABot-M0 (Yang et al., 2026)	60.4	67.9	86.4	96.2	91.6	86.4	82.6	80.5
PILOT (Ma et al., 2026)	76.0	70.0	84.0	92.0	85.0	78.0	82.0	81.0
GF-VLA	68.9	**71.4**	**90.8**	95.1	95.8	90.1	81.3	**83.8**

4.1 Experimental Setup

Simulation Benchmarks. We evaluate GF-VLA on three complementary simulation benchmarks covering standard end-effector manipulation, robustness to observation and state changes, and high-dimensional bimanual control. **LIBERO** (Liu et al., 2023) includes Spatial, Object, Goal, and Long suites for a Franka Panda arm with end-effector control. **LIBERO-Plus** (Fei et al., 2025) extends LIBERO with perturbations to camera viewpoints, robot initial states, language instructions, lighting, backgrounds, sensor noise, and object layouts, yielding 10,030 perturbed task instances. We directly evaluate the LIBERO-trained checkpoint on all 10,030 instances without additional training. **RoboCasa-GR1** (Bjorck et al., 2025) comprises 24 tabletop manipulation tasks for the GR-1 humanoid, requiring coordinated control of two dexterous hands through high-dimensional joint-space actions. We train separate policies for LIBERO and RoboCasa-GR1 and use 50 rollouts per task on both benchmarks. All ablations and analyses in sec. 4.3 use RoboCasa-GR1.

Real World Tasks. We evaluate GF-VLA on two bimanual tasks on ARX-Lift2S and one single-arm task on Realman RM75. **Object Packing (ARX)** evaluates long-horizon manipulation with a final bimanual closing stage. The robot sequentially places a tennis ball, a toy carrot, and a Rubik's cube into a box, then uses both arms to close the left and right hinged lids inward. **Bricks Disassembly (ARX)** evaluates contact-rich manipulation across three assembled configurations. The robot separates the bricks and places them into the color-matched compartments of a tray, requiring precise contact handling and placement. **Hanger Placement (Realman RM75)** evaluates precise spatial manipulation. The robot aligns a hanger with a designated hook and places it securely onto the hook. Platform-specific geometry targets and training settings are detailed in app. A.

Metrics. We report success rate (SR), the percentage of successful evaluation episodes. We report UV error and depth error to measure consistency between traces decoded from motion latents

Table 3: Average success rates (%) on RoboCasa-GR1. Bold indicates the best performing model, and underline indicates the runner-up model.

Method	SR
ZR-0 (Li et al., 2026)	69.3
GR00T-N1.5 (Bjorck et al., 2025)	48.2
GR00T-N1.6 (NVIDIA GEAR Team, 2025)	47.6
VP-VLA (Wang et al., 2026c)	53.8
JoyAI-RA (Zhang et al., 2026c)	63.2
DIAL (Chen et al., 2026b)	<u>70.2</u>
PILOT (Ma et al., 2026)	62.6
ABot-M0 (Yang et al., 2026)	58.3
GF-VLA	**71.9**

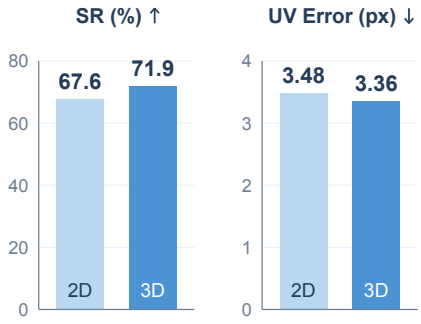

Figure 3: Motion forcing analysis.

and the robot's subsequently executed motion. UV error measures image-plane distance in pixels; depth error measures camera-frame depth differences in millimeters. Lower errors indicate closer agreement. Definitions and aggregation procedures are provided in app. B.3.

4.2 Simulation Results

LIBERO. As shown in tab. 1, GF-VLA achieves the highest average SR of 98.40%, with 99.8% on Object, 99.2% on Goal, 98.8% on Spatial, and 95.8% on Long. On Object, motion geometry forcing captures interaction structure that transfers across object instances. On Goal, where object and spatial configurations remain fixed while task instructions change, the result shows that task semantics are grounded in task-specific motion geometry before action generation. The Spatial result points to scene geometry forcing helping robot motion adapt to changes in object relationships, while the Long result indicates that jointly learned motion and scene geometry remain effective over multi-stage manipulation.

LIBERO-Plus. As shown in tab. 2, GF-VLA achieves the highest overall zero-shot SR of 83.8%, exceeding the next-best method by 1.5 percentage points. It records the best SR under Robot and Language perturbations, reaching 71.4% and 90.8% in the two settings. Under Robot perturbations, motion geometry compactly encodes task semantics from language and visual observations, allowing the policy to adjust movement from different initial robot poses. Under Language perturbations, the learned motion geometry remains stable when the same task is phrased in different ways. Under Camera perturbations, GF-VLA reaches 68.9% SR, consistent with scene geometry forcing providing spatial context across viewpoint changes. It also reaches 95.1% under lighting and 95.8% under background perturbations. Under sensor noise, GF-VLA achieves 90.1% SR, indicating that dual geometry forcing keeps action generation grounded in task-relevant motion and scene structure despite corrupted observations.

RoboCasa-GR1. As shown in tab. 3, GF-VLA achieves the highest SR of 71.9%, exceeding DIAL by 1.7 percentage points. In this setting, motion geometry describes hand trajectories in 3D space, while the policy generates high-dimensional bimanual joint-space actions. We define each hand's motion reference point as the midpoint between its thumb and index fingertips, extending motion forcing beyond gripper-based manipulation to a humanoid with dexterous hands. The resulting geometric trajectory provides an intermediate description of hand movement, while the Action Expert maps it to joint-space control. GF-VLA maintains its advantage in this setting, showing that dual geometry forcing remains effective when the supervised motion geometry and final actions lie in different action spaces. Together with scene geometry learned at the action-chunk endpoint, this result gives GF-VLA the best score on the benchmark and demonstrates the effectiveness of dual geometry forcing for high-dimensional control.

4.3 Ablation Studies and Analysis

Motion Geometry Forcing. We compare 2D and 3D motion forcing to test whether supervising motion beyond the image plane improves action generation. The 2D variant uses UV traces, while

Table 4: Scene forcing analysis.

Current	Future	SR (%) ↑	UV Error (px) ↓	Depth Error (mm) ↓
×	×	66.8	3.46	8.67
✓	×	67.6	3.74	9.34
✓	✓	67.9	3.47	8.47
×	✓	**71.9**	**3.36**	**8.09**

Table 5: Dual forcing ablation.

Motion	Scene	SR (%) ↑
×	×	65.6
✓	×	66.8
×	✓	70.8
✓	✓	**71.9**

the 3D variant uses UVD traces. Both variants reconstruct scene depth at the action-chunk endpoint for scene geometry forcing and use the same trace sampling and Action Expert. As shown in fig. 3, 3D motion forcing raises SR on Robocasa-GR1 from 67.6% to 71.9%, a gain of 4.3 percentage points. UV error also decreases from 3.48 to 3.36 pixels, suggesting that incorporating depth into motion forcing can also improve agreement between predicted and executed image-plane motion.

Scene Geometry Forcing. We compare current depth forcing, future depth forcing, and their combination while keeping 3D motion forcing fixed (tab. 4). The targets describe scene geometry at the current observation and the action-chunk endpoint. Current depth forcing raises SR from 66.8% to 67.6% but increases both errors. Combining both targets improves on current depth forcing, reaching 67.9% SR with lower errors. Future depth forcing alone achieves the highest SR of 71.9%, exceeding motion forcing alone by 5.1 percentage points. It also yields the lowest UV and depth errors of 3.36 pixels and 8.09 mm.

We believe this advantage comes from aligning scene geometry learning with the motion to be executed. Reconstructing depth at the action-chunk endpoint encourages the model to anticipate the scene after that movement, providing spatial context for learning the corresponding motion trace.

We also examine feature alignment, commonly used in latent reasoning, as an alternative implementation of future depth forcing. Experimental results show that depth reconstruction achieves higher SR and lower depth error than DA3 feature alignment; detailed results are provided in app. C.2.

Dual Geometry Forcing. We compare the base VLA, motion forcing alone, scene forcing alone, and their combination to assess the contribution of each geometry objective (tab. 5). The base VLA uses only action supervision and achieves 65.6% SR. Motion forcing alone raises SR to 66.8%, while scene forcing alone reaches 70.8%. Combining both objectives achieves the highest SR of 71.9%, exceeding the base VLA by 6.3 percentage points and scene forcing alone by 1.1 percentage points. The additional gain over either objective alone supports jointly constraining scene and motion geometry for action learning.

Beyond task success, we visualize scene and motion predictions across consecutive action chunks in fig. 4. Each column pairs the current observation with predicted scene depth at the chunk endpoint and the UV projection of the decoded UVD trace; darker blue trace points indicate smaller camera-frame depth. In this task, the robot places a cup in a drawer and then closes it. The predicted depth maps depict the drawer and robot arm. The motion traces capture a shift in the active hand: the left hand places the cup while the right remains still; then the left remains still as the right hand closes the drawer. This example shows how the two geometry representations reflect the changing scene and robot movement during a continuous manipulation sequence.

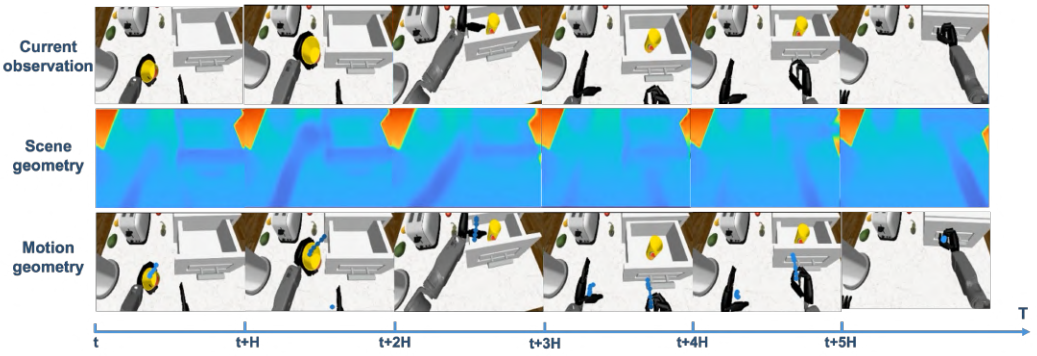

Figure 4: Visualization of dual geometry predictions for the task "Pick up the cup, place it in the drawer, and close the drawer."

4.4 Real-World Experiments

We compare GF-VLA with QwenGR00T (Baseline) (StarVLA Community, 2026; Bjorck et al., 2025), which uses a Qwen3.5-4B backbone and a flow-based Action Expert without geometry branches. Easy conditions match the demonstration settings, while Hard conditions introduce a changed background, distractor bricks, and unseen hook positions for the three tasks, respectively (fig. 5). Each method is evaluated in 30 trials per task and condition. Tab. 6 reports success rates; further details are in app. B.2. Under Easy conditions, GF-VLA reaches 93.3% SR on Object Packing and 63.3% on both Bricks Disassembly and Hanger Placement, exceeding the baseline by 10.0–13.3 pp. These gains cover long-horizon bimanual execution and fine-grained contact and placement. Under Hard conditions, GF-VLA outperforms the baseline on all three tasks by 10.0–16.7 pp. With a changed background, Object Packing reaches 80.0% SR versus the baseline's 63.3%. Distractor bricks can prompt an incorrect disassembly sequence, while unseen hook positions cause the largest Easy-to-Hard drop for both methods, showing that precise placement at unfamiliar locations remains challenging.

Table 6: Real-world task success rates (%).

	Baseline		GF-VLA	
Task	Easy	Hard	Easy	Hard
Object Packing	80.0	63.3	**93.3**	**80.0**
Bricks Disassembly	53.3	36.7	**63.3**	**50.0**
Hanger Placement	50.0	23.3	**63.3**	**33.3**

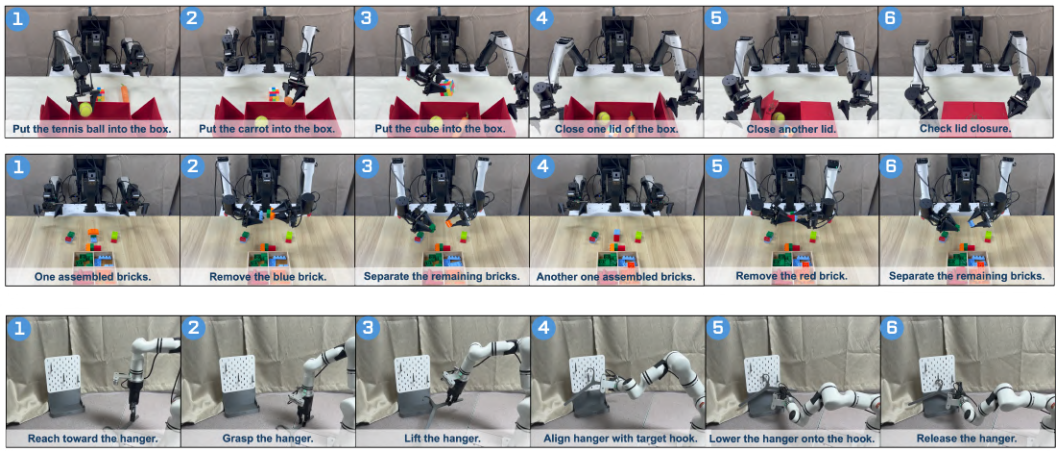

Figure 5: Real-robot tasks and example rollouts. Top: Object Packing on ARX-Lift2S. Middle: Bricks Disassembly on ARX-Lift2S. Bottom: Hanger Placement on Realman RM75.

5 Conclusion

We introduce GF-VLA, a vision-language-action framework that couples 3D motion geometry with future scene geometry for action generation. Motion geometry forcing supervises time-indexed motion latents with a sparse UVD trace, while scene geometry forcing supervises scene latents through depth reconstruction at the action-chunk endpoint. Motion queries read these future-depth-supervised scene latents, and the resulting motion latents condition the Action Expert alongside native vision-language features. Across three simulation benchmarks and three real-world tasks, GF-VLA achieves a mean success rate of 98.4% on LIBERO, an overall zero-shot success rate of 83.8% on LIBERO-Plus, and a mean success rate of 71.9% on RoboCasa-GR1. On the real-world tasks, it improves average success rate over the baseline by 12.2 percentage points under Easy conditions and 13.3 percentage points under Hard conditions. These results demonstrate the value of coupling motion and scene geometry for VLA action generation. However, task-level gains are uneven, and prediction–execution errors increase at later waypoints within an action chunk. Future work will explore how to make more effective use of the information encoded by dual geometry forcing and extend the approach to navigation and mobile manipulation.

AI USE STATEMENT

We used generative AI tools to improve manuscript clarity and translation. We did not use them to discuss the method design, generate data, or produce experimental measurements. The author take responsibility for the final text, claims, and results.

ETHICS STATEMENT

To our knowledge, this study raises no specific ethics concerns. It uses public simulation benchmarks and controlled tabletop robot experiments, without human-subject data or sensitive personal information.

REPRODUCIBILITY STATEMENT

The method and main evaluation are described in secs. 3 and 4. Apps. A and B provide geometric target preparation, training settings, and evaluation protocols; apps. C and D provide additional results and case studies.

REFERENCES

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 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. doi: 10.48550/arXiv.2410.24164. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Xinzhe Chen, Sihua Ren, Liqi Huang, Haowen Sun, Mingyang Li, Xingyu Chen, Zeyang Liu, and Xuguang Lan. OASIS: Observation-action space alignment via SE(3) trajectory prediction for robotic manipulation. *arXiv preprint arXiv:2605.25829*, 2026a. doi: 10.48550/arXiv.2605.25829. URL https://arxiv.org/abs/2605.25829.

Yi Chen, Yuying Ge, Hui Zhou, Mingyu Ding, Yixiao Ge, and Xihui Liu. DIAL: Decoupling intent and action via latent world modeling for end-to-end vla. *arXiv preprint arXiv:2603.29844*, 2026b. URL https://arxiv.org/abs/2603.29844.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. doi: 10.48550/arXiv.2510.13626. URL https://arxiv.org/abs/2510.13626.

Zeyuan He, Bowen Yang, Zhirui Fang, Keru Zhou, Lei Jiang, Jingjing Qian, Fan Mo, Junchi Yan, Philip Torr, Xiu Li, Li Jiang, and Jialin Yu. ELAN4D: Embodiment-Centric 4D Supervision for Vision-Language-Action Models via Plug-and-Play Adaptation. *arXiv preprint arXiv:2605.30484*, 2026.

Dongyoon Hwang, Byungkun Lee, Dongjin Kim, Hyojin Jang, Hoiyeong Jin, Jueun Mun, Minho Park, Hojoon Lee, Hyunseung Kim, and Jaegul Choo. 3D HAMSTER: Bridging planning and control in hierarchical vision language action models through 3d trajectory guidance. *arXiv preprint arXiv:2606.31329*, 2026.

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. doi: 10.48550/arXiv.2406.09246. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Seungjae Lee, Yoonkyo Jung, Jusuk Lee, Jonghun Shin, Amir Hossein Shahidzadeh, Yao-Chih Lee, H. Jin Kim, Jia-Bin Huang, and Furong Huang. μ_0: A scalable 3d interaction-trace world model. *arXiv preprint arXiv:2606.13769*, 2026.

Haoyang Li, Guanlin Li, Youhe Feng, Chen Zhao, Zhuoran Wang, Yang Li, Qizhe Wei, Shifeng Bao, Haitao Shen, Yihan Zhao, Tong Yang, and Jing Zhang. Training vision-language-action models with dense embodied chain-of-thought supervision. *arXiv preprint arXiv:2606.30552*, 2026.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025a.

Yixuan Li, Yuhui Chen, Mingcai Zhou, Haoran Li, Zhengtao Zhang, and Dongbin Zhao. QDepth-VLA: Quantized Depth Prediction as Auxiliary Supervision for Vision-Language-Action Models. *arXiv preprint arXiv:2510.14836*, 2025b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. doi: 10.48550/arXiv.2210.02747. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023. doi: 10.48550/arXiv.2306.03310. URL https://arxiv.org/abs/2306.03310.

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-Horizon Manipulation via Trace-Conditioned VLA Planning. *arXiv preprint arXiv:2604.21924*, 2026.

Xiangkai Ma, Yue Ma, Junjie Wang, Sheng Xu, Mingyang Li, Han Zhang, Yuzheng Zhuang, Wenzhong Li, and Zhihao Yuan. Decoupling intention from trajectory: A representational deduction framework for world action models. *arXiv preprint arXiv:2608.06994*, 2026. URL https://arxiv.org/abs/2608.06994.

NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots. NVIDIA Research project page, December 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_6/.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James

Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025. doi: 10.48550/arXiv.2512.06963. URL https://arxiv.org/abs/2512.06963.

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing. *arXiv preprint arXiv:2604.05014*, 2026.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model, 2026a. URL https://arxiv.org/abs/2602.10098.

Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. GeoVLA: Empowering 3D representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025. doi: 10.48550/arXiv.2508.09071. URL https://arxiv.org/abs/2508.09071.

Nan Sun, Yuan Zhang, Yongkun Yang, Wentao Zhao, Peiyan Li, Jun Guo, Wenxuan Song, Pengxiang Ding, Runze Suo, Yifei Su, Xin Xiao, Xinghang Li, and Huaping Liu. Revisiting embodied chain-of-thought for generalizable robot manipulation. *arXiv preprint arXiv:2606.03784*, 2026b.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Chenyi Wang, Xinkai Wang, Bokai Lin, Jialin Tian, Fucheng Zhang, Cewu Lu, and Lixin Yang. Track4Action: Distilling world-centric 3d tracker into vision-language-action policies. *arXiv preprint arXiv:2608.03727*, 2026a. doi: 10.48550/arXiv.2608.03727. URL https://arxiv.org/abs/2608.03727.

Xinkai Wang, Chenyi Wang, Yifu Xu, Mingzhe Ye, Fucheng Zhang, Jialin Tian, Xinyu Zhan, Lifeng Zhu, Cewu Lu, and Lixin Yang. LaMP: Learning Vision-Language-Action Policy with 3D Scene Flow as Latent Motion Prior. *arXiv preprint arXiv:2603.25399*, 2026b.

Zixuan Wang, Yuxin Chen, Yuqi Liu, Jinhui Ye, Pengguang Chen, Changsheng Lu, Shu Liu, Bei Yu, and Jiaya Jia. VP-VLA: Visual Prompting as an Interface for Vision-Language-Action Models. *arXiv preprint arXiv:2603.22003*, 2026c.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. ABot-M0: VLA Foundation Model for Robotic Manipulation with Action Manifold Learning. *arXiv preprint arXiv:2602.11236*, 2026.

Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. DepthVLA: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025.

Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, Jinyi Liu, Pengyi Li, Yan Zheng, and Jianye Hao. From seeing to doing: Bridging reasoning and decision for robotic manipulation, 2026a. URL https://arxiv.org/abs/2505.08548.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, Hongyao Tang, and Jianye Hao. Embodied-r1: Reinforced embodied reasoning for general robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 8984–9009, 2026b.

Yifu Yuan, Yaoting Huang, Xianze Yao, Yutong Li, Shuoheng Zhang, Linqi Han, Pengyi Li, Jiangeng Sun, Wenting Jia, Zhao Zhang, et al. Embodied-r1. 5: Evolving physical intelligence via embodied foundation models. *arXiv preprint arXiv:2606.11324*, 2026c.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Yanpeng Zhou, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, et al. 4d-vla: Spatiotemporal vision-language-action pretraining with cross-scene calibration. *Advances in Neural Information Processing Systems*, 38: 33914–33937, 2026a.

Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3d with language instruction. *arXiv preprint arXiv:2606.18558*, 2026b. URL https://arxiv.org/abs/2606.18558.

Tianle Zhang, Zhihao Yuan, Dafeng Chi, Peidong Liu, Dongwei Li, Kejun Hu, Likui Zhang, Junnan Nie, Ziming Wei, Zengjue Chen, Yili Tang, Jiayi Li, Zhiyuan Xiang, Mingyang Li, Tianci Luo, Hanwen Wan, Ao Li, Linbo Zhai, Zhihao Zhan, Xiaodong Bai, Jiakun Cai, Peng Cao, Kangliang Chen, Siang Chen, Yixiang Dai, Shuai Di, Yicheng Gong, Chenguang Gui, Yucheng Guo, Peng Hao, Qingrong He, Haoyang Huang, Kunrui Huang, Zhixuan Huang, Shibo Jin, Yixiang Jin, Anson Li, Dongjiang Li, Jiawei Li, Ruodai Li, Yihang Li, Yuzhen Li, Jiaming Liang, Fangsheng Liu, Jing Long, Mingxi Luo, Xing Pan, Hui Shen, Xiaomeng Tian, Daming Wang, Song Wang, Junwu Xiong, Hang Xu, Wanting Xu, Zhengcheng Yu, He Zhang, Jiyao Zhang, Lin Zhao, Chen Zhou, Nan Duan, Yuzheng Zhuang, and Liang Lin. JoyAI-RA 0.1: A Foundation Model for Robotic Autonomy. *arXiv preprint arXiv:2604.20100*, 2026c.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A Vision-Language-Action Model Dreamed with Comprehensive World Knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

A Implementation Details

This section specifies how the geometric targets are constructed and summarizes the training settings used in our experiments.

A.1 Geometric Target Preparation

The released LIBERO and RoboCasa-GR1 training data do not provide the depth maps and camera parameters needed for our geometry targets. We restore demonstration episodes from the corresponding open-source HDF5 simulator records, replay their recorded states, and rerender camera views while retaining the original actions. This produces camera parameters and metric depth aligned with each frame. We use these records to construct UVD motion targets and scene-depth targets at action-chunk endpoints. As in DIAL (Chen et al., 2026b), the future observation is aligned with a demonstrated action interval; the simulator replay additionally supplies the metric geometry used here.

Motion Targets. In LIBERO, we use the gripper center of the Franka Panda arm as the motion reference point. In RoboCasa-GR1, we use the midpoint between the thumb and index finger positions of each GR-1 hand. For a demonstrated reference point $(x_s^{\mathrm{w}}, y_s^{\mathrm{w}}, z_s^{\mathrm{w}})$ in simulation, we obtain its main-view UVD coordinates using the camera extrinsics and image-adjusted intrinsics:

$$\begin{bmatrix} u_s d_s \\ v_s d_s \\ d_s \end{bmatrix} = \mathbf{K}_s \begin{bmatrix} \mathbf{R}_s & \mathbf{t}_s \end{bmatrix} \begin{bmatrix} x_s^{\mathrm{w}} \\ y_s^{\mathrm{w}} \\ z_s^{\mathrm{w}} \\ 1 \end{bmatrix}. \tag{11}$$

Here, $\mathbf{R}_s$ and $\mathbf{t}_s$ map world coordinates to the camera frame, and $\mathbf{K}_s$ accounts for image preprocessing. The third coordinate gives camera-frame depth d_s. We normalize UV for training and retain depth in meters. For real-world demonstrations, we use the center of each ARX-Lift2S gripper and the Realman RM75 gripper as motion reference points. We project them into the main RGB view on ARX-Lift2S and the fixed RGB view on Realman RM75 using the corresponding camera calibration.

Scene Targets. We use scene depth at the endpoint of each action chunk as the supervision target. LIBERO supervises the agent and wrist views, and its motion queries can read scene latents from both. RoboCasa-GR1 supervises the main view. For real-world demonstrations, endpoint depth comes from the primary camera on ARX-Lift2S and from the exterior and wrist cameras on Realman RM75.

A.2 Training Configuration

GF-VLA uses a Qwen3.5-4B (Qwen Team, 2026) VLM backbone and a flow-based Action Expert in all settings; the simulation configuration is summarized in tab. 7.

Table 7: Simulation training settings for GF-VLA.

Setting	LIBERO	RoboCasa-GR1
Training steps	60,000	100,000
Global batch size	64	128
Action-chunk length	8	16
Action dimension	7	29
Trace points	4	6 per hand
Scene-query views	Agent, wrist	Main
Scene queries per view	8	8
VLM learning rate	10^{-5}	10^{-5}
Action Expert learning rate	10^{-4}	10^{-4}
Warmup steps	5,000	5,000
Minimum learning rate	5×10^{-7}	5×10^{-7}
Future depth weight per view	0.15	0.15

The Action Expert is a 16-block DiT-B with hidden width 1024. We use AdamW with cosine decay after warmup. Geometry losses use Smooth-L1 with $\beta = 1$; the absolute and relative UVD terms use weights 0.62 and 0.062, and future-depth forcing uses 0.15 per view in both settings. LIBERO applies it to agent and wrist views, whereas RoboCasa-GR1 applies it only to the main view.

B Evaluation Protocols

B.1 Simulation Evaluation

LIBERO uses 40 tasks with 50 rollouts per task, giving 2,000 trials. Its four suites are Spatial, Object, Goal, and Long, and their SRs are averaged equally. LIBERO-Plus evaluates 10,030 instances under seven perturbation categories: Camera, Robot, Language, Light, Background, Noise, and Layout. The Language category changes the instruction while the other categories are represented by visual or embodiment changes; it is therefore described in the protocol but is not shown as a separate image panel in fig. 6. RoboCasa-GR1 uses 24 tasks with 50 rollouts per task, giving 1,200 trials; fig. 6 shows representative examples from its sampled tabletop tasks. At evaluation, we use four Euler integration steps to generate each action chunk and execute the full chunk before the next policy decision. All simulation evaluations use seed 7.



Figure 6: Representative benchmark scenes from LIBERO, LIBERO-Plus, and RoboCasa-GR1.

B.2 Real-World Evaluation Protocol

Demonstrations and Evaluation Conditions. We train each method on the same demonstrations for a given task. Object Packing has 50 demonstrations, Bricks Disassembly has 150 across three assembly configurations, and Hanger Placement has 100 across three hook positions. Easy evaluation resembles data collection. Hard evaluation changes the tablecloth for Object Packing, adds distractor bricks for Bricks Disassembly, and moves the hook to unseen positions for Hanger Placement. Fig. 7 shows the platforms, cameras, and task objects. Fig. 8 illustrates the evaluation changes, and tab. 8 summarizes the data and trial counts.

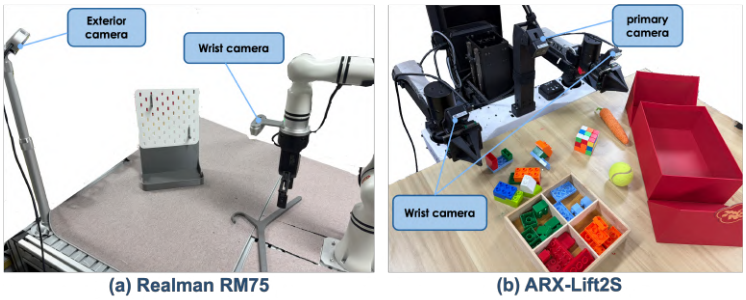

Figure 7: Real-world experimental setups. (a) Realman RM75 for Hanger Placement, showing the exterior and wrist cameras, hanger, and hooks. (b) ARX-Lift2S for Object Packing and Bricks Disassembly, showing the primary and wrist cameras and the task objects.

Policy and Execution. ARX-Lift2S uses three RGB views and 14-dimensional joint-and-gripper actions; Realman RM75 uses fixed and wrist RGB views and seven-dimensional end-effector-and-gripper actions. QwenGR00T and GF-VLA share the Qwen3.5-4B backbone and flow-based Action Expert and are fine-tuned on the same demonstrations. Their task-specific action-chunk lengths and fine-tuning steps are given in tab. 8. For all three tasks, each predicted action chunk is executed in full before the next policy decision. The motion trace spans each action chunk, and the future scene target corresponds to its endpoint.

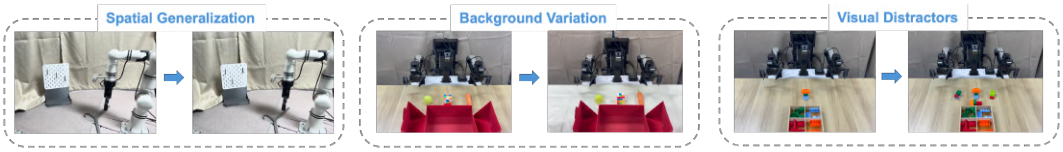

Figure 8: Easy and Hard real-world evaluation conditions. Each pair shows collection-like conditions on the left and the corresponding Hard condition on the right: unseen hook positions for Hanger Placement, background variation for Object Packing, and visual distractors for Bricks Disassembly.

Table 8: Real-world demonstration and evaluation settings.

Setting	Object Packing	Bricks Disassembly	Hanger Placement
Robot platform	ARX-Lift2S	ARX-Lift2S	Realman RM75
Demonstrations	50	150	100
Fine-tuning steps	40,000	60,000	40,000
Action-chunk length	30	40	20
Easy trials per method	30	30	30
Hard trials per method	30	30	30

B.3 Trace–Action Consistency

At each policy decision, we match a decoded motion waypoint to the robot reference point reached at the same sampled time during action execution. Let $(\hat{u}_i, \hat{v}_i, \hat{d}_i)$ and $(u_i^{\text{exec}}, v_i^{\text{exec}}, d_i^{\text{exec}})$ be their predicted and executed UVD coordinates. UV is normalized to $[0, 1]$, and depth is measured in meters. Let W and H denote the image width and height. The pointwise errors are

$$e_i^{\text{UV}} = \frac{(W-1)|\hat{u}_i - u_i^{\text{exec}}| + (H-1)|\hat{v}_i - v_i^{\text{exec}}|}{2}, \qquad e_i^{\text{depth}} = 1000|\hat{d}_i - d_i^{\text{exec}}|. \tag{12}$$

UV error is therefore the mean absolute difference across the two pixel coordinates, while depth error is in millimeters. We average valid matched points across decisions and hands within each episode, then average episodes within each task and tasks equally. Points not executed before episode termination are excluded. These metrics assess agreement between predicted and executed motion, not scene-depth reconstruction.

C Additional Quantitative Results

C.1 Number of Motion Latents

To study the trade-off among UVD trace sparsity, task performance, and inference cost, we vary the number of trace points per hand while fixing the action-chunk length at $H = 16$ and keeping the other training settings unchanged. Each trace point corresponds to one motion latent, so denser

sampling increases the conditioning sequence and its computational cost. We compare 3, 6, and 13 points per hand, corresponding to 6, 12, and 26 motion latents in the bimanual setting.

As shown in Table 9, the SR increases from 69.9% to 71.9% when the number of points grows from 3 to 6, but decreases to 69.6% with 13 points. Six points per hand therefore provide the best balance in this comparison: the sparse setting provides fewer temporal anchors for the motion within a chunk, while the denser setting adds motion latents without improving task completion. We use six points per hand in the main model.

Table 9: Number of motion latents on RoboCasa-GR1.

Points / hand	Motion latents	SR (%) ↑
3	6	69.9
6	12	**71.9**
13	26	69.6

C.2 Additional Scene-Supervision Results

To examine whether scene geometry is better learned through metric reconstruction or latent feature alignment, we retain UVD motion forcing and replace future-depth reconstruction with alignment to a frozen DA3 teacher. The teacher processes the RGB image at the action-chunk endpoint and provides features from layer 23 with dimension 2048. Its 16×16 patch features are pooled into a 2×4 grid, producing eight target features for the eight scene latents. The student scene latents are projected to the same dimension and trained with a cosine-distance loss weighted by 0.15. The numerical depth decoder is disabled in this variant, and the endpoint RGB image is used only by the teacher during training.

As shown in Table 10, feature alignment improves SR from 66.8% for UVD only to 69.7%, while future metric-depth reconstruction reaches 71.9%. Thus, both scene-supervision targets improve task completion, but metric depth provides an additional 2.2 percentage-point gain over feature alignment and a 5.1-point gain over UVD only. The two variants have the same rounded UV error of 3.36 pixels, while metric-depth reconstruction obtains a lower depth error of 8.09 mm than feature alignment at 8.18 mm. Feature alignment therefore provides useful scene information to the latent representation, whereas metric depth more directly constrains the camera-frame spatial quantity required by UVD motion learning.

Table 10: Future-supervision results. UV error and depth error measure prediction–execution consistency.

Future geometry learning	SR (%) ↑	UV error (px) ↓	Depth error (mm) ↓
None	66.8	3.46	8.67
DA3 features	69.7	3.36	8.18
Future metric depth	**71.9**	3.36	**8.09**

C.3 Within-Chunk Prediction–Execution Consistency

We further inspect how the full GF-VLA maintains prediction–execution consistency within an action chunk. For each future waypoint, we match the predicted UVD point to the robot reference point reached at the corresponding execution offset and aggregate the errors over the RoboCasa-GR1 evaluation. We omit the pre-execution point and report offsets 3, 6, 10, 13, and 16. The two curves in fig. 9 report the resulting UV and depth errors in waypoint order.

Across the reported future offsets, UV error grows from 2.58 pixels at offset 3 to 4.54 pixels at offset 16. Depth error follows the same overall trend, increasing from 6.72 mm to 10.05 mm. The errors therefore grow roughly with the temporal distance of the predicted waypoint, consistent with control deviation accumulating over the open-loop portion of an action chunk. Near-term waypoints remain more closely aligned with execution, whereas later waypoints are more exposed to accumulated

deviation. This diagnostic complements the aggregate consistency reported in the main text by showing how agreement evolves over the local motion interval.

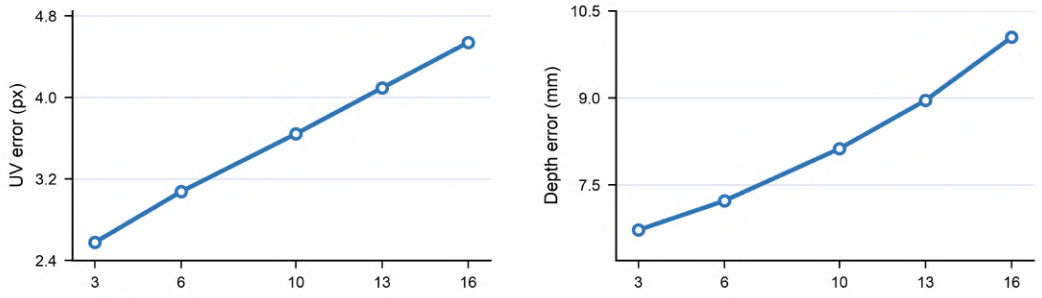

Figure 9: Within-chunk prediction–execution consistency of the full GF-VLA on RoboCasa-GR1. The two panels report UV error and depth error at matched waypoint offsets; lower values indicate closer agreement.

D Case Study

For each comparison on RoboCasa-GR1, we examine two tasks with larger gains and two with smaller gains. In tasks marked "Novel," "From" and "To" denote the object's source and destination. The manipulated object is sampled from novel object categories, so specific object-placement instances can vary across episodes.

Table 11: Selected RoboCasa-GR1 task-level success rates (%). The table compares 2D and 3D motion forcing.

Task	2D	3D
Largest gains		
Milk → Microwave	42	**70**
Bottle → Cabinet	52	**70**
Smallest gains		
Novel Cuttingboard → Tieredbasket	**72**	56
Novel Placemat → Bowl	**86**	78

The 2D-versus-3D cases in Table 11 show that the benefit of adding metric depth is task dependent. The retained largest-gain cases are Milk → Microwave and Bottle → Cabinet, where 3D outperforms 2D by 28 and 18 percentage points. The retained smallest-gain cases move in the opposite direction, so the aggregate gain should not be read as uniform across all tasks.



Figure 10: Qualitative comparison of 2D and 3D motion forcing for the task "Pick up the bottle, place it in the cabinet, and close the cabinet."

Fig. 10 illustrates how 3D motion forcing affects placement in an enclosed space. The 2D policy, shown in red, places the bottle too shallowly and prevents the cabinet door from closing. The 3D policy, shown in green, places it farther inside and closes the door. This contrast highlights the importance of spatial placement for subsequent manipulation.

Table 12: Selected RoboCasa-GR1 task-level success rates (%) for scene forcing with 3D motion fixed. Current and future columns indicate the scene geometry forcing used by each variant.

Task	Current depth	Future depth	SR (%)
Largest gains			
Novel Tray → Plate	×	×	78
	✓	×	74
	×	✓	**94**
	✓	✓	82
Cup → Drawer	×	×	34
	✓	×	32
	×	✓	**50**
	✓	✓	46
Smallest gains			
Novel Tray → Tieredbasket	×	×	56
	✓	×	**62**
	×	✓	50
	✓	✓	50
Wine → Cabinet	×	×	50
	✓	×	44
	×	✓	50
	✓	✓	**58**

Table 12 isolates the effect of scene-geometry forcing while keeping the 3D motion representation fixed. Among the retained largest-gain cases, future forcing reaches 94% on Novel Tray → Plate and 50% on Cup → Drawer. The retained smallest-gain cases show why the scene target remains task dependent: for Novel Tray → Tieredbasket, current-only forcing reaches 62%, whereas future-only and current-plus-future reach 50%; for Wine → Cabinet, the combined setting reaches 58% while future-only reaches 50%. These differences are consistent with the varying placement geometry across Novel episodes and show why the choice of scene target matters for learning motion geometry.

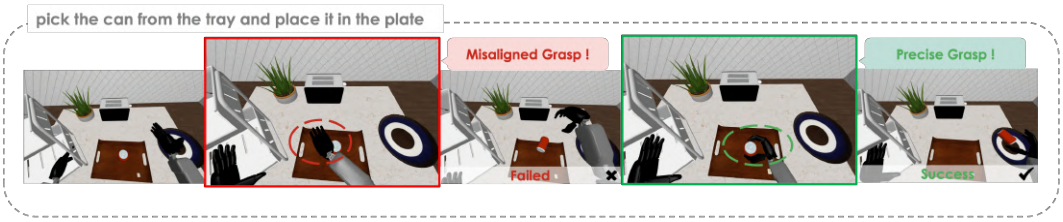

Figure 11: Qualitative comparison of current and future depth forcing for the task "Pick the can from the tray and place it in the plate."

With UVD motion forcing fixed, fig. 11 compares current and future depth forcing. The current-depth variant, shown in red, approaches the can with a misaligned grasp and fails to pick it up from the tray. The future-depth variant, shown in green, aligns the hand with the can, lifts it, and places it in the plate. This case illustrates how future depth forcing can assist motion geometry learning for accurate hand–object interaction.

Table 13: Selected RoboCasa-GR1 task-level success rates (%) for dual geometry forcing. Motion and scene columns indicate the forcing used by each variant.

Task	Motion forcing	Scene forcing	SR (%)
Largest gains			
Cup → Drawer	×	×	34
	✓	×	34
	×	✓	32
	✓	✓	**50**
Milk → Microwave	×	×	58
	✓	×	60
	×	✓	56
	✓	✓	**70**
Smallest gains			
Novel Tray → Tieredbasket	×	×	56
	✓	×	56
	×	✓	**70**
	✓	✓	50
Novel Placemat → Bowl	×	×	**92**
	✓	×	72
	×	✓	86
	✓	✓	78

Table 13 examines the two geometry-forcing objectives together. The retained largest-gain cases are Cup → Drawer and Milk → Microwave, where the joint setting reaches 50% and 70%, respectively. The retained smallest-gain cases again show task dependence: for Novel Tray → Tieredbasket, scene-only forcing reaches 70% but the joint setting reaches 50%; for Novel Placemat → Bowl, the task is already strong without forcing at 92%, and the added constraints do not improve it. These examples show that dual geometry forcing is most useful when a task requires both motion and scene information, while its gain is not uniform across the placement instances encountered in Novel episodes.

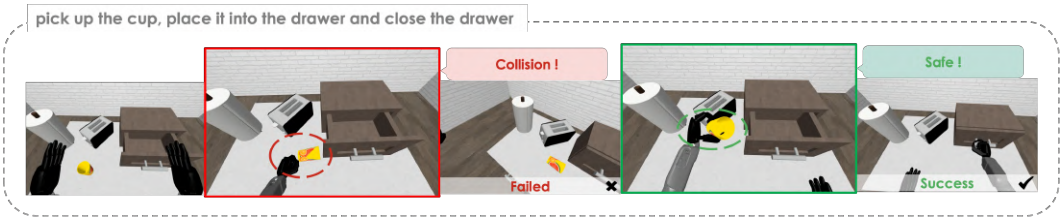

Figure 12: Qualitative comparison with and without UVD motion forcing for the task "Pick up the cup, place it into the drawer and close the drawer."

With future depth forcing retained, fig. 12 compares the policy without UVD motion forcing with the full GF-VLA. In the red-framed rollout without motion forcing, the hand collides with the cup during approach, and the task fails. GF-VLA, shown in green, successfully grasps the cup, places it in the drawer, and closes the drawer. This contrast highlights the additional guidance provided by motion geometry for object interaction when scene geometry is already learned.