3D HAMSTER: Bridging Planning and Control in Hierarchical Vision Language Action Models through 3D Trajectory Guidance

Dongyoon Hwang[1*], Byungkun Lee[1*], Dongjin Kim[1*], Hyojin Jang[1], Hoiyeong Jin[1], Jueun Mun[2], Minho Park[1], Hojoon Lee[1], Hyunseung Kim[1,3], and Jaegul Choo[1†]

***Abstract*— Hierarchical Vision-Language-Action (VLA) models decouple high-level planning from low-level control to improve generalization in robot manipulation. Recent work in this paradigm uses 2D end-effector trajectories predicted by a Vision-Language Model (VLM) as explicit guidance for a downstream policy. However, state-of-the-art low-level policies operate in 3D metric space on point clouds, and feeding them 2D guidance that lacks depth forces each waypoint to be assigned the depth of whatever scene surface lies beneath it, producing geometrically distorted trajectories. We propose *3D HAMSTER*, a hierarchical framework that closes this gap by having the planner directly output metrically reliable 3D trajectories. We augment a VLM with a dedicated depth encoder and a dense depth reconstruction objective to predict 3D waypoint sequences, which are directly integrated into a pointcloud-based low-level policy. Across 3D trajectory prediction, simulation, and real-world manipulation, *3D HAMSTER* consistently outperforms proprietary VLMs and 2D-guided baselines, with the largest gains under appearance-altering shifts and unseen language, spatial, and visual conditions. The project page is available at `https://davian-robotics.github.io/3D_HAMSTER/`.**

I. INTRODUCTION

A long-standing challenge in robot manipulation is bridging the gap between high-level semantic reasoning and low-level motor control. The strong semantic understanding capabilities of Vision-Language Models (VLMs) [1]–[3] have inspired the development of end-to-end Vision-Language-Action (VLA) architectures that directly map visual observations and language instructions to continuous actions [4]–[7]. However, end-to-end VLA architectures, also referred to as *monolithic* models [5]–[7], have shown limited performance due to the scarcity of robot demonstration data, which remains costly to collect at scale on physical hardware. Fine-tuning on this limited data erodes the broad generalizability of the underlying VLM, leaving these models vulnerable to out-of-distribution (OOD) visual shifts across novel objects, viewpoints, and environments [8].

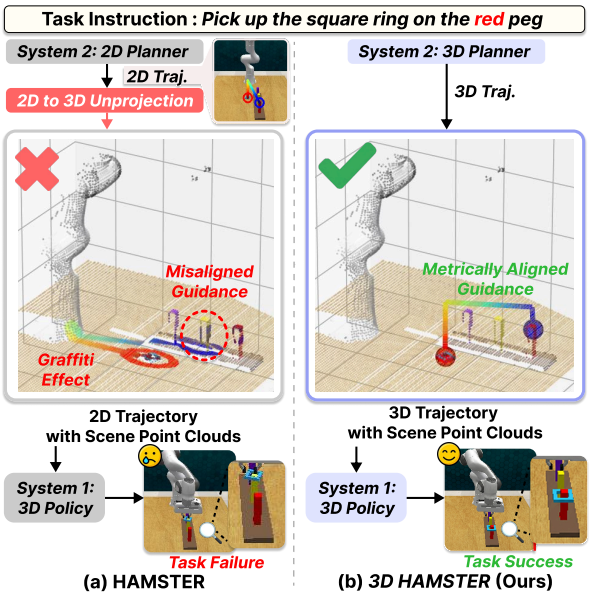

Fig. 1. Comparison of 2D and 3D guidance in hierarchical VLAs. (a) 2D planners create representational misalignment: unprojecting 2D plans yields flawed 3D guidance, leading to brittle execution. (b) Our 3D-aware planner generates metrically reliable 3D guidance, establishing a shared metric space for robust manipulation.

To better leverage the generalization capability of VLMs, hierarchical VLA frameworks [9]–[11] have been proposed as a compelling alternative that explicitly decouples semantic reasoning from low-level motor control. Specifically, they utilize a VLM as a high-level planner that produces 2D keypoints on a camera image (System 2), while a low-level controller receives this guidance to generate motor commands (System 1). This decoupling provides a critical scaling advantage. Because the planner predicts visual targets rather than robot-specific actions, it can be trained on abundant non-robot data encompassing spatial reasoning, visual grounding, 2D bounding boxes, and general VQA, preserving the broad generalizability of the underlying VLM.

While this 2D formulation is a natural fit for VLM-based planners, recent research on low-level policies has increasingly favored 3D-native architectures that operate on point clouds, consistently outperforming 2D alternatives in spatial precision and robustness to viewpoint changes [12]–[15]. This creates a fundamental representational misalignment in current hierarchical VLAs: the planner would reason in 2D pixel coordinates, while the controller operates in 3D space.

For instance, HAMSTER [9], pairs a 2D VLM planner with a pointcloud-based 3D controller (Fig. 1a). To bridge

*Equal contribution, †Corresponding author.

This research was supported by a grant from KRAFTON AI and the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT).

[1]Dongyoon Hwang, Byungkun Lee, Dongjin Kim, Hyojin Jang, Hoiyeong Jin, Minho Park, Hojoon Lee, Hyunseung Kim, and Jaegul Choo are with the Kim Jaechul Graduate School of AI, KAIST, Seoul, Republic of Korea (e-mail: godnpeter@kaist.ac.kr).

[2]Jueun Mun is with the Graduate School of Artificial Intelligence, POSTECH, Pohang, Republic of Korea.

[3]Hyunseung Kim is with KRAFTON AI, Seoul, Republic of Korea.

Links: GitHub Code | HF Models | Project Page

the gap between 2D plans and 3D execution, the planner's 2D waypoints must be lifted into 3D by sampling depth from the scene surface at each pixel location. However, since the planner itself provides no depth information, the result is determined entirely by whatever geometry happens to lie beneath each waypoint. This produces a *graffiti effect* (Fig. 1a): the trajectory clings to the scene surface rather than passing freely through 3D space, making it difficult for the controller to distinguish the intended path from the geometry itself. If, instead, the planner directly outputs 3D coordinates, no such projection is needed: the trajectory exists in the same metric space as the controller from the start, faithfully reflecting the planner's intent (Fig. 1b).

Fortunately, recent VLMs have acquired substantial 3D spatial knowledge through large-scale geometry-rich pretraining, making 3D-aware planning increasingly feasible [2], [16]. Building on this foundation, we propose *3D HAMSTER*, a hierarchical VLA framework that trains a VLM backbone to predict metrically reliable 3D end-effector trajectories in (u, v, d) form, where (u, v) are image-plane coordinates and d is metric depth, producing guidance directly aligned with the operating space of 3D low-level policies. Since RGB features alone cannot reliably recover metric depth from a single view, we augment the VLM with a dedicated depth encoder and introduce a dense depth reconstruction objective that regularizes the model's internal representations to preserve faithful scene geometry, complementing sparse trajectory supervision with a scene-level 3D prior. To maintain broad generalization, we train the planner on a mixture of 3D capability data and 2D preservation data. The predicted trajectories are then transformed into world coordinates and executed by a pointcloud-based low-level policy, ensuring a unified metric interface from planning through execution.

We evaluate *3D HAMSTER* across three settings. (1) On *DroidSpatial-Bench*, a benchmark we construct from held-out DROID pick-and-place episodes [17], our depth-encoder-augmented VLM produces more accurate 3D trajectories than strong baselines including Gemini-3.0-Pro and RoboBrain 2.5. (2) On 11 tasks from the Colosseum simulation benchmark [18], which stress-tests policies under 14 perturbation axes, 3D trajectory guidance consistently outperforms 2D alternatives, with the largest gains under appearance-altering shifts such as lighting and texture changes. (3) On a real Franka Panda arm across three tasks of increasing spatial precision (button pressing, pouring, and pick-and-place), 3D guidance improves performance across four generalization axes: unseen language, spatial references, visual conditions, and their combination. Our contributions are as follows:

- A framework that closes the 2D-3D representational gap in hierarchical VLAs by having the planner output 3D trajectories directly consumable by pointcloud-based policies.
- A training recipe combining a depth encoder, dense reconstruction loss, and a curated data mixture that enables a pretrained VLM to generate metric 3D trajectories.
- Experimental validation across three complementary settings showing that aligning planning and execution in 3D metric space yields robust manipulation under distribution shifts where 2D guidance fails.

II. RELATED WORK

A. Hierarchical Vision-Language-Action (VLA) Models

Existing hierarchical VLA frameworks decouple high-level planning from low-level control through VLM-friendly intermediate representations such as language [19], 2D keypoints [20], and 2D end-effector trajectories [9]–[11]. However, since fine-grained manipulation fundamentally operates in 3D space, these 2D representations introduce ambiguity under 3D-sensitive shifts such as occlusion or viewpoint changes. This limitation has motivated a shift toward 3D-aware planning. Concurrent to our work, RoboTracer [21] and RoboBrain 2.5 [22] take a step in this direction by producing depth-aware keypoint sequences in (u, v, d) form. RoboTracer further incorporates depth input and predicts metric scale via a regression-supervised decoder, but remains a standalone motion planner that does not integrate its predicted trajectories with a downstream controller for closed-loop manipulation. In contrast, *3D HAMSTER* enforces scene-level geometric understanding through a dense depth reconstruction loss and investigates how to effectively couple the predicted 3D trajectories with a pointcloud-based low-level policy, closing the full loop from 3D-aware planning to 3D-native execution.

B. Vision-Language Models (VLMs) for Spatial Reasoning

VLMs are increasingly used for spatial prediction in embodied robotics, with outputs evolving from point estimates [20], [23]–[26] to trajectory-level guidance [21], [22], [27]. Beyond 2D spatial reasoning, recent VLMs have also acquired rough 3D spatial knowledge through large-scale pretraining on geometry-rich data, as evidenced by capabilities such as 3D bounding box prediction in Qwen3-VL [2] and joint 3D scene reconstruction and spatial question answering in G^{2}VLM [16]. Other works go further by injecting explicit geometric priors into VLMs through 3D grounding data [28], 3D geometry encoders [29], or 3D vision-language-action instruction tuning [30]. These results collectively demonstrate that modern VLMs possess a meaningful foundation of 3D spatial understanding. However, prior robotic planning methods have not fully leveraged this pretrained 3D knowledge, largely confining VLM planners to 2D outputs. Our work shows that this latent 3D capability can be effectively channeled into metrically reliable trajectory prediction for robotic manipulation through targeted fine-tuning and architectural augmentation.

C. 3D Low-Level Manipulation Policies

Earlier manipulation policies such as Diffusion Policy [31] operate on 2D image observations and struggle with depth ambiguities and perspective distortions in contact-rich tasks [32]. The field has shifted toward 3D-native architectures that operate directly on metric representations such as point clouds. DP3 [12] adopts pointcloud-based state representations within diffusion-based frameworks. Act3D [13] and RVT-2 [33] leverage 3D feature fields and multi-view

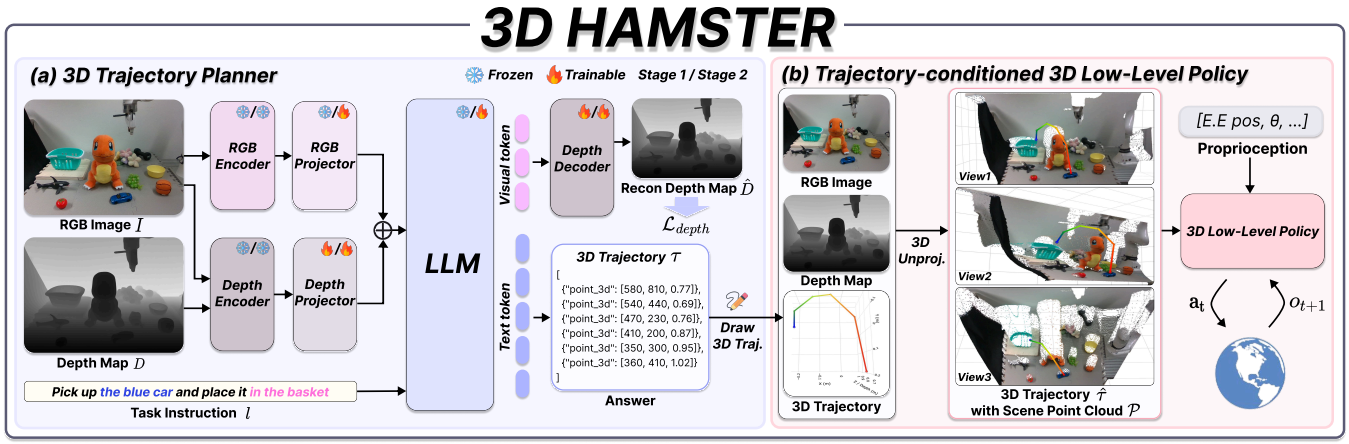

Fig. 2. Overview of *3D HAMSTER*. The framework decouples semantic planning and motor execution with two-stage training strategy: Stage 1 aligns depth features with the VLM space using a dense reconstruction loss ($\mathcal{L}_{depth}$) while preserving VLM capabilities; Stage 2 fine-tunes for trajectory prediction. The 3D trajectory planner fuses RGB and depth to generate metrically reliable 3D trajectories, which the trajectory-conditioned 3D low-level policy executes as robust closed-loop actions ($\mathbf{a_t}$) from the scene point cloud.

rendering for precise spatial prediction. 3D Diffuser Actor [14] unifies diffusion policies with 3D scene representations through a relative position denoising transformer. 3D FlowMatch Actor (3DFA) [15] extends 3DDA by replacing diffusion with rectified flow matching for faster and more efficient end-effector trajectory prediction. The consistent gains of these 3D controllers motivate our choice of a pointcloud-based low-level policy, and more broadly, the need for a high-level planner that produces guidance in the same 3D space.

III. METHOD

A. 3D HAMSTER

a) Problem Formulation: Given calibrated RGB-D cameras and a natural language instruction, our goal is to produce motor actions that accomplish the described manipulation task. At each timestep, the robot observes an RGB image $I \in \mathbb{R}^{H\times W\times 3}$, a depth map $D \in \mathbb{R}^{H\times W}$, and a task instruction l. We decompose this into a high-level planner that predicts an end-effector trajectory $\tau = \{(u_t, v_t, d_t)\}_{t=1}^{T}$ in pixel coordinates (u_t, v_t) with metric depth d_t, where T is the number of waypoints and t indexes each waypoint along the trajectory, and a low-level policy $\pi_{\text{low}}(\mathcal{P}, \tau) = \{\mathbf{a_t}\}_{t=1}^{T_a}$ that produces a chunk of T_a actions from the current point cloud observation $\mathcal{P}$ conditioned on τ, executing closed-loop control at each re-planning step.

b) Overall Framework: As illustrated in Fig. 2, *3D HAMSTER* consists of a 3D trajectory planner and a trajectory-conditioned 3D low-level policy. The 3D trajectory planner (Fig. 2a) takes a single RGB image I, a depth map D, and a task instruction l as input, and autoregressively generates a metrically reliable end-effector trajectory τ. To produce geometrically faithful predictions, we augment the VLM with a dedicated depth encoder that produces depth features alongside the RGB features from the vision encoder. These two feature sets are then fused into unified visual tokens and passed to the LLM backbone, which is further regularized by a dense depth reconstruction loss to preserve metrically accurate scene geometry. The trajectory-conditioned 3D low-level policy (Fig. 2b) then unprojects τ into world coordinates using known camera intrinsics and extrinsics, appends the resulting 3D waypoints to the scene point cloud constructed from RGB-D observations, and predicts a chunk of T_a actions $\{\mathbf{a_t}\}_{t=1}^{T_a}$ through rectified flow matching [34] over this unified 3D representation.

B. 3D Trajectory Planner

We build on Qwen3-VL [2], which possesses broad 3D spatial knowledge from large-scale pretraining, including the ability to predict 3D bounding boxes from a single image. However, this knowledge was acquired through object-level localization tasks and does not directly transfer to generating coherent sequences of 3D waypoints for end-effector trajectory prediction. Without targeted adaptation, the base model achieves near-zero accuracy on this task (Table II, row 1 of *ours*). We bridge this gap through three complementary components: (i) a curated training mixture, (ii) a dedicated depth encoder, and (iii) a dense depth reconstruction loss, whose individual contributions we isolate in Sec. IV-B.

a) Training Data: We train the planner on a mixture of eight data sources grouped into two categories (Table I), designed to channel the base VLM's 3D knowledge into metric trajectory prediction without eroding its existing vision-language understanding. *(i) 3D capability data* (RGB + depth) consists of end-effector trajectories from real and simulated robot demonstrations and spatial reasoning data from RefSpatial [26]. Each trajectory sample pairs an RGB image and depth map with a (u, v, d) waypoint sequence and gripper actions. We include three supervision variants: 2D-only, 3D-only, and a chain-of-thought variant that first predicts the 2D trajectory and then lifts it to 3D, encouraging the model to learn consistent pixel-to-depth mappings. RefSpatial, originally a 2D-only dataset, is augmented with generated depth maps and extended to include depth-aware spatial reasoning and vacant-space localization, providing additional supervision for the depth encoder. Ground-truth depth is used for RLBench [35] and DROID [17]; for

TABLE I
3D TRAJECTORY PLANNER TRAINING DATA COMPOSITION.

Source	Env.	Modality	Tasks	Size
3D capability data				
RLBench [35]	Sim	RGB-D	2D / 3D / 2D→3D	606K
DROID [17]	Real	RGB-D	2D / 3D / 2D→3D	123K
InternData-M1 [36]	Sim	RGB-D	2D / 3D / 2D→3D	1.5M
RefSpatial [26]	Mix	RGB-D	Spatial QA / Vacant loc.	2.2M
Preservation data				
RoboPoint [20]	Sim	RGB	2D pointing	666K
PixMo [38]	Real	RGB	2D pointing	171K
LVIS [39]	Real	RGB	2D Bbox det.	138K
Honey-1M [40]	Web	RGB	General VQA	749K

InternData-M1 [36] and RefSpatial, we generate metric depth using MoGe-2 [37]. *(ii) Preservation data* (RGB only) anchors the base model's original vision-language capabilities, preventing them from being overwritten during trajectory fine-tuning. This includes RoboPoint [20] for point-based referring, PixMo [38] for indoor pointing, LVIS [39] for bounding-box detection, and Honey-1M [40] for general VQA. These samples are processed entirely through the base VLM, bypassing the depth pathway.

b) Depth Encoder: While the training data provides supervision for metric depth prediction, RGB features alone cannot reliably recover metric depth from a single view. We therefore augment the VLM with a dedicated depth pathway (Fig. 2a). The RGB image I is encoded into visual tokens by the pretrained visual encoder. In parallel, the depth map D is processed by a separately initialized depth encoder to produce depth tokens that capture geometric cues complementary to the RGB features. Both token sequences are projected into the LLM embedding space through their own dedicated projectors, fused element-wise, and then passed to the transformer backbone alongside the tokenized instruction l. Given this fused visual, depth, and language context, the model autoregressively generates a trajectory $\tau = \{(u_t, v_t, d_t)\}_{t=1}^{T}$.

c) Depth Reconstruction Loss: Simply providing depth tokens does not guarantee that the LLM's hidden states retain metrically faithful depth information. Since the model outputs the trajectory as a sequence of text tokens, relying solely on the standard autoregressive language modeling loss $\mathcal{L}_{\text{LM}}$ provides only a sparse gradient signal, which may cause the depth encoder's geometric features to degrade during fine-tuning. To prevent this, we route the depth tokens z_D through a lightweight decoder f_{dec} to reconstruct the full depth map $\hat{D} = f_{\text{dec}}(z_D)$, supervised by an ℓ_1 loss:

$$\mathcal{L}_{\text{depth}} = \|D - \hat{D}\|_1. \tag{1}$$

The total training loss combines autoregressive trajectory prediction with this dense reconstruction signal:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda\, \mathcal{L}_{\text{depth}}, \tag{2}$$

where $\lambda = 0.1$. While trajectory supervision is sparse and task-specific, the reconstruction loss provides a scene-level geometric prior that encourages metrically consistent depth understanding across varying objects and viewpoints.

d) Two-stage Training: To integrate the depth pathway without disrupting the pretrained VLM, we train in two stages (Fig. 2a). In stage 1 (depth alignment stage), we freeze the RGB encoder, depth encoder, and LLM backbone, training only the depth projector and the depth decoder. This aligns the depth representation with the pretrained visual-language space while preserving existing capabilities. In stage 2 (task fine-tuning stage), we freeze both the RGB and depth encoders to retain their learned representations and fine-tune the remaining parameters for trajectory prediction.

C. Trajectory-conditioned 3D Low-level Policy

We adopt 3DFA [15], a pointcloud-based policy that predicts actions via rectified flow matching, as our low-level backbone. The planner's trajectory $\tau = \{(u_t, v_t, d_t)\}_{t=1}^{T}$ is first unprojected to a world coordinate trajectory $\hat{\tau} = \{(X_t, Y_t, Z_t)\}_{t=1}^{T}$ using known camera intrinsic matrix K and extrinsic parameters $R, \mathbf{t_{cam}}$:

$$\mathbf{p}_{\text{cam}} = d \cdot K^{-1}[u, v, 1]^{\top}, \quad \mathbf{p}_{\text{world}} = R\,\mathbf{p}_{\text{cam}} + \mathbf{t_{cam}}. \tag{3}$$

The central challenge is how to condition this policy on the resulting 3D trajectory so that guidance and scene geometry operate in a shared representation.

a) Trajectory-Scene Fusion: The world-frame trajectory $\hat{\tau}$ is appended to the scene point cloud $\mathcal{P}$ (Fig. 2b), so that trajectory guidance and visual observations coexist in a unified 3D representation. Each waypoint is color-coded by its temporal position along the trajectory, enabling the policy to distinguish the progression of the planned motion. To further allow the policy to learn distinct strategies for following the trajectory versus understanding the scene, we add learnable modality embeddings to each point feature:

$$\tilde{f}_j = f_j + \begin{cases} e_{\text{traj}} & \text{if } j \in \mathcal{I}_{\text{traj}}, \\ e_{\text{scene}} & \text{otherwise}, \end{cases} \tag{4}$$

where f_j is the feature of point j, $\mathcal{I}_{\text{traj}}$ is the set of trajectory point indices, and $e_{\text{traj}}, e_{\text{scene}}$ are learned embeddings. Finally, since the policy subsamples the full point cloud for computational efficiency, we enforce that all trajectory points are preserved during subsampling by replacing the lowest-priority scene points with any dropped trajectory points. This guarantees that the guidance signal is never lost regardless of scene complexity.

b) Action Prediction: Given the modality-tagged point cloud, the policy learns a velocity field v_θ under a rectified flow matching formulation. A noised action sequence $\mathbf{a}^i = (1-i)\,\mathbf{a}^0 + i\,\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ and $i \in [0, 1]$, is mapped back to the clean action sequence $\mathbf{a}^0$. The training objective is:

$$\mathcal{L}_{\text{policy}} = \|v_\theta(\mathbf{a}^i, \tilde{f}, i) - (\mathbf{a}^0 - \epsilon)\|_2^2 + \mathcal{L}_{\text{BCE}}(g_\theta, g^*), \tag{5}$$

where the first term supervises the velocity field and the second supervises the gripper open/close action. At inference, the policy denoises from Gaussian noise to a clean action chunk of T_a steps. Since trajectory guidance and scene share a common metric coordinate frame, the controller

Fig. 3. End-to-end manipulation evaluation setups. (a) Simulation environments from the Colosseum benchmark, showcasing various visual and physical perturbations. (b) Real-world setup using a Franka Panda arm equipped with an external RGB-D camera

executes the plan directly without the implicit 2D-to-3D lifting required by 2D guidance approaches [9].

IV. EXPERIMENTS

Our experiments address the following research questions:

- *RQ1:* How important is depth for 3D end-effector trajectory prediction, and what adaptations enable a VLM to generate metric waypoints?
- *RQ2:* Does 3D trajectory guidance improve manipulation robustness under controlled visual and physical perturbations in RLBench?
- *RQ3:* Do the benefits of 3D guidance transfer to real-world manipulation across diverse generalization axes?

A. Experimental Setup

a) Evaluation: We evaluate in three settings: 3D trajectory prediction, simulation, and real-world manipulation.

For trajectory prediction (*RQ1*), we propose *DroidSpatial-Bench*, constructed from 148 held-out DROID [17] pick-and-place episodes. Given an RGB-D image and a language instruction, the model predicts a 3D trajectory. We evaluate whether the predicted grasp and placement points each fall within $\delta \in \{5, 10\}$ cm of ground truth, and report start-position, end-position, and combined (*Both*) accuracies.

For simulated manipulation (*RQ2*), we use the Colosseum benchmark [18] (Fig. 3a), which extends RLBench [35] with 14 perturbation axes spanning object appearance, scene context, camera pose, and their aggregate. Following HAMSTER [9], we select 11 front-camera-visible tasks and report success rates over 25 episodes per perturbation.

For real-world manipulation (*RQ3*), we deploy a Franka Panda arm with an external RGB-D camera (Fig. 3b) on three task families: *button pressing* (3 buttons), *pouring* (2 cups, 3 bowls), and *pick-and-place* (10 objects, 3 bowls). Each is evaluated under four generalization axes: *language* (unseen synonyms), *spatial* (relative verbal references, and unseen object heights), *visual* (novel textures, lighting, and distractors), and *multiple* (all combined). Rather than binary success, we adopt a continuous score in $[0, 1]$: pick-and-place and pouring award 25% at each of four stages (reach, grasp, transport, place/pour), while button pressing awards 50% each for positioning and actuation. All results are averaged over a total of 25 evaluation rollouts.

b) Baselines: For trajectory prediction (*RQ1*), we compare against four proprietary VLMs (Sonnet-4.6, GPT-5.2, Gemini-3.0-Pro) and the open-source RoboBrain-2.5-8B [22]. RoboTracer [21] is excluded as it has not been publicly released. For end-to-end manipulation (*RQ2*, *RQ3*), all methods share the same 3DFA [15] low-level policy trained on identical demonstration data, isolating the effect of the guidance signal. We compare: (i) 3DFA without guidance, (ii) 3DFA with 2D trajectories from HAMSTER [9], and (iii) 3DFA with 3D trajectories from *3D HAMSTER*. For real-world experiments, we additionally include $\pi_{0.5}$ [6], a state-of-the-art monolithic VLA, fine-tuned on the same demonstration dataset as a baseline.

TABLE II

3D TRAJECTORY PREDICTION ACCURACY (%) ON *DroidSpatial-Bench*. **BOLD** = BEST PER COLUMN.

Model	Input	$\delta = 5$ cm			$\delta = 10$ cm		
		Start	End	Both	Start	End	Both
Proprietary API models							
Sonnet-4.6	RGB	1.4	8.8	0.7	6.1	16.2	2.0
GPT-5.2	RGB	6.8	29.7	2.7	29.7	45.3	16.2
Gemini-3.0-Pro	RGB	29.1	44.0	16.2	43.2	56.1	29.7
Open-source models							
RoboBrain-2.5-8B	RGB	61.5	58.1	39.2	80.4	74.3	60.1
***3D HAMSTER** (ours)*							
Qwen3-VL-8B	RGB	0.7	9.5	0.7	0.7	14.9	0.7
+ 3D Traj. Data	RGB	50.0	50.0	27.7	71.6	72.3	50.0
+ Depth encoder	RGBD	62.8	62.2	**42.6**	**83.8**	75.0	62.8
+ $\mathcal{L}_{\text{depth}}$	RGBD	**63.5**	**66.2**	41.9	80.4	**82.4**	**65.5**

c) Implementation Details: For the 3D trajectory planner, the VLM backbone is Qwen3-VL-8B-Instruct [2], with the depth encoder and decoder initialized from LingBot-Depth [41]. Pixel coordinates are normalized to $[0, 1000]$ and depth is predicted in meters. Stage 1 trains the depth projector and decoder only; Stage 2 keeps both encoders frozen and applies LoRA [42] (rank 64) to the LLM while fully training the projectors and decoder. Both stages run for 1 epoch (lr $= 1\times10^{-4}$, warmup 0.03, batch 256, $8\times$ H100). For the low-level policy, we use 3DFA [15] with RGB-D, language, and proprioceptive inputs to predict action chunks of length 20. In simulation, the policy trains on 100 demonstrations per task for 500k steps (batch 64). In the real world, we collect 300/144/108 episodes for pick-and-place/pouring/button pressing and train for 300k steps (batch 64, $4\times$ H100). All use lr $= 1\times10^{-4}$. For $\pi_{0.5}$ [6], we fine-tune for 50k steps (lr $= 1\times10^{-5}$, batch 64), limiting training duration to preserve the base model's pre-trained knowledge.

B. RQ1: 3D Trajectory Prediction Accuracy

a) Depth-augmented VLM outperforms proprietary and open-source baselines: Table II reports trajectory prediction

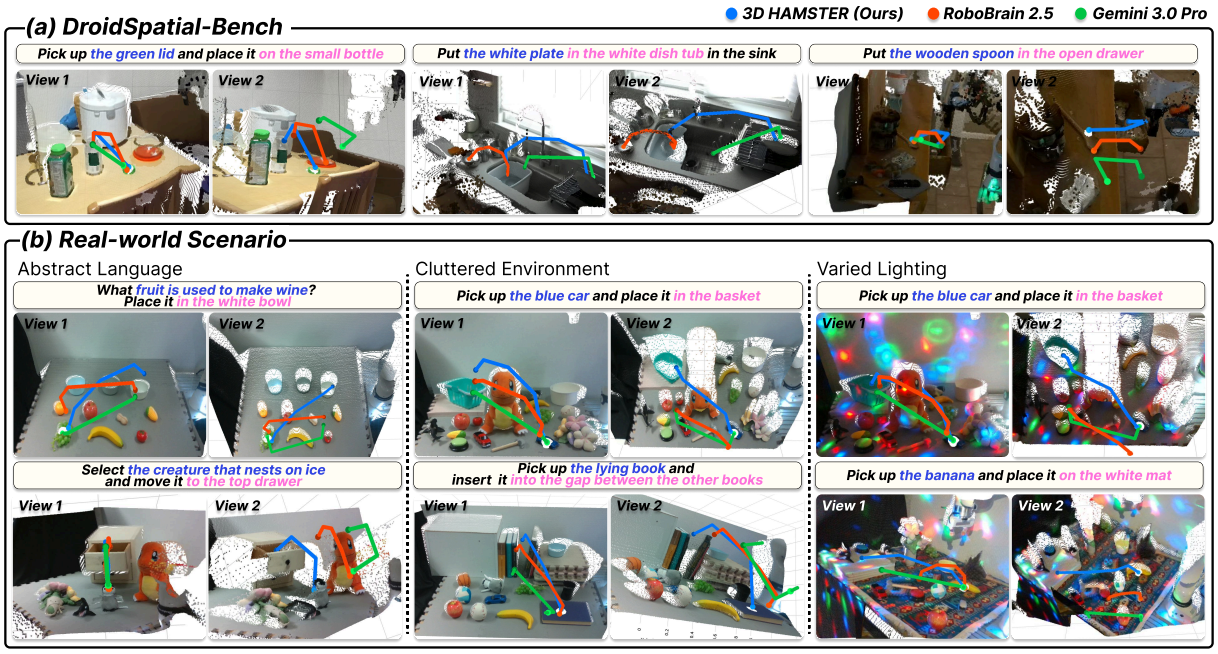

Fig. 4. Qualitative comparison of 3D trajectory predictions. Each trajectory is shown from two viewpoints: baseline predictions that appear plausible in View 1 reveal significant depth errors in View 2, whereas *3D HAMSTER* remains metrically consistent across both views.

TABLE III

PER-VARIATION SUCCESS RATES (%) AVERAGED ACROSS TASKS ON COLOSSEUM [18]. **BOLD** = BEST PER COLUMN.

Method	None	MO	RO	Light	Table	Distract	BG Tex	RLB Var	Cam Pose	All Var	Avg.
3DFA	53.8	39.0	27.2	38.0	30.3	36.4	43.3	50.0	46.8	0.8	36.6
3DFA + HAMSTER	49.5	38.1	28.8	38.8	**42.3**	40.8	43.3	50.5	48.4	**7.2**	38.8
3DFA + *3D HAMSTER*	**62.9**	**49.4**	**36.3**	**54.4**	39.6	**44.8**	**52.0**	**52.5**	**49.2**	**7.2**	**44.8**

accuracy on *DroidSpatial-Bench*. Proprietary VLMs perform poorly at 3D metric trajectory prediction despite their strong general vision-language capabilities: even the best, Gemini-3.0-Pro, achieves only 16.2% on the strict 5 cm *Both* metric, confirming that off-the-shelf VLMs lack metric depth reasoning. RoboBrain-2.5-8B provides a much stronger comparison point, reaching 39.2% (5 cm *Both*) and 60.1% (10 cm *Both*) with RGB input alone. However, *3D HAMSTER* surpasses it across all metrics, with the largest improvements on end-position accuracy (66.2% vs. 58.1% at δ=5 cm; 82.4% vs. 74.3% at δ=10 cm), where accurate depth prediction is most critical for downstream placement success.

b) Training data, depth encoding, and reconstruction each provides gain: The bottom block of Table II isolates each component by incrementally building upon Qwen3-VL-8B. (i) *3D trajectory dataset supervision* based on our multi-task mixture data provides a substantial boost (5 cm *Both*: 0.7%→27.7%), confirming that targeted training is required to channel the base VLM's general 3D knowledge into effective trajectory prediction. *(ii) Depth encoder* yields further substantial gains (5 cm *Both*: 27.7%→42.6%, 10 cm *Both*: 50.0%→62.8%), indicating that RGB alone struggles in reliably recovering metric depth. *(iii) Depth reconstruction loss* improves end-point accuracy (10 cm *End*: 75.0%→82.4%), where depth drift across waypoints is greatest. This suggests that sparse trajectory supervision alone cannot maintain depth fidelity over the full sequence; the dense reconstruction loss counteracts this drift by anchoring depth representations to a scene-level geometric prior.

c) Qualitative results for in- and out-of-distribution scenes: Fig. 4 visualizes predicted trajectories from *3D HAMSTER*, RoboBrain 2.5, and Gemini 3.0 Pro. On *DroidSpatial-Bench* (Fig. 4a), baseline predictions appear reasonable from View 1 but reveal substantial depth errors when viewed from a second angle, whereas *3D HAMSTER* remains metrically consistent across both viewpoints. This gap widens in Fig. 4b, where all models are evaluated zero-shot on real-world scenes unseen during training. Baseline trajectories lose spatial coherence entirely, while *3D HAMSTER* continues to produce geometrically plausible paths, demonstrating that the depth-augmented planner generalizes its metric 3D reasoning beyond the training distribution.

C. RQ2: Robustness Under Systematic Distribution Shifts

a) 2D guidance harms in-distribution performance; 3D guidance improves it: Table III shows that *3D HAMSTER* achieves the highest average success rate (44.8%), improving over 2D guidance (HAMSTER) by 6.0% and over the unguided baseline by 8.2%. Notably, 2D guidance actually *degrades* unperturbed performance relative to no guidance (49.5% vs. 53.8%), indicating that projecting plans into 2D pixel coordinates introduces geometric ambiguity that harms the 3D low-level controller even on in-distribution data. In

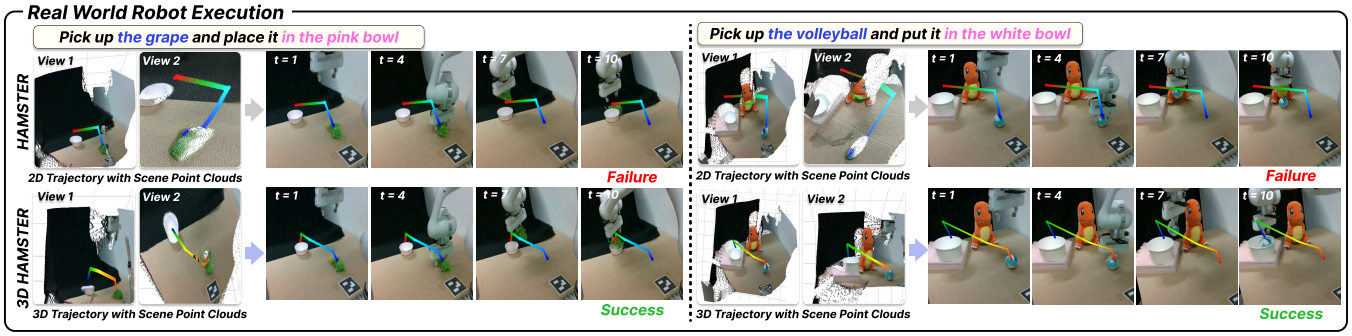

Fig. 5. Real-world execution rollouts. HAMSTER's 2D trajectories cling to the scene surface (top row, graffiti effect), causing failure. *3D HAMSTER*'s 3D trajectories remain metrically reliable (bottom row), enabling success.

contrast, 3D guidance improves the unperturbed condition to 62.9%, confirming that metrically reliable trajectories align naturally with the policy's 3D action space.

b) 3D guidance provides both appearance invariance and geometric robustness: 3D guidance yields the largest gains over 2D on lighting (+15.6%), manipulated object (MO, +11.3%), and background texture (BG Tex., +8.7%) perturbations. These axes directly corrupt the color and texture features that 2D planners depend on for implicit depth inference, whereas 3D trajectories specify the path in metric coordinates independently of visual appearance. Notably, MO and RO perturbations in Colosseum alter not only object color and texture but also object size, meaning the task geometry itself changes. That 3D guidance still outperforms 2D under these combined appearance and geometry shifts suggests its benefit extends beyond appearance invariance, providing metrically stable waypoints even under geometric variation. However, under the All Var. condition, where all perturbations compound simultaneously, both guided methods achieve only 7.2%. While this still substantially exceeds the unguided baseline (0.8%), the 2D and 3D planners converge, indicating that when severe visual corruption overwhelms the VLM's ability to ground the target object, guidance dimensionality becomes irrelevant.

D. RQ3: Real-World Transfer Across Generalization Axes

a) 3D guidance consistently improves real-world success, especially under out-of-distribution shifts: Table IV summarizes real-world results. *3D HAMSTER* achieves the highest average success across all three task families (80% button pressing, 68% pouring, 62% pick-and-place), outperforming both 2D guidance (HAMSTER: 60%, 45%, 46%) and the monolithic $\pi_{0.5}$ (74%, 41%, 40%). While $\pi_{0.5}$ matches or exceeds *3D HAMSTER* in-distribution (e.g., 100% on button pressing and pick-and-place), it degrades sharply under distribution shifts, dropping to 15% on pouring and 15% on pick-and-place under spatial variation. This confirms that monolithic VLAs overfit to training conditions despite strong in-distribution performance. The advantage of 3D over 2D guidance is most pronounced under visual shifts (button pressing: 100% vs. 80%; pouring: 65% vs. 35%) and spatial shifts (button pressing: 50% vs. 20%; pouring: 65% vs. 40%), where implicit depth inference from altered appearance breaks down. The largest overall gain appears

TABLE IV

REAL-WORLD SUCCESS RATES (%). **BOLD** = BEST PER CELL.

	In-D	Lang	Spat	Vis	Mult	Avg.
Button Pressing						
$\pi_{0.5}$	**100**	80	40	90	60	74
3DFA	90	30	0	30	40	38
3DFA + HAMSTER	80	50	20	80	**70**	60
3DFA + *3D HAMSTER*	**100**	**90**	**50**	**100**	60	**80**
Pouring						
$\pi_{0.5}$	60	15	35	**65**	30	41
3DFA	80	45	50	50	25	50
3DFA + HAMSTER	75	45	40	35	30	45
3DFA + *3D HAMSTER*	**95**	**75**	**65**	**65**	**40**	**68**
Pick-and-Place						
$\pi_{0.5}$	**100**	35	15	20	30	40
3DFA	65	45	5	25	10	30
3DFA + HAMSTER	70	75	35	15	35	46
3DFA + *3D HAMSTER*	90	**95**	**40**	**35**	**50**	**62**

in pouring (68% vs. 45%), which demands precise height control when tilting the cup, a geometric requirement that 3D waypoints directly encode. Under compounding shifts (multiple), 3D guidance degrades gracefully across all tasks, consistently outperforming or matching all baselines.

b) 3D guidance executes the planner's intent faithfully; 2D guidance distorts it: Fig. 5 compares real-world execution rollouts of HAMSTER and *3D HAMSTER* on two pick-and-place tasks. For HAMSTER, the 2D trajectory appears plausible from View 1 but View 2 reveals the *graffiti effect*: since the 2D planner provides no depth, each waypoint is assigned the depth of the underlying scene surface, pinning the trajectory to the point cloud rather than tracing a free path through 3D space. The low-level policy cannot distinguish this surface-bound guidance from the scene geometry itself, resulting in task failure in both examples. In contrast, *3D HAMSTER*'s 3D trajectories remain metrically consistent across both viewpoints, clearly separated from the scene surface and faithfully encoding the intended pick-and-place motion. The low-level policy tracks these waypoints accurately, achieving successful execution in both tasks. This comparison directly shows the core advantage of operating in a shared 3D metric space: the controller receives guidance that exists in its own coordinate frame, eliminating the geometric distortion inherent in 2D-to-3D lifting.

V. CONCLUSION

We presented *3D HAMSTER*, a hierarchical framework that aligns high-level planning and low-level control in metric 3D space. A pretrained VLM augmented with a depth encoder and a dense depth reconstruction loss generates metrically reliable (u, v, d) trajectories, which are directly integrated into a pointcloud-based low-level policy. Our experiments show that (1) each component of the depth-augmented planner provides complementary gains in 3D trajectory prediction accuracy, outperforming Gemini-3.0-Pro and RoboBrain-2.5 on *DroidSpatial-Bench*; (2) 3D guidance consistently outperforms 2D alternatives in simulation, with the largest improvements under perturbations that corrupt visual appearance or alter object geometry; and (3) these benefits transfer to real-world manipulation, where 3D guidance improves performance across language, spatial, and visual generalization axes, with the most substantial gains in tasks demanding precise depth control such as pouring. Several limitations suggest directions for future work. First, our framework requires explicit depth maps from an RGB-D sensor; integrating monocular depth estimation would remove this hardware dependency. Second, the planner operates from a single viewpoint, making it vulnerable to heavy occlusion; multi-view fusion would improve robustness in cluttered scenes. Finally, our evaluation is limited to single-arm tabletop tasks; extending to mobile manipulation and bimanual coordination would test the generality of 3D trajectory guidance in more complex settings.

REFERENCES

[1] L. Beyer, A. Steiner, A. S. Pinto *et al.*, "Paligemma: A versatile 3b vlm for transfer," *arXiv preprint arXiv:2407.07726*, 2024. 1

[2] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng *et al.*, "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025. 1, 2, 3, 5

[3] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr *et al.*, "Flamingo: a visual language model for few-shot learning," in *NeurIPS*, 2022. 1

[4] B. Zitkovich, T. Yu, S. Xu, P. Xu *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *CoRL*, 2023. 1

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao *et al.*, "Openvla: An open-source vision-language-action model," in *CoRL*, 2024. 1

[6] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail *et al.*, "pi0.5: a vision-language-action model with open-world generalization," in *CoRL*, 2025. 1, 5

[7] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan *et al.*, "GR00T N1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025. 1

[8] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian *et al.*, "Libero-plus: In-depth robustness analysis of vision-language-action models," *arXiv preprint arXiv:2510.13626*, 2025. 1

[9] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu *et al.*, "Hamster: Hierarchical action models for open-world robot manipulation," in *ICLR*, 2025. 1, 2, 5

[10] G. Ma, S. Wang, Z. Zhang, S. Yu, and H. Tang, "Generalvla: Generalizable vision-language-action models with knowledge-guided trajectory planning," *arXiv preprint arXiv:2602.04315*, 2026. 1, 2

[11] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang, "Thinkact: Vision-language-action reasoning via reinforced visual latent planning," in *NeurIPS*, 2025. 1, 2

[12] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *RSS*, 2024. 1, 2

[13] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation," in *CoRL*, 2023. 1, 2

[14] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," in *CoRL*, 2024. 1, 3

[15] N. Gkanatsios, J. Xu, M. Bronars, A. Mousavian, T.-W. Ke *et al.*, "3d flowmatch actor: Unified 3d policy for single-and dual-arm manipulation," *arXiv preprint arXiv:2508.11002*, 2025. 1, 3, 4, 5

[16] W. Hu, J. Lin, Y. Long, Y. Ran, L. Jiang, Y. Wang *et al.*, "G^2vlm: Geometry grounded vision language model with unified 3d reconstruction and spatial reasoning," *arXiv preprint arXiv:2511.21688*, 2025. 2

[17] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024. 2, 3, 4, 5

[18] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox, "The colosseum: A benchmark for evaluating generalization for robotic manipulation," in *RSS*, 2024. 2, 5, 6

[19] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng, "Dexvla: Vision-language model with plug-in diffusion expert for general robot control," *arXiv preprint arXiv:2502.05855*, 2025. 2

[20] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali *et al.*, "Robopoint: A vision-language model for spatial affordance prediction for robotics," in *CoRL*, 2024. 2, 4

[21] E. Zhou, C. Chi, Y. Li, J. An, J. Zhang, S. Rong *et al.*, "Robotracer: Mastering spatial trace with reasoning in vision-language models for robotics," *arXiv preprint arXiv:2512.13660*, 2025. 2, 5

[22] H. Tan, E. Zhou, Z. Li, Y. Xu, Y. Ji *et al.*, "Robobrain 2.5: Depth in sight, time in mind," *arXiv preprint arXiv:2601.14352*, 2026. 2, 5

[23] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta *et al.*, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," *arXiv preprint arXiv:2402.07872*, 2024. 2

[24] P. Sundaresan, S. Belkhale, D. Sadigh, and J. Bohg, "Kite: Keypoint-conditioned policies for semantic manipulation," in *ICML*, 2023. 2

[25] F. Liu, K. Fang, P. Abbeel, and S. Levine, "Moka: Open-world robotic manipulation through mark-based visual prompting," *RSS*, 2024. 2

[26] E. Zhou, J. An, C. Chi, Y. Han, S. Rong, C. Zhang *et al.*, "Roborefer: Towards spatial referring with reasoning in vision-language models for robotics," in *NeurIPS*, 2025. 2, 3, 4

[27] J. Gu, S. Kirmani, P. Wohlhart *et al.*, "Rt-trajectory: Robotic task generalization via hindsight trajectory sketches," in *ICRL*, 2024. 2

[28] Y. Wang, L. Ke, B. Zhang, T. Qu, H. Yu, Z. Huang *et al.*, "N3d-vlm: Native 3d grounding enables accurate spatial reasoning in vision-language models," *arXiv preprint arXiv:2512.16561*, 2025. 2

[29] D. Zheng *et al.*, "Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors," *NeurIPS*, 2025. 2

[30] J. Huang, S. Yong, X. Ma, X. Linghu, P. Li, Y. Wang *et al.*, "An embodied generalist agent in 3d world," in *ICML*, 2024. 2

[31] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2025. 2

[32] R. Wolf, Y. Shi, S. Liu, and R. Rayyes, "Diffusion models for robotic manipulation: a survey," *Frontiers in Robotics and AI*, 2025. 2

[33] A. Goyal, V. Blukis, J. Xu, Y. Guo *et al.*, "Rvt-2: Learning precise manipulation from few demonstrations," in *RSS*, 2024. 2

[34] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022. 3

[35] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020. 3, 4, 5

[36] I.-M. contributors, "Interndata-m1," https://github.com/InternRobotics/InternManip, 2025. 4

[37] R. Wang, S. Xu, Y. Dong, Y. Deng, J. Xiang, Z. Lv *et al.*, "Moge-2: Accurate monocular geometry with metric scale and sharp details," *arXiv preprint arXiv:2507.02546*, 2025. 4

[38] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park *et al.*, "Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models," in *CVPR*, 2025. 4

[39] A. Gupta, P. Dollár, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," in *CVPR*, 2019. 4

[40] Y. Zhang, B. Ni, X.-S. Chen, H.-R. Zhang, Y. Rao, H. Peng *et al.*, "Bee: A high-quality corpus and full-stack suite to unlock advanced fully open mllms," *arXiv preprint arXiv:2510.13795*, 2025. 4

[41] B. Tan, C. Sun, X. Qin, H. Adai, Z. Fu *et al.*, "Masked depth modeling for spatial perception," *arXiv preprint arXiv:2601.17895*, 2026. 5

[42] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, 2022. 5