VEGA: VISUAL ENCODER GROUNDING ALIGNMENT FOR SPATIALLY-AWARE VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Precise spatial reasoning is fundamental to robotic manipulation, yet the visual backbones of current Vision-Language-Action (VLA) models are predominantly pretrained on 2D images without explicit 3D geometric supervision, which can limit their ability to capture fine-grained spatial relationships. Prior work partially addresses this limitation by aligning LLM-level visual tokens with features from 3D-aware foundation models. However, our probes show more localized attributions around task-relevant regions at the visual encoder output, along with larger action-prediction errors under encoder-level masking at higher tested masking ratios than under masking at the probed LLM layer. These observations motivate direct geometric supervision of the visual representations supplied to downstream action prediction. We propose VEGA (Visual Encoder Grounding Alignment), a lightweight framework that grounds spatial knowledge directly at the visual encoder output, avoiding the need to select an internal LLM layer for alignment. VEGA uses frozen spatial teachers obtained by fine-tuning visual backbones with multi-view-consistent feature targets rendered from 3D Gaussian representations. A lightweight projector aligns encoder outputs with teacher features through a patch-level cosine distance loss, jointly optimized with the action prediction objective. The teacher and alignment projector are used only during training, leaving the policy architecture and inference cost unchanged. Extensive experiments on RoboTwin 2.0 and real-world manipulation tasks demonstrate that VEGA achieves higher average success rates than existing implicit spatial grounding baselines, with improvements on both OpenVLA-OFT and $\pi_{0.5}$ highlighting its applicability across VLA architectures.

1 INTRODUCTION

Robotic manipulation requires precise spatial reasoning about objects, contacts, and geometric relationships in the 3D physical world (Song et al., 2025; Landsiedel et al., 2017). Vision-Language-Action (VLA) models have demonstrated strong instruction-following capabilities by leveraging large-scale vision-language pretraining (Kim et al., 2024; 2025; Black et al., 2024; Intelligence et al., 2025; Zitkovich et al., 2023; Bjorck et al., 2025). However, their visual backbones are predominantly pretrained on 2D images or image-text pairs, typically without explicit 3D geometric supervision (Liu et al., 2023; Chen et al., 2023; Karamcheti et al., 2024). Understanding relative spatial relationships and estimating object distances and sizes remain challenging for vision-language models (Chen et al., 2024; Kamath et al., 2023). For robotic manipulation, accurate spatial representations are important for localizing interaction targets and predicting actions in 3D space (Shridhar et al., 2023; Goyal et al., 2023; Yuan et al., 2024). These observations motivate enriching VLA visual representations with geometric supervision to better support precise manipulation.

To bridge this gap, existing approaches generally pursue either explicit or implicit spatial grounding. Figure 1 illustrates explicit geometry inputs and alternative supervision locations for implicit representation alignment. The first incorporates explicit 3D information through sensor-derived depth maps or point clouds (Sun et al., 2025; Bhat et al., 2025), geometry estimated from RGB images (Qu et al., 2025), or spatially annotated training data (Zhen et al., 2024; Huang et al., 2023). These approaches face practical challenges related to sensing and calibration, estimation accuracy,

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.

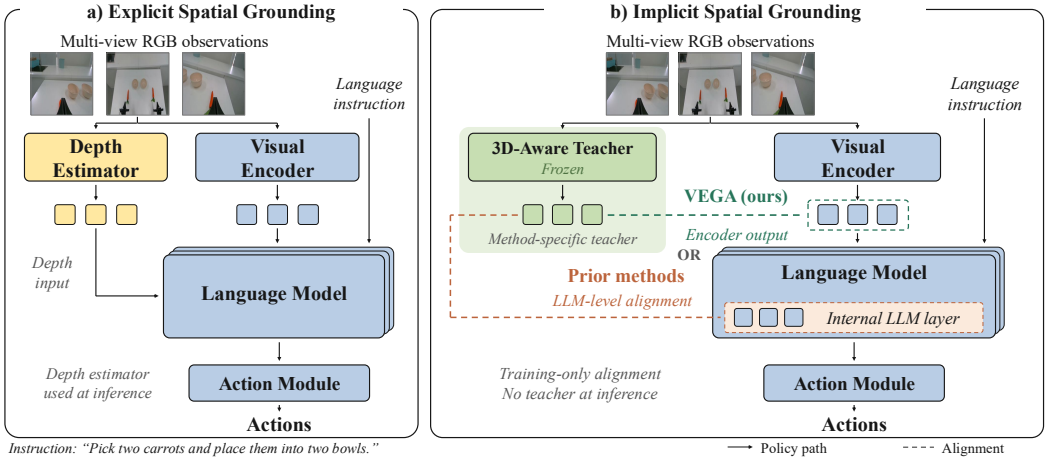

Figure 1: **Representative spatial grounding strategies for VLA models. (a)** Explicit grounding augments policy inputs with estimated geometry, retaining the depth estimator at inference. **(b)** Implicit grounding via representation alignment supervises internal LLM visual tokens in prior methods, whereas VEGA aligns visual encoder outputs with a FiT3D-adapted teacher. The dashed paths indicate alternative alignment sites, with method-specific frozen teachers used only during training. Both panels use multi-view RGB observations and a language instruction.

or additional data preparation. The second pursues implicit spatial grounding by transferring features from pretrained 3D foundation models, such as VGGT (Wang et al., 2025), through feature fusion (Lin et al., 2025) or representation alignment (Li et al., 2025; Sun et al., 2026; Guo et al., 2025). Feature fusion retains an auxiliary geometry encoder at inference, whereas alignment uses the geometry teacher only during training. However, LLM-level alignment requires choosing internal supervision layers through empirical evaluation or heuristic rules (Li et al., 2025; Sun et al., 2026), placing the alignment objective downstream of the visual encoder rather than directly on its output representations. This motivates exploring direct geometric supervision at the visual encoder output to strengthen the spatial representations supplied to downstream action prediction.

To address this limitation, we propose **VEGA** (**V**isual **E**ncoder **G**rounding **A**lignment), a lightweight framework that directly aligns VLA visual encoder outputs with features from frozen 3D-aware teachers. VEGA applies geometric supervision to the upstream visual representations used for action prediction, avoiding the need to select internal LLM layers for alignment. We construct the spatial teachers by fine-tuning the corresponding visual backbones using FiT3D (Yue et al., 2024), with multi-view-consistent feature supervision derived from 3D Gaussian Splatting (Kerbl et al., 2023). During training, a lightweight alignment projector maps visual encoder outputs into the teacher feature space, and a patch-level cosine distance loss is jointly optimized with the action prediction objective. The teacher and alignment projector are used only during training, leaving the policy's inference architecture and computational cost unchanged. Experiments on RoboTwin 2.0 (Chen et al., 2025) and real-world manipulation tasks demonstrate higher average success rates than the evaluated implicit spatial grounding baselines. Improvements on both OpenVLA-OFT (Kim et al., 2025) and $\pi_{0.5}$ (Intelligence et al., 2025) support VEGA's applicability across VLA architectures.

In summary, our main contributions are as follows:

- We motivate encoder-level spatial supervision through attribution visualization and ROI-token masking, highlighting localized encoder-level attribution patterns and the sensitivity of action prediction to perturbations of upstream visual representations.
- We propose VEGA, a lightweight framework that aligns visual encoder outputs with frozen FiT3D-adapted teacher features through an auxiliary alignment projector. The teacher and projector are used only during training, introducing no additional inference cost.
- Experiments on RoboTwin 2.0 and real-world manipulation tasks demonstrate higher average success rates than the evaluated implicit spatial grounding baselines, with improvements on both OpenVLA-OFT and $\pi_{0.5}$ supporting VEGA's applicability across VLA architectures with different visual backbones.

2 Related Work

2.1 Vision-Language-Action Models

Vision-language-action (VLA) models integrate visual perception, language understanding, and action prediction for robotic control. RT-1 (Brohan et al., 2022) demonstrated scalable language-conditioned visuomotor learning, while RT-2 (Zitkovich et al., 2023) adapted pretrained vision-language models (VLMs) to generate robot action tokens. Open-source generalist policies, including Octo (Team et al., 2024) and OpenVLA (Kim et al., 2024), use large-scale cross-embodiment datasets (O'Neill et al., 2024) to support multi-task learning and adaptation to new robots. OpenVLA combines DINOv2 (Oquab et al., 2023) and SigLIP (Zhai et al., 2023) visual features with a language backbone, while OpenVLA-OFT (Kim et al., 2025) improves performance and inference efficiency through continuous action regression, parallel decoding, and action chunking. π_0 (Black et al., 2024) and GR00T N1 (Bjorck et al., 2025) combine pretrained VLMs with flow-matching action generation, while $\pi_{0.5}$ (Intelligence et al., 2025) extends π_0 through heterogeneous co-training for open-world generalization. VLM-based policies commonly inherit visual encoders pretrained on 2D images or image-text pairs without explicit 3D geometric supervision. VEGA complements these advances by introducing 3D-aware representation alignment at the visual encoder output.

2.2 Spatial Grounding for VLA Models

Recent work explores explicit and implicit spatial grounding for VLA models to improve spatial understanding. Explicit methods incorporate 3D positional encodings derived from estimated depth (Qu et al., 2025) or point-cloud representations (Li et al., 2026; Sun et al., 2025). Sensor-based approaches depend on depth availability and quality, while RGB-based geometry estimation adds computation and can propagate prediction errors. Implicit methods transfer geometric knowledge through features from 3D-aware foundation models such as VGGT (Wang et al., 2025). Evo-0 (Lin et al., 2025) fuses VGGT features with visual encoder features through cross-attention, retaining the geometry encoder at inference. Spatial Forcing (Li et al., 2025) and GLaD (Guo et al., 2025) align LLM hidden states corresponding to visual tokens with geometric teacher features during training. ROCKET (Sun et al., 2026) extends alignment to multiple layers using a shared projector and a training-free layer-selection strategy. These methods place the alignment objective on LLM hidden states rather than directly on visual encoder outputs. VEGA applies training-only alignment directly to visual encoder outputs, using frozen teachers obtained through FiT3D adaptation of the corresponding visual backbones (Yue et al., 2024). This design avoids internal LLM-layer selection while preserving the policy's inference architecture and cost.

3 Method

3.1 Preliminaries

Vision-Language-Action Models. VLA models map visual observations and language instructions to robot actions. Given multi-view images $\mathcal{I} = \{I_k\}_{k=1}^{K}$ and an instruction $\mathcal{L}$, a visual encoder E_{vis} extracts visual features $\mathbf{F}^V = E_{\text{vis}}(\mathcal{I})$, while the instruction is tokenized and embedded. The resulting visual and language representations are processed by a multimodal backbone and an action prediction module. The policy is trained on demonstrations $\mathcal{D}$ using

$$\mathcal{L}_{\text{action}}(\theta) = \mathbb{E}_{(\mathcal{I},\mathcal{L},A_{\text{gt}})\sim\mathcal{D}}\left[\ell_{\text{action}}(\theta;\mathcal{I},\mathcal{L},A_{\text{gt}})\right], \tag{1}$$

where θ denotes the policy parameters, A_{gt} is the demonstrated action sequence, and ℓ_{action} denotes the architecture-specific training objective, such as continuous action regression (Kim et al., 2025) or flow matching (Black et al., 2024; Intelligence et al., 2025). Our work focuses on strengthening the geometric representations produced by the visual encoder to support action prediction.

3D-Aware Visual Encoder Fine-Tuning. FiT3D (Yue et al., 2024) enhances 2D visual foundation models through multi-view-consistent feature supervision derived from 3D Gaussian Splatting (Kerbl et al., 2023). In the first stage, a feature-augmented 3D Gaussian representation is optimized for each scene using posed multi-view images, with RGB reconstruction and feature supervision from a pretrained 2D encoder. This shared 3D representation aggregates features across

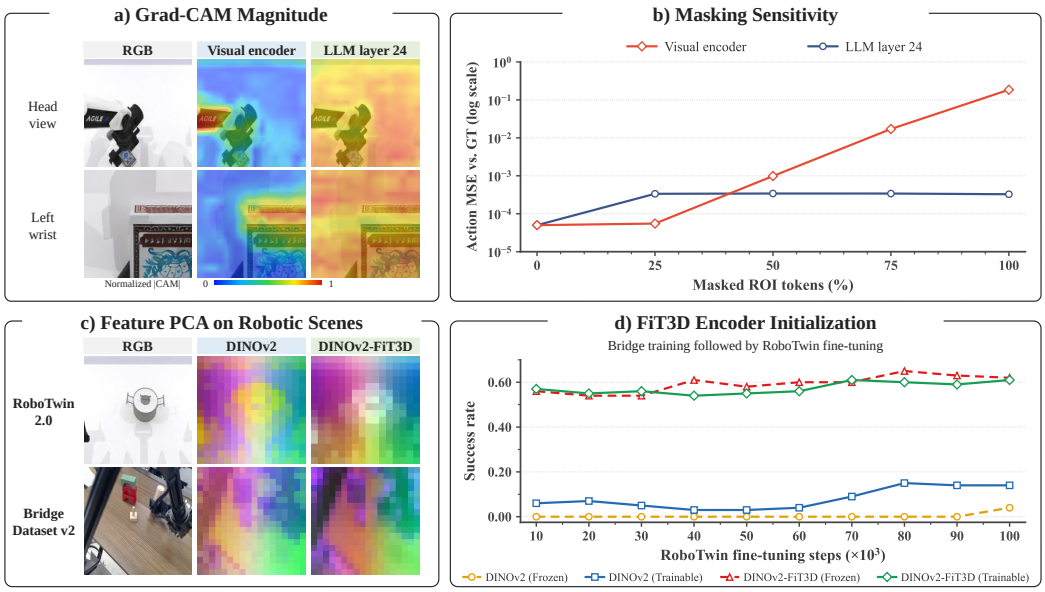

Figure 2: **Motivating encoder-level spatial supervision.** **(a)** Absolute pre-ReLU Grad-CAM values with per-representation normalization. **(b)** Action-prediction MSE against ground truth under ROI-token masking in normalized action coordinates. **(c)** PCA visualizations comparing DINOv2 and DINOv2-FiT3D features on RoboTwin 2.0 and Bridge Dataset v2. **(d)** Success rates of DINOv2 and DINOv2-FiT3D under frozen and trainable encoder settings, evaluated on Move Playingcard Away during RoboTwin fine-tuning after Bridge training.

viewpoints. In the second stage, feature maps rendered from the fixed scene representations supervise fine-tuning of the 2D encoder:

$$\mathcal{L}_{\text{FiT3D}} = \left\| E_{\text{2D}}(I) - \mathbf{F}_I^{\text{3D}} \right\|_1 , \tag{2}$$

where E_{2D} is the encoder being fine-tuned and $\mathbf{F}_I^{\text{3D}}$ is the fixed feature target rendered at the viewpoint associated with I and resized to match the encoder's output resolution. This procedure transfers multi-view geometric information into an encoder that operates on individual RGB images. VEGA uses the FiT3D-adapted encoder as a frozen spatial teacher for visual encoder alignment.

3.2 Motivation

Spatial Supervision at the Visual Encoder. Inspired by Luo et al. (2026), we probe an OpenVLA-OFT baseline (Kim et al., 2025) at the visual encoder and LLM layer 24, the alignment location used by Spatial Forcing (Li et al., 2025). We visualize the absolute values of pre-ReLU Grad-CAM, using the ℓ_2 norm of the normalized predicted action chunk as the target. For each representation, we normalize these magnitudes by their maximum across all camera views. In the illustrated example, encoder-level attribution magnitudes concentrate around the robot arm and manipulated object, whereas those at LLM layer 24 are more spatially diffuse, as shown in Fig. 2(a). This localization suggests that the visual encoder is a suitable site for imposing geometric constraints on action-relevant visual representations.

On the same observation, we mask region-of-interest (ROI) tokens at the visual encoder output or LLM layer 24, and measure action-prediction MSE against ground truth in normalized action coordinates, as shown in Fig. 2(b). At the higher tested masking fractions (50–100%), encoder-level masking yields larger errors than layer-24 masking. This greater sensitivity suggests that improving the upstream visual representations could benefit downstream action prediction. Together, the localized attribution patterns and the sensitivity to encoder-level perturbations motivate our choice to apply direct geometric supervision at the visual encoder output.

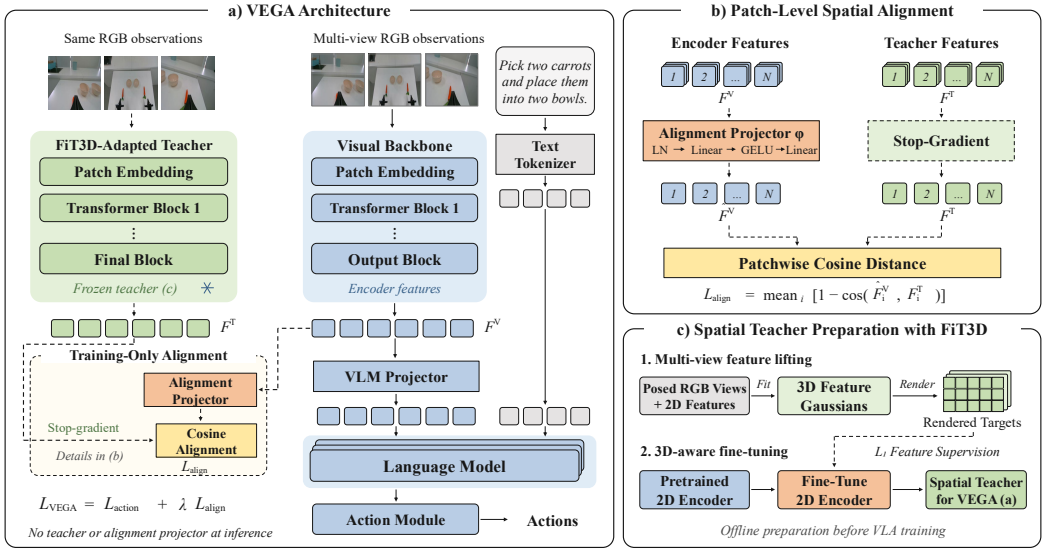

Figure 3: **Overview of the VEGA training framework.** **(a)** VEGA combines action supervision with geometric supervision at the visual encoder output. **(b)** A lightweight alignment projector maps encoder features into the frozen teacher's feature space, where the mean cosine distance between corresponding patches is minimized. **(c)** The spatial teacher is prepared offline through FiT3D adaptation using rendered 3D feature targets. The teacher and alignment projector are used only during training and removed at inference, introducing no additional computation.

3D-Aware Features for Robotic Manipulation. We investigate DINOv2-FiT3D as a source of geometric supervision. FiT3D acquires 3D-aware representations from indoor scenes (Yeshwanth et al., 2023) without using robotic data. In Fig. 2(c), PCA visualizations on RoboTwin 2.0 (Chen et al., 2025) and Bridge Dataset v2 (Walke et al., 2023) show qualitatively more spatially coherent feature patterns for FiT3D than for DINOv2.

To assess the utility of FiT3D initialization for action prediction, we compare DINOv2 and DINOv2-FiT3D initializations under frozen and trainable settings. All four configurations undergo OpenVLA pretraining on Bridge, followed by OFT fine-tuning on Move Playingcard Away in RoboTwin 2.0. In Fig. 2(d), both FiT3D variants outperform their DINOv2 counterparts across the evaluated fine-tuning checkpoints, while the frozen and trainable FiT3D variants exhibit similar learning curves.

These results suggest that FiT3D features provide useful visual representations for robotic manipulation. This motivates using FiT3D as a feature teacher for an existing pretrained VLA, introducing geometric supervision through encoder-level alignment without replacing its visual encoder or repeating Bridge training.

3.3 Visual Encoder Spatial Alignment

VEGA introduces geometric supervision directly at the visual encoder output through patch-level alignment with a frozen 3D-aware teacher (Fig. 3). Let E_{vis} denote the policy's visual encoder and E_{tea} the spatial teacher prepared offline through FiT3D adaptation.

Given an image I, the visual encoder and teacher produce patch-level features

$$\mathbf{F}^V = E_{\text{vis}}(I), \qquad \mathbf{F}^T = E_{\text{tea}}(I), \tag{3}$$

where $\mathbf{F}^V \in \mathbb{R}^{N \times d_V}$ and $\mathbf{F}^T \in \mathbb{R}^{N \times d_T}$ contain features at N corresponding spatial locations. Here, d_V and d_T denote the visual encoder and teacher feature dimensions, respectively. For multi-view observations, patch correspondences are defined within each camera view.

A lightweight alignment projector ϕ, consisting of LayerNorm followed by a two-layer MLP with a GELU activation between the linear layers, maps visual encoder features into the teacher feature space to facilitate patch-level feature alignment:

$$\hat{\mathbf{F}}^V = \phi(\mathbf{F}^V), \qquad \hat{\mathbf{F}}^V \in \mathbb{R}^{N \times d_T}. \tag{4}$$

We minimize the average cosine distance between corresponding patch features:

$$\mathcal{L}_{\text{align}} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{\hat{\mathbf{F}}_i^V \cdot \mathbf{F}_i^T}{\|\hat{\mathbf{F}}_i^V\|_2 \|\mathbf{F}_i^T\|_2}\right). \tag{5}$$

The overall training objective combines alignment with the original action-training objective:

$$\mathcal{L}_{\text{VEGA}} = \mathcal{L}_{\text{action}} + \lambda \mathcal{L}_{\text{align}}, \tag{6}$$

where λ controls the strength of geometric supervision. We optimize the trainable policy parameters and the alignment projector while keeping the teacher fixed. The alignment projector operates in an auxiliary training branch. At inference, the teacher and alignment projector are removed, leaving the policy's inference architecture unchanged and introducing no additional computation.

4 Experiments

4.1 Experimental Setup

Simulation Environment. We evaluate VEGA on RoboTwin 2.0 (Chen et al., 2025) using the AgileX ALOHA embodiment. Following prior spatial grounding studies (Li et al., 2025; Sun et al., 2026), we consider six tasks: *Move Playingcard Away*, *Turn Switch*, *Click Bell*, *Beat Block Hammer*, *Lift Pot*, and *Place Shoe*. Under the official protocol, policies are trained on 50 clean demonstrations per task and evaluated over 100 trials per task in each of the Easy and Hard settings. Easy uses clean scenes, while Hard randomizes scene clutter, background textures, lighting, and tabletop heights. We report per-task success rates and their mean across the six tasks.

Real-World Environment. We further evaluate VEGA with OpenVLA-OFT and $\pi_{0.5}$ on a physical AgileX ALOHA platform across four tasks: *Close Laptop*, *Handover Cucumber*, *Pick Dual Carrots into Dual Bowls*, and *Pick Dual Flowers into Vase*. For each task, we collect 100 demonstration trajectories through teleoperation and use the same demonstrations to fine-tune all evaluated policies. Each policy is evaluated over 20 trials per task, and we report per-task success rates.

Implementation Details. We implement VEGA on OpenVLA-OFT (Kim et al., 2025) and $\pi_{0.5}$ (Intelligence et al., 2025), using the corresponding RoboTwin 2.0 training configurations (Chen et al., 2025) as the starting point. For $\pi_{0.5}$, we adapt the SigLIP encoder using rank-32 LoRA in place of full fine-tuning, while freezing the original SigLIP weights. The corresponding $\pi_{0.5}$ baseline uses the same vision adaptation scheme without the alignment loss. We set the alignment coefficient to $\lambda = 0.1$ for OpenVLA-OFT and $\lambda = 0.01$ for $\pi_{0.5}$. We train OpenVLA-OFT on four NVIDIA H100 GPUs and $\pi_{0.5}$ on a single H100, using one training seed per configuration. Our main simulation results use evaluation seed 0.

4.2 Simulation Experiments

Main Results. Table 1 summarizes the results on six RoboTwin 2.0 tasks. With OpenVLA-OFT, VEGA achieves the highest six-task average success rates of 67.5% on Easy and 30.7% on Hard, surpassing OFT+SF by 3.3 and 2.8 percentage points, respectively. With $\pi_{0.5}$, VEGA improves the average success rate over the matched baseline from 54.7% to 57.5% on Easy and from 23.7% to 26.5% on Hard, also outperforming the corresponding SF variant in both settings. These results support VEGA's effectiveness for robotic manipulation across both VLA architectures under clean and domain-randomized conditions. Results across three evaluation seeds appear in Appendix C.1.

Training and Data Efficiency. We evaluate efficiency using OpenVLA-OFT on Move Playingcard Away under the Easy setting. As shown in Fig. 4(a), VEGA achieves 69% success at 10k steps, matching the baseline at 70k steps, and reaches 77% at 100k steps versus 70% for the baseline. Figure 4(b) shows that VEGA outperforms the baseline across four demonstration fractions from 25% to 100%, achieving 41% versus 31% success with only 25% of the data.

Table 1: **Quantitative comparison on RoboTwin 2.0.** Success rates under Easy and Hard settings. Green and yellow indicate the best and second-best values in each column, including ties. AVG denotes the mean across all six tasks; — indicates unavailable results.

Method	Move Card Away		Turn Switch		Click Bell		Beat Block		Lift Pot		Place Shoe		**AVG**	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
DP (Chi et al., 2025)	0.47	0.00	0.36	0.01	0.54	0.00	0.42	0.19	0.39	0.08	0.08	0.01	0.377	0.048
RDT (Liu et al., 2024)	0.43	0.11	0.35	0.15	0.79	0.08	0.77	0.37	0.71	0.09	0.04	0.04	0.515	0.140
π_0 (Black et al., 2024)	0.53	0.22	0.27	0.23	0.42	0.02	0.43	0.21	0.82	0.38	0.15	0.00	0.437	0.177
π_0 + SF (Li et al., 2025)	0.60	0.33	0.38	0.24	0.46	0.07	0.46	0.35	0.90	0.41	0.21	0.03	0.502	0.238
π_0 + ROCKET (Sun et al., 2026)	0.73	0.35	0.25	0.21	—	—	0.50	0.04	—	—	0.25	0.00	—	—
$\pi_{0.5}$ (Intelligence et al., 2025)	0.74	0.27	0.20	0.10	0.88	0.72	0.46	0.08	0.84	0.12	0.16	0.13	0.547	0.237
$\pi_{0.5}$ + SF (Li et al., 2025)	0.66	0.29	0.19	0.12	0.94	0.73	0.61	0.09	0.73	0.13	0.18	0.08	0.552	0.240
$\pi_{0.5}$ + **VEGA (Ours)**	0.75	0.34	0.21	0.15	0.91	0.75	0.49	0.11	0.89	0.15	0.20	0.09	0.575	0.265
OpenVLA-OFT (Kim et al., 2025)	0.70	0.34	0.20	0.08	0.95	0.46	0.02	0.06	0.98	0.33	0.51	0.09	0.560	0.227
OFT + SF (Li et al., 2025)	0.76	0.32	0.39	0.20	0.97	0.48	0.36	0.10	0.91	0.44	0.46	0.13	0.642	0.278
OFT + **VEGA (Ours)**	0.77	0.43	0.39	0.24	0.97	0.53	0.38	0.11	0.98	0.28	0.56	0.25	0.675	0.307

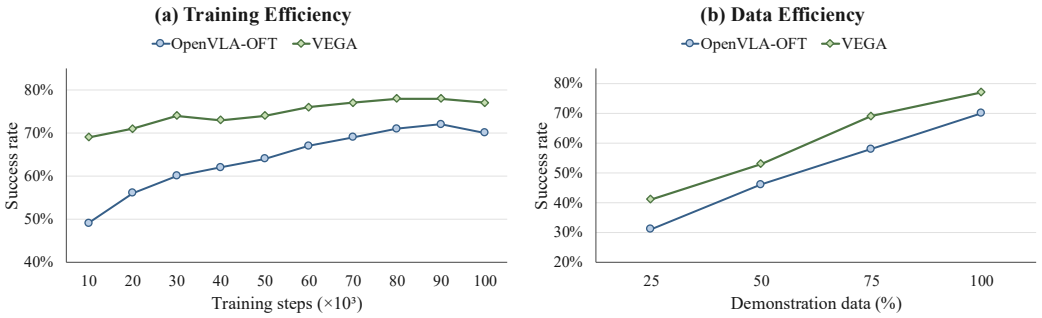

Figure 4: **Training and data efficiency.** Success rates of OpenVLA-OFT with and without VEGA across **(a)** training steps and **(b)** demonstration data fractions.

4.3 Ablation Studies and Representation Analysis

Effect of Teacher Choice and Alignment Site. Table 2 compares three teachers and two alignment sites on OpenVLA-OFT. Moving alignment from LLM layer 24 to the encoder reduces VGGT's average success rates from 64.2% to 60.2% on Easy and from 27.8% to 15.7% on Hard. For DINOv2-FiT3D, the same change improves these averages from 59.2% to 67.5% and from 20.0% to 30.7%. At the encoder output, FiT3D also outperforms the original DINOv2 teacher, which achieves 59.3% on Easy and 25.8% on Hard. These results highlight the joint importance of teacher choice and alignment site, with VEGA achieving the highest averages in both settings.

Depth Probing. We use depth probing to assess the geometric information encoded in the visual representations of OpenVLA-OFT and its VEGA variant. As shown in Table 3, VEGA reduces AbsRel across all six tasks, lowering the average error from 0.0168 to 0.0150, while improving success rates on most tasks. Together, these results suggest a positive association between depth decodability and manipulation performance, providing complementary evidence of improved geometric representations. Figure 5 presents qualitative comparisons between probe predictions and simulator ground-truth depth across camera views.

Effect of Alignment Loss Coefficient λ. We evaluate $\lambda \in \{0.05, 0.1, 0.2\}$ on OpenVLA-OFT. As shown in Fig. 6, $\lambda = 0.1$ achieves the highest six-task mean success rates among the tested values. Increasing λ to 0.2 lowers the Easy average to 61.3%, while decreasing it to 0.05 lowers the Hard average to 21.7%. We therefore use $\lambda = 0.1$ for OpenVLA-OFT.

Table 2: **Ablation of teacher choice and alignment site.** Success rates on six RoboTwin 2.0 tasks under Easy and Hard settings using OpenVLA-OFT. Green and yellow indicate the best and second-best values in each column, including ties. AVG denotes the six-task mean.

Teacher	Alignment Site	Move Card Away		Turn Switch		Click Bell		Beat Block		Lift Pot		Place Shoes		AVG	
		Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
None (baseline)	—	0.70	0.34	0.20	0.08	0.95	0.46	0.02	0.06	0.98	0.33	0.51	0.09	0.560	0.227
VGGT	LLM layer 24	0.76	0.32	0.39	0.20	0.97	0.48	0.36	0.10	0.91	0.44	0.46	0.13	0.642	0.278
VGGT	Visual encoder output	0.70	0.29	0.30	0.19	0.95	0.18	0.11	0.06	0.93	0.07	0.62	0.15	0.602	0.157
DINOv2	LLM layer 24	0.70	0.34	0.31	0.12	0.97	0.63	0.17	0.10	0.86	0.31	0.10	0.01	0.518	0.252
DINOv2	Visual encoder output	0.72	0.40	0.35	0.21	0.95	0.51	0.22	0.05	0.94	0.26	0.38	0.12	0.593	0.258
DINOv2-FiT3D	LLM layer 24	0.71	0.53	0.35	0.14	0.92	0.28	0.22	0.10	0.91	0.05	0.44	0.10	0.592	0.200
DINOv2-FiT3D	**Visual encoder output**	0.77	0.43	0.39	0.24	0.97	0.53	0.38	0.11	0.98	0.28	0.56	0.25	0.675	0.307

Table 3: **Depth probing on RoboTwin 2.0.** We compare OpenVLA-OFT with and without VEGA in Easy setting. AbsRel ↓ and Success Rate ↑; AVG is the six-task mean.

Task	OpenVLA-OFT		OFT + VEGA	
	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑
Move Card Away	0.0104	0.70	0.0103	0.77
Turn Switch	0.0307	0.20	0.0224	0.39
Click Bell	0.0074	0.95	0.0073	0.97
Beat Block	0.0129	0.02	0.0127	0.38
Lift Pot	0.0134	0.98	0.0126	0.98
Place Shoe	0.0258	0.51	0.0248	0.56
AVG	0.0168	0.560	**0.0150**	**0.675**

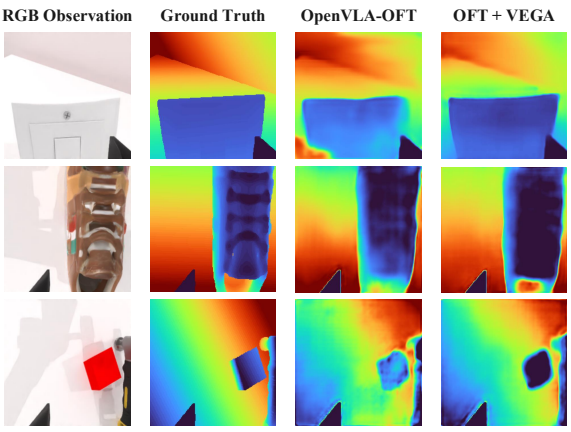

Figure 5: **Qualitative depth probing.** Rows, top to bottom: Turn Switch, Place Shoe, and Beat Block.

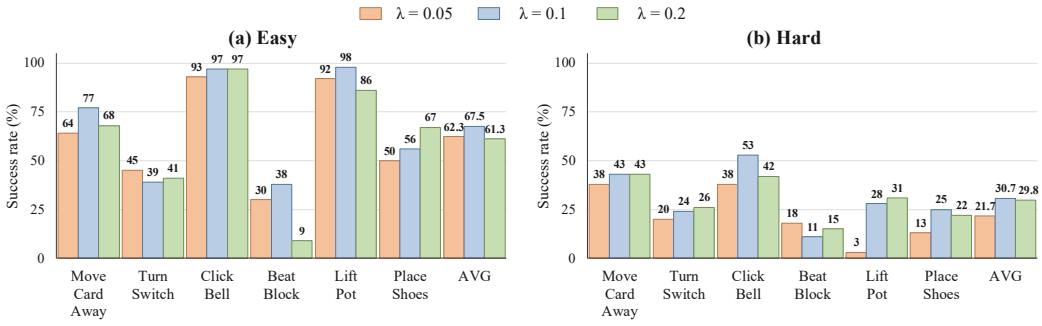

Figure 6: **Effect of alignment loss coefficient λ.** Per-task and six-task mean success rates of OpenVLA-OFT with VEGA on RoboTwin 2.0 under Easy and Hard settings.

4.4 Real-World Experiments

Result Analysis. Table 4 reports real-world success rates, with qualitative examples in Fig. 7. VEGA improves the average success rate from 47.5% to 60.0% on OpenVLA-OFT and from 60.0% to 68.8% on $\pi_{0.5}$, outperforming the matched baselines on all four tasks. On both architectures, VEGA also exceeds the corresponding SF variant by 5.0 percentage points on average, achieving higher success rates on three tasks and matching SF on *Pick Dual Carrots into Dual Bowls*. These results complement the simulation evaluations and support the effectiveness of encoder-level spatial supervision across different VLA architectures in real-world manipulation. Additional real-world results on a multi-stage domino task are reported in Appendix C.2.

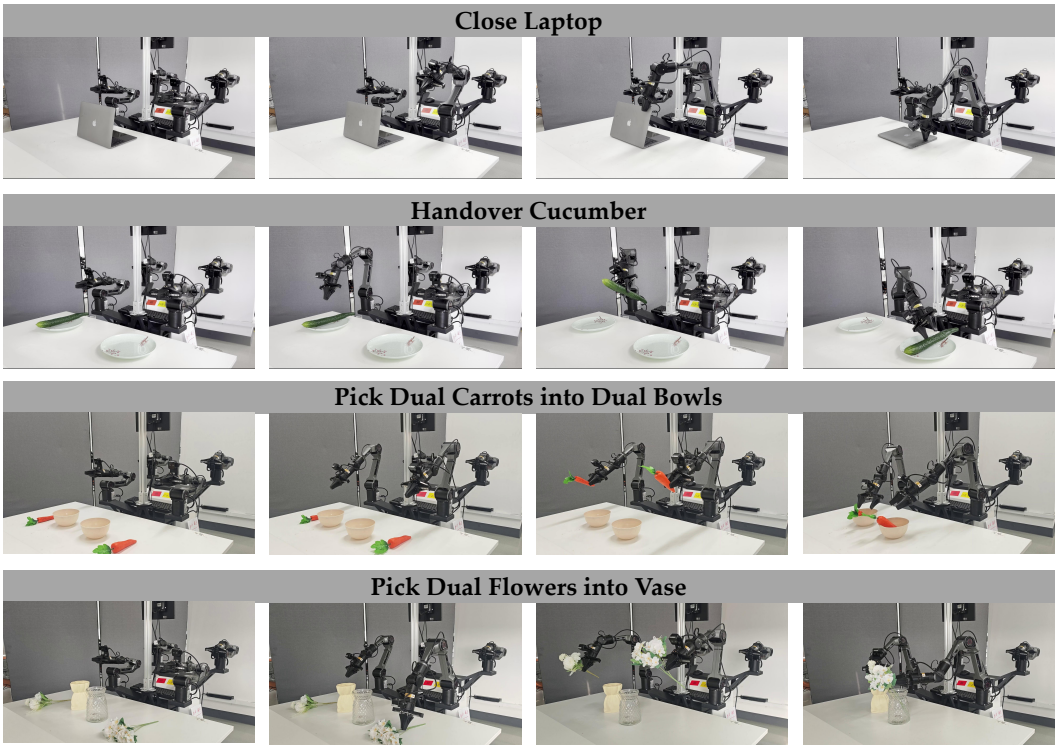

Figure 7: **Real-world manipulation with VEGA.** Keyframe sequences of four tasks, with each row showing execution from the initial state to task completion.

Table 4: **Quantitative comparison on real-world manipulation tasks.** Success rates over 20 trials per task. AVG denotes the mean across all four tasks.

Method	Single-Arm Tasks		Bimanual Tasks		AVG
	Close Laptop	Handover Cucumber	Pick Dual Carrots into Dual Bowls	Pick Dual Flowers into Vase	
OpenVLA-OFT (Kim et al., 2025)	0.65	0.60	0.50	0.15	0.475
OFT + SF (Li et al., 2025)	0.70	0.70	0.60	0.20	0.550
OFT + **VEGA (Ours)**	0.80	0.75	0.60	0.25	0.600
$\pi_{0.5}$ (Intelligence et al., 2025)	0.85	0.70	0.60	0.25	0.600
$\pi_{0.5}$ + SF (Li et al., 2025)	0.85	0.75	0.70	0.25	0.638
$\pi_{0.5}$ + **VEGA (Ours)**	0.90	0.80	0.70	0.35	0.688

5 Conclusion

In this work, we presented VEGA, a lightweight framework for implicit spatial grounding in VLA models. VEGA directly aligns visual encoder outputs with features from frozen FiT3D-adapted teachers through an auxiliary projector, removing the need to select internal LLM layers for alignment. The teacher and alignment projector are used only during training, leaving the policy architecture and inference cost unchanged. Experiments on RoboTwin 2.0 and real-world manipulation tasks demonstrate higher average success rates than the evaluated implicit spatial grounding baselines. Improvements on both OpenVLA-OFT and $\pi_{0.5}$ support VEGA's applicability across VLA architectures, while analyses on OpenVLA-OFT show gains in training and data efficiency. Our ablations also indicate that the benefits of encoder-level alignment depend on teacher choice, highlighting the importance of compatibility between teacher and visual encoder representations. Future work will investigate this compatibility, explore alternative spatial teachers, and extend evaluation to more diverse robotic tasks and embodiments.

AI USE STATEMENT

We used generative AI tools solely for language polishing and grammar correction. The authors reviewed all AI-assisted revisions to ensure that they preserved the intended meaning and technical accuracy. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work investigates spatial representation learning for robotic manipulation, with evaluations in simulation and on a physical robotic platform. Errors in learned policies may lead to unintended robot motions or object interactions, and improved task success rates do not constitute a guarantee of safe operation. Deployment beyond the evaluated settings should therefore include application-specific safety validation, appropriate operational constraints, and mechanisms for human intervention. Potential biases and coverage limitations in pretrained models and demonstration data should also be considered when applying VEGA to new environments, tasks, or embodiments.

REPRODUCIBILITY STATEMENT

The teacher preparation procedure and visual encoder alignment objective are described in Sections 3.1 and 3.3, respectively. Section 4.1 documents the datasets, demonstration counts, evaluation protocols, backbone adaptation settings, and training configurations. Sections 4.2, 4.3, and 4.4 present the experimental comparisons, ablation configurations, and evaluation results. We provide our implementation in the supplementary material and will publicly release the source code to facilitate reproduction and further research.

REFERENCES

Vineet Bhat, Yu-Hsiang Lan, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. 3d cavla: Leveraging depth and 3d context to generalize vision language action models for unseen tasks. *arXiv preprint arXiv:2505.05800*, 2025.

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, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14455–14465, 2024.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Xi Chen, Xiao Wang, Lucas Beyer, Alexander Kolesnikov, Jialin Wu, Paul Voigtlaender, Basil Mustafa, Sebastian Goodman, Ibrahim Alabdulmohsin, Piotr Padlewski, et al. Pali-3 vision language models: Smaller, faster, stronger. *arXiv preprint arXiv:2310.09199*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Minghao Guo, Meng Cao, Jiachen Tao, Rongtao Xu, Yan Yan, Xiaodan Liang, Ivan Laptev, and Xiaojun Chang. Glad: Geometric latent distillation for vision-language-action models. *arXiv preprint arXiv:2512.09619*, 2025.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_0.5$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9161–9175, 2023.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Christian Landsiedel, Verena Rieser, Matthew Walter, and Dirk Wollherr. A review of spatial reasoning and interaction for real-world robotics. *Advanced Robotics*, 31(5):222–242, 2017.

Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

Tao Lin, Gen Li, Yilei Zhong, Yanwen Zou, Yuxin Du, Jiting Liu, Encheng Gu, and Bo Zhao. Evo-0: Vision-language-action model with implicit spatial understanding. *arXiv preprint arXiv:2507.00416*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

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

Yulin Luo, Hao Chen, Zhuangzhe Wu, Bowen Sui, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Qiuxuan Feng, Jiale Yu, Shuo Gu, et al. Look before acting: Enhancing vision foundation representations for vision-language-action models. *arXiv preprint arXiv:2603.15618*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

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

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15768–15780, 2025.

Guoheng Sun, Tingting Du, Kaixi Feng, Chenxiang Luo, Xingguo Ding, Zheyu Shen, Ziyao Wang, Yexiao He, and Ang Li. Rocket: Residual-oriented multi-layer alignment for spatially-aware vision-language-action models. *arXiv preprint arXiv:2602.17951*, 2026.

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.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12–22, 2023.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In *European Conference on Computer Vision*, pp. 57–74. Springer, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Additional Experimental Setup

A.1 Spatial Teacher Preparation

We prepare DINOv2 and SigLIP spatial teachers using FiT3D (Yue et al., 2024) on ScanNet++ (Yeshwanth et al., 2023). The scene-disjoint training and validation partitions contain 230 scenes (140,451 images) and 50 scenes (30,636 images), respectively. We use anonymized DSLR RGB images and camera parameters, without semantic or depth supervision.

Feature Gaussian fitting. For each backbone, we independently fit a feature-augmented Gaussian representation for every training and validation scene. Each scene is initialized from SfM points and optimized using Adam for 30,000 iterations, sampling one view per iteration from all available within-scene views. The backbone remains frozen. Each Gaussian carries 64-dimensional features, whose rendered maps are projected to the backbone feature dimension through a bias-free 3×3 convolution. We jointly optimize Gaussian geometry, appearance, features, and the projection using

$$\mathcal{L}_{\text{stage1}} = 0.8\mathcal{L}_{\text{RGB},L_1} + 0.2(1 - \text{SSIM}) + \mathcal{L}_{\text{feature},L_1}. \tag{7}$$

Backbone fine-tuning. We freeze the fitted Gaussians and projections, initialize each backbone from its pretrained weights, and fine-tune all backbone parameters on training scenes using a mask-weighted L_1 loss against rendered feature targets. Validation scenes are used only for loss evaluation at this stage. We use AdamW with a constant learning rate of 10^{-5}, weight decay 10^{-4}, batch size 2, gradient norm clipping at 0.1, and random seed 42. Training lasts one epoch (70,225 updates), dropping the incomplete final batch. Horizontal flips are applied jointly to images, targets, and masks with probability 0.5.

Backbones and checkpoints. We use LVD-142M-pretrained DINOv2 ViT-L/14 with four register tokens and WebLI-pretrained SigLIP So400m/14. Inputs are resized from 584×876 to 574×868 and use ImageNet RGB normalization. Features are spatial patch tokens after the final LayerNorm, excluding DINOv2's CLS and register tokens. Their dimensions are $1024 \times 41 \times 62$ and $1152 \times 41 \times 62$, respectively. Gaussian rendering uses 584×876 resolution; feature targets are bilinearly resized to the required resolution in each stage. We use the final checkpoints after 30,000 fitting iterations and one fine-tuning epoch, without validation-based selection. Both teachers remain frozen during VEGA policy training.

A.2 Controlled Pretraining Setup

The controlled experiment follows a two-stage procedure. We first pretrain OpenVLA on the Bridge mixture using its action-token prediction objective. We then fine-tune the pretrained models on RoboTwin 2.0 using OFT with continuous action regression. Figure 9 reports metrics from the OpenVLA pretraining stage, while Fig. 2(d) reports success rates during subsequent OFT fine-tuning.

For the controlled pretraining experiment used in Sec. 3.2, we initialize the model from the Prismatic-DINO-SigLIP-224px+7B base VLM and train it on the Bridge mixture from Open X-Embodiment (O'Neill et al., 2024). We use four NVIDIA H100 80GB GPUs with fully sharded data parallelism (FSDP). The per-device batch size is 32, and gradient accumulation is disabled, yielding an effective global batch size of 128.

We optimize the model with AdamW using a constant learning rate of 2×10^{-5}, zero weight decay, and no warmup. The AdamW parameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We clip gradients to a maximum norm of 1.0, enable gradient checkpointing and BF16 mixed-precision training, and perform FSDP gradient reduction in FP32. Image augmentation is disabled, and all runs in this controlled pretraining experiment use random seed 7.

The training configuration allows up to 1000 epochs and does not use validation-based early stopping. The reported checkpoint corresponds to approximately 8 epochs. For the FiT3D variant, all data, optimization, freezing, and training settings are kept identical to the matched baseline; only the initialization of the DINOv2 branch is replaced with the corresponding DINOv2-FiT3D checkpoint before pretraining.

A.3 Baseline Method Details

We compare VEGA with diffusion-based policies, general-purpose VLA policies, and implicit spatial grounding methods. The baselines are grouped according to their policy architecture and spatial supervision strategy.

A.3.1 Diffusion Policies

- **DP (Chi et al., 2025):** Diffusion Policy formulates action generation as a conditional denoising process and models a distribution over continuous action sequences. We use it as a representative diffusion-based imitation learning baseline.
- **RDT (Liu et al., 2024):** Robotics Diffusion Transformer uses a Transformer-based diffusion architecture to generate robot action trajectories conditioned on multimodal observations and task instructions.

A.3.2 Vision-Language-Action Policies

- π_0 **(Black et al., 2024):** π_0 is a general-purpose VLA model that combines pretrained vision-language representations with continuous action generation for robotic control.
- $\pi_{0.5}$ **(Intelligence et al., 2025):** $\pi_{0.5}$ extends the π_0 framework through heterogeneous co-training to improve generalization across tasks, embodiments, and environments.
- **OpenVLA-OFT (Kim et al., 2025):** OpenVLA-OFT adapts the OpenVLA model to robotic manipulation through optimized fine-tuning. It uses continuous action regression, parallel decoding, and action chunking to improve policy training and inference efficiency.

A.3.3 Implicit Spatial Grounding Variants

- **Spatial Forcing (SF) (Li et al., 2025):** SF transfers geometric information from a pretrained 3D-aware teacher to a VLA by aligning visual tokens at a selected internal LLM layer during training. The teacher and alignment components are used only during training, allowing RGB-only inference. We evaluate SF on π_0, $\pi_{0.5}$, and OpenVLA-OFT, denoted as $\pi_0 + \mathrm{SF}$, $\pi_{0.5} + \mathrm{SF}$, and OFT+SF, respectively.
- $\pi_0 + \mathrm{ROCKET}$ **(Sun et al., 2026):** ROCKET performs multi-layer representation alignment with a shared projector and uses training-free layer selection to transfer spatial information from a 3D-aware vision foundation model.

Unlike these methods, which impose geometric supervision on internal LLM representations, VEGA applies the alignment objective directly at the visual encoder output.

A.4 Projector Architecture Details

The alignment projector ϕ operates independently on each patch token and maps visual encoder features into the corresponding teacher feature space. For a batch of visual features $\mathbf{F}^V \in \mathbb{R}^{B\times N\times d_V}$, the projector is defined as

$$\hat{\mathbf{F}}^V = \mathbf{W}_2\,\mathrm{GELU}\big(\mathbf{W}_1\,\mathrm{LN}(\mathbf{F}^V) + \mathbf{b}_1\big) + \mathbf{b}_2, \tag{8}$$

where LN is applied independently to the last feature dimension, N is the number of patch tokens, and $\hat{\mathbf{F}}^V \in \mathbb{R}^{B\times N\times d_T}$. The hidden dimension of the projector is denoted by d_H.

For OpenVLA-OFT, the aligned features are the 1024-dimensional patch features from the DINOv2 branch. The corresponding DINOv2-FiT3D teacher also outputs 1024-dimensional features. We use $d_V = d_T = d_H = 1024$, so the projector maps $\mathbb{R}^{B\times N\times 1024}$ to $\mathbb{R}^{B\times N\times 1024}$. It contains 2,101,248 trainable parameters, including the LayerNorm parameters. Its linear weights are initialized with Xavier uniform initialization, and all linear biases are initialized to zero.

For $\pi_{0.5}$, the SigLIP visual encoder and its FiT3D-adapted teacher have feature dimension 1152. The projector uses $d_V = d_T = 1152$ and $d_H = 1024$, with the mapping

$$\mathbb{R}^{B\times 256\times 1152} \longrightarrow \mathbb{R}^{B\times 256\times 1152}.$$

It contains 2,363,776 parameters. The projector is implemented with Flax NNX defaults: linear kernels use LeCun-normal initialization, linear biases are initialized to zero, and LayerNorm scale and bias are initialized to one and zero, respectively. The projector parameters are stored in float32.

In both architectures, the teacher encoder is fully frozen and its features are stop-gradient targets. The projector is optimized jointly with the trainable policy parameters. For $\pi_{0.5}$, this includes the vision LoRA parameters, the PaliGemma and action-expert LoRA parameters, and other parameters allowed by the configured training filter. The teacher and projector are used only in the auxiliary training branch and are removed at inference, leaving the policy's inference architecture unchanged.

A.5 Depth Probing Protocol

Data and splits. For each of the six RoboTwin 2.0 tasks, we re-render 50 clean demonstration episodes using their original seeds and trajectories, retaining one in every four sampled frames. We obtain synchronized RGB and camera-space Z-depth from the head and two wrist cameras. RGB and depth are resized to 224×224 using bicubic and nearest-neighbor interpolation, respectively, with depth measured in meters. Using split seed 2026, we partition each task into 35/8/7 training/validation/test episodes, keeping all frames and views of an episode together and sharing the split across methods and architectures. These trajectories were used for policy fine-tuning, so this split provides held-out episodes for probe training, rather than unseen data for the policy backbone.

Feature extraction. We probe OpenVLA-OFT and $\pi_{0.5}$ checkpoints at 100k and 30k training steps, respectively, freezing the visual encoders and their existing LoRA weights. For OpenVLA-OFT, we extract DINOv2 patch tokens from the penultimate block, apply the terminal LayerNorm, remove CLS/register tokens, and bypass language-conditioned FiLM. For $\pi_{0.5}$, we extract SigLIP patch tokens after the final LayerNorm and before the output projection. The feature maps are $16 \times 16 \times 1024$ and $16 \times 16 \times 1152$, respectively. RGB preprocessing uses ImageNet normalization for OpenVLA-OFT and scaling to $[-1, 1]$ for $\pi_{0.5}$.

Probe training. We train a separate decoder for each task, architecture, and policy variant, shared across the three cameras. A single-layer feature map passes through LayerNorm and a 256-channel 1×1 convolution, followed by 96-channel residual convolutional fusion at spatial scales 64×64, 32×32, 16×16, and 8×8, and upsampling to 224×224 log-depth. An affine output transform uses log-depth statistics computed only from valid training pixels. Within each architecture, baseline and VEGA probes use initialization seed 0 and identical data-shuffling orders.

We use AdamW with batch size 8, learning rate 3×10^{-4}, weight decay 10^{-4}, cosine decay to 5% of the initial learning rate, and gradient norm clipping at 1.0. The objective combines valid-pixel log-depth L_1 loss with a 0.1-weighted horizontal/vertical gradient-matching loss over valid neighboring pixel pairs. Training runs for up to 30 epochs, with early stopping after six epochs without improvement in validation AbsRel. The checkpoint with the lowest validation AbsRel is tested.

Evaluation. Valid pixels have finite ground-truth depth within $[0.05, 5.0]$ meters. Predicted log-depth is clipped to $[\log 0.05, \log 5.0]$ and exponentiated, without scale or shift alignment. For each task, AbsRel is aggregated over all valid pixels $\mathcal{V}$ across test frames and cameras:

$$\mathrm{AbsRel} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \frac{|\hat{d}_i - d_i|}{d_i}. \tag{9}$$

No additional foreground or robot masks are applied.

A.6 Real-World Robot Hardware Setup

We conduct real-world experiments on the AgileX Cobot Magic platform, which comprises four 6-DoF AgileX Piper robotic arms arranged as two leader–follower pairs. Each arm is equipped with a parallel gripper with an 85 mm stroke. The follower arms execute actions under joint position control. The platform is equipped with three RGB-D cameras: a head-mounted Intel RealSense D435 providing a global view of the workspace, and two Intel RealSense D405 cameras mounted

on the left and right follower wrists. All policies use the RGB streams from these three cameras as visual inputs.

A.7 Details of Real-World Data Collection

We collect demonstrations for four manipulation tasks, comprising two single-arm tasks and two bimanual tasks. For each task, we collect 100 demonstration trajectories through leader–follower teleoperation and use the same demonstrations to fine-tune all evaluated policies. Each trajectory contains time-synchronized recordings of the follower arms' joint positions and RGB streams from all three cameras. We vary the initial object positions on the tabletop across demonstrations. The tasks are described below.

1. **Close Laptop.** The robot uses a single arm to close an open laptop screen. This task requires precise spatial perception to locate the screen and hinge, as well as smooth and controlled motion to avoid damaging the articulated structure during contact.
2. **Handover Cucumber.** The robot grasps a cucumber from one plate and places it onto another using a single arm. This task requires accurate object localization and a smooth transfer trajectory to ensure stable grasping and precise placement without dropping the object.
3. **Pick Dual Carrots into Dual Bowls.** Each arm simultaneously grasps a carrot and places it into the nearest corresponding bowl. This bimanual task requires synchronized motion planning and spatial reasoning to correctly associate each carrot with its target bowl, while executing two independent manipulation sequences in parallel without inter-arm interference.
4. **Pick Dual Flowers into Vase.** Each arm independently grasps a flower and inserts it into its corresponding vase. This bimanual task requires coordinated motion planning and spatial reasoning, as the robot must simultaneously manage two independent manipulation sequences while avoiding inter-arm interference.

B Additional Analysis

B.1 Feature Representation Analysis

Figure 8 compares patch-level representations across the original DINOv2, DINOv2-FiT3D, and the DINOv2 branch of OpenVLA-7B after VLA training.

The PCA visualizations show that DINOv2-FiT3D generally produces more spatially coherent feature patterns than the original DINOv2, particularly in scenes containing multiple foreground objects. The DINOv2 branch of OpenVLA-7B exhibits patterns that are qualitatively closer to DINOv2-FiT3D than to the original DINOv2, suggesting that the trained VLA representation shares geometric structure with the FiT3D-adapted representation.

The ARI matrices provide a complementary quantitative comparison. The DINOv2-FiT3D–OpenVLA pair has the highest pairwise ARI in three of the four scenes and the highest mean off-diagonal ARI across scenes, 0.79 compared with 0.73 for DINOv2–DINOv2-FiT3D and 0.72 for DINOv2–OpenVLA. These results indicate that the spatial groupings of the trained VLA representation are, on average, more consistent with those of DINOv2-FiT3D than with those of the original DINOv2. This compatibility provides additional support for using DINOv2-FiT3D as the spatial teacher in VEGA.

B.2 Pretraining Dynamics Analysis

Figure 9 shows the training loss and action-token accuracy of OpenVLA with the original DINOv2 initialization and the FiT3D-initialized variant during Bridge pretraining. The four variants follow closely matched trajectories in both metrics and reach comparable values by the end of the reported run. Thus, under this pretraining configuration, initializing the DINOv2 branch from a FiT3D checkpoint does not introduce an evident optimization issue or measurable degradation in the monitored training metrics.

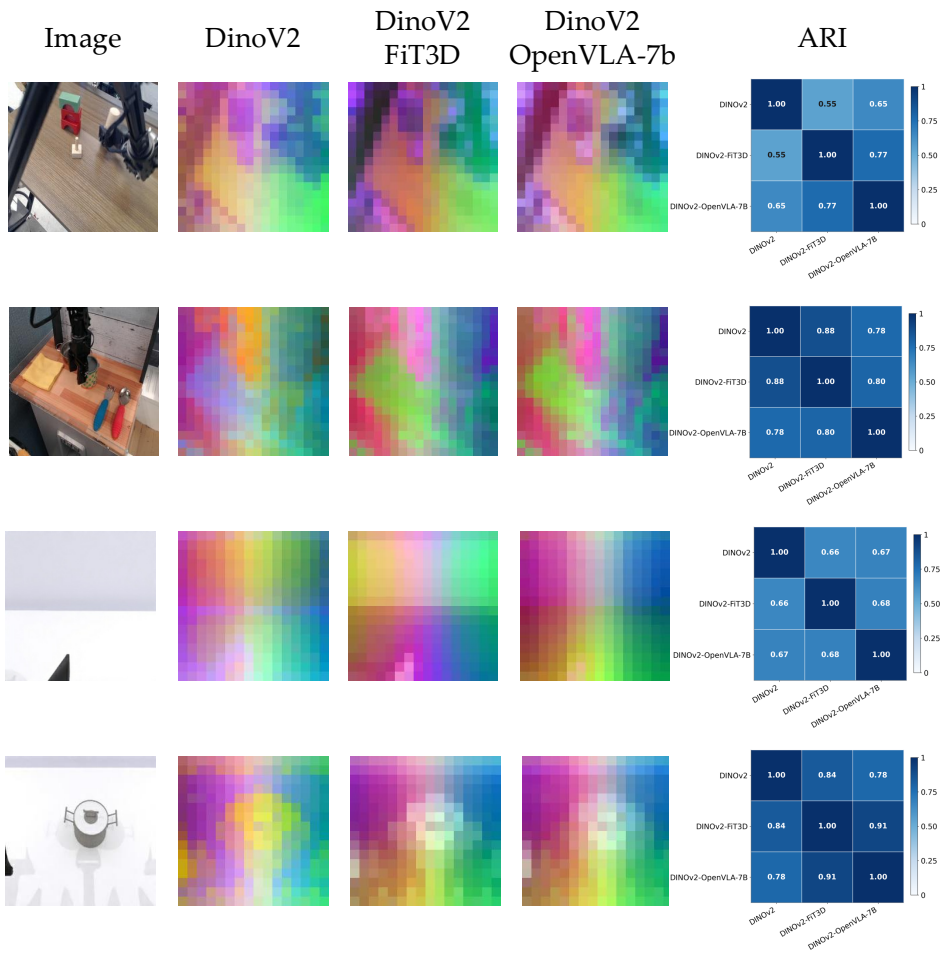

Figure 8: **Feature representation analysis across encoder variants.** Each row shows a scene from the robotic manipulation domain. Columns 2–4 show PCA visualizations of patch-level features from DINOv2, DINOv2-FiT3D, and the DINOv2 branch of OpenVLA-7B, respectively. Column 5 shows pairwise ARI matrices computed from KMeans cluster assignments of patch features, where larger off-diagonal values indicate greater agreement between spatial groupings.

C Additional Experiment Results

C.1 Results Across Evaluation Seeds

We evaluate the baseline, Spatial Forcing, and VEGA for both OpenVLA-OFT and $\pi_{0.5}$ using evaluation seeds $\{0, 1, 2\}$. For each configuration, we keep the trained checkpoint fixed and vary only the evaluation seed. Table 5 reports the mean and sample standard deviation across these evaluations. For AVG, we first compute the six-task mean for each evaluation seed and then aggregate these means across seeds.

On OpenVLA-OFT, VEGA achieves average success rates of 67.1% on Easy and 31.6% on Hard, exceeding Spatial Forcing by 2.8 and 4.5 percentage points, respectively. On $\pi_{0.5}$, VEGA improves the corresponding averages from 54.3% to 57.0% on Easy and from 23.4% to 26.6% on Hard relative to the baseline. It also surpasses the corresponding Spatial Forcing variant by 1.2 and 3.3 percentage points, respectively.

VEGA therefore achieves the highest mean six-task success rates within each architecture under both settings. These results reinforce the main experimental findings and provide additional evidence that VEGA's average performance gains persist when evaluated across different random seeds.

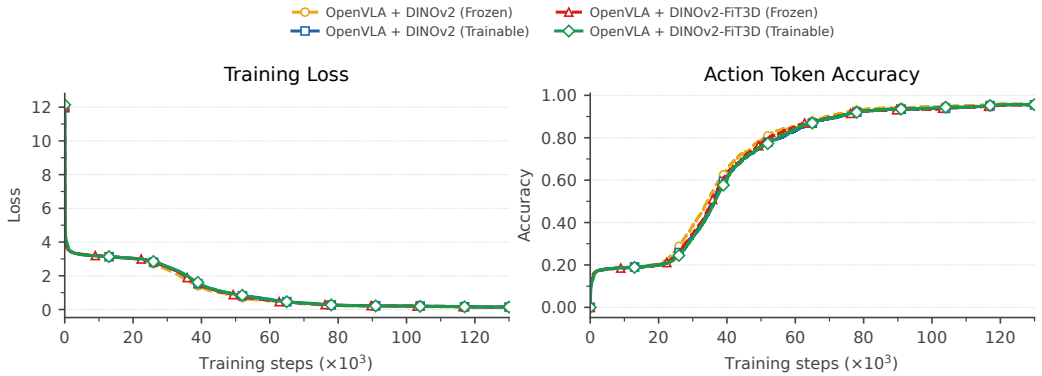

Figure 9: **OpenVLA pretraining dynamics on Bridge.** Training loss and action-token accuracy during OpenVLA pretraining with the original DINOv2 and DINOv2-FiT3D initializations, each under frozen and trainable settings. The four configurations exhibit similar training trajectories and comparable final values. The pretrained models are subsequently fine-tuned on RoboTwin 2.0 using optimized finetuning.

Table 5: **Quantitative comparison on RoboTwin 2.0 across evaluation seeds.** Success rates are reported as mean ± sample standard deviation across evaluation seeds $\{0, 1, 2\}$, with one fixed trained checkpoint per configuration. Green and yellow indicate the best and second-best mean values in each row, including ties. AVG aggregates the per-seed six-task means.

Task	**Setting**	$\pi_{0.5}$			OpenVLA OFT		
		Base	+SF	+VEGA	Base	+SF	+VEGA
Move Card Away	Easy	0.743 ± 0.015	0.680 ± 0.017	0.740 ± 0.010	0.677 ± 0.021	0.757 ± 0.015	0.760 ± 0.010
	Hard	0.260 ± 0.026	0.287 ± 0.035	0.360 ± 0.017	0.330 ± 0.010	0.303 ± 0.021	0.430 ± 0.040
Turn Switch	Easy	0.187 ± 0.015	0.180 ± 0.026	0.197 ± 0.012	0.187 ± 0.015	0.400 ± 0.026	0.410 ± 0.017
	Hard	0.100 ± 0.030	0.113 ± 0.006	0.130 ± 0.020	0.080 ± 0.040	0.197 ± 0.015	0.233 ± 0.031
Click Bell	Easy	0.863 ± 0.021	0.947 ± 0.021	0.893 ± 0.015	0.953 ± 0.025	0.970 ± 0.010	0.953 ± 0.021
	Hard	0.700 ± 0.020	0.730 ± 0.040	0.763 ± 0.012	0.480 ± 0.020	0.470 ± 0.026	0.553 ± 0.021
Beat Block	Easy	0.467 ± 0.021	0.620 ± 0.026	0.493 ± 0.025	0.010 ± 0.010	0.383 ± 0.021	0.367 ± 0.015
	Hard	0.097 ± 0.015	0.073 ± 0.015	0.107 ± 0.035	0.070 ± 0.010	0.110 ± 0.010	0.107 ± 0.015
Lift Pot	Easy	0.840 ± 0.010	0.737 ± 0.031	0.880 ± 0.026	0.977 ± 0.015	0.903 ± 0.021	0.980 ± 0.010
	Hard	0.140 ± 0.020	0.103 ± 0.023	0.153 ± 0.025	0.347 ± 0.021	0.440 ± 0.040	0.303 ± 0.021
Place Shoe	Easy	0.160 ± 0.020	0.187 ± 0.031	0.217 ± 0.021	0.520 ± 0.026	0.443 ± 0.015	0.553 ± 0.021
	Hard	0.107 ± 0.021	0.090 ± 0.010	0.080 ± 0.010	0.083 ± 0.021	0.103 ± 0.023	0.267 ± 0.015
AVG	Easy	0.543 ± 0.003	0.558 ± 0.009	0.570 ± 0.006	0.554 ± 0.006	0.643 ± 0.005	0.671 ± 0.006
	Hard	0.234 ± 0.009	0.233 ± 0.007	0.266 ± 0.011	0.232 ± 0.005	0.271 ± 0.009	0.316 ± 0.010

C.2 Additional Real-World Evaluation on Domino Manipulation

We further evaluate $\pi_{0.5}$, $\pi_{0.5}$ + SF, and $\pi_{0.5}$ + VEGA on *Domino Arrangement and Toppling*. Following the same data collection and evaluation protocol as our other real-world tasks, we collect 100 demonstration trajectories through teleoperation and use the same demonstrations to fine-tune all three policies. Each policy is evaluated over 20 trials. The robot must grasp four initially flat dominoes, place them upright in a row, and push an end domino to trigger a cascade. Success requires completing the arrangement and toppling all four dominoes through the resulting chain

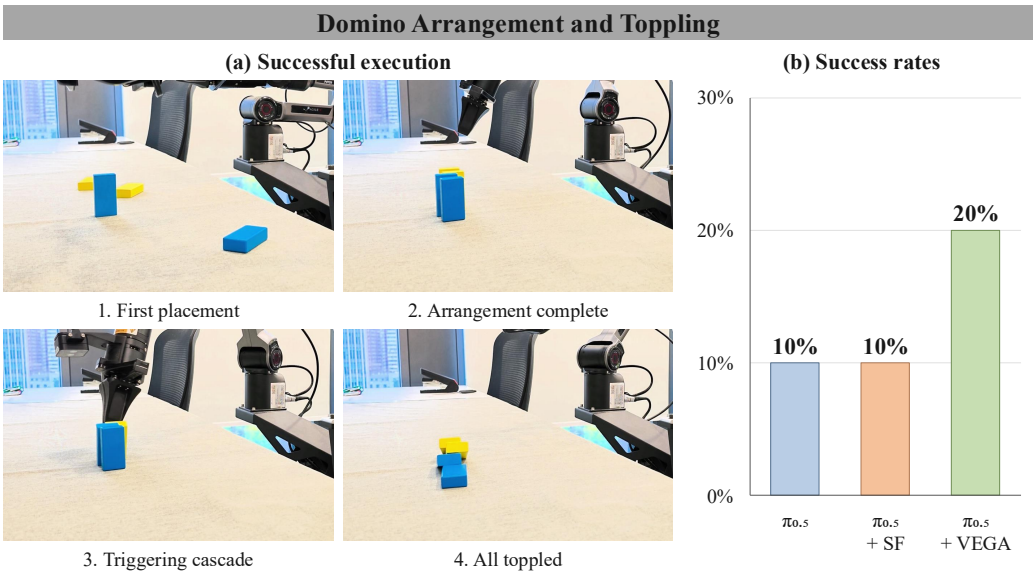

Figure 10: **Real-world evaluation on Domino Arrangement and Toppling. (a)** Keyframes from a successful execution, showing domino placement, the completed arrangement, and the final toppling sequence. **(b)** Task success rates of $\pi_{0.5}$, $\pi_{0.5}$ + SF, and $\pi_{0.5}$ + VEGA.

Table 6: **Depth probing for $\pi_{0.5}$.** Comparison of the baseline and VEGA across six RoboTwin 2.0 tasks. We report AbsRel ↓ and success rate (SR) ↑, with SR evaluated under the Easy setting. Better results are shown in bold.

Method	Move Card Away		Turn Switch		Click Bell		Beat Block		Lift Pot		Place Shoes	
	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑	AbsRel ↓	SR ↑
$\pi_{0.5}$	0.0100	0.74	0.0233	0.20	0.0068	0.88	0.0124	0.46	0.0127	0.84	0.0254	0.16
$\pi_{0.5}$ + **VEGA (Ours)**	**0.0099**	**0.75**	**0.0225**	**0.21**	**0.0066**	**0.91**	**0.0119**	**0.49**	**0.0126**	**0.89**	**0.0240**	**0.20**

reaction. This task requires accurate object reorientation and relative placement across multiple stages while maintaining the stability of previously placed dominoes.

Figure 10(a) illustrates a successful execution, from the first placement to the final cascade. As shown in Fig. 10(b), VEGA succeeds in 4 of 20 trials (20%), compared with 2 of 20 trials (10%) for both the baseline $\pi_{0.5}$ and its SF variant, yielding an improvement of 10 percentage points over each. These results extend our evaluation to a multi-stage real-world task involving sequential placement and physical interactions between objects. Nevertheless, the low absolute success rates indicate that reliable execution of the complete sequence remains challenging.

C.3 Additional Depth Probing

We further evaluate depth decodability for $\pi_{0.5}$ and its VEGA variant. Table 6 reports AbsRel and success rates across six RoboTwin 2.0 tasks. VEGA reduces AbsRel on all six tasks, lowering the mean error from 0.0151 to 0.0146. It also improves the success rate on every task, increasing the six-task mean from 54.7% to 57.5%. These consistent improvements provide complementary evidence that encoder-level spatial alignment enhances depth-related geometric representations in a second VLA architecture with a different visual backbone.

C.4 Simulation Visualization

Figure 11 presents representative executions of VEGA on all six RoboTwin 2.0 manipulation tasks. Each row contains a sequence of keyframes showing the progression from the initial scene to the

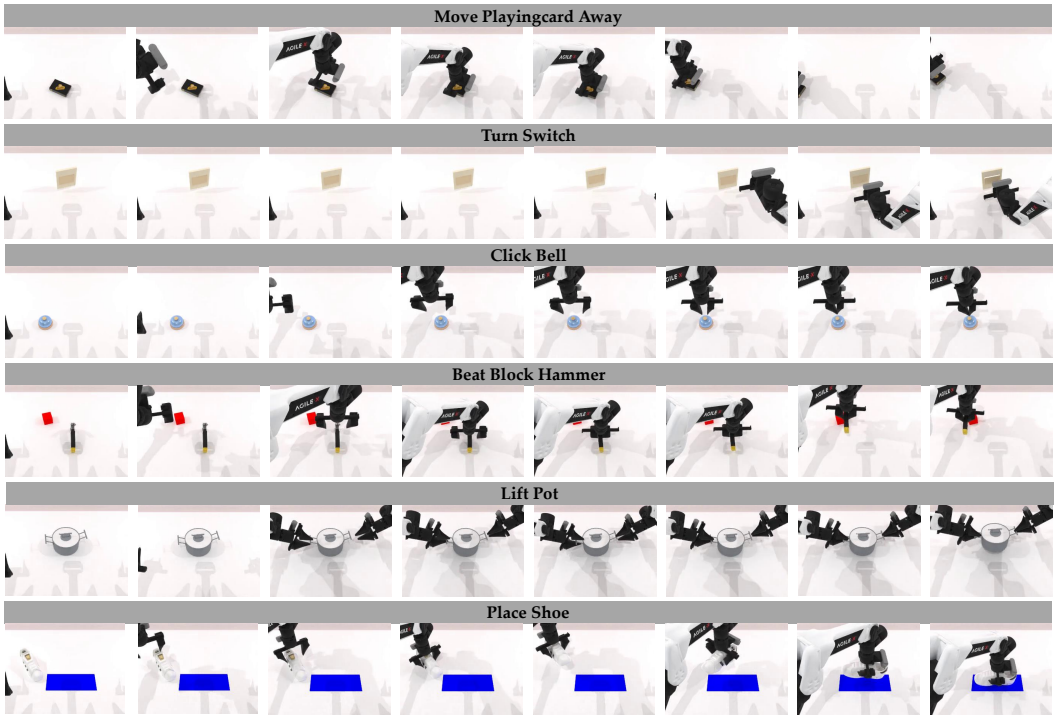

Figure 11: **Qualitative visualization of VEGA on RoboTwin 2.0.** Representative keyframe sequences for six manipulation tasks: *Move Playingcard Away*, *Turn Switch*, *Click Bell*, *Beat Block Hammer*, *Lift Pot*, and *Place Shoe*. Each row shows the temporal progression from the initial state to task completion.

final task state. The visualizations provide qualitative examples of VEGA's task execution across manipulation scenarios with different interaction patterns and object configurations.

D Real-World Demonstrations

We provide a supplementary video demonstrating VEGA on four real-world manipulation tasks, including two single-arm tasks, *Close Laptop* and *Handover Cucumber*, and two bimanual tasks, *Pick Dual Carrots into Dual Bowls* and *Pick Dual Flowers into Vase*. The video illustrates representative policy executions across tasks with different object configurations and interaction requirements, complementing the quantitative results reported in the main paper.

E Failure Case Analysis

We analyze failures from the real-world evaluations of VEGA on OpenVLA-OFT and $\pi_{0.5}$. Table 7 reports failure counts across four tasks, with 20 trials per task. We assign each failed trial to one primary category: incorrect contact location, grasp-related placement failure, tabletop collision, or timeout. Across the 80 trials, OpenVLA-OFT with VEGA records 32 failures (40.0%), while $\pi_{0.5}$ with VEGA records 25 failures (31.25%). Figure 12 illustrates representative examples of the first three categories.

Type I: Incorrect Contact Location. In *Close Laptop*, the gripper occasionally approaches an unintended side of the display and fails to establish the contact needed to close the lid. In the illustrated failure, the policy does not recover before reaching the maximum step limit. Incorrect-contact failures account for three trials with OpenVLA-OFT and two with $\pi_{0.5}$ on this task. These cases may reflect errors in localizing the intended contact region or selecting an appropriate approach trajectory, together with limited recovery from an unsuccessful interaction.

Table 7: **Failure-mode analysis of VEGA.** Failure counts over 20 trials per task, with failure rates in the Total column. Each failed trial is assigned to one primary category.

Task	Incorrect Contact Location	Grasp-Related Placement Failure	Tabletop Collision	Timeout	Total Failures
OpenVLA-OFT + VEGA					
Close Laptop	3	0	0	1	4 (20%)
Handover Cucumber	2	1	1	1	5 (25%)
Pick Dual Carrots into Dual Bowls	2	2	0	4	8 (40%)
Pick Dual Flowers into Vase	1	2	8	4	15 (75%)
$\pi_{0.5}$ **+ VEGA**					
Close Laptop	2	0	0	0	2 (10%)
Handover Cucumber	2	2	0	0	4 (20%)
Pick Dual Carrots into Dual Bowls	1	2	0	3	6 (30%)
Pick Dual Flowers into Vase	1	3	6	3	13 (65%)

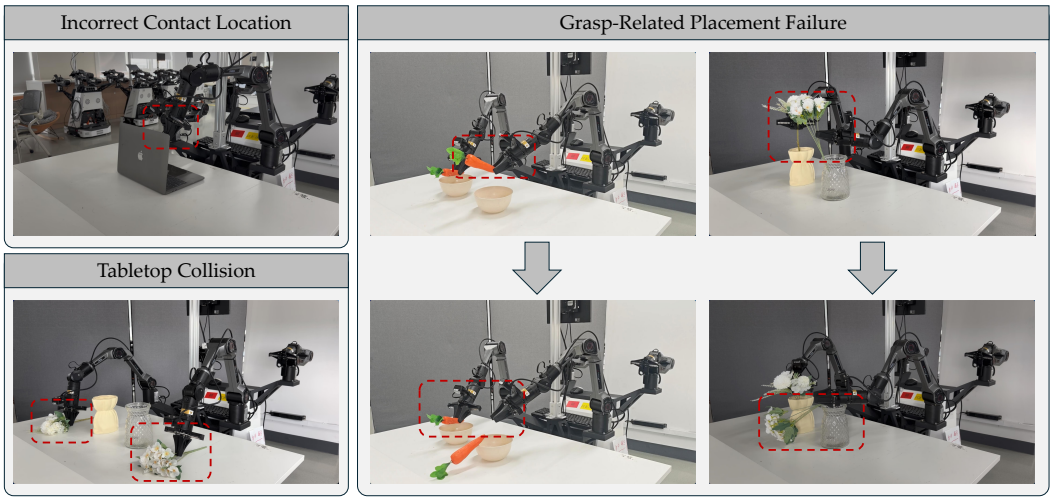

Figure 12: **Representative failure cases in real-world manipulation.** Top left: an incorrect approach to the laptop display. Right: placement failures following unfavorable grasps. Bottom left: tabletop contact during flower grasping. Red dashed boxes highlight the relevant regions, and arrows indicate temporal progression.

Type II: Grasp-Related Placement Failure. This category occurs in *Handover Cucumber* and both bimanual tasks. In *Pick Dual Carrots into Dual Bowls*, grasping a carrot near its base rather than its center changes its position relative to the gripper. In the observed failures, the carrot subsequently misses the bowl or falls out during placement. Similarly, in *Pick Dual Flowers into Vase*, an off-center grasp leaves the flower tilted, hindering alignment with the narrow vase opening. These cases suggest that the policy does not always compensate for variations in the object pose established during grasping when executing subsequent placement actions.

Type III: Tabletop Collision. Tabletop collisions occur primarily in *Pick Dual Flowers into Vase*, accounting for eight failures with OpenVLA-OFT and six with $\pi_{0.5}$. The flower stems lie close to the tabletop, and the robot occasionally contacts the table during grasping, triggering a protective shutdown of the actuators. One additional tabletop collision occurs in *Handover Cucumber* with OpenVLA-OFT. These failures highlight the difficulty of maintaining sufficient clearance during near-surface manipulation, particularly when grasping thin objects.

Type IV: Timeout. Timeout denotes trials that reach the maximum step limit without an identifiable failure belonging to the other three categories. Trials with an identified contact, placement, or collision failure remain in that category even if they subsequently reach the step limit. We record ten timeouts with OpenVLA-OFT and six with $\pi_{0.5}$. Most occur in the two bimanual tasks: four per task with OpenVLA-OFT and three per task with $\pi_{0.5}$.

These observations highlight limitations that geometric feature alignment alone does not fully address. Although VEGA supervises visual representations, its alignment objective does not explicitly constrain contact selection, object pose after grasping, or clearance during execution. Combining encoder-level alignment with grasp-aware placement, collision avoidance, and failure recovery may help address these remaining challenges.

F LIMITATIONS

VEGA's benefits depend on the choice of spatial teacher and its compatibility with the policy's visual encoder, as indicated by our teacher ablations. Moreover, encoder-level alignment supervises visual representations without explicitly constraining contact selection, object pose after grasping, or clearance during execution. Our failure analysis and the low success rates on the domino task highlight the challenges in reliable multi-stage manipulation and failure recovery. Combining spatial alignment with mechanisms that address these aspects is a promising direction. Finally, our evaluation covers two VLA architectures and a limited set of manipulation tasks. Further evaluation is needed to establish generalization across diverse tasks, environments, and robotic embodiments.