GeoRefiner: Geometry-Conditioned Plug-and-Play Action Refinement for Vision-Language-Action Models

Haijier Chen[1,2], **Xiaoquan Sun**[1,3], **Bo Chen**[1,3], **Ziyi Ding**[4], **Jiahui Chen**[1,3], **Jiarun Zhu**[1,3], **Huanzhang Hu**[1,3], **Dongchen Zheng**[1,3], **Shoujian Zhang**[2], **Mingqi Yuan**[1,3]*, **Jiayu Chen**[1,3]*

[1]INFIFORCE, Hangzhou, China
[2]Wuhan University, Wuhan, China
[3]The University of Hong Kong, Hong Kong SAR, China
[4]Tsinghua University, Beijing, China

Abstract

Vision-Language-Action (VLA) models have achieved notable progress in general-purpose robotic manipulation. However, their visual representations often lack explicit modeling and supervision of 3D spatial structure and geometric relations, which can lead to degraded performance under environmental perturbations and insufficient precision for contact-rich, fine-grained tasks. While recent geometry-aware VLA methods incorporate geometric information into policy learning, they are often tightly coupled to backbone-specific architectures and cannot be easily adapted to different base VLAs. To address this limitation, we introduce GeoRefiner, a geometry-conditioned plug-and-play framework for inference-time action refinement that operates downstream of the base model without modifying or retraining it. It leverages task-relevant geometric features to predict residual action corrections and employs a staged training strategy that progressively establishes geometry-aware representations and enables lightweight adaptation to new systems. Extensive experiments across multiple benchmarks and diverse base VLA models demonstrate that GeoRefiner consistently improves performance, supports cross-architecture reuse, and can be adapted to new systems with limited additional training.

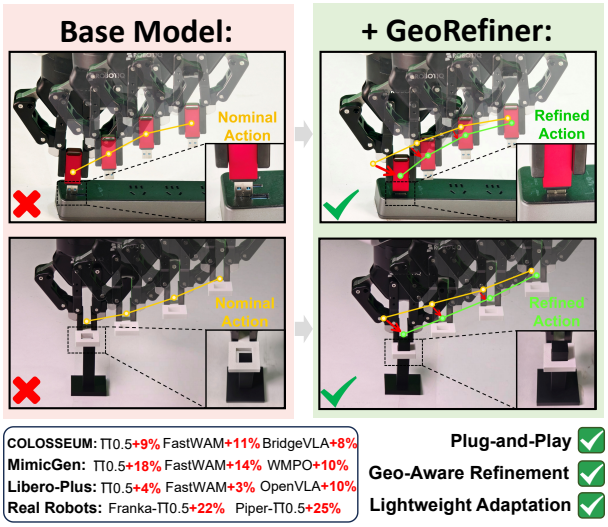

Figure 1: GeoRefiner serves as a plug-and-play downstream module that refines nominal actions from heterogeneous base VLAs without modifying or retraining them. Lightweight benchmark adapter fine-tuning supports adaptation across simulation benchmarks and robot platforms, and GeoRefiner achieves consistent gains across these settings.

Introduction

Vision-Language-Action (VLA) models have emerged as a promising paradigm for general-purpose robotic manipulation by integrating visual perception, language understanding, and action generation within a unified framework (Brohan et al. 2024; Kim et al. 2024; Black et al. 2024). Built on pretrained vision-language models (VLMs) or multimodal large language models (MLLMs) (Chen et al. 2025), these models inherit broad visual-semantic knowledge, enabling strong instruction-following and cross-task generalization capabilities in robotic manipulation.

Despite this progress, current VLA models face a fundamental limitation: their visual backbones are typically pretrained on large-scale 2D image-text data without explicit supervision of 3D spatial structure and geometric information required for robotic manipulation. As a result, their visual representations often rely more heavily on appearance-correlated cues than on stable and transferable geometric features. This limitation manifests in two main settings. First, changes in object appearance or size, background, and lighting can disrupt the estimation of target locations, spatial relations, and action trajectories (Pumacay et al. 2024). Second, fine-grained tasks such as grasping, insertion, assembly, and precise placement require accurate reasoning about local geometry, contact regions, and relative poses; even small errors can lead to execution failure (Mandlekar et al. 2023). These issues reduce the reliability of VLA models in complex and changing environments.

To address this limitation, recent work has incorporated specialized 3D vision models into VLA architectures (Zhen

*Corresponding author.

et al. 2024; Qu et al. 2025). Existing geometry-aware VLA models typically use depth, point clouds, 3D representations, or geometry tokens to improve spatial perception, demonstrating the value of geometric information for robotic manipulation. However, these components are usually embedded within the VLA backbone and depend on model-specific architectures and internal feature interfaces. Adapting these components to a new base VLA therefore often requires redesigning the fusion mechanism, modifying the network, and performing additional training. This limits the reuse of geometry-aware capabilities across different VLA models, benchmarks, and robot platforms. A unified mechanism that can enhance multiple pretrained VLA models without modifying their internal architectures is still lacking.

We introduce GeoRefiner, a geometry-conditioned, plug-and-play action refinement framework for VLA models. Our key observation is that the actions produced by a base VLA often capture the correct task intent, but execution may still fail because of local errors in position, orientation, or contact under environmental perturbations or in fine-grained manipulation settings. Rather than modifying or retraining the base VLA, GeoRefiner is deployed as an independent module downstream of the base VLA and applies bounded geometry-conditioned residual corrections to the nominal actions. This design decouples geometry-aware enhancement from a specific VLA backbone and provides a reusable action refinement mechanism that can be adapted to different systems. We evaluate GeoRefiner on COLOSSEUM, MimicGen, and LIBERO-Plus, as well as on two real-world robot platforms. The results show consistent improvements across multiple base VLA models under environmental perturbations and on fine-grained manipulation tasks. Lightweight benchmark adapter fine-tuning enables cross-system adaptation by updating only 0.9M parameters. An overview of GeoRefiner is shown in Fig. 1. Our key contributions are:

- We propose GeoRefiner, a plug-and-play, geometry-aware action refinement framework that can be attached to heterogeneous base VLAs without modifying or retraining them, while supporting lightweight adaptation across benchmarks and robot platforms.
- We introduce a geometry-conditioned gated residual refinement mechanism that uses task-relevant geometric information to apply bounded corrections to nominal actions.
- We validate the effectiveness and adaptability of GeoRefiner on multiple simulation benchmarks and real-world robot platforms. We will release the code to support future research.

Related Work

Geometry-Aware VLA Models Recent advances in VLA modeling have introduced generalist policies such as OpenVLA (Kim et al. 2024), $\pi_{0.5}$ (Intelligence et al. 2025), and GR00T (Bjorck et al. 2025). As physical interactions become more complex, the need to model 3D scene structure and geometric relations has motivated two broad directions in geometry-aware VLA research. One directly incorporates depth maps, point clouds, or 3D positional representations to explicitly model scene structure and spatial relations (Zhen et al. 2024; Qu et al. 2025; Li et al. 2026a; Sun et al. 2025), while the other learns implicit geometric priors from RGB observations through auxiliary depth supervision, representation alignment, or spatial foundation models (Lin et al. 2025; Li et al. 2025b,a; Zhang et al. 2025). 3D-MIX (Yu et al. 2026) systematically investigates several strategies for integrating geometric features into VLA models and uses semantically conditioned gated fusion to support different VLA architectures. Despite these different integration strategies, existing geometry-aware methods generally inject geometric features into the visual backbone, multimodal token stream, or action expert, as illustrated in Fig. 2(a). Their geometry-aware enhancements therefore remain coupled to the internal representations and interfaces of the base VLA. Adapting these methods to new architectures often requires modifying the model and redesigning the feature-fusion mechanism. In contrast, GeoRefiner enables geometry-aware enhancement across different base VLAs without modifying their architectures.

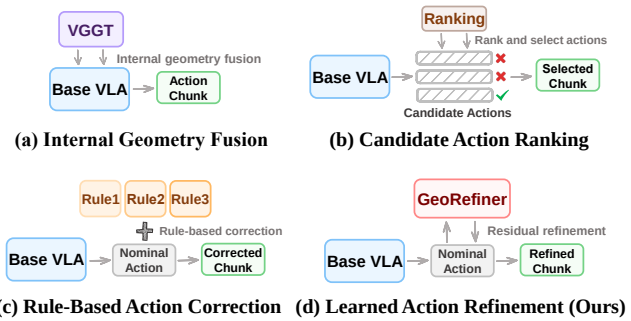

Figure 2: Architectural comparison of representative VLA enhancement paradigms. Unlike prior approaches based on internal geometry fusion, candidate action ranking, or rule-based action correction, GeoRefiner performs learned residual refinement downstream of the base VLA without modifying the base VLA, enabling cross-architecture reuse.

Plug-and-Play Enhancement for VLA Models Recent work has explored plug-and-play enhancement from several perspectives, including observation intervention, inference-time verification, and action correction, with the goal of improving deployment performance without modifying or retraining the base VLA. At the action level, one representative direction samples, searches over, or verifies multiple action candidates and selects a more reliable action at test time (Kwok et al. 2025, 2026; Zhao et al. 2026; Li et al. 2026c), as illustrated in Fig. 2(b). Another direction filters or modifies the actions produced by the base VLA according to predefined task stages, dynamics models, or safety constraints, as shown in Fig. 2(c), thereby improving control and safety during execution (Chandra, Damodaran, and Wang 2026; Hu et al. 2025). A small number of studies have also investigated learned action correction. These methods primarily address action-chunk latency or use explicit object poses to improve sim-to-real transfer (Sendai et al. 2025; Kim et al. 2026). In contrast, GeoRefiner adopts a plug-and-play

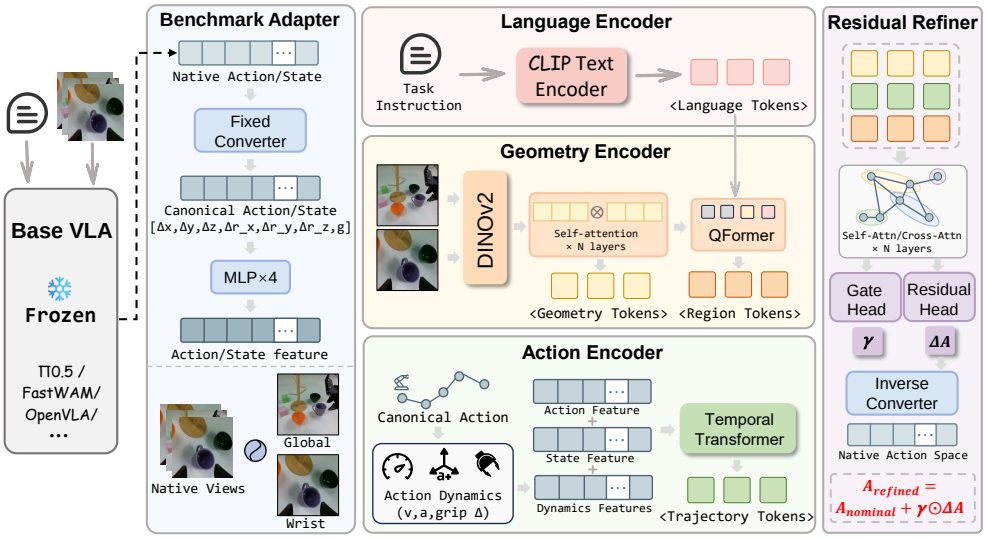

Figure 3: Architecture of GeoRefiner. GeoRefiner refines the nominal actions produced by a base VLA through three components. The benchmark adapter maps actions, states, and observations into a unified canonical space. The language-geometry-action encoder fuses task semantics, geometric cues, and action dynamics. The geometry-conditioned residual refiner then predicts gated and bounded residuals to correct geometry-related action errors.

approach, directly predicting residual corrections to the nominal actions via a learned refinement module, as illustrated in Fig. 2(d).

Method

We propose GeoRefiner, a geometry-conditioned plug-and-play action refinement framework for VLA models. GeoRefiner first uses a benchmark adapter to map heterogeneous system inputs from different benchmarks and robot platforms into a unified canonical space. The language-geometry-action encoder then extracts task-relevant geometric representations from two-view RGB observations while jointly encoding language semantics and action trajectory dynamics. Based on these representations, the geometry-conditioned residual refiner predicts bounded action residuals together with refinement gates, allowing the model to selectively correct geometrically misaligned components of the nominal action. The overall architecture is shown in Fig. 3.

Benchmark Adapter

GeoRefiner operates across heterogeneous benchmarks and robot platforms with different native action spaces, state representations, and visual observation formats. We therefore introduce a benchmark adapter that maps heterogeneous inputs into a shared canonical space, decoupling GeoRefiner from benchmark-specific data formats.

The action and state adapters follow the same structure: a fixed converter followed by a lightweight MLP calibration module. The fixed converter applies benchmark-specific transformation rules to convert native action and state representations into end-effector-centric canonical representations. The canonical action at timestep t is defined in the robot-base Cartesian frame as

$$\mathbf{a}_t = [\Delta x, \Delta y, \Delta z, \Delta r_x, \Delta r_y, \Delta r_z, g], \tag{1}$$

where the first six dimensions represent the translational and rotational motion of the end effector, and g denotes the gripper command. The MLP calibration modules then project the canonical action and state into an action feature $\mathbf{F}^a$ and a state feature $\mathbf{F}^s$, respectively. These learnable projections reduce distributional gaps in action and state representations across different systems.

For visual observations, the view adapter applies fixed view selection, resizing, and normalization operations to construct a two-view RGB input consisting of a global view and a wrist view. After GeoRefiner predicts the refined actions in the canonical space, an inverse action converter maps them back to the corresponding native action space for execution.

Language-Geometry-Action Encoder

The language-geometry-action encoder jointly represents task semantics, visual geometry, and action trajectory dynamics, providing the contextual information required for action refinement. For task semantics, we use a lightweight CLIP text encoder (Radford et al. 2021) to encode the task instruction into language tokens $\mathbf{F}^l$.

Geometry Encoder We use DINOv2 (Oquab et al. 2023) as the visual encoder to extract global-view tokens $\mathbf{X}^g$ and wrist-view tokens $\mathbf{X}^w$ from the global and wrist observations $\mathbf{I}^g$ and $\mathbf{I}^w$, respectively. The two token sequences are concatenated and processed by a self-attention transformer to model geometric relations across views, producing geometry tokens $\mathbf{G}$.

In addition to modeling cross-view geometry, effective action refinement benefits from localizing task-relevant regions, such as target objects and manipulable object parts.

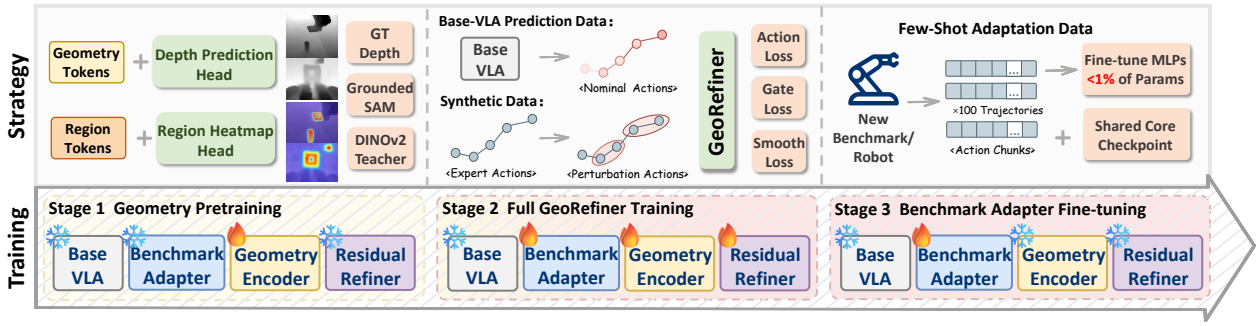

Figure 4: Overview of the three-stage training strategy. GeoRefiner is trained in three stages: geometry pretraining, full GeoRefiner training, and adapter fine-tuning. Stage 1 trains the geometry encoder with depth and region supervision. Stage 2 trains the complete GeoRefiner using synthetic perturbation data and base-VLA prediction data. Stage 3 freezes the GeoRefiner core and fine-tunes only the benchmark adapter for a new benchmark or robot platform.

We therefore employ a QFormer (**?**) with learnable region queries. Conditioned on the geometry tokens $\mathbf{G}$ and language tokens $\mathbf{F}^l$, the region queries interact with visual geometry and task semantics through cross-attention, yielding language-guided region tokens $\mathbf{R}$.

Action Encoder Given the nominal action chunk $\mathbf{A}_{\text{nominal}}$, we construct action dynamics features containing the current action, gripper velocity and acceleration, and gripper transition. These features describe motion patterns and phase transitions within the nominal trajectory. The action dynamics features are fused with the action feature $\mathbf{F}^a$ and state feature $\mathbf{F}^s$ produced by the benchmark adapter. A lightweight temporal transformer encoder then processes the fused sequence to obtain trajectory tokens $\mathbf{T}$.

Geometry-Conditioned Residual Refiner

The geometry-conditioned residual refiner applies conservative corrections to the nominal action using visual geometry and temporal action information. We concatenate the geometry tokens $\mathbf{G}$ and language-guided region tokens $\mathbf{R}$ to form the geometry context $\mathbf{C}$. The trajectory tokens $\mathbf{T}$ serve as queries and interact with $\mathbf{C}$ through stacked self-attention and cross-attention blocks, producing the fused action representation $\mathbf{F}$.

To avoid unnecessary corrections, the gate head is applied token-wise to $\mathbf{F}$ and outputs one refinement gate for each timestep and action dimension:

$$\boldsymbol{\gamma} = \text{sigmoid}\left(\text{MLP}_{\text{gate}}\left(\text{LayerNorm}(\mathbf{F})\right)\right). \tag{2}$$

A temporal residual block models local dependencies in $\mathbf{F}$ to obtain $\mathbf{H}$, from which the bounded residual is predicted:

$$\Delta\mathbf{A} = s \cdot \tanh\left(\text{MLP}_{\text{res}}\left(\text{LayerNorm}(\mathbf{H})\right)\right), \tag{3}$$

where s controls the maximum residual magnitude. The refined action is then computed as

$$\mathbf{A}_{\text{refined}} = \mathbf{A}_{\text{nominal}} + \boldsymbol{\gamma} \odot \Delta\mathbf{A}, \tag{4}$$

where $\odot$ denotes element-wise multiplication. This gated residual formulation preserves the nominal trajectory while selectively controlling the magnitude of each correction.

Three-Stage Training

GeoRefiner is trained in three stages to progressively learn geometry-aware representations, develop residual-refinement capabilities, and enable cross-system adaptation. Stage 1 performs geometry pretraining, enabling the geometry encoder to extract scene structure and task-relevant regions from RGB observations. Stage 2 trains the complete GeoRefiner to perform geometry-conditioned residual correction. Stage 3 adapts the model to a new benchmark or robot platform by fine-tuning only a small number of benchmark adapter parameters. The training procedure is illustrated in Fig. 4.

Geometry Pretraining In stage 1, we train the geometry encoder while freezing the remaining components of GeoRefiner. To provide explicit geometric supervision, we introduce two temporary auxiliary heads: a depth prediction head and a region heatmap head. The depth prediction head uses a DPT-style decoder (Ranftl, Bochkovskiy, and Koltun 2021) to decode the geometry tokens $\mathbf{G}$ into a dense depth map. The region heatmap head uses the region tokens $\mathbf{R}$ to attend to $\mathbf{G}$ and converts the resulting attention maps into task-relevant region predictions.

The stage 1 loss is defined as:

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{region}} + \mathcal{L}_{\text{dino}}. \tag{5}$$

Here, $\mathcal{L}_{\text{depth}}$ is a Smooth L1 loss between the predicted and ground-truth depth maps. $\mathcal{L}_{\text{region}}$ combines binary cross-entropy and Dice losses to supervise the task-relevant region heatmaps, using pseudo-labels generated by Grounded-SAM (Ren et al. 2024). $\mathcal{L}_{\text{dino}}$ is a cosine loss that aligns the learned visual features with those produced by a frozen DINOv2 teacher encoder, preventing the pretrained visual representations from degrading under geometric supervision.

Full GeoRefiner Training In stage 2, we jointly train the complete GeoRefiner, including the benchmark adapter, language-geometry-action encoder, and geometry-conditioned residual refiner. Stage 2 trains GeoRefiner to correct errors in nominal actions using expert actions as supervision. We therefore construct nominal actions from two

Model	Avg.	Clean	MO-Color	MO-Size	MO-Texture	RO-Color	RO-Size	RO-Texture	Light-Color	Table-Color	Distractor
$\pi_{0.5}$	55.6	66.3	43.8	64.7	47.6	56.9	45.3	61.8	53.1	70.4	46.1
+GeoRefiner	64.3	72.0	57.4	72.1	55.7	64.4	58.2	70.6	60.3	76.4	55.9
FastWAM	50.3	65.9	38.6	58.4	43.2	52.7	39.5	55.8	46.1	61.3	41.5
+GeoRefiner	61.5	73.8	54.8	67.8	53.9	64.3	54.9	65.9	55.9	69.8	53.9
BridgeVLA	65.8	73.1	60.5	69.3	63.5	63.8	61.7	68.4	69.7	75.7	51.8
+GeoRefiner	73.3	77.9	72.2	75.5	70.5	69.7	72.6	75.5	76.1	80.9	62.1

Table 1: Results on the COLOSSEUM benchmark. We report success rates (%) under the clean setting and nine environmental perturbations. GeoRefiner reduces performance degradation across diverse environmental perturbations for multiple base VLAs.

Model	Avg.	Coffee	ST	TPA	Square
$\pi_{0.5}$	36.3	45.5	51.1	20.4	28.2
+GeoRefiner	54.5(+18.2)	60.2	68.0	43.0	46.8
FastWAM	41.5	55.8	49.2	28.7	32.3
+GeoRefiner	55.0(+13.5)	66.0	62.0	46.3	45.7
WMPO	47.1	61.7	56.3	37.5	32.8
+GeoRefiner	56.7(+9.6)	68.8	64.9	50.2	42.9

Table 2: Results on the MimicGen benchmark. We report success rates (%) on four contact-sensitive, fine-grained tasks. ST and TPA denote Stack Three and Three-Piece Assembly. GeoRefiner achieves an average gain of 13.8 percentage points across three base VLAs.

Model	Avg.	Cam.	Init	Lang.	Light	BG.	Noise	Layout
$\pi_{0.5}$	82.6	70.6	78.2	77.4	92.4	91.1	84.5	83.8
+GeoRefiner	86.4(+3.8)	77.4	83.1	82.7	93.6	92.5	88.0	87.5
FastWAM	84.7	72.9	79.8	80.2	93.8	93.2	86.4	86.3
+GeoRefiner	87.8(+3.1)	78.5	84.1	84.7	94.6	94.1	89.2	89.4
OpenVLA-OFT	69.7	58.2	30.6	75.5	89.2	91.1	71.7	71.9
+GeoRefiner	79.8(+10.1)	74.0	59.9	86.8	90.2	91.5	78.2	78.0

Table 3: Results on the LIBERO-Plus benchmark. We report success rates (%) across seven evaluation variations. With a single 0.9M benchmark adapter fine-tuned on 100 trajectories, GeoRefiner improves all three base VLAs, including OpenVLA-OFT, which is held out from all stages of GeoRefiner training.

sources. The first source is synthetic perturbation data, generated by adding random errors to expert actions to construct nominal actions. The second is base-VLA prediction data, in which the nominal actions are obtained from the actual predictions of base VLA models.

The stage 2 loss is defined as:

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{action}} + \lambda_{\text{gate}}\mathcal{L}_{\text{gate}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}. \quad (6)$$

Here, $\mathcal{L}_{\text{action}}$ is a dimension-weighted Smooth L1 loss between the refined and expert actions, computed over the position, rotation, and gripper components. $\mathcal{L}_{\text{gate}}$ is a mean ℓ_1 penalty on the refinement gate $\boldsymbol{\gamma}$ to discourage unnecessary corrections, while $\mathcal{L}_{\text{smooth}}$ penalizes the ℓ_2 norm of the difference between residuals at adjacent timesteps to encourage temporal smoothness. λ_{gate} and λ_{smooth} denote the weights of the gate regularization and temporal smoothness terms, respectively.

Adapter Fine-tuning In stage 3, the GeoRefiner core is frozen, and only the MLP layers in the benchmark adapter are fine-tuned. These layers align the native action and state representations of a new system with GeoRefiner's canonical feature space, accounting for differences in action scales, state distributions, and feature distributions across benchmarks and robot platforms. Stage 3 uses only the stage 2 action loss $\mathcal{L}_{\text{action}}$.

Experiments

Experimental Setup

Training Details GeoRefiner is trained in three stages. In stage 1, we pretrain the geometry encoder using 120k two-view RGB observations, including 90k samples from simulation and 30k samples collected on the Piper and Franka platforms. Simulator-provided depth is used as supervision for the simulation data, while pseudo-depth generated by Depth Anything (Yang et al. 2024) is used for the real-world data. For both data sources, task-relevant region heatmaps are generated using Grounded-SAM. In stage 2, synthetic perturbation data are constructed by adding action errors to 4400 expert trajectories collected from COLOSSEUM (Pumacay et al. 2024), MimicGen (Mandlekar et al. 2023), Piper, and Franka. Base-VLA prediction data are collected from $\pi_{0.5}$ (Intelligence et al. 2025), FastWAM (Yuan et al. 2026), WMPO (Zhu et al. 2025), and BridgeVLA (Li et al. 2026b) across COLOSSEUM, MimicGen, Piper, and Franka. During this stage, the benchmark adapter associated with each benchmark or robot platform is jointly trained with the shared GeoRefiner core. Stage 3 adapts GeoRefiner to LIBERO-Plus (Fei et al. 2025), which is not used during stage 2. We freeze the GeoRefiner core and fine-tune the corresponding benchmark adapter for 2k steps using synthetic perturbation data constructed from 100 LIBERO-Plus trajectories. GeoRefiner has approximately 400M parameters, of which only 0.9M are trainable in stage 3. Each benchmark or real-world platform uses a separate set of benchmark adapter parameters to handle its action, state, and observation formats, while all systems share a single GeoRefiner core. Within the same system, all base VLAs share the same benchmark adapter parameters. The adapters for COLOSSEUM, MimicGen, Piper, and Franka are trained during stage 2, whereas the LIBERO-Plus adapter is trained separately in stage 3.

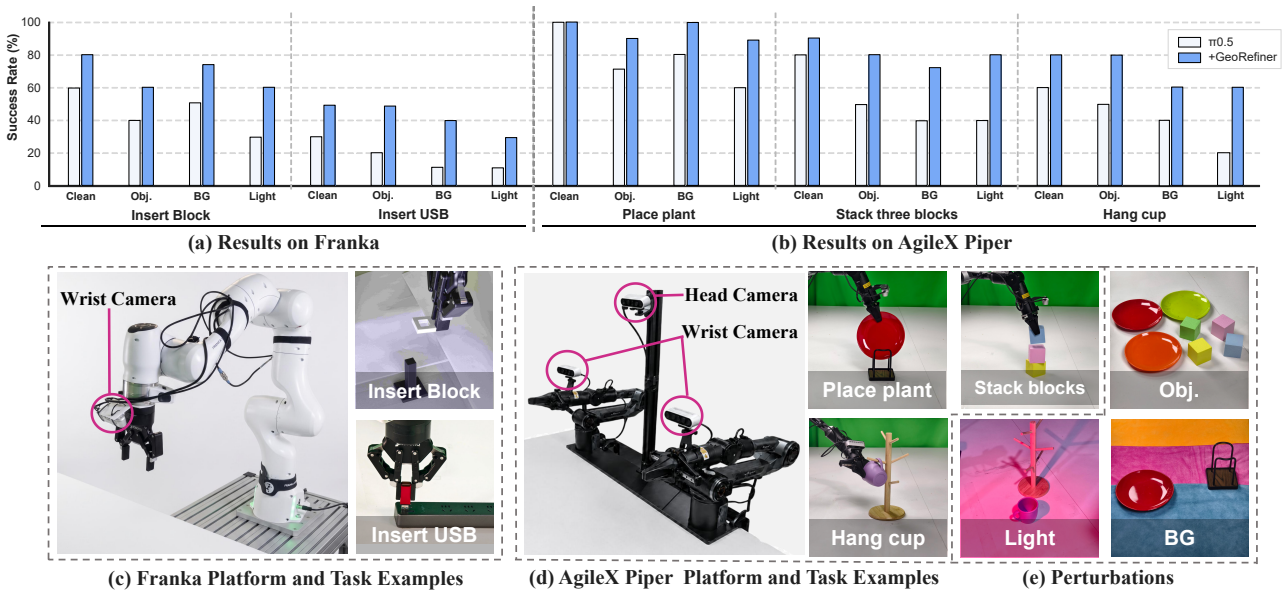

Figure 5: Real-world evaluation on Franka and AgileX Piper. We evaluate GeoRefiner on five real-world tasks across the two robot platforms. Panels (a) and (b) report success rates under clean and perturbed conditions. Panels (c) and (d) show the platforms and task examples, and (e) illustrates the evaluated perturbations.

Benchmarks and Platforms We evaluate GeoRefiner on three simulation benchmarks: COLOSSEUM, MimicGen, and LIBERO-Plus. COLOSSEUM assesses action refinement under variations in object properties, background, texture, and lighting. MimicGen includes contact-sensitive, fine-grained manipulation tasks to evaluate action refinement in grasping, assembly, and placement. LIBERO-Plus is excluded from stage 2 training and is used to evaluate whether GeoRefiner can adapt to a new benchmark through lightweight adapter fine-tuning.

We also conduct real-world experiments on five tasks across the Franka and AgileX Piper robot platforms, as shown in Fig. 5(c)–(d). Each task is evaluated under four conditions, illustrated in Fig. 5(e): clean, object perturbation (Obj.), background perturbation (BG), and lighting perturbation. Object perturbation changes the properties or placement of the target objects, background perturbation modifies the appearance of the workspace background, and lighting perturbation changes the scene illumination.

Baselines and Evaluation Protocol For the simulation experiments, we evaluate GeoRefiner with five base action models: $\pi_{0.5}$, FastWAM, WMPO, BridgeVLA, and OpenVLA-OFT (Kim, Finn, and Liang 2025). WMPO and BridgeVLA are leading methods on MimicGen and COLOSSEUM, respectively. While $\pi_{0.5}$ represents a conventional VLA, FastWAM follows the world-action-model (WAM) paradigm, allowing us to evaluate GeoRefiner on embodied action models beyond standard VLAs. No data generated by OpenVLA-OFT are used to train GeoRefiner, allowing us to evaluate adaptation to an unseen base VLA. For all real-world experiments, we use $\pi_{0.5}$ as the base VLA. For COLOSSEUM, MimicGen, and LIBERO-Plus, we follow the task configurations, fine-tuning settings, and evaluation protocols used by BridgeVLA, WMPO, and OpenVLA-OFT, respectively. All base VLAs compared within each benchmark are trained using the same data and settings. For each real-world task, $\pi_{0.5}$ is fine-tuned using 200 demonstration trajectories. We conduct 50 independent trials for each task under each evaluation condition and report the success rate. In all experiments, GeoRefiner takes only global and wrist RGB observations, robot state, task instruction, and the base VLA's nominal action as input.

Simulation Results

GeoRefiner consistently improves all evaluated base VLAs across the three simulation benchmarks. As shown in Tab. 1, environmental perturbations generally reduce base-VLA performance relative to the clean setting. GeoRefiner mitigates this degradation across models by using task-relevant geometry to refine nominal actions. As shown in Tab. 2, GeoRefiner yields an average improvement of 13.8 percentage points across the three base VLAs on MimicGen. These tasks require precise control over end-effector position and orientation, as well as contact interactions, because small terminal errors can cause failure. By identifying task-relevant manipulation regions and locally refining the nominal actions, GeoRefiner reduces end-effector alignment errors in fine-grained manipulation. As shown in Tab. 3, GeoRefiner improves all three base VLAs on LIBERO-Plus by fine-tuning a single benchmark adapter using 100 trajectories. In particular, it achieves a 10.1 percentage-point gain on OpenVLA-OFT, although no OpenVLA-OFT-generated data were used in any stage of GeoRefiner training. These results show that GeoRefiner supports cross-architecture reuse and can be adapted to new benchmarks with limited data and parameter updates.

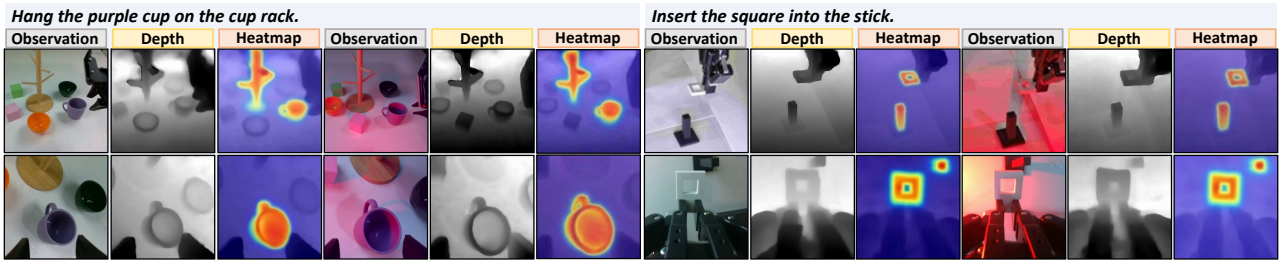

Figure 6: Geometry visualization under real-world perturbations. We use the depth prediction head and region heatmap head employed during stage 1 to visualize the geometric representations of real-world observations. The figure shows two real-world tasks under clean conditions and lighting perturbations.

	Cam.	Init	Lang.	Light	···	Avg.
Training Strategy						
w/o Stage 3	+5.9	+14.6	+3.8	+0.0	···	+4.3
Stage 3 w/ 50 Traj.	+11.2	+27.7	+9.4	+0.5	···	+8.7
Stage 3 w/ 100 Traj. (Ours)	+15.8	+29.3	+11.3	+1.0	···	+10.1
Stage 3 w/ 200 Traj.	+16.4	+29.1	+12.8	+1.0	···	+10.3
Stage 2 w/ OpenVLA-OFT	**+17.9**	**+29.6**	**+13.2**	**+1.7**	···	**+12.7**
Architecture						
w/o Refinement Gate	+13.2	+26.3	+7.6	-2.3	···	+7.9
w/o Region Tokens	+4.9	+18.4	+5.9	+0.0	···	+3.2
Full GeoRefiner (Ours)	**+15.8**	**+29.3**	**+11.3**	**+1.0**	···	**+10.1**

Table 4: Ablation studies on LIBERO-Plus. Using OpenVLA-OFT as the base VLA, we report GeoRefiner's absolute success-rate gains in percentage points.

Real-World Results

As shown in Fig. 5(a)–(b), GeoRefiner improves the success rates of $\pi_{0.5}$ across all tasks and evaluation conditions on both Franka and AgileX Piper. The gains are particularly pronounced for the fine-grained insertion tasks on Franka, where GeoRefiner corrects small positional and orientational errors in the nominal actions that would otherwise prevent successful execution. Under object, background, and lighting perturbations, GeoRefiner restores performance to approximately the clean level of the base VLA and exceeds it in several settings, suggesting that its geometry-aware representations make action predictions more robust to environmental changes.

To further examine the learned geometric representations, we apply the depth prediction head and region heatmap head from stage 1 to produce depth and region visualizations for real-world task observations. As shown in Fig. 6, the predicted depth structure remains stable under substantial lighting changes, while GeoRefiner continues to attend to task-relevant objects and manipulation regions. This shows that the learned representations are less sensitive to appearance variations while preserving useful geometric context for action refinement under perturbations. The consistent improvements on both robot platforms further demonstrate the applicability of GeoRefiner to different robot embodiments.

Ablation Studies

We conduct ablation studies on LIBERO-Plus using OpenVLA-OFT as the base VLA. The results are reported in Tab. 4. The *w/o Stage 3* setting uses no LIBERO-Plus data for additional adaptation but still achieves an average gain of 4.3 percentage points, indicating that the shared GeoRefiner core retains some zero-shot transfer capability. In the *Stage 3 w/ 50/100/200 Traj.* settings, the benchmark adapter is fine-tuned using the corresponding number of trajectories. Performance improves as more adaptation data are used and begins to saturate beyond 100 trajectories, suggesting that 100 trajectories provide a reasonable balance between adaptation cost and performance. In the *Stage 2 w/ OpenVLA-OFT* setting, LIBERO-Plus data generated by OpenVLA-OFT are added to the stage 2 training data to jointly train the complete GeoRefiner. This setting achieves the best overall performance, showing that direct exposure to the visual distribution of the target setting and the base VLA's action error patterns further improves refinement performance. We also remove two key model components. Without the refinement gate, the average gain decreases, and GeoRefiner leads to a performance drop under the lighting perturbation. This suggests that, without explicit control over the correction magnitude, the model may over-correct nominal actions that are already reliable. Removing the region tokens also reduces the average gain, indicating that global geometry tokens alone are insufficient for precisely identifying target objects, contact regions, and manipulation boundaries.

Conclusion

We presented GeoRefiner, a geometry-conditioned plug-and-play action refinement framework for VLA models. Given the nominal actions produced by a base VLA, GeoRefiner uses a benchmark adapter to align heterogeneous system inputs and combines geometric information, task semantics, and action trajectory dynamics to predict gated and bounded residual corrections. This design preserves the original action intent while correcting local errors in position, orientation, and contact, without modifying or retraining the base VLA. Through a three-stage training strategy, GeoRefiner progressively learns geometry-aware representations and develops residual-refinement capabilities while enabling lightweight cross-system adaptation. Experiments across multiple benchmarks, base VLA models, and real-world robot platforms

demonstrate its effectiveness and adaptability, while the ablation studies quantify the effects of the key model components and training stages. Despite these results, GeoRefiner is designed for local action refinement and assumes that the base VLA produces nominal actions consistent with the intended task; high-level semantic or planning errors therefore remain beyond its current scope.

References

Bjorck, J.; Castañeda, F.; Cherniadev, N.; Da, X.; Ding, R.; Fan, L.; Fang, Y.; Fox, D.; Hu, F.; Huang, S.; et al. 2025. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*.

Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*.

Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C.; et al. 2024. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv. org/abs/2307.15818*, 1: 2.

Chandra, N.; Damodaran, S.; and Wang, L. 2026. PhysVLA: Towards Physically-Grounded VLA for Embodied Robotic Manipulation. *arXiv preprint arXiv:2606.13886*.

Chen, H.; Xu, B.; Zhang, S.; Liu, H.; Lin, J.; and Wang, J. 2025. Vid-LLM: A Compact Video-based 3D Multimodal LLM with Reconstruction-Reasoning Synergy. *arXiv preprint arXiv:2509.24385*.

Fei, S.; Wang, S.; Shi, J.; Dai, Z.; Cai, J.; Qian, P.; Ji, L.; He, X.; Zhang, S.; Fei, Z.; et al. 2025. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*.

Hu, S.; Liu, Z.; Liu, S.; Cen, J.; Meng, Z.; and He, X. 2025. VLSA: Vision-Language-Action Models with Plug-and-Play Safety Constraint Layer. *arXiv preprint arXiv:2512.11891*.

Intelligence, P.; Black, K.; Brown, N.; Darpinian, J.; Dhabalia, K.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; et al. 2025. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*.

Kim, K.; Saito, N.; Kim, H.; Ikeuchi, K.; Choo, J.; and Matsushita, Y. 2026. Object-Centric Residual RL for Zero-Shot Sim-to-Real VLA Enhancement. *arXiv preprint arXiv:2606.18953*.

Kim, M. J.; Finn, C.; and Liang, P. 2025. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Kwok, J.; Agia, C.; Sinha, R.; Foutter, M.; Li, S.; Stoica, I.; Mirhoseini, A.; and Pavone, M. 2025. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*.

Kwok, J.; Zhang, X.; Xu, M.; Liu, Y.; Mirhoseini, A.; Finn, C.; and Pavone, M. 2026. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. *arXiv preprint arXiv:2602.12281*.

Li, C.; Wen, J.; Peng, Y.; Peng, Y.; and Zhu, Y. 2026a. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3): 2506–2513.

Li, F.; Song, W.; Zhao, H.; Wang, J.; Ding, P.; Wang, D.; Zeng, L.; and Li, H. 2025a. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*.

Li, P.; Chen, Y.; Wu, H.; Ma, X.; Wu, X.; Huang, Y.; Wang, L.; Kong, T.; and Tan, T. 2026b. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *Advances in Neural Information Processing Systems*, 38: 63635–63673.

Li, Y.; Chen, Y.; Zhou, M.; Li, H.; Zhang, Z.; and Zhao, D. 2025b. QDepth-VLA: quantized depth prediction as auxiliary supervision for vision-language-action models. *arXiv preprint arXiv:2510.14836*.

Li, Z.; Liu, J.; Dong, Z.; Teng, T.; Rouxel, Q.; Caldwell, D.; and Chen, F. 2026c. Towards deploying vla without fine-tuning: Plug-and-play inference-time vla policy steering via embodied evolutionary diffusion. *IEEE Robotics and Automation Letters*, 11(5): 6234–6241.

Lin, T.; Li, G.; Zhong, Y.; Zou, Y.; Du, Y.; Liu, J.; Gu, E.; and Zhao, B. 2025. Evo-0: Vision-language-action model with implicit spatial understanding. *arXiv preprint arXiv:2507.00416*.

Mandlekar, A.; Nasiriany, S.; Wen, B.; Akinola, I.; Narang, Y.; Fan, L.; Zhu, Y.; and Fox, D. 2023. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Pumacay, W.; Singh, I.; Duan, J.; Krishna, R.; Thomason, J.; and Fox, D. 2024. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*.

Qu, D.; Song, H.; Chen, Q.; Yao, Y.; Ye, X.; Ding, Y.; Wang, Z.; Gu, J.; Zhao, B.; Wang, D.; et al. 2025. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PmLR.

Ranftl, R.; Bochkovskiy, A.; and Koltun, V. 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, 12179–12188.

Ren, T.; Liu, S.; Zeng, A.; Lin, J.; Li, K.; Cao, H.; Chen, J.; Huang, X.; Chen, Y.; Yan, F.; et al. 2024. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*.

Sendai, K.; Alvarez, M.; Matsushima, T.; Matsuo, Y.; and Iwasawa, Y. 2025. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*.

Sun, L.; Xie, B.; Liu, Y.; Shi, H.; Wang, T.; and Cao, J. 2025. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*.

Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; and Zhao, H. 2024. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10371–10381.

Yu, B.; Lian, S.; Lin, X.; Shen, Z.; Wei, Y.; Liu, H.; Wu, C.; Yuan, H.; Wang, B.; Huang, C.; et al. 2026. 3d-mix for vla: A plug-and-play module for integrating vggt-based 3d information into vision-language-action models. *arXiv preprint arXiv:2603.24393*.

Yuan, T.; Dong, Z.; Liu, Y.; and Zhao, H. 2026. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*.

Zhang, Z.; Li, H.; Dai, Y.; Zhu, Z.; Zhou, L.; Liu, C.; Wang, D.; Tay, F. E.; Chen, S.; Liu, Z.; et al. 2025. From spatial to actions: Grounding vision-language-action model in spatial foundation priors. *arXiv preprint arXiv:2510.17439*.

Zhao, G.; Guo, L.; Zhu, J.; Fu, J.; Mei, Y.; Cao, B.; Jiang, J.; He, X.; and Liu, J. 2026. VeriSpace: Spatially Grounded Action Verification for Vision-Language-Action Models. *arXiv preprint arXiv:2606.10568*.

Zhen, H.; Qiu, X.; Chen, P.; Yang, J.; Yan, X.; Du, Y.; Hong, Y.; and Gan, C. 2024. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*.

Zhu, F.; Yan, Z.; Hong, Z.; Shou, Q.; Ma, X.; and Guo, S. 2025. Wmpo: World model-based policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.09515*.