VLA Models Are More Generalizable Than You Think: Revisiting Physical and Spatial Modeling

Supplementary Material

7. Real-World Experiments

To validate the practical efficacy of our method beyond simulation, we established a physical tabletop manipulation environment centered around a Franka Emika Panda robot. This section details our experimental setup, data collection pipeline, task design, and the deployment of our Feature Linear Adaptation (FLA) method. Please refer to the accompanying video in the supplementary material for visualization.

7.1. Experimental Setup and Data Collection

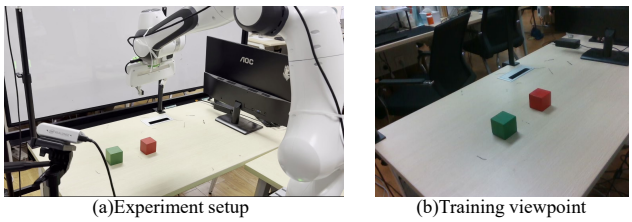
The robot setup features a 7-DoF Franka Emika Panda arm equipped with a standard parallel-jaw gripper. The system operates within an 8-dimensional configuration and action space (7 joint positions + 1 gripper state). To capture comprehensive visual information, we employ two fixed camera views: a **3rd-person static camera** providing a global view of the workspace, and a **wrist-mounted camera** providing egocentric observations. The physical configuration is illustrated in Fig. 6.

For data collection, we utilized **GELLO** [32], a general, low-cost, and intuitive teleoperation framework. This allowed for high-quality human demonstrations to be collected efficiently. We utilized the $\pi_{0.5}$ architecture as our base policy.

7.2. Evaluation Tasks

We designed five distinct real-world manipulation tasks ranging from basic object interaction to complex articulated object manipulation. These tasks were selected to rigorously evaluate the model's spatial grounding capabilities and precision under novel viewpoints:

1. **"Pick up the red block and stack it on the green block"**: A multi-stage task that assesses precision and depth perception. Successfully stacking the block requires the model to maintain geometric consistency throughout the trajectory, a challenge often exacerbated by viewpoint shifts.
2. **"Pull out the top drawer"**: This task involves manipulating a prismatic joint with frictional resistance. It evaluates the policy's robustness in maintaining contact and exerting force in the correct direction under a novel visual perspective.
3. **"Press the green button"**: A fine-grained control task requiring high spatial precision. The target is small relative to the scene, making it highly sensitive to the misalignment of spatial features usually caused by camera viewpoint changes.
4. **"Pick up the red block on the table"**: This task evaluates the fundamental spatial grounding capability of the model. It requires the agent to correctly identify the target object based on color semantics and accurately estimate its 3D position from the new camera viewpoint.
5. **"Close the microwave oven door"**: This task tests the model's ability to interact with articulated objects. It requires understanding the kinematic constraints of the door and executing a precise push motion.



(a)Experiment setup (b)Training viewpoint

Figure 6. **Real-World Experimental Setup.** (a) Our hardware environment features a Franka Emika Panda robot teleoperated via the GELLO framework, equipped with both a third-person static camera and a wrist-mounted camera. (b) The *Novel Camera Viewpoint* used for one-shot adaptation. This viewpoint introduces a significant spatial shift compared to the standard pre-training distribution, serving as the testbed for our Feature Linear Adaptation (FLA) method.

7.3. Adaptation Protocol

We evaluate the robustness of the pre-trained $\pi_{0.5}$ policy when transferred from a source viewpoint to a significantly different novel viewpoint (used in training, as shown in Fig. 6).

Consistent with our simulation results, we employ **Feature Linear Adaptation (FLA)** for domain adaptation. Specifically, we apply FLA with a rank of $r = 32$ to the linear layers of the Vision Transformer (ViT) encoder. The adaptation is performed in a **one-shot** manner: the model is updated using only a single human demonstration collected from the new viewpoint via GELLO. This setup tests whether our lightweight adaptation method can effectively realign the spatial representations of the physical world with minimal data and parameter overhead. We set the batch size to 32 and train for 750 steps.

Following the one-shot adaptation, we deploy the policy in a closed-loop control setting utilizing continuous dual-camera observations. We observe strong robustness to spa-

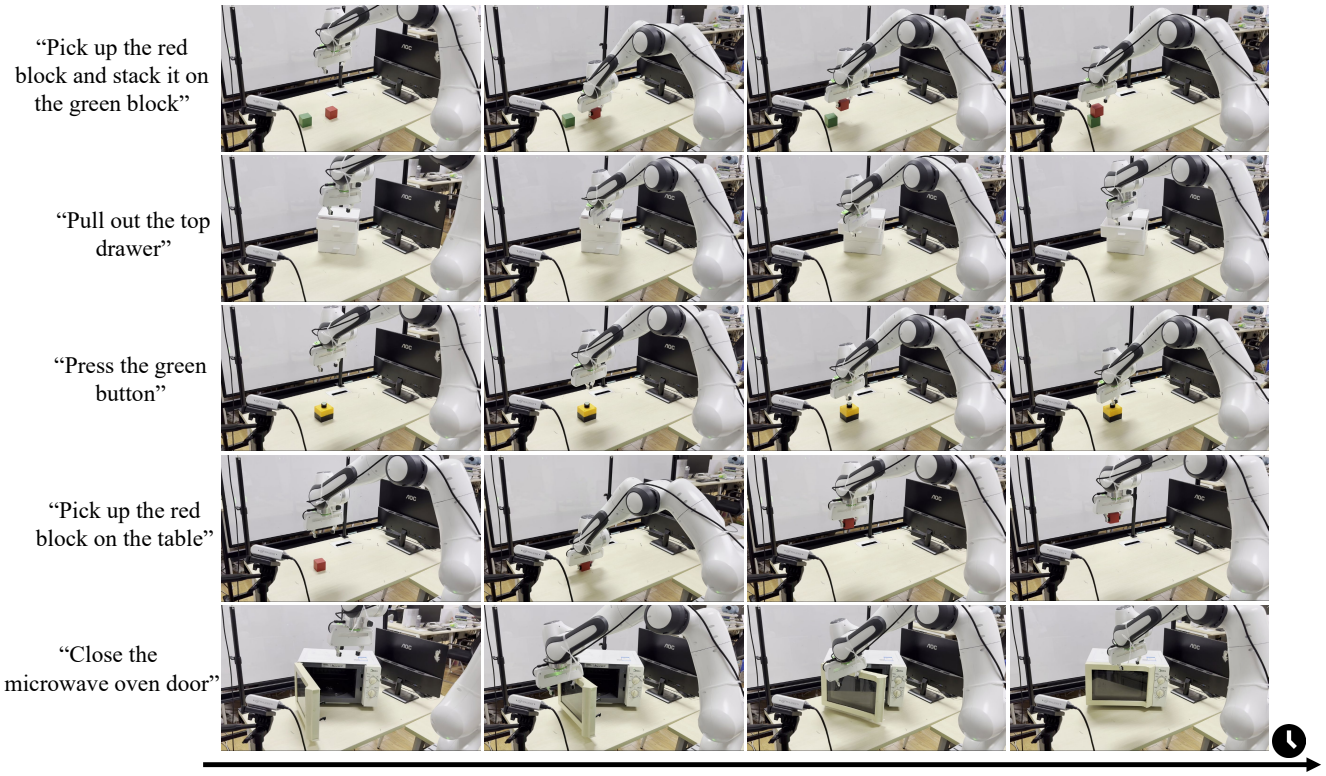


Figure 7. **Qualitative Results on Real-World Tasks.** Sample rollouts of the five evaluation tasks executed by our policy after one-shot FLA adaptation. The tasks include: (1) Pick up the red block, (2) Stack the red block on the green block, (3) Close the microwave oven door, (4) Press the green button, and (5) Pull out the top drawer. Despite the substantial visual discrepancy introduced by the novel viewpoint, the adapted policy successfully recovers spatial grounding and executes precise manipulation in a closed-loop manner.

tial misalignment and consistent success across all five tasks (see Fig. 7). These results empirically demonstrate that lightweight spatial adaptation is sufficient to bridge the domain gap between pre-trained distributions and novel real-world settings, effectively recovering the model's manipulation capabilities despite substantial visual discrepancies.

7.4. Robustness to Imperfect Demonstrations

During real-world data collection, human teleoperation can often yield imperfect or sub-optimal trajectories. As shown in Fig. 9(Adaptation Demonstrations), the one-shot demonstrations used for adaptation occasionally include imperfect executions, such as awkward grasps or slightly delayed alignments. Despite these imperfections in the source data, the Feature Linear Adaptation (FLA) method remains robust. For instance, FLA achieves a 90.7% success rate on the $\pi_{0.5}$ baseline even when adapted using these sub-optimal trajectories. This indicates that our lightweight spatial adaptation primarily realigns the visual embedding space and is highly robust to minor kinematic imperfections in the demonstration data.

7.5. Robustness to Dynamic Objects

While our primary evaluations focus on static visual perturbations (e.g., camera viewpoints, lighting), real-world deployments often encounter dynamic environments. As illustrated in Fig. 8, we simulate a dynamic scenario by manually disturbing the target object's position (the blue block) during the robot's execution phase. Because our adapted policy operates in a closed-loop manner, it successfully perceives the displacement and adjusts its trajectory on the fly to grasp the moving object. This demonstrates that our adaptation framework is highly effective and robust not only against static visual shifts but also in dynamic environments.

8. Theoretical Analysis

In this section, we provide theoretical justification for why lightweight vision–level adapters such as the Feature Token Modulation (FTM) and the Feature Linear Adaptation (FLA) are capable of restoring policy performance under large visual distribution shifts (e.g., camera viewpoint changes). We formalize the relationship between visual representation drift and policy degradation, and show that

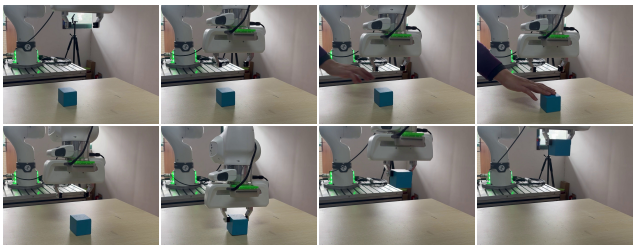


Figure 8. **Dynamic objects.** FLA successfully adjusts trajectories in real-time against manual target disturbances.

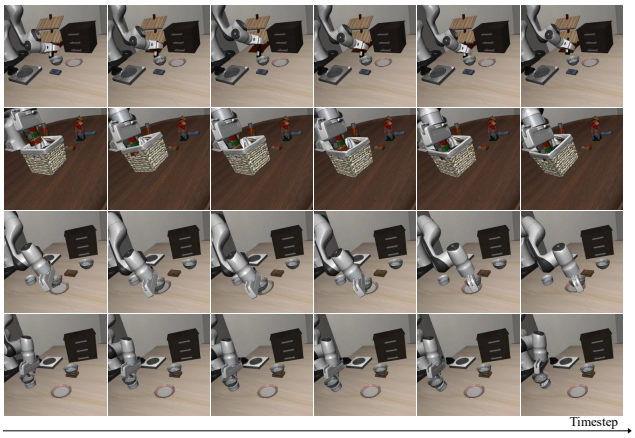


Figure 9. **Imperfect examples.** FLA maintains high success rates even with sub-optimal demonstrations.

small affine or low–rank corrections to the vision encoder are sufficient to recover the original policy behavior under mild and empirically verifiable assumptions.

8.1. Preliminaries

A vision–language–action (VLA) model is decomposed into a vision encoder f_v, a language encoder f_ℓ, and an autoregressive policy decoder g:

$$\hat{a}_t \sim g(a_{<t}; [z; \ell]), \qquad z = f_v(v), \quad \ell = f_\ell(l),$$

where z denotes the visual tokens passed to the decoder.

Let $\mathcal{D}_s$ denote the source (training) visual distribution and $\mathcal{D}_t$ the target distribution after camera/viewpoint shifts. Let

$$Z_s = f_v(\mathcal{D}_s), \qquad Z_t = f_v(\mathcal{D}_t)$$

be the corresponding distributions of visual tokens.

Throughout this section, we assume the action decoder's conditioning context $(a_{<t}, \ell)$ is fixed, and we write $g_{a|z}$ for the induced action distribution conditioned on visual token z.

We use $d_{\mathrm{TV}}(\cdot, \cdot)$ to denote total variation distance.

8.2. Assumptions

We state the assumptions required for our theoretical results.

A1 (Locally Lipschitz Policy). There exists $L > 0$ such that for all z, z' in a neighborhood of interest,

$$d_{\mathrm{TV}}\big(g_{a|z},\, g_{a|z'}\big) \le L\|z - z'\|.$$

A2 (Task Semantic Invariance). The underlying optimal action distribution does not change between $\mathcal{D}_s$ and $\mathcal{D}_t$; i.e., degradation in policy performance arises solely from drift in visual representations.

A3 (Locally Affine or Low–Rank Structure). The mapping from Z_t to Z_s can be well-approximated by an affine or low–rank linear transformation in the neighborhood of interest. This corresponds to the FTM(Sec. 3.3) and FLA(Sec. 3.4) parameterizations:

$$\text{FTM:} \qquad \hat{z} = (1 + \gamma) \odot z + \beta,$$

$$\text{FLA:} \qquad W' = W + \Delta W, \qquad \Delta W = BA,$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$ with $r \ll d$.

These assumptions are mild and can be empirically verified.

8.3. Main Results

We now formalize three theorems that connect representation drift with policy degradation, and show why affine and low–rank corrections are sufficient for recovery.

Theorem 1 (Policy degradation is upper bounded by representation drift)**.** *Let*

$$z_s^\star = \mathbb{E}_{v \sim \mathcal{D}_s}\big[f_v(v)\big]$$

be a representative source-domain token. Under Assumption A1,

$$\mathbb{E}_{v \sim \mathcal{D}_t}\big[d_{\mathrm{TV}}\big(g_{a|f_v(v)},\, g_{a|z_s^\star}\big)\big] \le L\, \mathbb{E}_{v \sim \mathcal{D}_t}\left[\|f_v(v) - z_s^\star\|\right].$$

Proof. By A1, for any z, z',

$$d_{\mathrm{TV}}(g_{a|z}, g_{a|z'}) \le L\|z - z'\|.$$

Taking $z = f_v(v)$ and $z' = z_s^\star$, and then taking expectation over $v \sim \mathcal{D}_t$ yields the desired inequality. □

Interpretation. Performance degradation under viewpoint perturbations is controlled by the drift of visual representations. If Z_t deviates significantly from Z_s, the policy shifts accordingly.

Theorem 2 (Existence of affine corrections (justification of FTM)). *Suppose A1 and A3 hold, and that there exists an affine map $A(z) = Mz + b$ (e.g., $M = \mathrm{diag}(1+\gamma)$ and $b = \beta$ as in FTM) such that*

$$\mathbb{E}_{v\sim\mathcal{D}_t}\|Mf_v(v) + b - z_s^\star\|^2 \le \epsilon^2.$$

Then the post-adaptation policy satisfies

$$\mathbb{E}_{v\sim\mathcal{D}_t} d_{\mathrm{TV}}\big(g_{a|A(f_v(v))},\, g_{a|z_s^\star}\big) \le L\epsilon.$$

Proof. Apply A1 with $z' = z_s^\star$:

$$d_{\mathrm{TV}}\big(g_{a|A(z)},\, g_{a|z_s^\star}\big) \le L\,\|A(z) - z_s^\star\|.$$

Taking $z = f_v(v)$ and expectation over $v \sim \mathcal{D}_t$ gives

$$\begin{aligned}\mathbb{E}d_{\mathrm{TV}} &\le L\,\mathbb{E}\|A(f_v(v)) - z_s^\star\| \\ &\le L\sqrt{\mathbb{E}\|A(f_v(v)) - z_s^\star\|^2} \\ &\le L\epsilon\end{aligned}$$

where we used Jensen's inequality. □

Interpretation. If target-domain tokens lie within an affine transformation of source-domain tokens, then FTM—which implements a per-channel affine transform—can provably restore the original policy up to error $\mathcal{O}(\epsilon)$.

Theorem 3 (Low-rank corrections approximate optimal linear shift (justification of FLA)). *Let $\Delta W^\star$ denote the optimal linear correction to a vision layer (in any fixed Frobenius-norm sense). Let its singular values be $\sigma_1 \ge \cdots \ge \sigma_d$. Let ΔW_r be the best rank-r approximation of $\Delta W^\star$. Then by the Eckart–Young theorem,*

$$\|\Delta W^\star - \Delta W_r\|_F^2 = \sum_{i=r+1}^{d} \sigma_i^2.$$

If the representation drift is corrected using ΔW_r via

$$W' = W + \Delta W_r = W + BA,$$

then under A1,

$$d_{\mathrm{TV}}\big(g_{a|z'},\, g_{a|z_s^\star}\big) \le L\,\|z' - z_s^\star\|,$$

and the residual representation error is controlled by the tail energy $\sum_{i>r}\sigma_i^2$.

Proof. The Eckart–Young theorem states that ΔW_r is the optimal rank-r approximation of $\Delta W^\star$ under the Frobenius norm and yields exactly the stated error. This error propagates linearly through the affected network layer, producing a representation error proportional to $\|\Delta W^\star - \Delta W_r\|_F$. Applying A1 then bounds the resulting deviation in policy outputs. □

Interpretation. If the optimal visual correction $\Delta W^\star$ is (approximately) low-rank, then FLA—which imposes a rank-r structure—recovers the correction with error proportional to the spectrum tail. This explains why FLA achieves performance comparable to full LoRA with orders of magnitude fewer parameters.

8.4. Combined Error Bound

Combining Theorems 1–3, for any adaptation module A_ϕ (affine or low-rank), we obtain:

$$\mathbb{E}_{v\sim\mathcal{D}_t} d_{\mathrm{TV}}\big(g_{a|A_\phi(f_v(v))},\, g_{a|z_s^\star}\big) \le L\,\mathbb{E}\|A_\phi(f_v(v)) - z_s^\star\|.$$

Thus, the goal of adaptation is to minimize the right-hand side. Under A3, small-parameter modules such as FTM or FLA are expressive enough to achieve this, thereby restoring the source-domain policy.

9. Qualitative Analysis: Visual Embedding Alignment

To empirically validate our hypothesis that performance degradation under novel viewpoints stems from spatial misalignment, we visualized the distribution of visual tokens before and after adaptation. We extracted visual feature tokens from the SigLIP encoder using trajectories collected from two distinct domains: the source domain (original training camera) and the target domain (novel camera viewpoint). We employed t-Distributed Stochastic Neighbor Embedding (t-SNE) to project these high-dimensional embeddings into a 2D space, as shown in Figure 10.

In the Zero-Shot setting (Figure 10(a)), we observe a severe **embedding drift**. The visual tokens from the novel viewpoint (red) form a cluster that is completely isolated from the source domain manifold (blue). This significant domain gap explains the catastrophic failure of the pretrained policy, as the frozen physical modeling module receives inputs that lie outside its valid operating region.

In contrast, after applying our one-shot **Feature Linear Adaptation (FLA)** (Figure 10(b)), the target distribution (green) undergoes a structured transformation. While preserving its internal geometric consistency, the target manifold is projected to closely adjoin the source manifold. This **manifold alignment** effectively bridges the domain gap, creating a continuous latent space that enables the frozen action expert to generalize to the new viewpoint without requiring full parameter finetuning.

10. Benchmark Details

The original LIBERO benchmark [21] assesses knowledge transfer in multitask and lifelong robot learning for manipulation tasks. It consists of four task suites, and each suite contains 10 tasks.

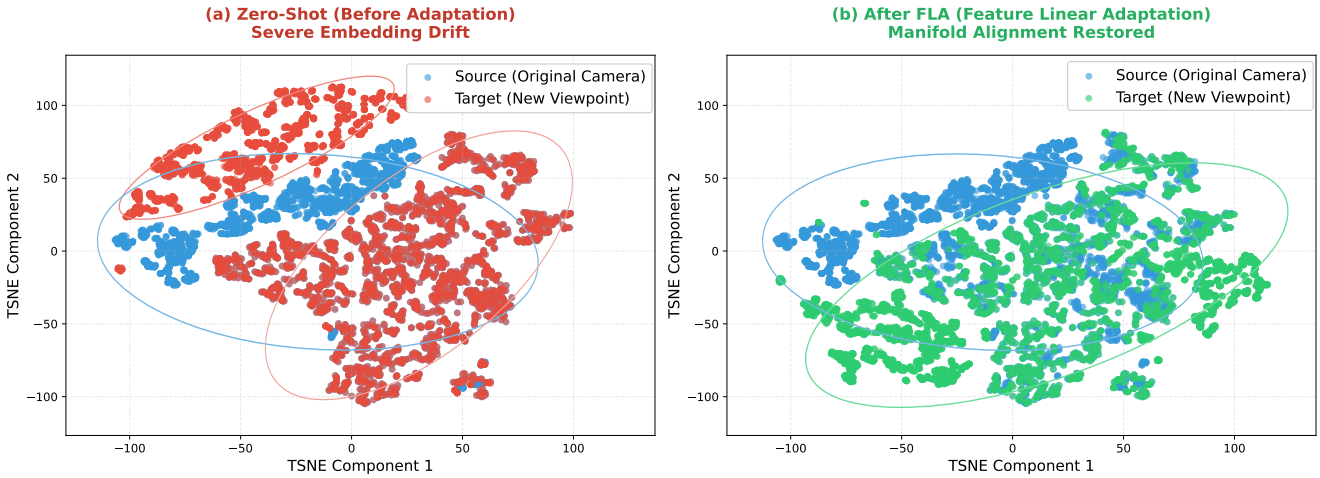


Figure 10. **Visualization of Visual Token Embeddings via t-SNE.** (a) **Before Adaptation:** A significant domain gap is observed between the source (blue) and target (red) embeddings, indicating severe spatial misalignment caused by viewpoint shifts. (b) **After FLA:** Our method projects the target embeddings (green) to align with the source manifold. This *manifold alignment* restores the connectivity of the feature space, allowing the frozen policy to function correctly without requiring the distributions to perfectly overlap.

LIBERO-Spatial, which tests generalization to novel spatial layouts while keeping task and object types fixed;

LIBERO-Goal, which introduces new tasks while maintaining the same objects and layouts;

LIBERO-Object, which evaluates performance with novel object types under the same tasks and layouts; and

LIBERO-Long, which presents a broader variety of objects, layouts, and backgrounds.

10.1. Robustness Extensions

To rigorously evaluate the robustness of Vision-Language-Action (VLA) models against New Spatial Domains shifts beyond the original scope, we extended the standard LIBERO benchmark by implementing a multi-axis perturbation engine. Our implementation draws upon the evaluation protocols defined in Adapt3R [31] for geometric shifts and LIBERO-Plus [8] for visual domain randomization. The benchmark systematically introduces variations across four distinct dimensions: Camera Pose, Illumination, Background Texture, and Sensor Noise.

10.1.1. Camera Pose Variations

Following the methodology in Adapt3R [31], we assess geometric robustness by altering the extrinsic parameters of the primary observation camera (referred to as *agentview*). We implemented a custom `ControlEnv` wrapper that manipulates the camera's spatial position and orientation quaternions within the MuJoCo physics state during environment initialization. We support two variation modes:

Continuous Orbital Rotation. To test viewpoint invariance, we generate camera poses that orbit the robot's workspace. Let $\mathbf{p}_{cam}$ be the initial camera position and $\mathbf{p}_{eef}$ be the position of the robot's end-effector. We define a rotation radius $r = \|\mathbf{p}_{cam}^{xy} - \mathbf{p}_{eef}^{xy}\|_2$. Given a perturbation angle θ, the new position $\mathbf{p}'_{cam}$ is computed as:

$$\mathbf{p}'_{cam} = \begin{bmatrix} r\cos(\theta) + \mathbf{p}_{eef}^{x} \\ r\sin(\theta) + \mathbf{p}_{eef}^{y} \\ \mathbf{p}_{cam}^{z} \end{bmatrix}$$

The orientation is updated by composing the base rotation R_{ref} with a Z-axis rotation $R_z(\theta)$ to ensure the camera remains focused on the workspace center.

Discrete Perturbations. For standardized New Spatial Domains evaluation, we defined three discrete difficulty levels (*Small*, *Medium*, *Large*). These apply fixed translation offsets $\Delta\mathbf{p}$ and specific quaternion rotations to simulate progressively severe sensor misalignments.

Fig. 11 shows the success rate of different viewpoints gained by discrete translation. Our method can efficiently adapt to camera variations across scales (e.g., large).

10.1.2. Illumination Perturbations

To evaluate robustness against illumination variations, we leverage the procedural scene generation pipeline from LIBERO-Plus [8]. In contrast to standard 2D image-space augmentations, we procedurally modify the internal MuJoCo scene configurations to introduce physically grounded lighting changes.

- **Implementation:** We constructed a comprehensive library of scene definitions featuring diverse illumination profiles. To facilitate systematic evaluation, we implemented specialized environment classes that seamlessly load these pre-generated configurations.

- **Variation Factors:** We manipulate four attributes: (1) **Diffuse Color** (RGB intensity), (2) **Light Direction** (vector $\mathbf{d} \in \mathbb{R}^3$), (3) **Specular Intensity** (surface highlights), and (4) **Cast Shadows** (toggling shadow rendering for occlusions).

10.1.3. Background Texture Perturbations

We implemented a high-fidelity texture substitution system to test robustness against semantic visual shifts. Unlike simple color jittering, we utilize Physically Based Rendering (PBR) assets.

- **Asset Library:** We utilized a curated dataset of high-resolution textures (up to 4K/6K) categorized into Natural (wood, stone), Industrial (metal, ceramic), and Fabric materials.
- **Dynamic Injection:** Upon environment initialization, we dynamically rebind the texture assets associated with the environmental boundaries (i.e., the floor and walls) within the simulation tree. This mechanism preserves the physical geometry of the scene while introducing significant variation in background visual styles.

10.1.4. Sensor Noise Injection

To simulate real-world camera imperfections, we implemented a real-time image degradation pipeline within the environment's `step()` function. This process intercepts the rendered RGB observation before it reaches the policy. We include five noise types with 10 severity levels each:

1. **Motion Blur:** Applies directional convolution kernels to emulate the visual artifacts resulting from high-speed camera dynamics or robot movement.
2. **Gaussian Blur:** Applies a Gaussian filter ($\sigma \in [1, 10]$) to simulate defocus.
3. **Zoom Blur:** Averages scaled image crops to simulate focal length changes.
4. **Fog:** Blends plasma fractal noise to simulate atmospheric scattering and contrast loss.
5. **Glass Blur:** Combines local pixel shuffling with Gaussian smoothing to simulate textured occlusions.

Table 6 summarizes the implementation scope for each perturbation dimension.

11. Model Details

We provide a rigorous description of the $\pi_{0.5}$ formulation, detailing the probabilistic decomposition, the derivation of the flow matching objective, and the specific architectural mechanisms used for timestep conditioning.

11.1. Probabilistic Formulation and Architecture

The $\pi_{0.5}$ model is designed to capture a joint distribution over high-level semantic outputs $\hat{l}$ (e.g., subtask descriptions, VQA answers) and low-level continuous action trajectories $a_{t:t+H}$, conditioned on multimodal observations

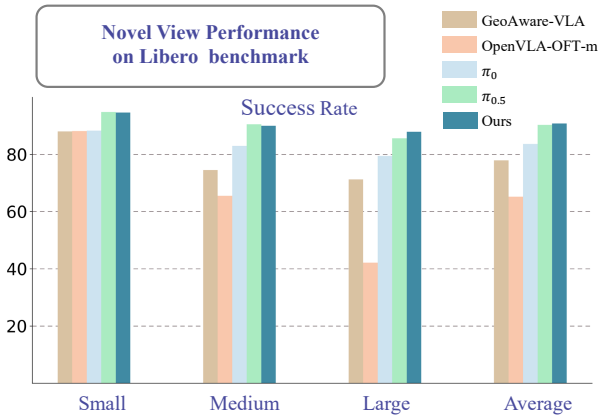


Figure 11. Success rates under Small, Medium, and Large camera viewpoint shifts, showing the stability and robustness of our adapted policy.

o_t and task instructions l. $\pi_{0.5}$ decomposes this joint distribution into a hierarchical inference process:

$$\pi_\theta(a_{t:t+H}, \hat{l} | o_t, l) = \underbrace{\pi_\theta(a_{t:t+H} | o_t, \hat{l})}_{\text{Action Expert (Flow)}} \cdot \underbrace{\pi_\theta(\hat{l} | o_t, l)}_{\text{VLM Backbone (Discrete)}}$$

This decomposition motivates our dual-stream architecture:

- **High-Level Policy ($\pi_\theta(\hat{l}|\cdot)$):** Parameterized by the PaliGemma 2B VLM backbone, optimized via cross-entropy to reason about subtasks and scene semantics.
- **Low-Level Policy ($\pi_\theta(a|\cdot)$):** Parameterized by the Action Expert, optimized via flow matching to generate precise control signals conditioned on the backbone's embeddings.

11.2. VLM Backbone Implementation

The backbone processes a unified sequence of tokens $x_{1:N}$ including images, text, and discrete proprioception.

- **Architecture:** $\pi_{0.5}$ utilize the PaliGemma architecture (Width=2048, Depth=18, Heads=18, MLP=16384).
- **Bidirectional Attention:** Unlike standard causal masking $M_{ij} = \mathbb{I}(j \leq i)$, $\pi_{0.5}$ employs bidirectional attention for the prefix sequence (images I_t and prompt l) to ensure full context visibility: $A(x_i, x_j) = 1$ for all $x_i, x_j \in$ Prefix.
- **Discrete Objective:** The discrete loss $\mathcal{L}_{\text{discrete}}$ is the standard negative log-likelihood over the target tokens:

$$\mathcal{L}_{\text{discrete}} = -\sum_{m=1}^{M} \log P_\theta(x_m | x_{1:m-1}, o_t, l)$$

This objective jointly supervises semantic text generation and discrete action prediction (via FAST tokens).

Table 6. Summary of the Robustness Benchmark Implementation.

Dimension	Implementation Method	Key Parameters
Camera Pose	Direct MuJoCo State Manipulation	Orbit θ, Offset $\Delta\mathbf{p}$, Quaternion q
Lighting	Procedural XML Generation	Diffuse, Specular, Direction, Shadows
Texture	Dynamic Asset Swapping	PBR Materials (Wood, Stone, Metal)
Sensor Noise	Real-time Post-processing	Blur (Motion/Gaussian/Zoom), Fog, Noise

11.3. Flow Matching Objective

To model the continuous action distribution $\pi_\theta(a_{t:t+H}|\cdot)$, $\pi_{0.5}$ employ Conditional Flow Matching (CFM).

Interpolation and Vector Field: $\pi_{0.5}$ defines a probability path that transforms a Gaussian noise distribution $p_0(\omega) = \mathcal{N}(\omega; 0, I)$ to the data distribution $p_1(a)$ (robot actions). $\pi_{0.5}$ constructs intermediate samples a^τ at timestep $\tau \in [0, 1]$ using linear interpolation between noise ω and data $a_{t:t+H}$:

$$a^\tau = \psi_t(\omega, a_{t:t+H}) = \tau a_{t:t+H} + (1-\tau)\omega$$

The objective is to learn a time-dependent vector field $v_t(a^\tau)$ that generates this flow. By differentiating the interpolation path with respect to τ, $\pi_{0.5}$ derives the target vector field:

$$u_t(a^\tau|a_{t:t+H}, \omega) = \frac{d}{d\tau}(\tau a_{t:t+H} + (1-\tau)\omega) = a_{t:t+H} - \omega$$

Note: Following the implementation in $\pi_{0.5}$, the network targets the direction $\omega - a_{t:t+H}$ (flow from data to noise) or vice versa. The loss function minimizes the error between the predicted field f_θ^a and this target.

Flow Matching Loss: The Action Expert is trained to regress this vector field:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\tau\sim p(\tau),\omega\sim\mathcal{N}(0,I),a\sim\mathcal{D}}\Big[||(\omega - a_{t:t+H}) - f_\theta^a(a^\tau, o_t, l, \tau)||^2\Big]$$

where the timestep τ is sampled from a Beta distribution $\text{Beta}(1.5, 1)$ to emphasize learning at low-noise levels ($s = 0.999$).

Inference (Integration): At inference time, $\pi_{0.5}$ approximates the integral of the predicted vector field using an ODE solver (Euler method) with $K = 10$ steps:

$$a^{\tau+\Delta\tau} \leftarrow a^\tau + \Delta\tau \cdot f_\theta^a(a^\tau, o_t, \hat{l}, \tau)$$

11.4. Action Expert Conditioning Mechanism

The Action Expert (300M parameters) is a Transformer distinct from the backbone. To strictly condition the continuous generation on the flow timestep τ, $\pi_{0.5}$ utilizes Adaptive RMSNorm.

Timestep Injection: The scalar timestep τ is first embedded via sinusoidal positional encodings $\phi(\tau)$ and an MLP W. This embedding modulates the normalization layer of each transformer block:

$$\text{AdaRMSNorm}(x, \tau) = y \cdot (1 + \gamma(\tau)) + \beta(\tau),$$
$$\text{where } y = \frac{x}{\|x\|_2}$$

Here, $\gamma(\tau)$ and $\beta(\tau)$ are learned scale and shift parameters projected from the timestep embedding. This ensures that temporal information is injected globally into the expert's feature space.

11.5. Attention Masking Strategy

To enable the joint optimization objective $\mathcal{L} = \mathcal{L}_{\text{discrete}} + \alpha\mathcal{L}_{\text{flow}}$, $\pi_{0.5}$ employs a structured attention mask $M \in \{0, 1\}^{N\times N}$:

1. **VLM $\rightarrow$ Expert:** The Action Expert attends to the VLM's visual and textual embeddings ($M_{expert,vlm} = 1$).
2. **Expert $\nrightarrow$ VLM:** The VLM backbone is prevented from attending to the Action Expert ($M_{vlm,expert} = 0$) to preserve the pre-trained semantic representations.
3. **Information Barrier:** Crucially, the Action Expert is masked from the discrete FAST action tokens ($M_{expert,FAST} = 0$). This conditional independence assertion forces the expert to infer $a_{t:t+H}$ solely from o_t and l, preventing information leakage from the discrete solution.

12. Training Details for baselines

In this section, we detail the training configurations for all baselines and our proposed methods (FTM and FLA). The evaluated baselines cover a broad spectrum of adaptation strategies, including parameter-efficient fine-tuning (LoRA), prompt learning, and visual backbone replacement. To ensure fair comparisons, all experiments utilize the same pre-trained checkpoints and evaluation protocols.

Before detailing specific hyperparameters, we analyze the training dynamics. As shown in Fig. 3, we compare the one-shot adaptation performance of our Feature Linear Adaptation (FLA) against LoRA and full fine-tuning across π_0 and $\pi_{0.5}$ architectures. FLA achieves faster convergence

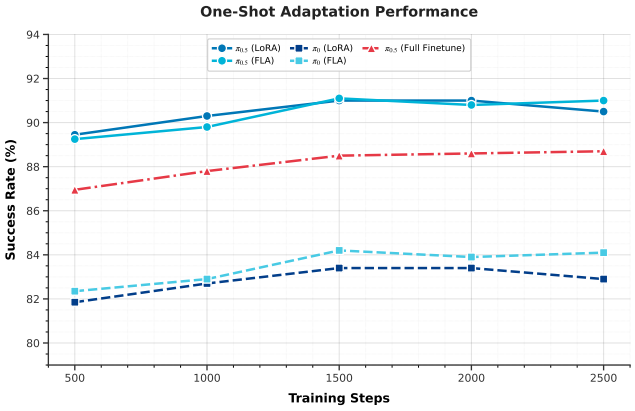


Figure 12. **Adaptation Stability.** Success rates across training steps. FLA remains stable and consistently outperforms baselines without overfitting.

and greater stability across training steps without overfitting. Notably, FLA reaches comparable or slightly superior peak success rates relative to LoRA, and consistently outperforms full fine-tuning, while updating only a fraction of the parameters.

We now outline the specific optimization and preprocessing configurations, beginning with the $\pi_{0.5}$ LoRA baseline.

$\pi_{0.5}$ LoRA Baseline

We fine-tune the $\pi_{0.5}$ backbone equipped with the pi05 action head using our custom dataset as shown in Fig. 3(a).

1. Model Configuration: To ensure parameter efficiency, we adopt Low-Rank Adaptation (LoRA) while keeping all non-LoRA weights frozen via the model-provided freeze filter. Specifically, we configure the VLM component using the Gemma 2B LoRA variant with a LoRA rank of 16, and the action expert using the Gemma 300M LoRA variant with a LoRA rank of 32, applying adaptation to both attention and feed-forward network (FFN) layers.

2. Optimization: The optimization process strictly adheres to the default VLA pretraining protocols. We utilize the AdamW optimizer with momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, a weight decay of 1×10^{-10}, and a global gradient norm clipping threshold of 1.0. The learning rate follows a cosine warmup schedule (1,000-step warmup), reaching a peak of 2.5×10^{-5} and decaying to a minimum of 2.5×10^{-6}.

3. Data Preprocessing: Visual observations are resized to 224×224 using a padding strategy (Resize with Pad).

Alternative Visual Backbones Baseline

To evaluate the efficacy of replacing the visual encoder with a robust geometric backbone, we implement a baseline following the GeoAware-VLA architecture as shown in Fig. 3(b).

1. Architecture: The standard visual encoder is replaced by a pre-trained Visual Geometry Grounded Transformer (VGGT), which serves as a frozen feature extractor. To align features, we introduce a trainable Feature Projection Layer that processes features from four evenly spaced intermediate layers via a 1D convolutional network, adaptive average pooling, and a final MLP projector to produce the visual embedding z_{vis}.

2. Optimization: The policy (excluding the frozen vision backbone) is trained using the AdamW optimizer with standard momentum parameters ($\beta_1 = 0.9, \beta_2 = 0.999$) and a weight decay of 1×10^{-4}. We employ a constant learning rate of 1×10^{-4} and a global batch size of 64.

3. Data Preprocessing: Visual observations are rendered at a resolution of 128×128 and subsequently center-cropped to 126×126 pixels to match the input requirements.

Prompt-based Adaptation Baseline

As illustrated in Fig. 3(c), we adopt a standard prompt tuning strategy by introducing learnable embedding tokens into the input sequence to implement the prompt learning baseline.

1. Mechanism: We instantiate learnable prompt tokens (initialized from a normal distribution with $\sigma = 0.02$) that are directly concatenated to the model inputs. These are divided into two groups: Prefix Prompts, which match the embedding width of the frozen PaliGemma encoder and are concatenated to the multimodal (image and language) sequence; and Suffix Prompts, which align with the action expert's width and are inserted between the proprioceptive state embedding and the action tokens. Unlike modulation-based approaches (e.g., FiLM), no auxiliary projection networks or affine transformations are applied. These tokens function purely as additional learnable context within the Transformer's attention mechanism. Consistent with the prompt tuning paradigm, the pre-trained backbone remains entirely frozen; optimization is restricted exclusively to the learnable prompt tokens and the action head.

2. Optimization: We utilize the AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-8}$, weight decay 1×10^{-10}) with a global gradient norm clipping of 1.0. Distinct from the LoRA baseline, the learning rate follows a schedule tuned for adaptation: a cosine schedule with a 500-step warmup, reaching a peak of 5×10^{-4} and decaying to 5×10^{-5} over a total of 2,000 iterations (batch size 32).

3. Data Preprocessing: Training is conducted on our custom dataset. Visual inputs undergo the same preprocessing pipeline as the LoRA baseline: frames are resized to 224×224 with padding.

Training Details for Our Solutions

We introduce two complementary, parameter-efficient

Table 7. Hyperparameters for our proposed adaptation methods (Feature Token Modulation and Feature Linear Adaptation).

Configuration	Feature Token Modulation (FTM)	Feature Linear Adaptation (FLA)
Optimizer	AdamW, clip norm 1.0	AdamW, clip norm 1.0
Optimizer Momentum	$\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 1e-8$	$\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 1e-8$
Weight Decay	1×10^{-10}	1×10^{-10}
Batch Size	32	32
Training Iterations	5,000 steps	1,500 steps
Cosine Warmup Steps	500	500
Peak Learning Rate	5×10^{-4}	5×10^{-4}
Cosine Decay Steps	5,000	2,000
Min LR (Decay Target)	5×10^{-5}	5×10^{-6}
Image Resize	224×224	224×224

adaptation strategies: **Feature Token Modulation (FTM)** and **Feature Linear Adaptation (FLA)**. Both methods focus on adapting pre-trained representations while keeping the majority of the VLA backbone frozen. Detailed hyperparameters are provided in Table 7. As illustrated in Fig. 12, FLA demonstrates strong training stability. The success rate consistently improves and plateaus across training steps without suffering from catastrophic overfitting, confirming its robustness to hyperparameter choices.

1. Feature Token Modulation (FTM): As illustrated in Fig. 3(d), FTM applies a global affine transformation to the visual embedding space. We instantiate two learnable tokens, global scale (γ) and shift (β) parameters. The VLA backbone is entirely frozen; we optimize *only* the learnable prompts. Optimization utilizes the AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 10^{-8}$, weight decay 1×10^{-10}) with a global gradient norm clipping of 1.0. The learning rate follows a cosine schedule with a 500-step warmup, reaching a peak of 5×10^{-4} and decaying to 5×10^{-5} over 5,000 iterations. The batch size is set to 32.

2. Feature Linear Adaptation (FLA): As shown in Fig. 3(e), FLA targets internal feature adaptation by injecting LoRA adapters ($\Delta W = BA$) into the linear layers (27 layers) of the SigLIP ViT encoder. We freeze the language model and action expert, updating only the LoRA weights of linear layer in ViT. Optimization settings generally mirror the FTM configuration (AdamW with consistent β and ϵ values), but the learning rate schedule is adjusted to decay to a lower minimum of 5×10^{-6}, and EMA is disabled. Training is conducted with float32 precision for 1,500 iterations using a global batch size of 32.

3. Common Preprocessing: For both strategies, visual inputs undergo the standard preprocessing pipeline: images are resized to 224×224.

Table 8. Success Rate (%) of FLA across different VLA models under novel camera viewpoints.

Model	Zero-shot	Rank=8	Rank=16	Rank=32	Rank=64
OpenVLA-OFT (One-Shot FLA)	50.3	89.0	88.0	88.0	85.6
π_0 (One-Shot FLA)	49.5	82.8	84.0	84.5	84.8
$\pi_{0.5}$ (One-Shot FLA)	48.5	91.05	90.8	91.2	90.6

13. Additional Ablation Studies on Adaptation Rank

In this section, we provide extended ablation results regarding the adaptation rank (r) used in our Feature Linear Adaptation (FLA) method. Table 8 presents the one-shot adaptation success rates across three distinct base architectures (OpenVLA-OFT, π_0, and $\pi_{0.5}$) under novel camera viewpoints. The results indicate that while higher ranks generally offer increased capacity for feature realignment, even modest ranks (e.g., $r = 16$) achieve highly competitive robustness with minimal parameter overhead.