SPARSE FEATURE ANALYSIS AND CONTROL OF VISION-LANGUAGE-ACTION MODELS UNDER VISUAL PERTURBATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have demonstrated strong manipulation performance, yet often degrade under visual perturbations such as camera, background-texture, lighting, and sensor-noise shifts. We ask which internal features make a VLA visually sensitive, what these features represent, and whether intervening on them produces the predicted change in behavior. To obtain localized analysis units, we decompose raw VLA activations using sparse autoencoders (SAEs), and introduce a shared-state paired perturbation protocol that isolates feature responses to visual shifts while controlling for physical state divergence. We then identify *compensator features*, which serve as action-coupled steering targets, and *detector features*, which predict perturbation-induced failures. Across autoregressive and diffusion-based VLA architectures, sparse features provide more localized perturbation-relevant units than raw residual dimensions, and the two roles form disjoint populations: detectors in the vision-language stream or early layers and compensators in the action expert or late layers. By relating features to the policy's own action modes, we find that compensators encode task-invariant motion primitives and that a perturbation weakens the policy's commitment to the primitive it is executing rather than switching it to another. Finally, we verify the interpretation by intervention: steering a single compensator feature shifts perturbed actions back toward the unperturbed action and modestly improves rollout success, and gating it with a detector makes it perturbation-specific. Together, these results identify, interpret, and verify the features behind visual sensitivity in VLAs.

1 INTRODUCTION

Vision-Language-Action (VLA) models (Kim et al., 2024; 2025; Brohan et al., 2023; Octo Model Team et al., 2024; Black et al., 2024; 2025) are a promising route toward generalist robot policies: they fine-tune pre-trained vision-language model (VLM) backbones Team et al. (2024); Bai et al. (2025); Wang et al. (2024) on demonstration data, inheriting broad semantic grounding while learning to emit robot actions. Although VLA models perform well in trained environments and tasks, they often degrade under visual perturbations such as camera, background-texture, sensor-noise, and lighting-condition shifts; recent studies further suggest that such performance may rely partly on memorization rather than robust task understanding (Fei et al., 2025; Zhou et al., 2025). Understanding these limitations requires interpreting how VLA models internally respond to perturbations. This motivates a central question: *which internal features make a VLA policy visually sensitive, what do they represent, and does intervening on them change behavior as predicted?*

However, analyzing perturbation-sensitive representations in VLA models is challenging for several reasons. First, VLA policies are intrinsically composite systems, coupling vision encoders, language backbones, and model-specific action heads. Second, the field lacks a unified architectural template: autoregressive VLA models (Kim et al., 2024; 2025; Brohan et al., 2023) and diffusion-based VLA models (Octo Model Team et al., 2024; Black et al., 2025; Liu et al., 2025; Chi et al., 2023) expose different internal representations and action-generation mechanisms. Third, robustness-relevant signals are often distributed across raw residual dimensions, making it difficult to localize features that respond to visual perturbations, relate to action changes, or predict failure-prone rollouts.

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.

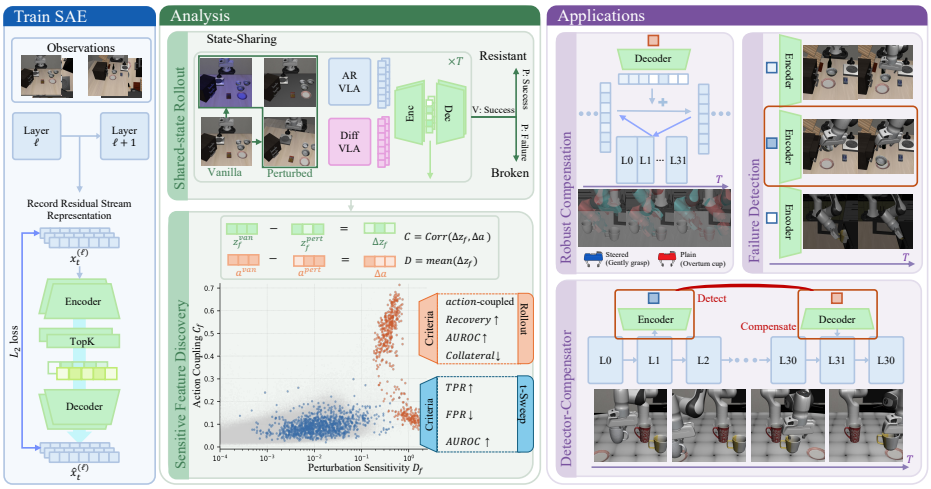


Figure 1: Overview. We train layer-wise SAEs and use shared-state paired perturbations to identify perturbation-sensitive features as compensators and detectors, interpret them through the policy's action modes, and verify the interpretation by steering, failure detection, and detector-gated steering under perturbations.

Sparse autoencoders (SAEs) provide a promising mechanism for decomposing dense neural activations into sparse feature representations that can serve as more localized analysis units (Gao et al., 2025; Bussmann et al., 2024; Rajamanoharan et al., 2024a;b; Bussmann et al., 2025). Recent work has shown that SAEs can reveal meaningful internal representations in vision (Kim et al., 2026; Lim et al., 2025; Stevens et al., 2025), language (Cunningham et al., 2024; Bricken et al., 2023; Templeton et al., 2024), and vision-language models (Pach et al., 2025; Zhang et al., 2025; Park et al., 2026). Yet VLA models pose a distinct challenge: their representations do not merely support perception or recognition, but are used inside policies that produce temporally extended robot behavior. Interpreting VLA policies with SAEs thus requires more than extracting meaningful features: they must be tested for whether they are perturbation-sensitive, coupled to action changes, predictive of rollout outcomes, and whether intervening on them changes behavior as predicted.

To this end, we introduce a pipeline that identifies, interprets, and experimentally verifies perturbation-sensitive features in VLA models. We begin by training layer-wise TopK SAEs (Gao et al., 2025) on internal VLA activations to obtain sparse feature representations that serve as localized analysis units. Rather than treating sparse features as static semantic units, we use them as behavior-linked variables for studying failures under visual distribution shift.

Given these sparse features, we propose a shared-state paired perturbation protocol that compares base and perturbed observations rendered from the same underlying physical state. This allows feature differences to be attributed to visual perturbations rather than trajectory divergence. Using this protocol, we score features by perturbation sensitivity, action coupling, and Broken-Resistant separability. We then categorize perturbation-sensitive features into two behaviorally relevant roles: compensator features, which are action-coupled features used as plausible steering targets, and detector features, which identify perturbation-induced failure-prone regimes.

Our contributions are as follows:

1. **Identification.** We propose a shared-state paired perturbation protocol and feature scores that identify compensator features and detector features in both autoregressive and diffusion-based VLAs, and show that the two roles form disjoint populations: detectors in the vision-language stream or early layers, compensators in the action expert or late layers.

2. **Interpretation.** Relating features to the policy's own action modes, we show that compensator features encode task-invariant motion primitives and that a visual perturbation weakens the policy's commitment to the primitive it is executing rather than switching it.

3. **Verification by intervention.** Steering a single compensator feature shifts perturbed actions back toward the unperturbed action and recovers more perturbed rollouts than control features, and gating it with a detector makes the intervention perturbation-specific.

Together, these results turn sparse feature analysis of VLA policies from post-hoc inspection into a pipeline whose interpretations are tested by intervention.

2 Related Work

Sparse autoencoders and mechanistic interpretability. Motivated by superposition in dense activations (Elhage et al., 2022), sparse autoencoders (SAEs) decompose neural representations into sparse and potentially interpretable features (Bricken et al., 2023). Early work introduced SAEs for residual-stream decomposition in language models (Cunningham et al., 2024; Bricken et al., 2023), while later work improved scalability with TopK sparse coding (Gao et al., 2025). SAE features have since been used to study feature-level computation and circuits in vision transformers (Kim et al., 2026) and language models (Marks et al., 2025), revealing structure hard to isolate from dense activations. More broadly, mechanistic interpretability and probing methods characterize internal representations through feature attribution, probes, or intermediate decoding (Belinkov, 2022; Belrose et al., 2023; Vig et al., 2020). However, applying them to VLA models is complicated by multimodal fusion, action-conditioned representations, and heterogeneous architectures.

Vision-language-action models and their interpretability. Visuomotor policies and pretrained visual representations laid the foundation for learning manipulation from demonstrations (Nair et al., 2022; Karamcheti et al., 2023; Zhao et al., 2023). Recent vision-language-action (VLA) models are largely divided into autoregressive VLAs (AR-VLA), which predict discretized action tokens that are detokenized into continuous trajectories (Kim et al., 2024; 2025; Brohan et al., 2023; Octo Model Team et al., 2024), and diffusion-based VLAs (Diff-VLA), which refine actions through denoising or flow matching conditioned on vision-language representations (Black et al., 2024; 2025; Liu et al., 2025; Chi et al., 2023). Despite strong manipulation performance, VLAs can degrade under distribution shifts and may partly rely on memorization rather than robust task understanding (Fei et al., 2025; Zhou et al., 2025). Interpretability for VLA policies is still emerging, with studies identifying motion-related neurons, sparse steering directions, and physically grounded behavior steering (Häon et al., 2025; Khan et al., 2025). Most closely related to ours, an SAE-based study shows that sparse VLA features capture memorized demonstrations and motion primitives, and are steerable through interventions (Swann et al., 2026). We build on this direction but ask where perturbation sensitivity is represented, what the sensitive features encode in terms of the policy's own actions, and whether intervening on them changes behavior as predicted.

3 Method

In this section, we describe the three parts of our pipeline. We identify perturbation-sensitive SAE features with a shared-state paired protocol and feature scores (§3.2 and 3.3), interpret them with an action-mode vocabulary (§3.4), and verify them with feature steering (§3.5).

3.1 Formulations

Task and representation formulation. We consider a language-conditioned robot policy under partial observability. At timestep $t \in \{0, 1, \ldots, T-1\}$, the robot receives a multimodal observation $o_t \in \mathcal{O}$ such as RGB observations and proprioception, and a language instruction $\mathcal{G}$, producing an action chunk $a_t \sim \pi_\theta(\cdot \mid o_t, \mathcal{G})$. The action chunk $a_t \in \mathbb{R}^{H \cdot d_{\text{act}}}$ has horizon H and action dimension d_{act}. We write $a \in \mathbb{R}^{T \times (H \cdot d_{\text{act}})}$ for the collection of $\{a_t\}_{t=0}^{T-1}$.

Sparse feature decomposition for analysis units. Recent work suggests that SAE-derived features can provide useful units for feature-level analysis and steering in VLA models Khan et al. (2025); Swann et al. (2026). Motivated by these findings, we train TopK SAEs Makhzani & Frey (2013); Gao et al. (2025) on layer-wise VLA activations to obtain localized representation-level analysis units.

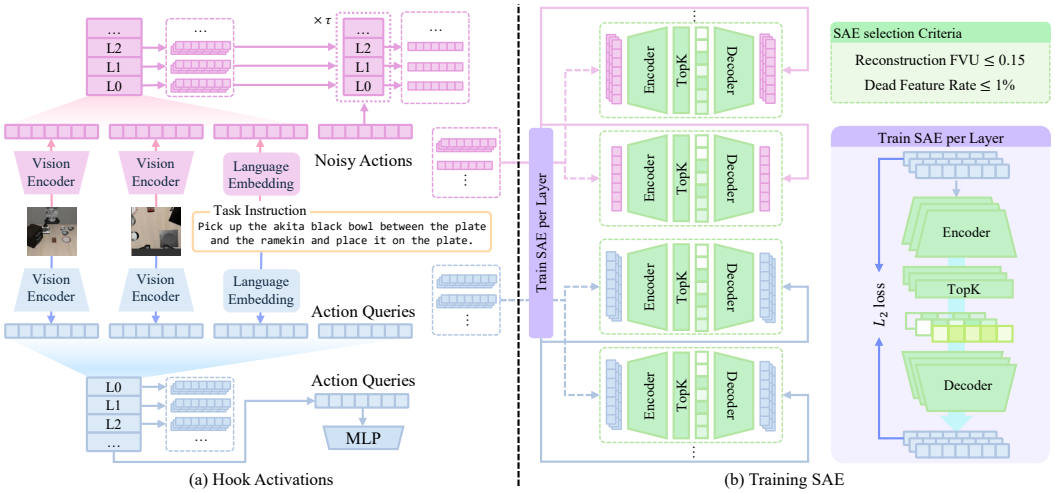


Figure 2: Layer-wise SAE training for VLA representations. We collect internal activations across layers, stabilize massive activations when needed, and train TopK SAEs to obtain sparse feature representations. SAE configurations are selected by reconstruction and dead-feature criteria.

Specifically, we first collect residual-stream activations from each layer of the VLA (Fig. 2(a)), and then train a sparse autoencoder on the collected activations for each layer (Fig. 2(b)).

Given a residual-stream activation $x_t \in \mathbb{R}^d$ at a layer, the TopK SAE encodes it as $z_t = \mathrm{TopK}(W^{\mathrm{enc}}(x_t - b^{\mathrm{pre}}))$ and reconstructs $\hat{x}_t = W^{\mathrm{dec}} z_t + b^{\mathrm{pre}}$, keeping the top-$K$ of d_{sae} features and minimizing $\|x_t - \hat{x}_t\|_2^2$. We retain SAEs with fraction of variance unexplained below 0.15 and dead-feature rate below 1% (Kim et al., 2026), and stabilize training against massive activations (Sun et al., 2024); training and selection details are in App. B. We suppress the layer index ℓ and write $z \in \mathbb{R}^{T \times d_{\mathrm{sae}}}$ for $\{z_t\}_{t=0}^{T-1}$.

3.2 Paired Perturbation Setup

Paired rollouts under base and perturbed conditions can rapidly diverge in physical state even from the same initial condition (Fei et al., 2025), so that feature differences reflect accumulated state changes rather than perturbation sensitivity. To remove this confound, we propose a shared-state paired perturbation protocol. We first execute a reference trajectory under base conditions and record the simulator state at each timestep. Then, for each recorded state s_t, we reset the environment to the same simulator configuration and render paired observations o_t^{base} and o_t^{pert} under base and perturbed conditions. Feature differences are thus measured at a fixed robot and object configuration.

We then compare feature activations between base and perturbed observations. Because perturbation responses may differ by outcome, we separately analyze *Resistant* rollouts, which remain successful under perturbation $(\mathrm{base} = \mathrm{success},\ \mathrm{pert} = \mathrm{success})$, and *Broken* rollouts, which fail only under perturbation $(\mathrm{base} = \mathrm{success},\ \mathrm{pert} = \mathrm{failure})$. Let $\mathcal{I}$ denote the set of paired timesteps.

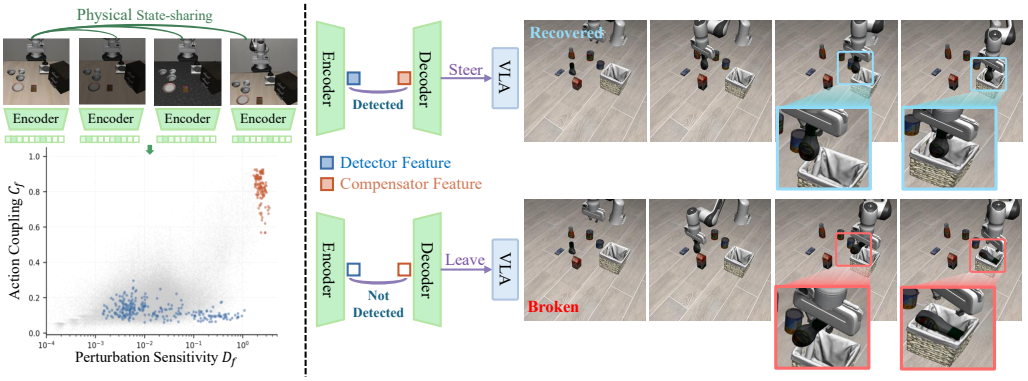
3.3 Feature Scores and Roles

For an SAE feature f, we define the feature shift as $\Delta z := z(o_t^{\mathrm{pert}}) - z(o_t^{\mathrm{base}})$ and the action shift as $\Delta a := a^{\mathrm{pert}} - a^{\mathrm{base}}$. We score each feature by perturbation sensitivity and action coupling:

$$M_f := \mathbb{E}_{t \in \mathcal{I}} \left[|\Delta z_{t,f}| \right], \qquad D_f := \frac{M_f}{\max_{f'} M_{f'} + \epsilon}, \qquad C_f := \max_i \left| \mathrm{Corr}\left(\Delta z_{:,f}, \Delta a_{:,i} \right) \right|. \tag{1}$$

Here, D_f is the feature-wise normalized perturbation response within each perturbation type, and C_f measures the strongest correlation between feature shifts and action shifts across action dimensions. We also compute A_f, the Mann–Whitney AUROC between *Broken* and *Resistant* rollouts, using the paired feature-response scores (Mann & Whitney, 1947; Hanley & McNeil, 1982).

We refer to a feature as *perturbation-sensitive* when its activation changes consistently between base and perturbed observations under the shared-state paired setup. Based on these scores, we categorize



(a) Perturbation Sensitive Feature Discovery (b) Detector-conditioned steering

Figure 3: (a) Perturbation-sensitive feature discovery. (b) Detector-conditioned steering uses a detector feature to gate a compensator feature for perturbation-specific steering.

perturbation-sensitive features into two roles. Compensator candidates are intervention directions for restoring perturbation-degraded behavior, whereas detector candidates respond in *Broken* rollouts.

Compensator candidates. Compensator candidates are perturbation-sensitive features that are also coupled to action changes, making them plausible intervention targets for restoring perturbation-degraded behavior. We rank features by the compensator score $A_f \cdot D_f \cdot C_f$, which combines group separation, perturbation sensitivity, and action coupling. For each perturbation type, we retain the top 0.1% candidates under this score and evaluate them with small-scale intervention rollouts on both *Broken* and *Resistant* initial states. We classify candidates as robust compensators when they recover *Broken* rollouts with limited collateral degradation, and as aggressive compensators when they yield stronger recovery but also degrade *Resistant* rollouts.

Detector candidates. Detector candidates are perturbation-sensitive features whose responses separate *Broken* and *Resistant* rollouts. Their role is diagnostic rather than corrective: they indicate failure-prone regimes rather than serve as steering directions. We rank features by A_f, which directly measures failure-side separation. For each perturbation type, we retain the top 0.1% detector candidates for failure detection and detector-compensator pairing.

3.4 Action Modes

To interpret what the discovered features represent, we describe them in terms of the policy's own actions. We cluster the commanded actions of each policy into a small set of action modes. Each forward pass is summarized by the horizon-mean of its action chunk (translation, rotation, gripper). We fit Ward clustering on forward passes from successful unsteered rollouts and choose the number of modes by bootstrap stability, which gives k=10 for $\pi_{0.5}$ and k=8 for OpenVLA-OFT; every other forward pass is assigned to the nearest centroid (App. J). We measure the mode selectivity of a feature as the AUROC of its activation for forward passes in a mode against all others. We define the *commitment* of a forward pass to a mode as the projection of its normalized action onto the unit centroid direction of that mode: it is large when the action points along the mode and moves far, and near zero when the action is orthogonal to the mode. Changes in commitment are reported as a percentage of the base commitment. Because the modes are clusters of commanded motions that recur across tasks, we also refer to them as motion primitives.

3.5 Feature Steering

After identifying compensator candidates, we test whether sparse features can serve as intervention interfaces rather than only analysis units. For a feature f at layer ℓ, let $W^{\text{dec}}_{:,f}$ denote the corresponding SAE decoder vector. Following prior feature-steering work (Templeton et al., 2024; Swann et al., 2026), we intervene on perturbed rollouts by adding this decoder direction to the residual dimensions: $\tilde{x}_t = x_t + \alpha W^{\text{dec}}_{:,f}$. Here, $\alpha \in \mathbb{R}$ controls the intervention strength and sign: positive values amplify the feature direction, whereas negative values suppress it.



Figure 4: Layer (OpenVLA-OFT, left) and stream and layer ($\pi_{0.5}$, right) of the top-5 detector and compensator candidates per perturbation.

Table 2: Deployed detector feature per perturbation (AUROC and feature id).

Perturbation	OpenVLA-OFT	$\pi_{0.5}$
Camera	0.776 (L15F6110)	0.828 (B:L02F0567)
Light	0.835 (L12F2024)	0.941 (B:L00F1823)
Background	0.876 (L01F3670)	0.813 (B:L00F0501)
Sensor	0.801 (L31F7245)	0.996 (B:L16F0279)

Detector-conditioned steering. Aggressive compensators can recover *Broken* rollouts but may harm *Resistant* ones when applied unconditionally. Thus, we gate aggressive compensators with failure detectors from the same perturbation type, *i.e.*, the compensator is activated only when the detector indicates a failure (Fig. 3(b)). Concretely, for *detector-conditioned steering*, f denotes the compensator feature and we replace α with αg_t, where g_t is a binary gate that takes value 1 when the detector feature is active at timestep t, and 0 otherwise. For each perturbation type, we select the detector-compensator pair based on expected benefit; details are in the supplementary material.

4 Identifying Perturbation-Sensitive Features

In this section, we evaluate two representative VLAs, OpenVLA-OFT[1] and $\pi_{0.5}$[2], in LIBERO (Liu et al., 2023) and LIBERO-Plus (Fei et al., 2025), whose camera, lighting, background-texture, and sensor-noise perturbations change the observation while preserving the task, the physical state, and the instruction. We first ask whether visual sensitivity can be isolated in individual features and where those features lie in the network; §5 and 6 then interpret and verify them. All experimental results are averaged over three trials. Features are named by layer and index; for $\pi_{0.5}$, the prefixes B: and E: denote the base stream and the action expert.

4.1 Comparing SAE Features with Raw Residual Dimensions

We first compare SAE features with raw residual dimensions under the same candidate discovery procedure defined in §3.3. For raw residual dimensions, we treat each residual activation as an individual feature and compute the same detector and compensator scores. For SAE features, we apply the same procedure to sparse feature activations.

Table 1: OpenVLA-OFT raw residual dimensions vs. SAE features.

Perturbation	Det. Top-0.1%		Comp. Top-0.1%	
	Raw	SAE	Raw	SAE
Camera	0.621	**0.729**	0.0077	**0.0399**
Light	0.724	**0.889**	0.0068	**0.0317**
Background	0.614	**0.730**	0.0098	**0.0306**
Sensor	0.700	**0.758**	0.0100	**0.0390**

Results. Table 1 reports the mean AUROC of the top 0.1% detector candidates and the mean compensator score of the top 0.1% compensator candidates. SAE features consistently outperform raw residual dimensions across perturbation types, suggesting that perturbation-relevant signals are more localized in the SAE basis. We therefore use SAE features hereafter.

4.2 Detectors and Compensators Are Disjoint Populations

Detectors and compensators are disjoint. For $\pi_{0.5}$, the top-0.1% detector and compensator candidate sets (62 and 41 features) have no feature in common. The separation follows the architecture: every deployed detector lies in the base stream and every deployed compensator in the action expert (Fig. 4). OpenVLA-OFT has a single stream, and there the split appears along depth: detector candidates come from early to middle layers, and compensator candidates from late layers (L23–L31). Detectors sit upstream, where vision is integrated with language; compensators sit where actions are produced.

[1] https://huggingface.co/moojink/openvla-7b-oft-finetuned-libero-spatial-object-goal-10

[2] OpenPI, config: `pi05_libero`, checkpoint: gs://openpi-assets/checkpoints/pi05_libero

We next examine whether detector features are predictive of rollout failure. For each perturbation category, we evaluate the top-0.1% detector candidates of §3.3 by their AUROC A_f between Broken and Resistant rollouts (App. C); the same rollouts serve for detector-compensator pairing.

Results. As summarized in Table 2, single sparse features can be highly predictive of perturbation-induced failures across both VLA architectures. For OpenVLA-OFT, detector AUROC scores range from 0.776 to 0.876, with the strongest separation under background texture perturbations. For $\pi_{0.5}$, detector AUROC scores range from 0.813 to 0.996, strongest under noise and lighting.

5 Interpreting the Features

Having identified distinct detector and compensator populations, we next ask what these features encode about the policy's behavior. We first examine whether compensators correspond to recurring action primitives, rather than task-specific visual or geometric variables, by relating their activations to the action modes defined in §3.4. We then contrast this behavioral interpretation with what detectors encode about perturbation-induced failures.

Results. Table 3 lists the mode each deployed feature is most selective for. Several compensators are close to single-feature readouts of one mode: E:L12F0801 for "reach forward with an open gripper" (AUROC 0.95), E:L00F0563 for "descend with an open gripper" (0.93), and L23F4722 for a lateral-and-lift primitive (0.98). The mapping does not depend on the task: the per-task AUROC of these features averages 0.95–0.97 and exceeds 0.6 in every base task where it occurs (11 to 38 per feature), which a feature tracking object pose or goal location would not pass. The compensators behave like motion primitives that the action expert executes wherever they are needed. Mode selectivity alone does not make a feature a compensator, since many expert features read out a mode with AUROC near 1 (App. J); what sets a compensator apart is that its mode is the one the perturbation disturbs.

Table 3: Most selective action mode, its AUROC, and commitment change Δc under perturbation (percent of base) per deployed feature.

Feature (role)	Action mode	AUROC	Δc (%)
$\pi_{0.5}$			
E:L12F0801 (comp.)	Reach $+x$ (Open)	0.95	-5%
E:L00F0563 (comp.)	Descend $-z$, (Open)	0.93	-7%
E:L02F1761 (comp.)	Lift $+z$, (Closed)	0.82	-6%
B:L16F0279 (det.)	Lift $+z$, $-y$, (Closed)	0.82	–
OpenVLA-OFT			
L23F4722 (comp.)	$-y$, $+z$ (Open)	0.98	-10%
L30F5719 (comp.)	Lift $+z$ (Open)	0.91	-8%
L15F6110 (det.)	Translate $+y$ (Open)	0.83	–

Perturbations weaken the executed primitive rather than replacing it. The last column of Table 3 restricts to paired states at which the base forward pass is executing the compensator's mode. The perturbed pass usually stays in the same mode: only 10–25% of pairs leave it (1.2 to 2.6 times the exit rate of other modes; App. J). What changes is how strongly the primitive is executed (Fig. 5, left). Commitment along the mode direction decreases in 70–90% of pairs, while commitment along the other mode directions is unchanged (App. J). For $\pi_{0.5}$ under sensor noise, commitment to the lift primitive of E:L02F1761 drops by 6% in 67% of pairs. For OpenVLA-OFT under camera shifts, commitment to the primitive of L23F4722 drops by 10% in 80% of pairs and is unaffected under lighting shifts, where this policy is most robust. A task-irrelevant visual change does not change what the policy does; it weakens its commitment to the primitive, which the compensator encodes.

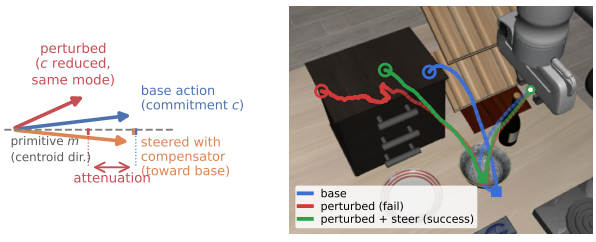


Figure 5: (Left) A perturbation lowers commitment c to the executed primitive, and steering the compensator shifts the action back toward the base action. (Right) End-effector trajectories of one rollout under a camera shift.

What detectors read. At shared states, detectors split into two kinds. The sensor-noise detector of $\pi_{0.5}$ (B:L16F0279) is nearly silent on the base view and turns on under noise (mean activation 0.00 to 0.05; active in 83% to 98% of states), and the size of this response separates Broken from Resistant

Table 4: Robust compensation with steering-basis ablation: success rate and the steered feature.

	OpenVLA-OFT				$\pi_{0.5}$	
Perturbation	Plain	SAE Robust Comp.	SAE Random	Raw Robust Comp.	Plain	SAE Robust Comp.
Camera	50.5%	**50.7%** (L23F4722)	50.6% (L26F6887)	48.5% (L31F1512)	70.8%	**71.7%** (E:L12F0801)
Light	85.3%	**88.7%** (L27F6345)	84.2% (L00F0890)	68.6% (L31F1512)	96.9%	**97.1%** (E:L02F1761)
Background	74.9%	**76.3%** (L28F0114)	74.3% (L26F8136)	66.3% (L31F1415)	96.6%	**97.2%** (E:L00F0563)
Sensor	69.6%	**70.0%** (L27F6345)	69.4% (L14F0420)	68.6% (L31F2533)	87.7%	**87.8%** (E:L14F1083)

states (AUROC 1.00). The camera, light, and background detectors do not respond to the perturbation on average: their activation rate is unchanged (0.64–1.00 before and after). What predicts failure is how much the perturbation disturbs them at a given state (unsigned change, AUROC 0.81–0.94), and this exceeds the shift of the whole layer (0.65–0.75), so the detector is a specific feature rather than a proxy for representational drift. The OpenVLA-OFT detectors behave like the second kind. Detectors are also less mode-selective (0.59–0.83) and less consistent across tasks than compensators (App. J). The detector's response does not track the immediate action change at the same state: after controlling for the layer-level shift, its correlation with the action change is 0.07–0.17 and with the loss of commitment near zero, whereas the compensator's response is action-coupled (0.44–0.55). Yet the detector separates Broken from Resistant states (0.81–1.00) far better than the action change itself (0.59–0.66). Detectors thus carry failure information that is not yet visible in the action.

6 Verifying by Intervention

If compensators encode the primitive that a perturbation weakens, adding a compensator direction back should re-assert it and, where the weakening caused the failure, recover the rollout. We test this prediction with direct compensator steering and then combine compensators with detector features to make the intervention selective to failure-prone states.

6.1 Robust Compensator Features

We steer the selected compensator features (§3.3) during perturbed rollouts and compare success rates against the unsteered plain policy. For comparison, we also steer randomly selected SAE features and raw residual dimensions selected by the same compensator scores.

Results. As shown in Table 4, compensator steering raises success under every perturbation type for both policies. Random SAE-feature steering does not produce the same consistent improvement, and raw residual compensation is less effective. Compensator features thus provide better intervention directions than arbitrary or raw-basis steering.

Table 5: Fixed-state steering test under camera shifts.

Policy	Steered feature	Cosine	Moved
OFT	L23F4722 (deployed)	**0.23**	**70%**
	Other features	$\leq$0.03	24–50%
$\pi_{0.5}$	E:L12F0801 (deployed)	**0.08**	**56%**
	Other features	$\leq$0.03	47–53%

Fixed-state effect of steering. At the initial state, the base, perturbed, and steered rollouts share the same physical state, so their first action chunks can be compared directly. For OpenVLA-OFT under camera shifts, the perturbation moves the first chunk away from the base action, and in 76% of states the two chunks stay in the same action mode. In these states ($n = 1{,}215$ Broken and $1{,}525$ Resistant), steering L23F4722 shifts the chunk by only 8% of the perturbation error, but the shift points back toward the base action (cosine 0.22; Table 5) and brings the chunk closer to it in 71% of Broken and 69% of Resistant states (controls 47–52%), never changing the action mode. This is specific to the deployed compensator: eight other SAE features steered on the same states (three other compensator candidates and five features outside the candidate set) have cosine 0.00–0.03 and move the action toward the base action in 24–50% of states, no better than chance. The correction per step is therefore small but consistently directed, and recovery accumulates over the rollout (Fig. 5, right); in about 30% of states the action moves further away, and where the perturbation has already changed the mode, steering does not restore it. For $\pi_{0.5}$, whose rollouts do not include a base arm, we ask whether the first action moves along the compensator's primitive. The shift appears where steering raises success the most: E:L00F0563 moves the action along its descend primitive in 80% of states under background shifts (cosine 0.33; controls 46–65%) and E:L12F0801 in 56% of 1,236 states under camera shifts (controls 47–53%; Table 5), and the shift size tracks the rollout gain.

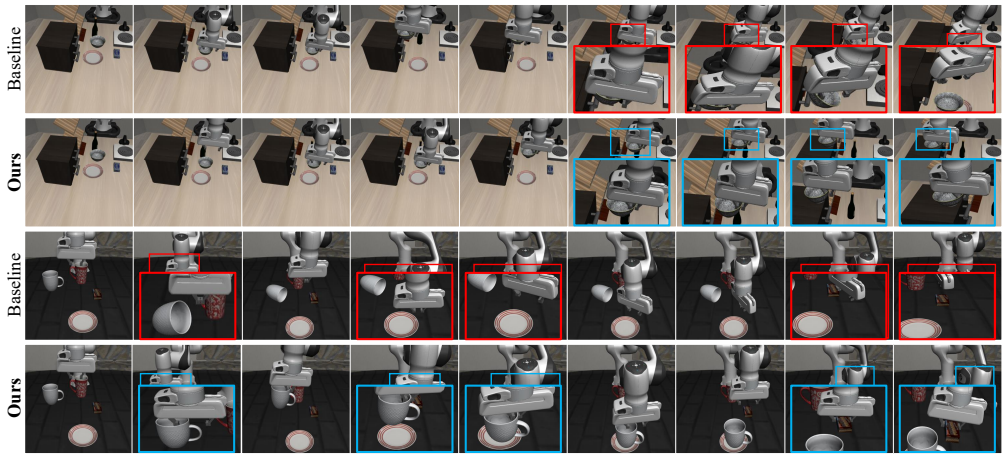


Figure 6: Detector-compensator steering on failure-prone rollouts under perturbation.

6.2 Detector-conditioned Steering

We next evaluate detector-conditioned steering (§3.5), which pairs a detector with a compensator and activates the compensator only when the detector indicates a failure-prone state.

Results. As shown in Table 6, detector-compensator pairing improves perturbed rollout success over the plain policy across both VLA architectures. Gating is useful because the compensator's per-step correction is small and can be misdirected (§6.1), while the detector identifies states where the perturbation has disrupted the upstream representation before the action changes (§5). Applying stronger compensation only at those states preserves the recovery benefit while reducing unnecessary intervention on Resistant rollouts. For $\pi_{0.5}$ Light and Background, Broken rollouts are rare, leaving little room for gating to improve aggregate success, so we retain robust-only steering instead (App. G). Detector gating thus makes compensation more selective and limits collateral effects. In Fig. 6, steering raises the gripper during transport to avoid a collision (top) and closes it earlier so the cup does not slip (bottom).

Table 6: Detector-conditioned steering: success rate of the plain policy and of the selected detector-compensator pair.

Perturbation	Model	Plain	Detector - Compensator
Camera	OpenVLA-OFT	50.5%	**52.5%** (L15F6110 - L30F5719)
	$\pi_{0.5}$	70.8%	**71.8%** (B:L02F0567 - E:L12F0259)
Light	OpenVLA-OFT	85.3%	**89.5%** (L12F2024 - L29F5719)
	$\pi_{0.5}$	96.9%	**97.1%** (Robust only, E:L02F1761)
Background	OpenVLA-OFT	74.9%	**76.3%** (Robust only, L28F0114)
	$\pi_{0.5}$	96.6%	**97.2%** (Robust only, E:L00F0563)
Sensor	OpenVLA-OFT	69.7%	**70.8%** (L31F7245 - L27F6345)
	$\pi_{0.5}$	87.7%	**89.1%** (B:L16F0279 - E:L16F0412)

7 Conclusion

We apply sparse autoencoders to vision-language-action models and identify two functional feature roles: detector features, which predict perturbation-induced failures from the vision-language stream, and compensator features, which encode task-invariant motion primitives in the action expert. Visual sensitivity appears as a weakening of the policy's commitment to the primitive it is executing, and steering a compensator feature shifts actions back toward the unperturbed action and modestly improves rollout success, with detector gating making it perturbation-specific. Across both VLA families, sparse features can be identified, interpreted, and tested by intervention.

Limitations. Our analysis requires controlled environments where states, perturbations, and rollouts can be matched, limiting direct application to unconstrained real-world settings. We evaluate on a single simulated benchmark and have not tested whether the discovered features transfer to real-world data or physical robots. Rollout stochasticity may also affect feature selection, detector estimates, and intervention outcomes despite averaging over trials. Future work will extend this analysis to real-world perturbations and deployed policies.

AI USE STATEMENT

We used generative AI tools in three ways. First, to aid and polish writing: grammar and word choice were checked with an LLM, and some paragraphs were drafted with LLM assistance and then rewritten by the authors. Second, for retrieval and discovery: an LLM was used to search for and organize related work, and every reference was verified by the authors against the original source. Third, to execute experiments: an LLM-based coding assistant was used to write and modify analysis and experiment scripts within our own codebase, and the resulting code and outputs were checked by the authors. No experimental results, figures, or claims were generated by an AI tool without author verification, and the research questions, methods, and interpretations are the authors' own. We take full responsibility for the content of this paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 2022.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. arXiv preprint arXiv:2303.08112, 2023.

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.

Kevin Black, Noah Brown, Karl Pertsch, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

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. In *Robotics: Science and Systems (RSS)*, 2023.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. Transformer Circuits Thread, 2022.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations (ICLR)*, 2025.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982. doi: 10.1148/radiology.143.1.7063747.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Conference on Robot Learning (CoRL)*, 2025.

Siddharth Karamcheti, Suraj Nair, Annie S. Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. In *Robotics: Science and Systems (RSS)*, 2023.

Momin Ahmad Khan, Novak Boskov, Fatima M. Anwar, and Manzoor A. Khan. Controlling vision–language–action policies through sparse latent directions. In *NeurIPS Workshop on Mechanistic Interpretability*, 2025.

Jinyeong Kim, Junhyeok Kim, Yumin Shim, Joohyeok Kim, Sunyoung Jung, and Seong Jae Hwang. Interpreting vision transformers via residual replacement model. *Advances in Neural Information Processing Systems*, 2026.

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. In *Conference on Robot Learning (CoRL)*, 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.

Hyesu Lim, Jinho Choi, Jaegul Choo, and Steffen Schneider. Sparse autoencoders reveal selective remapping of visual concepts during adaptation. In *International Conference on Learning Representations*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 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. In *International Conference on Learning Representations (ICLR)*, 2025.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

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. In *Robotics: Science and Systems (RSS)*, 2024.

Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *Advances in Neural Information Processing Systems*, 2025.

Sangha Park, Seungryong Yoo, Jisoo Mok, and Sungroh Yoon. Save: Sparse autoencoder-driven visual information enhancement for mitigating object hallucination. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2026.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024a.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Samuel Stevens, Wei-Lun Chao, Tanya Berger-Wolf, and Yu Su. Sparse autoencoders for scientifically rigorous interpretation of vision models. *arXiv e-prints*, pp. arXiv–2502, 2025.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling*, 2024.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe III Kennedy, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in VLA models. arXiv preprint arXiv:2603.19183, 2026.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Adam Jermyn, et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 sonnet. Transformer Circuits Thread, 2024.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Kaichen Zhang, Yifei Shen, Bo Li, and Ziwei Liu. Large multi-modal models can interpret features in large multi-modal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *[arXiv preprint arXiv:2510.03827]*, 2025.

Appendix: Table of Contents

Sparse Feature Analysis and Control of Vision-Language-Action Models

A Ablation Studies **14**

A.1 Number of selected features 14

A.2 Steering magnitude α 14

B SAE Training and Selection Details **14**

C Shared-State Perturbation Protocol Details **15**

C.1 Protocal 15

C.2 Perturbation families and pools. 17

C.3 Per-feature scores. 17

D Candidate Selection Details **17**

E Detector-Compensator Pairing details **18**

F Additional Experimental Results **19**

G Additional Reliability and Diagnostic Analyses **20**

H Compute and Implementation **21**

I Qualitative Results **21**

J Action-Mode Vocabulary **22**

J.1 Mode-selective features. 22

J.2 Per-task consistency of mode selectivity. 31

J.3 Mode-conditioned perturbation statistics. 31

A Ablation Studies

We ablate two design choices in the compensator intervention: the number of compensator features steered jointly (Sec. A.1) and the steering magnitude α (Sec. A.2). Both ablations use the canonical paired pools in Table 11, constructed by pairing LIBERO and LIBERO-Plus rollouts from identical simulator physics states. We report canonical success rates, together with recovery on Broken states and retention on Resistant states. The plain and deployed settings reproduce Table 14; the remaining settings are evaluated on the same pools.

A.1 Number of selected features

The main paper uses one compensator feature per perturbation to isolate the effect of a single sparse direction. We additionally evaluate joint steering of the top-k compensator candidates, with $k \in \{1, 2, 3, 5\}$, using the deployed steering magnitude $\alpha = -1$ and the same gating rule. The results in Table 7 show that adding features provides at most small gains and often reduces success. Light and Sensor peak at $k = 1$, while Camera and Background show only marginal changes. We therefore use single-feature steering in the main experiments.

Table 7: Multi-feature ablation on $\pi_{0.5}$. We report canonical success (%) when the top-k compensator features are steered jointly at $\alpha = -1$. The plain and $k = 1$ columns reproduce Table 14, which is our deployed setting. †Detector-gated detector-compensator setting for Camera and Sensor. Light and Background use robust-only steering, as in Table 14.

Perturbation	Plain	k=1	k=2	k=3	k=5
Camera†	70.8	71.8	**72.1**	71.5	70.7
Light	96.9	**97.1**	96.3	94.6	94.9
Background	96.6	**97.2**	96.8	96.2	97.0
Sensor†	87.7	**89.1**	88.3	86.9	87.4

A.2 Steering magnitude α

The compensator steers the residual stream along the decoder direction of a selected feature with magnitude α. In the main experiments, we set $\alpha = -1.0$ for all perturbations. We additionally conduct an ablation study over $\alpha \in -0.25, -0.5, -1.0, -2.0, -3.0$, with results reported in Table 8. The deployed value is at or near the best setting for the detector-conditioned steering. Although some robust-only cases peak at a neighboring value, but the differences are small. Larger magnitudes can over-steer; for example, on OpenVLA-OFT, Camera recovery degrades for $|\alpha| \geq 2.0$ as collateral increases. We therefore use $\alpha = -1.0$ as a global steering magnitude.

Table 8: Steering-magnitude ablation on $\pi_{0.5}$. We report canonical success (%) for each perturbation using the deployed compensator while varying α. The plain and $\alpha = -1.0$ columns reproduce Table 14; $\alpha = -1.0$ is the deployed setting. †Detector-gated detector-compensator setting for Camera and Sensor. Light and Background use robust-only steering, as in Table 14.

Perturbation	Plain	-0.25	-0.5	-1.0	-2.0	-3.0
Camera†	70.8	71.9	**72.2**	71.8	70.6	71.3
Light	96.9	96.3	95.1	**97.1**	96.0	95.7
Background	96.6	96.7	97.2	97.2	**97.5**	92.8
Sensor†	87.7	86.5	87.5	**89.1**	87.3	88.0

B SAE Training and Selection Details

This section describes the SAE training setup and selection criteria used to construct the layerwise sparse feature dictionaries for OpenVLA-OFT and $\pi_{0.5}$.

TopK SAE formulation. For a residual-stream activation $x_t \in \mathbb{R}^d$ at a given layer, the TopK SAE encodes

$$z_t = \mathrm{TopK}(W_{\mathrm{enc}}(x_t - b_{\mathrm{pre}})) \in \mathbb{R}^{d_{\mathrm{sae}}}, \tag{2}$$

and decodes

$$\hat{x}_t = W_{\mathrm{dec}} z_t + b_{\mathrm{pre}}. \tag{3}$$

The SAE is trained with the reconstruction loss $\|x_t - \hat{x}_t\|_2^2$. Residual-stream activations for SAE training are extracted from in-distribution LIBERO rollouts, and SAEs are trained independently for each layer and stream.

Massive activation stabilization. VLA residual streams contain a small number of massive coordinates whose norms are orders of magnitude larger than typical coordinates. These coordinates can dominate the L_2 reconstruction objective and obscure the quality of the remaining representation. We identify massive dimensions separately for each layer and exclude them when training and evaluation SAEs; the excluded-dimension lists are saved with each checkpoint to restore after SAE decoding. For $\pi_{0.5}$, most layers require excluding 0-4 dimensions, with the largest case being 89 dimensions at base L1.

Selection criteria. The target SAE quality criteria are $\mathrm{FVU} \leq 0.15$ and a dead-feature rate $\leq 1\%$. The dead-feature criterion is satisfied for every layer used in the experiments: all 32 OpenVLA-OFT SAEs have dead-feature rates below 1% with $k = 64$, and all $\pi_{0.5}$ SAEs, trained with $k = 16$, likewise have dead-feature rates below 1%. The FVU target is satisfied by a subset of layers: OpenVLA-OFT $9/32$, $\pi_{0.5}$ base $18/18$, and $\pi_{0.5}$ expert layers L0-L7 $(8/18)$. For deeper layers above this FVU target, we retain the strongest available SAE configuration instead of discarding the layer. Our downstream analysis relies on paired feature-shift contrasts and decoder-based steering directions, so exact reconstruction fidelity is not the only relevant criterion. Increasing TopK k can further reduce FVU, but it also reduces activation sparsity and gives only limited additional reconstruction gains in our setting. We therefore treat $\mathrm{FVU} \leq 0.15$ as a reconstruction-quality reference rather than a strict exclusion rule.

OpenVLA-OFT residual-stream SAEs. All OpenVLA-OFT SAEs use the same configuration: activation size 4096, dictionary size 8192 ($2\times$ expansion), TopK $k = 64$, and 3 training epochs. Table 9 lists the selected SAE for each layer.

$\pi_{0.5}$ dual-stream SAEs. $\pi_{0.5}$ has two residual streams. The base stream, denoted by the prefix `B:`, uses dictionary size 2048 ($1\times$ expansion) and TopK $k = 16$. The action-expert stream, denoted by the prefix `E:`, uses dictionary size 2048 ($2\times$ expansion), TopK $k = 16$, and 4 training epochs. All 18 base-stream SAEs satisfy $\mathrm{FVU} \leq 0.15$. In the action-expert stream, layers L0-L7 satisfy this target, while the deployed compensators are located in deeper, higher-FVU layers: L12F0259, L12F0801, L14F1083, and L16F0412. Tables 10a and 10b list the selected SAEs.

C Shared-State Perturbation Protocol Details

The shared-state perturbation protocol isolates the effect of visual perturbations from physical-state divergence and defines the canonical Broken/Resistant pools used in the analysis. We first run closed-loop rollouts under both base and perturbed renderings to assign outcome labels: Resistant denotes base success and perturbed success, while Broken denotes base success and perturbed failure. These closed-loop outcomes define the pools, but direct feature comparison across the two trajectories would be confounded. After a few steps, the robot and object configurations can diverge, so feature differences would reflect trajectory and state-action-history differences rather than the visual perturbation itself. Our shared-state replay removes this confound by using the closed-loop rollouts only for labeling, while feature and action shifts are computed from paired policy queries at identical simulator states.

C.1 Protocal

We first run a reference rollout under base rendering only. At each timestep t, we record the full simulator state s_t, including robot state, object poses, and any simulator variables needed to restore

Table 9: OpenVLA-OFT layerwise SAEs. The layer containing each main-paper feature is annotated. All layers are alive, with dead-feature rate $\leq 0.51\%$. The FVU target, FVU ≤ 0.15, is met by $9/32$ layers: L0-L7 and L31.

ℓ	Role in our paper	val FVU	dead %
0	-	0.144	0.00
1	Background detector (`F3670`)	0.131	0.00
2	-	0.123	0.00
3	-	0.125	0.00
4	-	0.127	0.00
5	-	0.130	0.00
6	-	0.133	0.00
7	-	0.144	0.00
8	-	0.153	0.00
9	-	0.164	0.00
10	-	0.175	0.00
11	-	0.187	0.00
12	Light detector / comp. (`F2024`)	0.206	0.00
13	-	0.217	0.00
14	-	0.234	0.00
15	Camera detector (`F6110`)	0.246	0.00
16	-	0.246	0.02
17	-	0.259	0.04
18	-	0.260	0.00
19	-	0.271	0.04
20	-	0.278	0.01
21	-	0.285	0.04
22	-	0.302	0.09
23	Camera robust comp. (`F4722`)	0.316	0.13
24	-	0.320	0.15
25	-	0.327	0.15
26	-	0.316	0.33
27	Sensor / Light robust comp. (`F6345`)	0.318	0.20
28	Background robust comp. (`F0114`)	0.303	0.34
29	Light det.-comp. (`F5670/F5719`)	0.256	0.49
30	Camera det.-comp. (`F5719`)	0.185	0.51
31	Sensor detector (`F7245`)	0.126	0.38

Table 10: $\pi_{0.5}$ SAE reconstruction quality.

(a) $\pi_{0.5}$ base-stream SAEs (all 18; detector layers in bold).

ℓ	val FVU	≤ 0.15	ℓ	val FVU	≤ 0.15
0	**0.027**	✓	9	0.066	✓
1	0.029	✓	10	0.082	✓
2	**0.027**	✓	11	0.098	✓
3	0.031	✓	12	0.138	✓
4	0.034	✓	13	0.123	✓
5	0.035	✓	14	0.098	✓
6	0.043	✓	15	0.050	✓
7	0.047	✓	**16**	**0.006**	✓
8	0.054	✓	17	0.020	✓

(b) $\pi_{0.5}$ action-expert SAEs (all 18; deployed compensator layers in bold).

ℓ	val FVU	≤ 0.15	ℓ	val FVU	≤ 0.15
0	0.050	✓	9	0.161	×
1	0.043	✓	10	0.165	×
2	**0.079**	✓	11	0.170	×
3	0.128	✓	**12**	**0.177**	×
4	0.128	✓	13	0.183	×
5	0.124	✓	**14**	**0.211**	×
6	0.131	✓	15	0.234	×
7	0.147	✓	**16**	**0.235**	×
8	0.153	×	17	0.234	×

the physical configuration, together with the policy action under base rendering. We then replay each saved state in a paired, one-step manner: for each recorded s_t, we reset the simulator to exactly the same physical state and render two observations, o_t^{base} and o_t^{pert}, corresponding to base and perturbed rendering. Both observations are encoded by the policy and the SAE at the same physical configuration, yielding $z(o_t^{\text{base}})$ and $z(o_t^{\text{pert}})$, as well as the policy-predicted action chunks a_t^{base} and a_t^{pert} at that state.

This gives paired feature shifts and action shifts as below:

$$\begin{aligned}\Delta z_t &= z(o_t^{\text{pert}}) - z(o_t^{\text{base}}) \\ \Delta a_t &= a_t^{\text{pert}} - a_t^{\text{base}}.\end{aligned} \tag{4}$$

Here a_t^{base} and a_t^{pert} are policy outputs queried at the same saved simulator state, not actions from two independently evolving trajectories. Thus, Δz_t and Δa_t measure observation-induced changes rather than compounding state drift. In implementation, each perturbed sample is matched to its base counterpart by the pairing key $(\text{suite}, \text{base_task}, \text{shared_state_idx})$, and the set I denotes all paired timesteps for a given perturbation family.

C.2 Perturbation families and pools.

We restrict to the four physics-invariant LIBERO-Plus perturbation families: Camera, Light, Background-texture, and Sensor-noise. These perturbations change the rendered observation or camera stream while leaving object poses, contacts, dynamics, and task initialization unchanged. For each perturbation family, the canonical paired pool consists of paired timesteps from base-success rollouts, grouped by suite and task and labeled by the outcome of the corresponding perturbed rollout.The outcome label is assigned at the rollout level and inherited by all paired timesteps from that rollout. We exclude base-failure rollouts because they do not isolate perturbation-induced degradation. Table 11 reports the resulting rollout-level pool sizes. These pools are used for feature discovery, scoring, and recovery/retention ablations, while deployed success rates are measured by executing the policy under the corresponding perturbation.

C.3 Per-feature scores.

Over the set I of paired timesteps, we compute three scores for each SAE feature f. First, perturbation sensitivity measures how strongly the feature responds to the perturbation:

$$M_f = \mathbb{E}_{t \in I}\left[\,|\Delta z_{t,f}|\,\right], \qquad D_f = \frac{M_f}{\max_{f'} M_{f'} + \epsilon}. \tag{5}$$

The normalized score $D_f \in [0, 1]$ makes sensitivity comparable across perturbation families. Second, action coupling measures whether the feature response tracks perturbation-induced action changes:

$$C_f = \max_i \left|\text{Corr}(\Delta z_{:,f}, \Delta a_{:,i})\right|. \tag{6}$$

We compute this as a Spearman rank correlation after rank-normalizing each feature-shift and action-shift dimension, and take the maximum absolute correlation over the action signature dimensions. For OpenVLA-OFT, the action signature is the 56-dimensional action chunk; for $\pi_{0.5}$, it is the corresponding action window used by the policy. Third, Broken-Resistant separability measures whether the feature distinguishes failing from surviving perturbed rollouts:

$$A_f = \text{AUROC}\big(\{\Delta z_{t,f} : t \in I_{\text{Broken}}\}, \{\Delta z_{t,f} : t \in I_{\text{Resistant}}\}\big), \tag{7}$$

computed as the Mann-Whitney AUROC between Broken and Resistant paired feature-response scores. Thus, $A_f = 0.5$ indicates chance-level separability, while values near 1 or 0 indicate strong separation in opposite directions. We use these quantities to rank perturbation-sensitive features for the detector and compensator analyses.

D Candidate Selection Details

This section details how each SAE feature is scored and how detector and compensator candidates are shortlisted before the screening rollout.

Top-0.1% selection. Compensator candidates are ranked by $S_f^{\text{cmp}} = A_f \cdot D_f \cdot C_f$; detector candidates by A_f. We retain the top 0.1% of each (OFT SAE: 262 of 32×8192; raw: 131 of 32×4096). A consistent asymmetry holds: for $\pi_{0.5}$ all top detectors are base-stream and all top compensators are expert-stream; for OFT detectors concentrate early and compensators late (L24-L30). The deployed $\pi_{0.5}$ robust compensators are in-pool but not the top-scored, they were selected by held-out rollout benefit among the top candidates, not by the offline score.

Table 11: Canonical perturbation pools (physics-invariant families) for OpenVLA-OFT and $\pi_{0.5}$. Broken = base-condition success but perturbed failure; Resistant = perturbed success.

	OpenVLA-OFT			$\pi_{0.5}$		
Perturbation	Broken	Resistant	Total	Broken	Resistant	Total
Camera Viewpoints	666	825	1491	468	1131	1599
Sensor Noise	343	1108	1451	198	1403	1601
Light Conditions	64	974	1038	29	691	720
Background Textures	133	863	996	38	1038	1076
Total	**1206**	**3770**	**4976**	**733**	**4263**	**4996**

Selection by screening rollout. The offline scores (A_f, S_f^{cmp}) only define the top-0.1% *candidate* shortlist; they do not by themselves determine the deployed features. The final detector and compensator features are chosen by a *screening rollout*: each shortlisted candidate, and for detector-compensator pairing each $(\text{detector}, \text{compensator}, \tau)$ combination, is evaluated in closed loop over the canonical perturbation pool. We retain the feature or pair that gives the largest rollout benefit over the plain policy. The offline score is thus a cheap pre-filter that shrinks the feature space to a tractable candidate set, while closed-loop screening is the actual deployment-selection criterion. This is why a deployed feature need not be the top-scored offline candidate, and why the ranked tables below report candidate *shortlists* rather than the single selected feature.

Ranked candidates. Tables 12a-13b list the top-5 candidates per perturbation. Detectors are ranked by A; compensators by $A \cdot D \cdot C$. Cell shows the score with the feature in parentheses.

Table 12: Detector candidate shortlists. Candidates are ranked by A.

(a) OpenVLA-OFT

Pert.	#1	#2	#3	#4	#5
Camera	**0.776** (L15F6110)	0.762 (L17F2733)	0.760 (L29F2032)	0.758 (L03F5017)	0.758 (L11F1156)
Light	**0.926** (L03F5256)	0.924 (L03F0745)	0.920 (L03F2776)	0.919 (L04F3553)	0.915 (L19F1713)
Backg.	**0.788** (L14F0645)	0.777 (L01F3670)	0.775 (L06F6572)	0.773 (L03F8086)	0.772 (L12F5832)
Sensor	**0.801** (L31F7245)	0.786 (L15F5451)	0.785 (L28F2466)	0.783 (L14F3589)	0.781 (L28F7631)

(b) $\pi_{0.5}$

Pert.	#1	#2	#3	#4	#5
Camera	**0.828** (B:L02F0567)	0.787 (B:L05F0306)	0.780 (B:L09F0557)	0.780 (B:L00F1006)	0.779 (B:L03F1720)
Light	**0.941** (B:L00F1823)	0.899 (B:L00F1875)	0.885 (B:L00F0570)	0.882 (B:L00F1407)	0.879 (B:L00F0501)
Backg.	**0.813** (B:L00F0501)	0.811 (B:L02F0710)	0.805 (B:L13F1577)	0.793 (B:L05F1446)	0.788 (B:L06F1446)
Sensor	**0.996** (B:L16F0279)	0.995 (B:L10F1069)	0.995 (B:L12F2023)	0.995 (B:L08F0520)	0.995 (B:L02F2043)

E Detector-Compensator Pairing details

This section details how a detector feature is paired with a compensator, and how the firing gate and steering magnitude are set.

Detector-conditioned steering. For a compensator feature f at layer ℓ with decoder column $W^{\text{dec}}_{:,f}$, steering adds $\tilde{x}_t = x_t + \alpha\, g_t\, W^{\text{dec}}_{:,f}$, where α controls the signed magnitude of the decoder-direction intervention and $g_t \in \{0, 1\}$ is a gate equal to 1 when the paired detector feature is active at step t. In all deployed experiments, we use $\alpha = -1.0$ unless otherwise stated. Unconditional steering ($g_t \equiv 1$) is the "robust-only" setting; detector gating restricts the compensator to failure-prone timesteps. For $\pi_{0.5}$ the detector is a base-stream feature and the compensator an expert-stream feature.

Pair selection. For each perturbation, we run the compensator always-on over the canonical pool while logging the activation of every candidate detector. We then sweep a firing threshold τ per

Table 13: Compensator candidate shortlists. Candidates are ranked by $A \cdot D \cdot C$.

(a) OpenVLA-OFT

Pert.	#1	#2	#3	#4	#5
Camera	**0.120** (L29F5719)	0.116 (L28F5565)	0.109 (L27F7281)	0.107 (L29F6041)	0.106 (L26F5651)
Light	**0.124** (L29F5670)	0.111 (L30F2625)	0.107 (L28F2105)	0.105 (L26F7581)	0.098 (L27F6742)
Backg.	**0.101** (L28F5565)	0.096 (L26F5651)	0.095 (L29F5719)	0.095 (L27F7281)	0.090 (L25F6158)
Sensor	**0.107** (L25F6557)	0.100 (L24F7160)	0.100 (L27F7160)	0.099 (L29F5719)	0.098 (L26F5977)

(b) $\pi_{0.5}$

Pert.	#1	#2	#3	#4	#5
Camera	**0.430** (E:L05F0688)	0.415 (E:L06F0802)	0.383 (E:L04F0536)	0.376 (E:L11F0066)	0.371 (E:L11F0801)
Light	**0.462** (E:L05F0688)	0.424 (E:L04F0536)	0.402 (E:L11F0801)	0.381 (E:L05F0377)	0.372 (E:L14F1698)
Backg.	**0.429** (E:L06F0802)	0.422 (E:L16F1083)	0.418 (E:L14F1083)	0.416 (E:L11F0066)	0.392 (E:L13F0066)
Sensor	**0.429** (E:L05F0688)	0.404 (E:L14F1083)	0.403 (E:L11F0066)	0.396 (E:L06F0802)	0.389 (E:L16F1083)

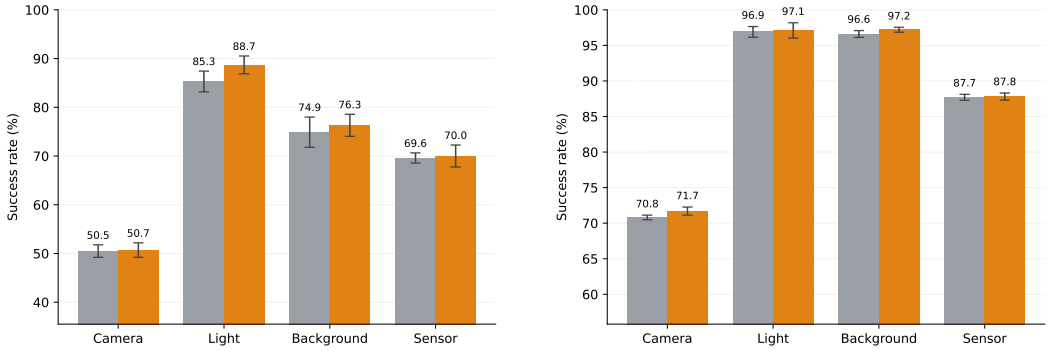
detector and select the $(\text{detector}, \tau)$ that maximizes gated success. We define expected benefit as gated success minus plain success. When no pairing beats the plain policy, as in high-baseline Light and Background cases, the robust-only compensator is retained. We additionally evaluate the stability of this selection with a 10-seed $50/50$ held-out cross-validation.

Separation of selection and evaluation. The deployment choices fixed by this screening are then evaluated on newly executed rollouts, so evaluation trajectories are not reused for selection. For α, the sweep in App. A.2 shows that $\alpha = -1$ performs at or near the best, so we fix it across conditions.

Steering hyperparameters. $\pi_{0.5}$ compensators use $\alpha = -1.0$ on the expert residual stream. The OFT graded gate normalizes the detector activation by a band (lo, hi) and fires at amplitude $a_{\max}$ when the normalized activation ≥ 0.5 (deployed Light band $(0, 0.0109)$, $a_{\max} = 0.5$).

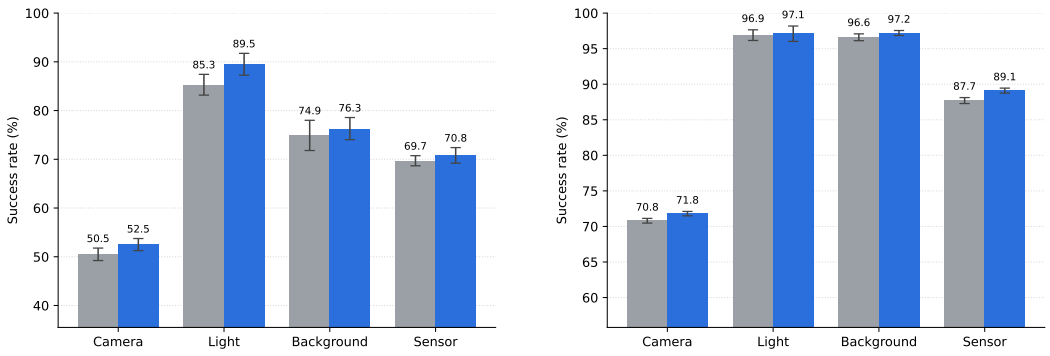
F Additional Experimental Results

Table 14 supplements the detector-compensator results in the main paper with trial-level variability. Success rates are averaged over three deployment trials, and the gain Δ denotes the improvement over the plain policy in percentage points. We report mean $\pm$ standard deviation across trials. Figures 7 and 8 show the corresponding robust-compensation and detector-compensator results with error bars.



(a) OpenVLA-OFT (b) $\pi_{0.5}$

Figure 7: Robust compensation with always-on steering compared with the plain policy under each visual perturbation. Bars show mean success over three trials, with error bars indicating standard deviation.



(a) OpenVLA-OFT (b) $\pi_{0.5}$

Figure 8: Deployed steering compared with the plain policy under each visual perturbation. Bars show mean success over three trials, with error bars indicating standard deviation. See Table 14.

Table 14: Deployed steering results corresponding to main-paper Table 6. Plain success and the gain Δ over the plain policy, in percentage points, are reported as mean $\pm$ standard deviation over three trials.

Perturbation	Model	Plain	Deployed intervention	Δ (pp)
Camera	OpenVLA-OFT	50.5 ± 1.3	52.5 (L15F6110 $\rightarrow$ L30F5719)	$+2.1 \pm 0.9$
	$\pi_{0.5}$	70.8 ± 0.3	71.8 (B:L02F0567 $\rightarrow$ E:L12F0259)	$+1.0 \pm 0.3$
Light	OpenVLA-OFT	85.3 ± 2.1	89.5 (L12F2024 $\rightarrow$ L29F5719)	$+4.2 \pm 0.9$
	$\pi_{0.5}$	96.9 ± 0.8	97.1 (Robust only, E:L02F1761)	$+0.2 \pm 1.1$
Background	OpenVLA-OFT	74.9 ± 3.1	76.3 (Robust only, L28F0114)	$+1.4 \pm 1.7$
	$\pi_{0.5}$	96.6 ± 0.5	97.2 (Robust only, E:L00F0563)	$+0.6 \pm 0.6$
Sensor	OpenVLA-OFT	69.7 ± 1.0	70.8 (L31F7245 $\rightarrow$ L27F6345)	$+1.1 \pm 1.4$
	$\pi_{0.5}$	87.7 ± 0.4	89.1 (B:L16F0279 $\rightarrow$ E:L16F0412)	$+1.4 \pm 0.7$

G Additional Reliability and Diagnostic Analyses

Confidence intervals over ten trials. Success rates in the main paper are averaged over three trials. To assess the stability of the reported gains across trials, we ran seven additional independent trials of the deployed $\pi_{0.5}$ configuration in Table 6, with the model, selected features, and α fixed, giving ten trials in total. We focus the additional trials on $\pi_{0.5}$, for which the absolute gains are generally smaller. Table 15 reports two-sided 95% Student's t confidence intervals for the trial-level gains. With $n = 10$, the intervals for Camera, Background, and Sensor exclude zero, while the interval for Light includes zero.

Table 15: Steering gains of $\pi_{0.5}$, in percentage points over the plain policy, with 95% confidence intervals for the original three trials and for ten trials.

Perturbation	Paper ($n = 3$)	Paper + additional trials ($n = 10$)
Camera	$+1.00\ [0.25,\ 1.75]$	$+0.86\ [0.26,\ 1.46]$
Light	$+0.20\ [-2.53,\ 2.93]$	$+0.20\ [-0.54,\ 0.94]$
Background	$+0.60\ [-0.89,\ 2.09]$	$+0.53\ [0.26,\ 0.80]$
Sensor	$+1.40\ [-0.34,\ 3.14]$	$+1.19\ [0.75,\ 1.63]$

Raw residual dimensions vs. SAE features for $\pi_{0.5}$. Table 1 reports the feature-basis comparison for OpenVLA-OFT. Table 16 repeats the same analysis for $\pi_{0.5}$. SAE features match or outperform raw residual dimensions for detection across all perturbations, with Camera nearly tied, and substantially outperform them in compensator scores, consistent with the OpenVLA-OFT trend.

Detector-gating diagnostics. Whether gating can add benefit over robust-only steering depends on the quality of the detector and on the imbalance between Resistant and Broken rollouts. Only

Table 16: $\pi_{0.5}$ raw residual dimensions vs. SAE features (top 0.1%): mean detector AUROC and mean compensator score.

Perturbation	Det. Top-0.1%		Comp. Top-0.1%	
	Raw	SAE	Raw	SAE
Camera	0.736	**0.737**	0.0108	**0.2470**
Light	0.733	**0.784**	0.0137	**0.2779**
Background	0.734	**0.770**	0.0125	**0.2393**
Sensor	0.868	**0.903**	0.0120	**0.2385**

Broken rollouts can contribute positive recovery to aggregate success. Table 17 lists, for each $\pi_{0.5}$ perturbation, the Resistant-to-Broken ratio of the canonical pool, the AUROC, true-positive rate, and false-positive rate of the deployed detector at its fixed threshold, and the resulting ceiling on the aggregate gain. The ceiling corresponds to the gain if every detected Broken rollout were recovered and no Resistant rollout were harmed. For Light and Background, the ceiling is below one percentage point because Broken rollouts are rare (R/B $\geq 23\times$). We therefore retain robust-only steering in these settings. At runtime, the fixed detector threshold gates the preselected compensator without using rollout outcomes.

Table 17: Detector-gating diagnostics for $\pi_{0.5}$: Resistant-to-Broken ratio, detector AUROC / TPR / FPR at the deployed threshold, and the ceiling on the aggregate gain in percentage points.

Perturbation	R/B	AUROC / TPR / FPR	Ceiling
Camera	$2.4\times$	0.828 / 0.80 / 0.32	$+1.33$
Light	$23.8\times$	0.941 / 0.76 / 0.07	$+0.59$
Background	$27.3\times$	0.813 / 0.86 / 0.43	$+0.72$
Sensor	$7.1\times$	0.996 / 0.97 / 0.03	$+1.77$

H Compute and Implementation

All experiments were run on a single node with $8\times$ NVIDIA H100 GPUs (80 GB each). Training the 32 layer-wise SAEs for OpenVLA-OFT required approximately 11 hours per layer, corresponding to approximately 350 GPU-hours in total. The full pipeline, including SAE training, feature scoring, perturbation scanning, and steering and detector-compensator rollouts, runs on the OpenVLA-OFT (`prismatic`) and `openpi`/JAX backbones with the LIBERO and LIBERO-Plus simulators.

Runtime overhead. On an NVIDIA A6000, detector-gated steering of $\pi_{0.5}$ increases mean inference latency from 305.98 ms to 308.14 ms ($+2.16$ ms), comparable to robust-only steering ($+2.28$ ms). Because the detector lies upstream of the compensator, detection and intervention occur within the same forward pass without an additional control-step delay.

I Qualitative Results

We provide step-matched "Baseline vs. Ours" rollout filmstrips for representative detector-compensator interventions. Each filmstrip compares the perturbed baseline with the steered policy from the same physical state, with the baseline shown on the top row and the steered rollout on the bottom row.

Figures 9 to 15 show OpenVLA-OFT examples across the four perturbation families: Camera, Light, Background, and Sensor noise. We include both recovery cases, where steering converts a Broken rollout into a success, and collateral-check cases, where steering is applied to Resistant rollouts to verify that success is preserved. Representative recoveries include raising the gripper during transport to avoid collision under camera perturbations and closing the gripper earlier to prevent object slippage under light perturbations.

Figures 16 and 17 show additional examples for $\pi_{0.5}$. Because the aggregate gains of $\pi_{0.5}$ are smaller, clean per-episode fail-to-success flips are less frequent, so we report representative recovery examples identified across rollout seeds.

J ACTION-MODE VOCABULARY

We construct an action-mode vocabulary separately for each policy using successful unsteered rollouts. Each forward pass is represented by the horizon-mean of its commanded action chunk, including translation, rotation, and gripper commands. Before clustering, translation and rotation dimensions are normalized using the pooled translation standard deviation, while the gripper dimension is normalized separately. We then apply Ward hierarchical clustering to $15{,}000$ sampled forward passes.

The number of clusters is selected by bootstrap stability. For each candidate k, we evaluate clustering stability over five resamples of $8{,}000$ forward passes using the adjusted Rand index. This procedure selects k=10 for $\pi_{0.5}$ and k=8 for OpenVLA-OFT. Tables 18 and 19 report the resulting centroids in raw controller units. The mode names summarize the dominant translational components and gripper state of each centroid. Every base, perturbed, and steered forward pass is assigned to its nearest centroid in the normalized action space.

For a feature f, we define its *mode selectivity* for mode m as the AUROC of its activation for forward passes assigned to m versus all remaining modes. For a forward pass assigned to mode m, we define its *commitment* to that mode as the projection of its normalized action onto the unit centroid direction of m. A large commitment therefore indicates that the commanded action strongly follows the direction represented by the mode.

Table 18: $\pi_{0.5}$ action modes (k=10). Centroids are reported in raw controller units, and *grip* denotes the mean gripper command.

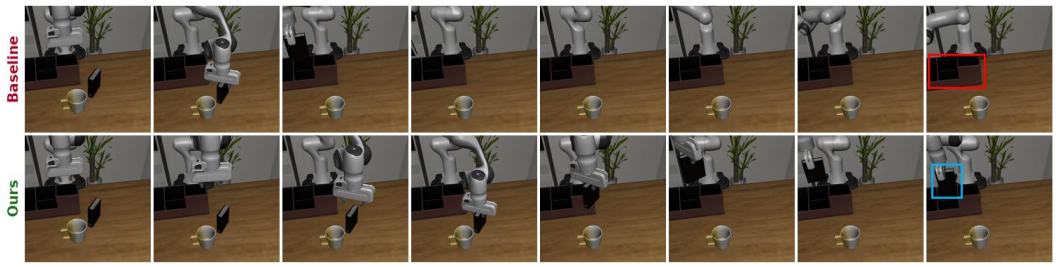
Mode	Name	dx	dy	dz	drx	dry	drz	grip
m_0	open / $+y, -x, -z$	-0.11	$+0.69$	-0.25	$+0.00$	-0.02	$+0.02$	-0.99
m_1	open / $-y, +x$	$+0.39$	-0.45	-0.03	$+0.01$	-0.00	-0.00	-0.99
m_2	open / $+x, +y$	$+0.79$	$+0.30$	-0.15	-0.01	-0.00	$+0.01$	-0.99
m_3	open / hold	$+0.12$	-0.03	$+0.03$	-0.01	$+0.02$	$+0.00$	-0.99
m_4	open / $-z, -x$	-0.15	$+0.02$	-0.62	$+0.01$	-0.02	$+0.00$	-0.97
m_5	close / $-z, -x$	-0.13	$+0.02$	-0.47	$+0.01$	-0.00	$+0.01$	$+0.82$
m_6	close / $+z$	$+0.13$	$+0.05$	$+0.07$	-0.01	$+0.03$	-0.01	$+0.99$
m_7	close / $+y$	$+0.02$	$+0.71$	-0.03	-0.00	-0.01	-0.00	$+0.99$
m_8	close / $+z, +y$	$+0.18$	$+0.33$	$+0.69$	-0.01	$+0.03$	-0.01	$+0.99$
m_9	close / $+z, -y, +x$	$+0.38$	-0.43	$+0.44$	$+0.04$	$+0.02$	-0.01	$+0.99$

Table 19: OpenVLA-OFT action modes (k=8). Centroids are reported in raw controller units, and *grip* denotes the mean gripper command.

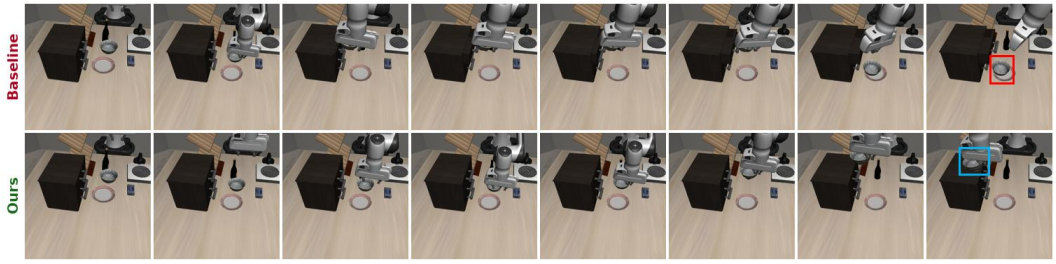
Mode	Name	dx	dy	dz	drx	dry	drz	grip
m_0	open / $-y, +z$	$+0.18$	-0.52	$+0.19$	$+0.03$	$+0.02$	-0.01	-0.00
m_1	open / $+y$	$+0.00$	$+0.70$	-0.01	-0.00	-0.01	-0.00	$+0.00$
m_2	open / $+z$	$+0.14$	$+0.21$	$+0.60$	-0.01	$+0.03$	-0.01	$+0.00$
m_3	open / $-z$	-0.03	$+0.06$	-0.24	$+0.00$	$+0.01$	$+0.01$	$+0.14$
m_4	close / $+z$	-0.00	$+0.08$	$+0.11$	-0.01	$+0.01$	-0.01	$+0.98$
m_5	close / $-z, -x$	-0.10	-0.00	-0.51	$+0.00$	-0.01	$+0.02$	$+0.98$
m_6	close / $+y, -z, -x$	-0.10	$+0.72$	-0.27	$+0.00$	-0.02	$+0.01$	$+0.99$
m_7	close / $+x, -y$	$+0.42$	-0.24	-0.06	$+0.01$	$+0.01$	-0.03	$+1.00$

J.1 MODE-SELECTIVE FEATURES.

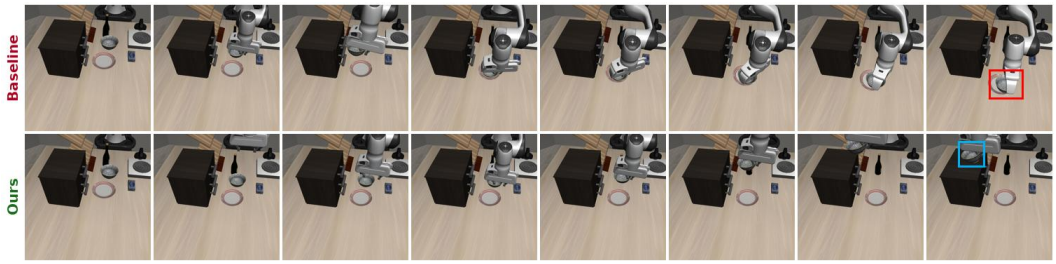
We first examine how strongly individual SAE features align with the action-mode vocabulary. Tables 20 and 21 report the three most mode-selective features for each mode, evaluated on shared-state base forward passes. Across both policies, many individual features strongly predict a particular



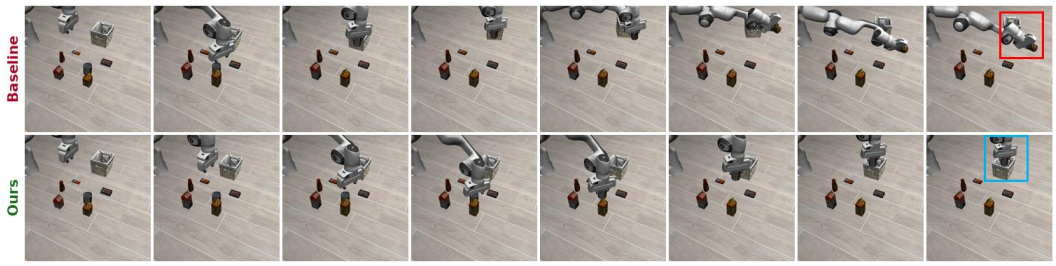
(a) OpenVLA-OFT - Camera, LIBERO-10. Task: “pick up the book and place it in the back compartment of the caddy.”



(b) OpenVLA-OFT - Camera, LIBERO-Goal. Task: “put the bowl on top of the cabinet.”

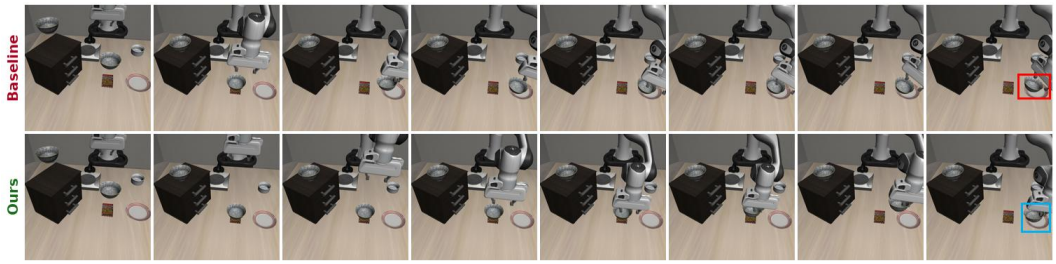


(c) OpenVLA-OFT - Camera, LIBERO-Goal. Task: “put the bowl on top of the cabinet.”

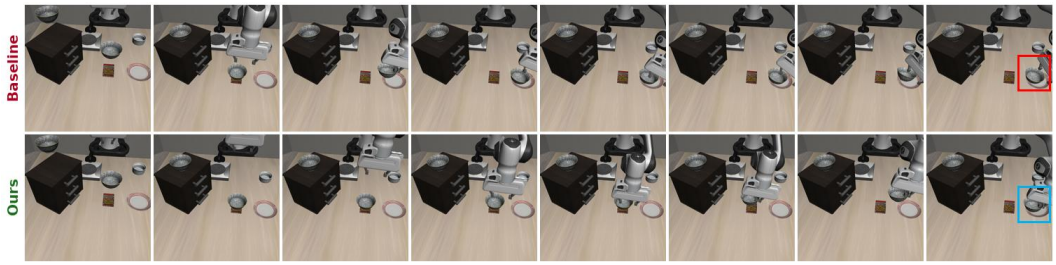


(d) OpenVLA-OFT - Camera, LIBERO-Object. Task: “pick up the tomato sauce and place it in the basket.”

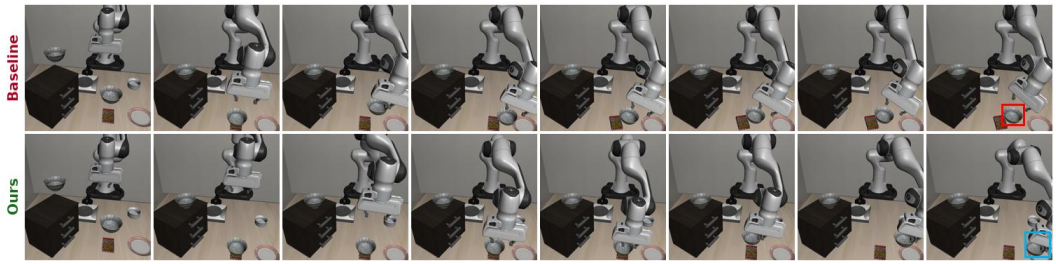
Figure 9: Qualitative examples under camera perturbation. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



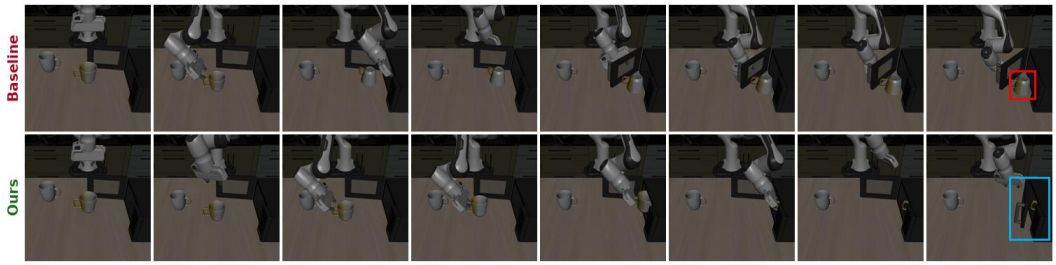
(a) OpenVLA-OFT - Camera, LIBERO-Spatial. Task: “pick up the black bowl on the cookie box and place it on the plate.”



(b) OpenVLA-OFT - Camera, LIBERO-Spatial. Task: “pick up the black bowl on the cookie box and place it on the plate.”

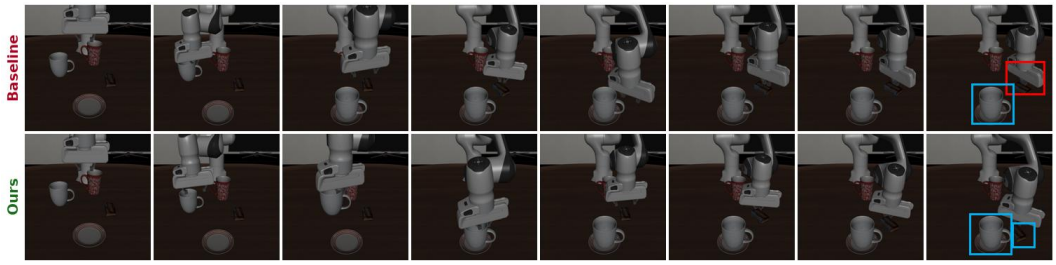


(c) OpenVLA-OFT - Camera, LIBERO-Spatial. Task: “pick up the black bowl on the cookie box and place it on the plate.”

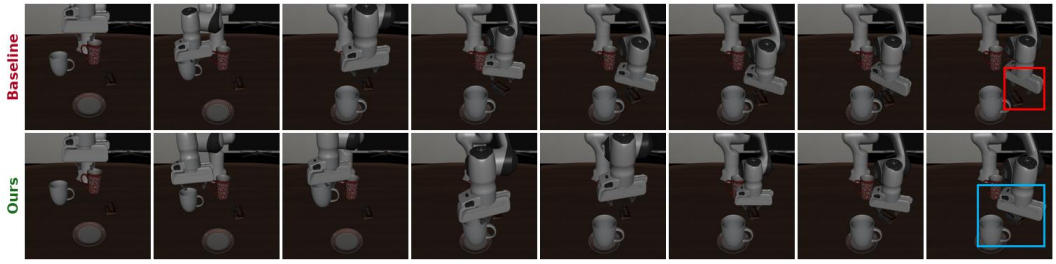


(d) OpenVLA-OFT - Light, LIBERO-10. Task: “put the yellow and white mug in the microwave and close it.”

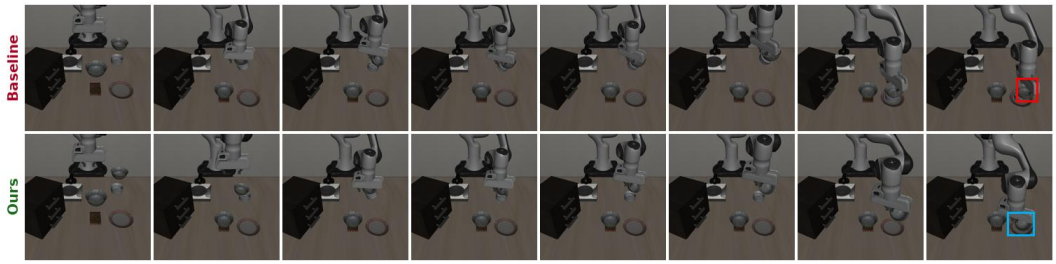
Figure 10: Additional qualitative examples under camera and light perturbations. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



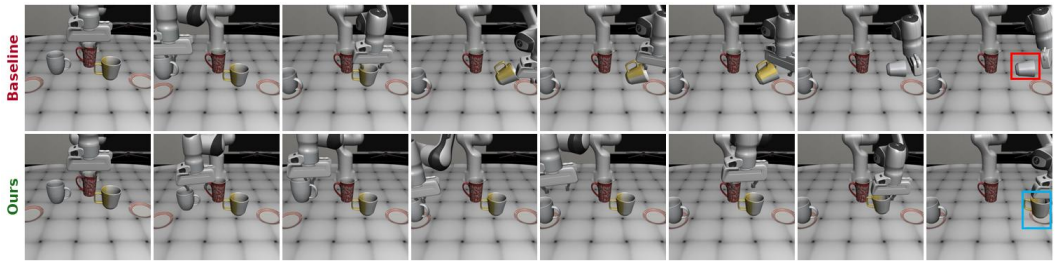
(a) OpenVLA-OFT - Light, LIBERO-10. Task: “put the white mug on the plate and put the chocolate pudding to the right of the plate.”



(b) OpenVLA-OFT - Light, LIBERO-10. Task: “put the white mug on the plate and put the chocolate pudding to the right of the plate.”

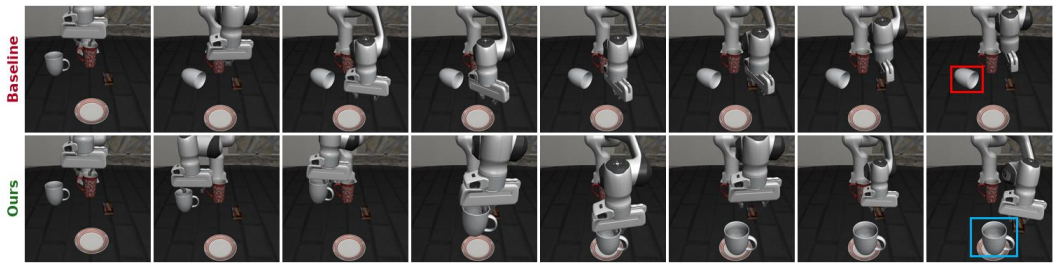


(c) OpenVLA-OFT - Light, LIBERO-Spatial. Task: “pick up the black bowl on the ramekin and place it on the plate.”

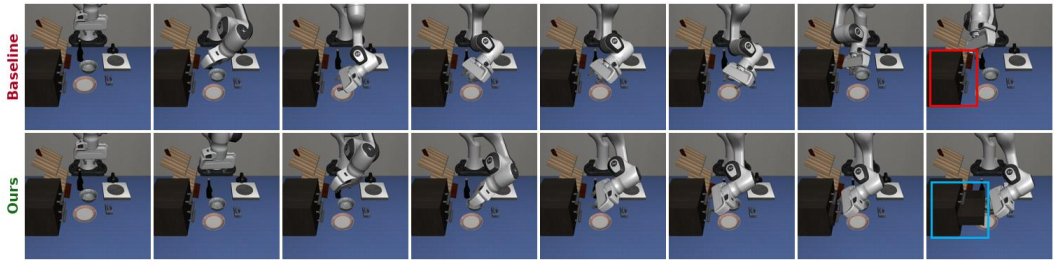


(d) OpenVLA-OFT - Background, LIBERO-10. Task: “put the white mug on the left plate and put the yellow and white mug on the right plate.”

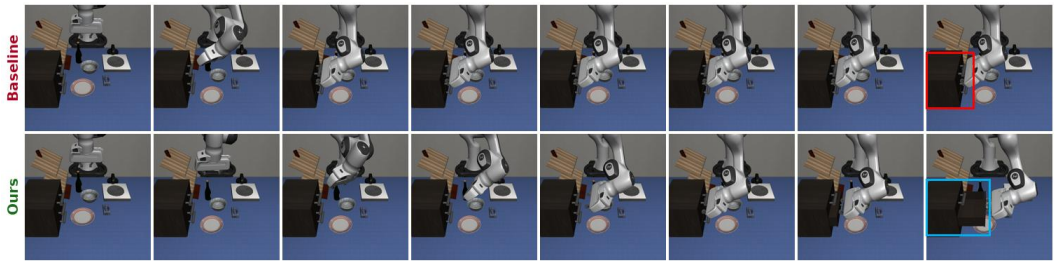
Figure 11: Qualitative examples under light and background perturbations. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



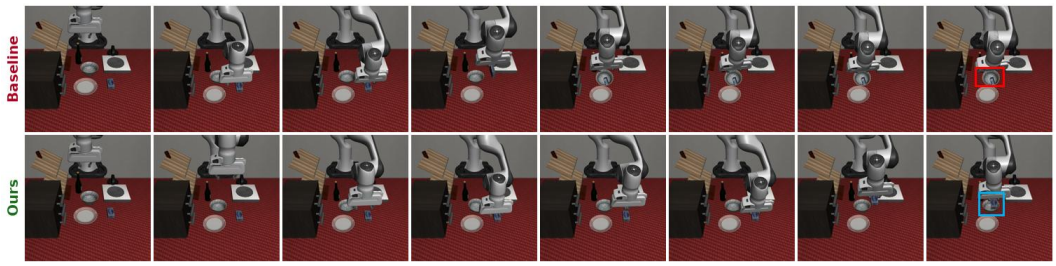
(a) OpenVLA-OFT - Background, LIBERO-10. Task: “put the white mug on the plate and put the chocolate pudding to the right of the plate.”



(b) OpenVLA-OFT - Background, LIBERO-Goal. Task: “open the middle drawer of the cabinet.”

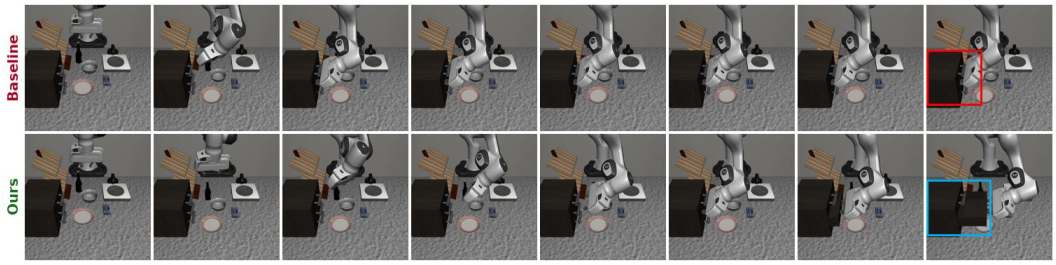


(c) OpenVLA-OFT - Background, LIBERO-Goal. Task: “open the middle drawer of the cabinet.”

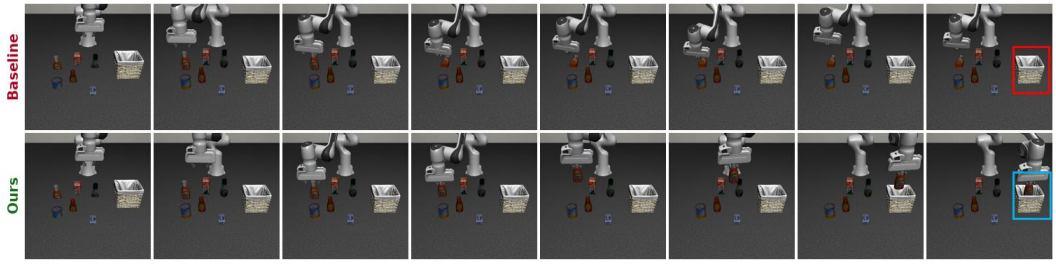


(d) OpenVLA-OFT - Background, LIBERO-Goal. Task: “put the cream cheese in the bowl.”

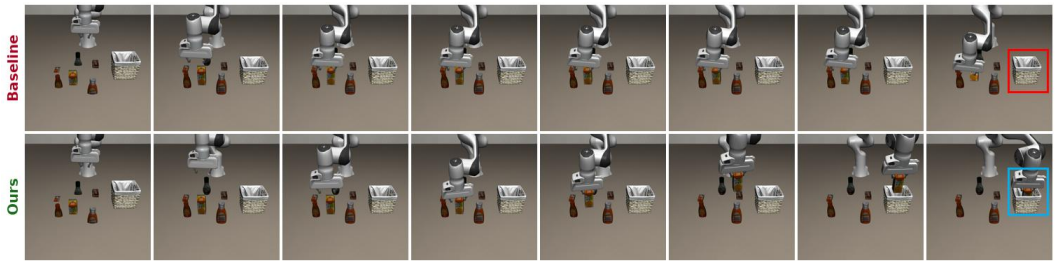
Figure 12: Qualitative examples under background perturbation. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



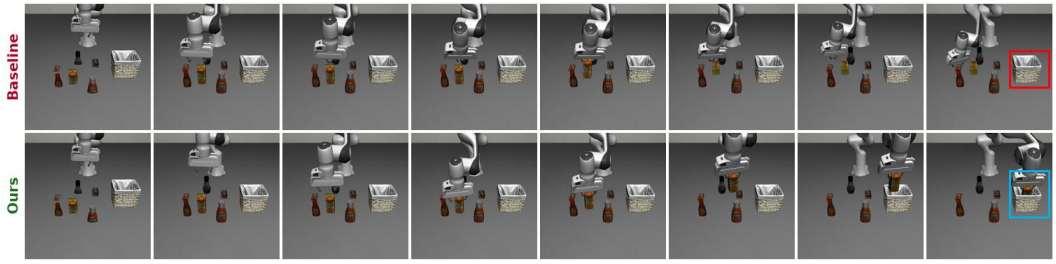
(a) OpenVLA-OFT - Background, LIBERO-Goal. Task: “open the middle drawer of the cabinet.”



(b) OpenVLA-OFT - Background, LIBERO-Object. Task: “pick up the ketchup and place it in the basket.”

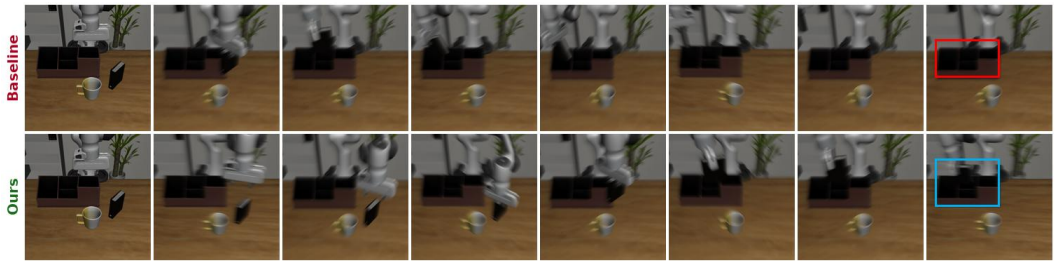


(c) OpenVLA-OFT - Background, LIBERO-Object. Task: “pick up the orange juice and place it in the basket.”

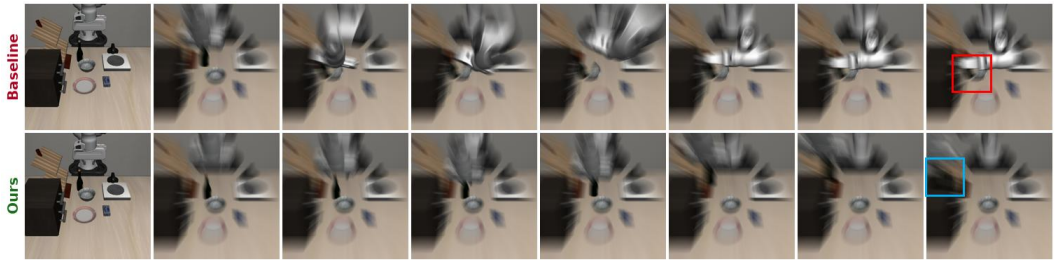


(d) OpenVLA-OFT - Background, LIBERO-Object. Task: “pick up the orange juice and place it in the basket.”

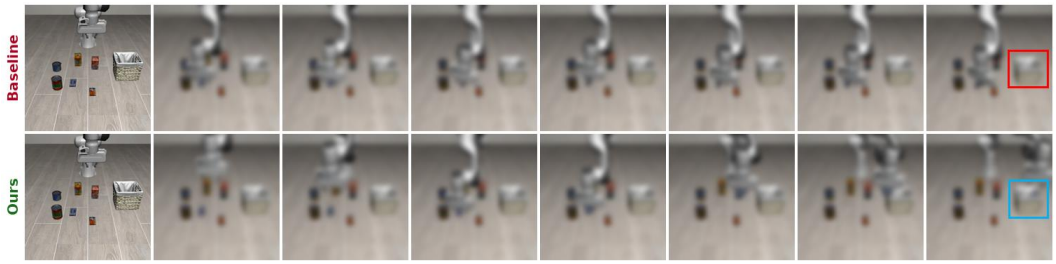
Figure 13: Additional qualitative examples under background perturbation. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



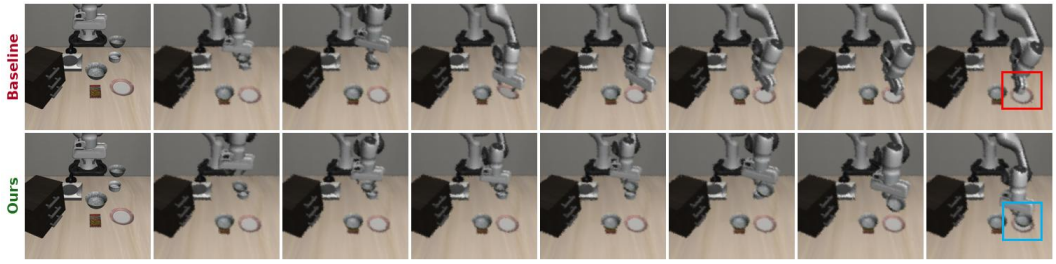
(a) OpenVLA-OFT - Sensor noise, LIBERO-10. Task: “pick up the book and place it in the back compartment of the caddy.”



(b) OpenVLA-OFT - Sensor noise, LIBERO-Goal. Task: “put the wine bottle on top of the cabinet.”



(c) OpenVLA-OFT - Sensor noise, LIBERO-Object. Task: “pick up the cream cheese and place it in the basket.”

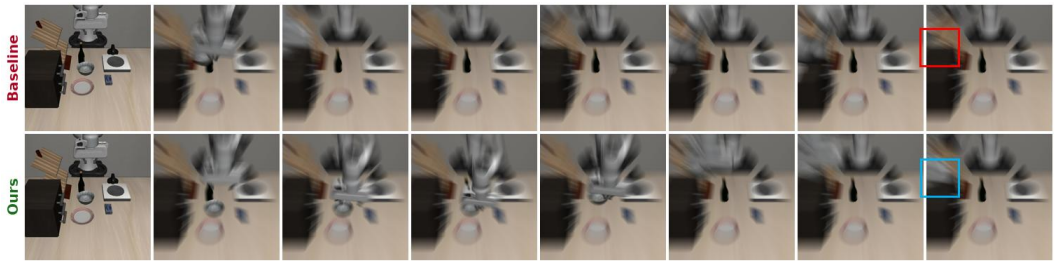


(d) OpenVLA-OFT - Sensor noise, LIBERO-Spatial. Task: “pick up the black bowl on the ramekin and place it on the plate.”

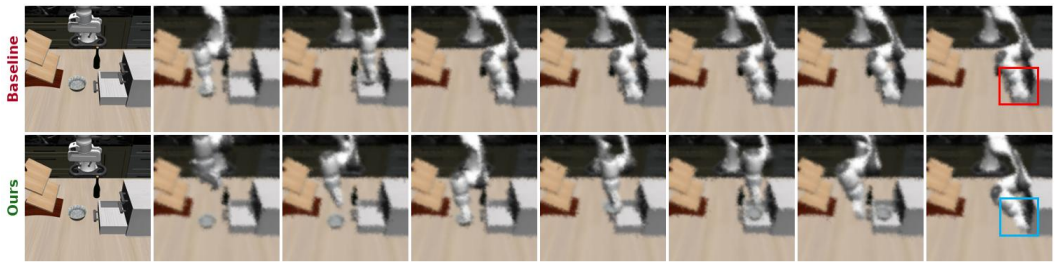
Figure 14: Qualitative examples under sensor noise perturbation. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



(a) OpenVLA-OFT - Sensor noise, LIBERO-10. Task: “put the black bowl in the bottom drawer of the cabinet and close it.”

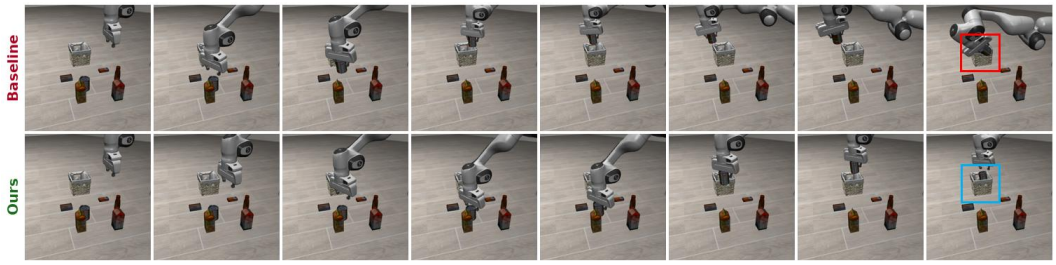


(b) OpenVLA-OFT - Sensor noise, LIBERO-Goal. Task: “put the bowl on top of the cabinet.”

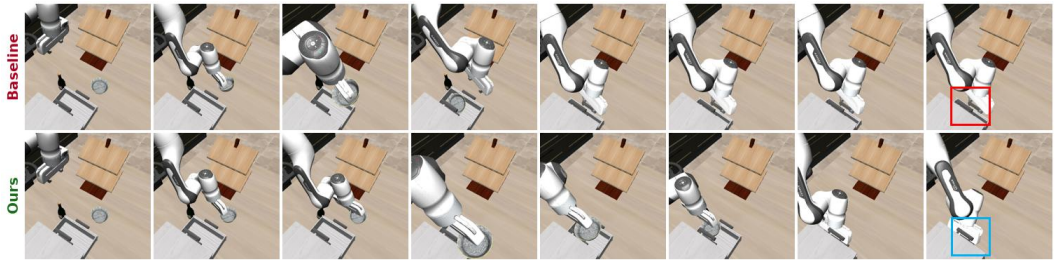


(c) OpenVLA-OFT - Sensor noise, LIBERO-10. Task: “put the black bowl in the bottom drawer of the cabinet and close it.”

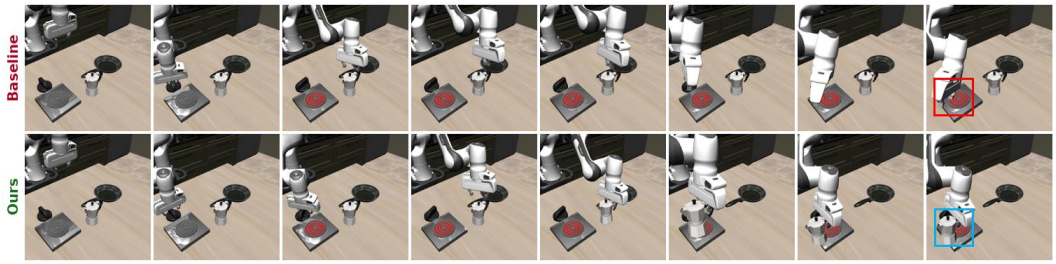
Figure 15: Additional qualitative examples for OpenVLA-OFT under sensor noise perturbation. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.



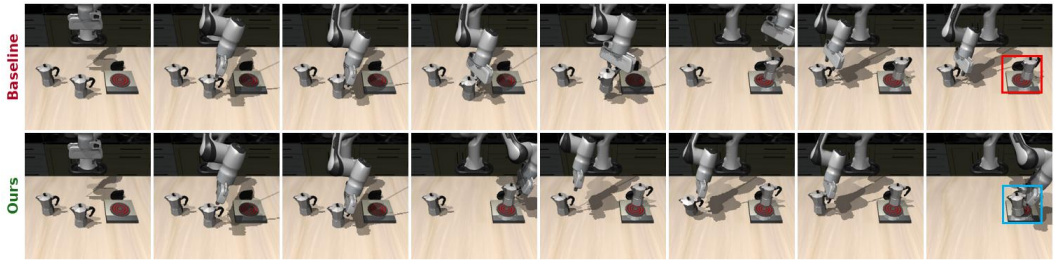
(a) $\pi_{0.5}$ - Camera, LIBERO-Object. Task: "pick up the tomato sauce and place it in the basket."



(b) $\pi_{0.5}$ - Camera, LIBERO-10. Task: "put the black bowl in the bottom drawer of the cabinet and close it."



(c) $\pi_{0.5}$ - Camera, LIBERO-10. Task: "turn on the stove and put the moka pot on it."



(d) $\pi_{0.5}$ - Light, LIBERO-10. Task: "put both moka pots on the stove."

Figure 16: Qualitative examples for $\pi_{0.5}$ under camera and light perturbations. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.

action mode, with the most selective features frequently reaching AUROC values near 1.0. Thus, high mode selectivity is not unique to compensators. Rather, mode selectivity provides an action-space interpretation of what a discovered feature represents. Compensators are distinguished by the additional requirements that they are perturbation-sensitive and that their perturbation-induced activation shifts are coupled to action changes.

Table 20: Top-3 mode-selective SAE features for each $\pi_{0.5}$ action mode. AUROC is computed on shared-state base forward passes.

Mode	#1	#2	#3
m_0	E:L01F1281 (1.00)	E:L17F0398 (1.00)	E:L00F1015 (1.00)
m_1	E:L03F0980 (0.99)	E:L04F0562 (0.98)	E:L01F1013 (0.98)
m_2	B:L14F0899 (1.00)	B:L13F0010 (1.00)	B:L17F1307 (0.99)
m_3	E:L00F0182 (0.94)	E:L06F0046 (0.92)	E:L00F0682 (0.92)
m_4	E:L00F0865 (0.99)	E:L00F0467 (0.99)	E:L02F1900 (0.98)
m_5	E:L00F0123 (0.99)	E:L01F1452 (0.98)	E:L06F0379 (0.97)
m_6	E:L03F0894 (0.94)	E:L00F0654 (0.93)	E:L01F0127 (0.93)
m_7	E:L02F2019 (1.00)	E:L00F0753 (0.99)	E:L16F0409 (0.99)
m_8	B:L11F0747 (0.98)	B:L16F1779 (0.98)	B:L10F1373 (0.98)
m_9	E:L14F1335 (1.00)	E:L03F1003 (0.99)	E:L04F0913 (0.99)

Table 21: Top-3 mode-selective SAE features for each OpenVLA-OFT action mode. AUROC is computed on shared-state base forward passes.

Mode	#1	#2	#3
m_0	L31F3291 (0.99)	L25F4494 (0.98)	L23F4722 (0.98)
m_1	L21F2117 (1.00)	L18F3312 (1.00)	L17F2391 (1.00)
m_2	L23F1977 (0.99)	L21F6218 (0.99)	L31F1231 (0.98)
m_3	L28F3237 (0.93)	L23F7158 (0.91)	L24F0051 (0.91)
m_4	L24F0114 (0.91)	L30F7616 (0.89)	L25F1186 (0.88)
m_5	L25F0114 (0.98)	L28F6936 (0.98)	L24F5215 (0.97)
m_6	L31F7760 (0.99)	L24F5470 (0.98)	L26F5629 (0.98)
m_7	L31F1935 (0.97)	L25F1521 (0.95)	L23F3327 (0.95)

J.2 Per-task consistency of mode selectivity.

We next test whether the action mode associated with a deployed feature is consistent across tasks rather than specific to a particular object configuration or goal. For each feature, we compute its mode-selectivity AUROC separately for every base task in which the feature occurs. Table 22 reports the mean and minimum per-task AUROC, together with the fraction of tasks for which the AUROC exceeds 0.6. The deployed compensators are consistently selective for the same action mode across tasks. For example, E:L12F0801, E:L00F0563, L23F4722, and L30F5719 exceed an AUROC of 0.6 in every task in which they occur. This consistency supports their interpretation as task-invariant motion primitives rather than features tied to individual objects, goals, or task instances. Detector features are less mode-selective and less consistent across tasks.

J.3 Mode-conditioned perturbation statistics.

Finally, we examine how visual perturbations affect execution of the motion primitive associated with each deployed compensator. For each feature, we restrict the analysis to shared-state pairs for which the base forward pass is assigned to the feature's most selective mode. We then measure how often the perturbed forward pass leaves that mode and how its commitment to the original mode changes.

Table 23 shows that perturbations usually preserve the executed action mode. Only 10–23% of paired forward passes leave the compensator's mode. Instead, the more consistent effect is a reduction in commitment to the same mode. Commitment decreases in 63–80% of paired states, with mean reductions ranging from 5% to 10% relative to the base commitment. These results support the

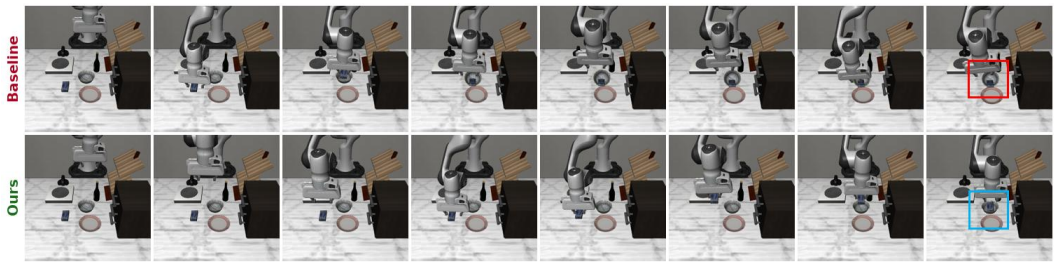
Table 22: Per-task consistency of mode selectivity for the deployed features. Task AUROC reports the mean and minimum mode-selectivity AUROC across the base tasks in which each feature occurs.

			AUROC		
Feature (role)	Pert.	Tasks	Mean	Min	> 0.6
$\pi_{0.5}$					
E:L12F0801 (comp.)	Camera	11	0.97	0.83	100%
E:L00F0563 (comp.)	Background	38	0.96	0.73	100%
E:L02F1761 (comp.)	Sensor	12	0.82	0.28	83%
B:L16F0279 (det.)	Sensor	22	0.71	0.43	77%
OpenVLA-OFT					
L23F4722 (comp.)	Camera	13	0.95	0.71	100%
L30F5719 (comp.)	Camera	37	0.93	0.66	100%
L15F6110 (det.)	Camera	24	0.82	0.51	96%

interpretation in Section 5: visual perturbations more often weaken the policy's commitment to the motion primitive it is already executing than replace it with a different primitive.

Table 23: Mode-conditioned perturbation statistics for deployed compensators. Statistics are computed over shared-state pairs whose base forward pass is assigned to the compensator's most selective mode. Mode exit is the fraction of perturbed passes that leave the mode, with the ratio relative to the exit rate of other modes in parentheses.

Feature	Pert.	Mode exit (ratio)	Commit. down	Δc
$\pi_{0.5}$				
E:L12F0801	Camera	23% (1.9×)	63%	−5%
E:L00F0563	Background	10% (1.4×)	79%	−7%
E:L02F1761	Sensor	19% (1.9×)	67%	−6%
OpenVLA-OFT				
L23F4722	Camera	20% (1.9×)	80%	−10%
L30F5719	Camera	13% (1.2×)	72%	−8%



(a) $\pi_{0.5}$ - Background, LIBERO-Goal. Task: "put the cream cheese in the bowl."



(b) $\pi_{0.5}$ - Sensor noise, LIBERO-10. Task: "put both the alphabet soup and the tomato sauce in the basket."

Figure 17: Additional qualitative examples for $\pi_{0.5}$ under background and sensor noise perturbations. For each example, the top row shows the perturbed baseline and the bottom row shows detector-compensator steering.