LIBERO-Plus: A Progressive Robustness Benchmark for Visual-Language-Action Models

Supplementary Material

A. Perturbation Dimensions

We conducted a comprehensive review of existing studies aimed at evaluating the generalization performance of VLA models, particularly those introducing new test suites such as COLOSSEUM [21], RL4VLA [17], AGNOSTOS [26], *etc.* The comparison of these methods is summarized in Table 1. Based on a systematic analysis of their task paradigms, environment construction, data collection pipelines, and evaluation dimension designs, this study ultimately identified seven core dimensions of perturbation: objects layout, environment background sampling, light variations, camera-view shifts, robotarm initialization perturbations, LLM-based language rewrites, and image noise, with the goal of testing model robustness and generalization ability across all modalities of input (vision, state, language). Each dimension contains multiple quantifiable sub-dimensions defined to enable fine-grained evaluation of model performance.

The perturbed examples are shown in Figures 15–20.

A.1. Objects Layout

This dimension is designed to test model robustness against object-level disturbances. It is further divided into two sub-dimensions:

- **O1: Confounding Objects.** Randomly add n additional unseen objects into the task scene. The object categories are drawn from a predefined set of 416 distractor objects. This perturbation is implemented by modifying the task description files (BDDL). In the benchmark, related perturbations are stored in BDDL files with an *add* suffix.
- **O2: Target Object Pose.** Apply random perturbations to the target object's initial position (x, y, z) and orientation (pitch, yaw, roll). This perturbation does not alter the target object itself and ensures that essential semantic relations to other objects remain unchanged (e.g., in the task *pick up the black bowl next to the cookie box and place it on the plate*, the relation to the cookie box determines the target object, and our modifications do not alter this constraint).

A.2. Background Textures

This dimension evaluates the model's ability to generalize to different background textures of the scene. It contains two sub-dimensions:

- **B1: Scene Theme.** Change the scene texture of the environment (e.g., from painted wall to brick wall). The new textures are sourced from a curated collection of 950 textures. This perturbation is implemented by modifying the scene XML definition files and registering new scene classes.
- **B2: Surface Appearance.** Randomly alter the texture of the working surface (e.g., tabletop or floor).

A.3. Light Conditions

This dimension evaluates the model's visual understanding under different lighting conditions. It includes four sub-dimensions, all implemented by modifying scene XML definition files:

- **L1: Diffuse.** The diffuse color, which defines the light color uniformly reflected by object surfaces (adjusted via RGB channels; e.g., 1 0 0 indicates red diffuse light, making objects appear reddish).
- **L2: Direction.** Change the direction of the parallel light source, which significantly affects color rendering and shading.
- **L3: Specular.** The intensity of the specular highlight on object surfaces (e.g., the bright spot reflected on metals). Larger values yield more distinct highlights, strongly influencing scene style.
- **L4: Shadows.** Boolean variable (true/false) indicating whether shadows of the robot arm and objects are cast in the scene.

A.4. Camera Viewpoints

This dimension tests the model's view-free representation and generalization ability by changing camera viewpoints. All perturbations are implemented by modifying the *Problem* class interface, with parameters derived from task filenames:

- **C1: Camera Distance.** Move the camera along its optical axis, changing the distance to the scene center. Camera distances are valued among $1.01\times$ to $2.00\times$ the original value.
- **C2: Spherical Position.** Perturb camera position on a sphere centered at the scene, altering azimuth $(\Delta\theta)$ and elevation $(\Delta\phi)$ within $15°$–$75°$ cones.
- **C3: Camera Orientation.** Fix the camera position but perturb its orientation (yaw, pitch, roll), valued within $2°$ to $10°$.

A.5. Robot Initial States

- **Initial Joint Angle.** Random perturbations are applied to the robot arm's initial joint positions (qpos). Perturbation

magnitudes are valued from 0.1 to 0.5. This perturbation is implemented by modifying the *Problem* class interface.

A.6. Language Instructions

This dimension employs large language models (LLMs) to rewrite original task instructions, testing model generalization and reasoning ability in natural language:

- **R1: Distraction.** Task instructions are rewritten into longer and more conversational forms that contain additional but task-irrelevant contextual cues.
- **R2: Common Sense.** Replacing the existing object descriptions with commonsense-based descriptions to test information extraction and filtering.
- **R3: Reasoning Chain.** For multi-step reasoning instructions, perturbations involve altering reasoning complexity.

Table 3. Examples of Language Instruction Rewriting

Sub-category	Examples
Original	push the plate to the front of the stove
R1	before turning on the burner, push the plate to the front of the stove
R2	propel the flat surface used for holding food toward the area designated for cooking heat adjustment
R3	make sure the plate ends up at the front of the stove

A.7. Sensor Noise

This dimension simulates real-world sensor imperfections to evaluate robustness under degraded input quality:

- **N1: Motion Blur.** Simulates blur caused by relative motion between camera and scene. Higher levels correspond to larger blur kernels, longer trajectories, and more severe blur.
- **N2: Gaussian Blur.** Simulates optical blur caused by defocus. Higher levels correspond to larger kernel size and standard deviation, resulting in smoother images with greater detail loss.
- **N3: Zoom Blur.** Simulates radial blur caused by rapid zoom during exposure. Higher levels increase zoom center and blur intensity, producing strong vignetting.
- **N4: Fog.** Simulates atmospheric interference such as fog or haze. Higher levels increase fog density and brightness, lowering image contrast and saturation.
- **N5: Glass Blur.** Simulates distortions and refractions caused by viewing through textured glass. Higher levels increase distortion amplitude and range, resulting in severe local pixel displacements.

Perturbation parameters are shown in Table 4.

B. Model Details

This appendix provides comprehensive descriptions of all models evaluated in our study, covering their architectural designs, training data sources, and key implementation specifications. We aim to offer sufficient transparency such that the reported results can be faithfully reproduced and compared against future work.

B.1. Model Overview

We evaluate a diverse set of vision-language-action (VLA) models that represent different design choices in terms of architecture and training strategy, enabling us to systematically analyze how different factors contribute to task performance and robustness. For each model, we summarize its backbone, modality encoders, fusion mechanisms, and decision heads.

B.2. OpenVLA [11] and OpenVLA-OFTs [12]

Base Architecture. OpenVLA adopts a modular vision-language architecture built on the Prismatic-7B VLM. The visual encoder is a 600M-parameter dual-backbone composed of SigLIP and DINOv2, whose outputs are concatenated along the channel dimension to enhance spatial reasoning capabilities crucial for robotic control. A lightweight two-layer MLP projector maps the fused visual features into the input space of a Llama2-7B language backbone, which integrates visual and textual inputs through cross-attention. This design enables OpenVLA to leverage both semantic understanding and spatial grounding for action prediction. To adapt the VLM backbone for robotic control, continuous robot actions are discretized into 256 bins per dimension and represented as tokens within the LLM vocabulary. The 256 least frequently used tokens of the Llama tokenizer are replaced by action tokens, and training proceeds with the standard next-token prediction objective applied to action sequences.

Training Strategy. The training pipeline consists of two stages: an initial pre-training followed by supervised fine-tuning. OpenVLA is pre-trained on the Open X-Embodiment (OpenX) dataset, which includes over 970k robot trajectories across multiple embodiments and tasks. The model is trained end-to-end with a cross-entropy loss applied exclusively to the action tokens. Unlike typical VLM practices, the vision encoder is fine-tuned rather than frozen, enabling the model to capture fine-grained spatial details crucial for robotic control.

Variants. In addition to the baseline OpenVLA models, we consider the OpenVLA-OFT family of variants:

- **OpenVLA-OFT:** A parallel decoding variant enabling simultaneous prediction of all actions in a single forward pass. It employs continuous action representations through a multi-layer MLP head and is trained with an L1 regression objective, resulting in faster inference

Table 4. Noise perturbation parameters.

ID	Noise Type	Key Parameters	Description of L1–L5
1	Motion Blur	Radius r, Gaussian kernel σ, angle θ	r and σ control blur strength (kernel size and spread). From weak blur ($r = 5, \sigma = 2$) to strong blur ($r = 35, \sigma = 20$). $\theta \sim U(-45°, 45°)$ determines blur direction.
2	Gaussian Blur	Standard deviation σ	σ controls the amount of smoothing. Small σ produces slight blur ($\sigma = 1$), large σ produces heavy blur ($\sigma = 10$).
3	Zoom Blur	Scaling factors $[s_{min}, s_{max}, step]$	Successive rescaling creates a zoom-like blur. Weak effect at $([1, 1.11, 0.01])$ and strong effect at $([1, 1.56, 0.03])$.
4	Fog	Density α, decay rate β	α controls fog thickness, β controls how quickly fog attenuates. Light fog ($\alpha = 0.5, \beta = 3.0$) $\rightarrow$ Dense fog with slow decay ($\alpha = 5.0, \beta = 1.3$).
5	Glass Blur	Gaussian blur σ, pixel displacement δ, iteration count	σ defines baseline blur, δ controls the displacement of pixels, and iterations determine the accumulation of distortions. Light blur with small displacements ($\sigma = 0.5, \delta = 1, iters = 3$) $\rightarrow$ Strong blur with large displacements ($\sigma = 2.5, \delta = 5, iters = 1$).

Table 5. Model HuggingFace repository addresses.

ID	Model Name	Checkpoint Address
1	OpenVLA	https://huggingface.co/openvla
2	OpenVLA-OFT	https://huggingface.co/moojink
3	OpenVLA-OFT_m	https://huggingface.co/moojink/openvla-7b-oft-finetuned-libero-spatial
4	NORA	https://huggingface.co/declare-lab
5	WorldVLA	https://huggingface.co/Alibaba-DAMO-Academy/WorldVLA
6	UniVLA	https://huggingface.co/qwbu/univla-7b-224-sft-libero
7	π_0	https://storage.googleapis.com/openpi-assets/checkpoints/pi0_libero
8	π_0-Fast	https://storage.googleapis.com/openpi-assets/checkpoints/pi0_fast_libero
9	RIPT-VLA	https://huggingface.co/tanshh97/RIPT_VLA

and more precise action generation, and it incorporates Feature-wise Linear Modulation (FiLM) to enhance language grounding.

- **OpenVLA-OFT_w:** A variant of OpenVLA-OFT that removes the first-person wrist camera input and retains only the third-person view. This model is trained from OpenVLA with the official OFT hyperparameters on four LIBERO benchmark suites for 150K steps using 8×A100 GPUs.
- **OpenVLA-OFT_m:** A mixed-training variant that adopts the official mix-SFT weights. Unlike suite-specific training, this model is jointly trained across all four LIBERO suites, enabling it to learn from a broader distribution of tasks and environments.

B.3. π_0 [2] and π_0-fast [20]

Base Architecture. The π_0 architecture is inspired by the Transfusion framework, which trains a single Transformer with multiple objective functions: a flow-matching loss for continuous output tokens and a cross-entropy loss for discrete tokens. Building upon this, π_0 implements two sets of transformer weights (one initialized from the VLM and a smaller action expert). The core model comprises a VLM base (PaliGemma) for semantic understanding of multimodal inputs (multiple RGB images, language instructions, and proprioceptive state q_t), and action tokens are projected

and routed to a smaller action-expert.

Training Strategy. The training follows a two-stage paradigm: (i) large-scale, diverse pre-training on a mixture dataset to learn broad capabilities and recovery behaviors; (ii) post-training on smaller, high-quality curated datasets to induce dexterity and fluent task execution. The pre-training mixture is carefully reweighted to avoid over-representation.

π_0-fast: Efficient Action Tokenization. The π_0-fast variant introduces the FAST tokenization method to compress action sequences. FAST combines a Discrete Cosine Transform (DCT) for converting temporal action trajectories into a sparse frequency-domain representation, followed by Byte-Pair Encoding (BPE) to losslessly compress the sparse DCT coefficient matrix into dense tokens.

B.4. Nora [9]

Base Architecture. NORA is a 3B-parameter general-purpose VLA model optimized for robotic tasks. It adopts the Qwen-2.5-VL-3B multimodal model as its backbone, chosen for its strong visual-semantic understanding capabilities, which enhance visual reasoning and action grounding. The model processes natural language task instructions and single-frame (as per its implementation) visual observations as input. It outputs discrete action sequences by employing the FAST+ tokenizer to discretize continuous action tokens.

Training Strategy. NORA is pre-trained on the Open X-Embodiment dataset, which includes trajectories from various robots performing diverse tasks. This phase aims to equip the model with broad robotic capabilities and strong generalization. Training was conducted on 8×H100 GPUs for approximately three weeks (totaling 4000 GPU hours), using the AdamW optimizer with a batch size of 256 over 1.1 million gradient steps. A linear warmup followed by cosine decay was applied to the learning rate.

B.5. WorldVLA [5]

Base Architecture. WorldVLA is an autoregressive action-world model that unifies visual-language-action (VLA) modeling and world modeling within a single, integrated framework. The core idea is to jointly learn a policy model for action generation and a world model for future state prediction, allowing the two components to mutually enhance each other. The model is initialized from Chameleon, a unified image understanding and generation model. It employs three tokenizers: a VQ-GAN-based image tokenizer, a BPE-based text tokenizer, and an action tokenizer that discretizes each dimension of the continuous robot action into 256 bins. All modalities (text, image, action) are discretized into tokens and modeled autoregressively within a unified sequence. A key architectural innovation is a customized attention mask for action generation that prevents the current action from attending to previous actions, thereby mitigating error propagation and enabling more robust parallel action chunk prediction.

Training Strategy. The model is trained on a mixture of action-modeling data and world-modeling data. The action-modeling data trains the model to generate action chunks given a language instruction and a history of image observations, using a loss computed only on the action tokens. The world-modeling data trains the model to predict the next image frame given the current image and action, using a loss computed only on the image tokens. This joint training strategy encourages the learning of shared representations: the world model acquires an understanding of environmental physics to aid task-relevant action generation, while the action model enhances visual understanding to support accurate frame prediction. The model is evaluated on the LIBERO benchmark suite, with training leveraging 90% of the successful trajectories for training and 10% for validation.

B.6. UniVLA [14]

Base Architecture. UniVLA is a universal visual-language-action model that operates in a discrete, task-centric latent action space to achieve cross-embodiment generalization. The architecture is built upon a pre-trained Prismatic-7B VLM, which integrates a fused visual encoder (SigLip and DINOv2), a projection layer, and an LLaMA-2 LLM. A key innovation is the extension of the LLM's vocabulary with special action tokens to represent quantized latent actions. The model takes a visual observation and a language instruction as input and autoregressively predicts a sequence of these latent action tokens. For deployment on specific robots, a lightweight action decoder head is added, which uses multi-head attention pooling to map the predicted latent actions into the robot's executable low-level control space.

Training Strategy. The training process involves three stages. First, a latent action model is trained in a self-supervised manner on large-scale video datasets to learn a discrete codebook of task-centric actions. This model uses a DINOv2-based reconstruction objective and conditions on language instructions to disentangle task-relevant dynamics from irrelevant visual changes. Second, the universal policy is pre-trained to predict these latent action tokens from observations and instructions, leveraging the generalizable representations of the pre-trained VLM. This approach compresses the action space dramatically, leading to significantly faster convergence compared to methods operating in raw action spaces. Finally, for downstream adaptation, the entire model is fine-tuned end-to-end with a combined loss for latent action prediction and low-level action regression, often using parameter-efficient methods like LoRA. A history-augmented input scheme, where past latent actions are fed back as context, is employed to en-

hance performance in long-horizon tasks.

B.7. RIPT-VLA [3]

Base Architecture. The base model for RIPT-VLA is OpenVLA-OFT, a continuous-action VLA model where the action head is typically trained with an L1 regression loss. To make this architecture compatible with reinforcement learning, which requires a probabilistic policy output, RIPT-VLA augments the model with a lightweight auxiliary head that predicts the scale parameter σ_θ for the action distribution. The policy is then treated as a factorized Laplace distribution (for L1 loss) with the original model output as the mean μ_θ and the new head's output as the scale. This allows for sampling actions and computing the log-probability $\log \pi_\theta(a_t|a_{<t}, c)$ in closed form, which is essential for policy gradient updates.

Training Strategy. RIPT-VLA introduces a third stage of Reinforcement Interactive Post-Training (RIPT) following the standard pre-training and supervised fine-tuning (SFT) stages. The strategy is centered on the Dynamic Sampling Leave-One-Out PPO (LOOP) framework. In the rollout collection phase, for a given context c_i, K trajectories are sampled from the current policy. The RLOO advantage estimation method is used to compute advantages from the sparse binary rewards. A key innovation is a dynamic rejection mechanism that filters out context samples where all K rollouts receive identical rewards (all successes or all failures), thus ensuring that the training batch contains meaningful learning signals. During the policy optimization phase, the PPO algorithm is applied to the collected rollouts to maximize the expected task success rate, with the policy update constrained by the probability ratio $r_i = \pi_\theta(a_i|c_i)/\pi_\psi(a_i|c_i)$ to ensure stable training. This iterative process of data collection and optimization allows the model to improve its performance through environment interaction, specifically targeting and overcoming failure modes encountered during deployment.

C. Perturbations and Benchmark Construction

C.1. Data Generation and Filtering

We began with the 40 evaluation tasks from LIBERO and generated 500 instances for each of the four generalization sub-tasks (Spatial, Object, Goal, Long) across the seven generalization dimensions, resulting in an initial set of 14,000 candidate tasks. These tasks were evaluated using several widely adopted baseline models to assess performance distributions, as summarized in Section 4.

Tasks that were solved by all models, or by a large majority, were removed to avoid ceiling effects. We further balanced the remaining tasks across augmentation sub-dimensions to prevent bias. The final test-only benchmark consists of 10,030 tasks spanning all seven dimensions.

C.2. Dataset Composition

Table 6 presents the final distribution of evaluation tasks across generalization dimensions and sub-task categories.

Table 6. Distribution of the evaluation dataset across dimensions and different categories.

	Spatial	Object	Goal	Long	Total
Camera	376	396	408	419	1599
Robot	350	398	409	393	1550
Language	354	390	410	383	1537
Light	292	297	279	274	1142
Background	258	248	281	289	1076
Noise	351	422	379	449	1601
Layout	312	425	403	385	1525
Total	2293	2576	2569	2592	10030

C.3. Difficulty Assessment

We evaluated the 10,030 tasks using four representative models—OpenVLA-OFT, π_0, π_0-fast, and UniVLA—and stratified task difficulty based on how many of these models succeeded on each instance:

- Level 1 (L1): solved by all four models;
- Level 2 (L2): solved by exactly three models;
- Level 3 (L3): solved by exactly two models;
- Level 4 (L4): solved by exactly one model;
- Level 5 (L5): solved by none.

Figure 9 illustrates the proportion of tasks at each difficulty level for every dimension.

C.4. Model Performance by Difficulty Level

We further analyzed how model accuracy varies with task difficulty. Figure 10 shows the success rates of each model across the five difficulty levels.

D. Training Dataset Construction

D.1. Dataset Overview

The generalized training dataset consists of over 20,000 successful trajectories, covering a wide range of task variations and environment configurations. Figure 11 shows the distribution of the 7-dimensional robot actions in the dataset. The plots are arranged from top to bottom and left to right, corresponding to the seven action dimensions, respectively. This visualization demonstrates the diversity and coverage of the actions captured in the generalized dataset.

The dataset includes six types of task variants and environment modifications: objects spanning, environment sampling, light variations, camera-view shifts, LLM-based

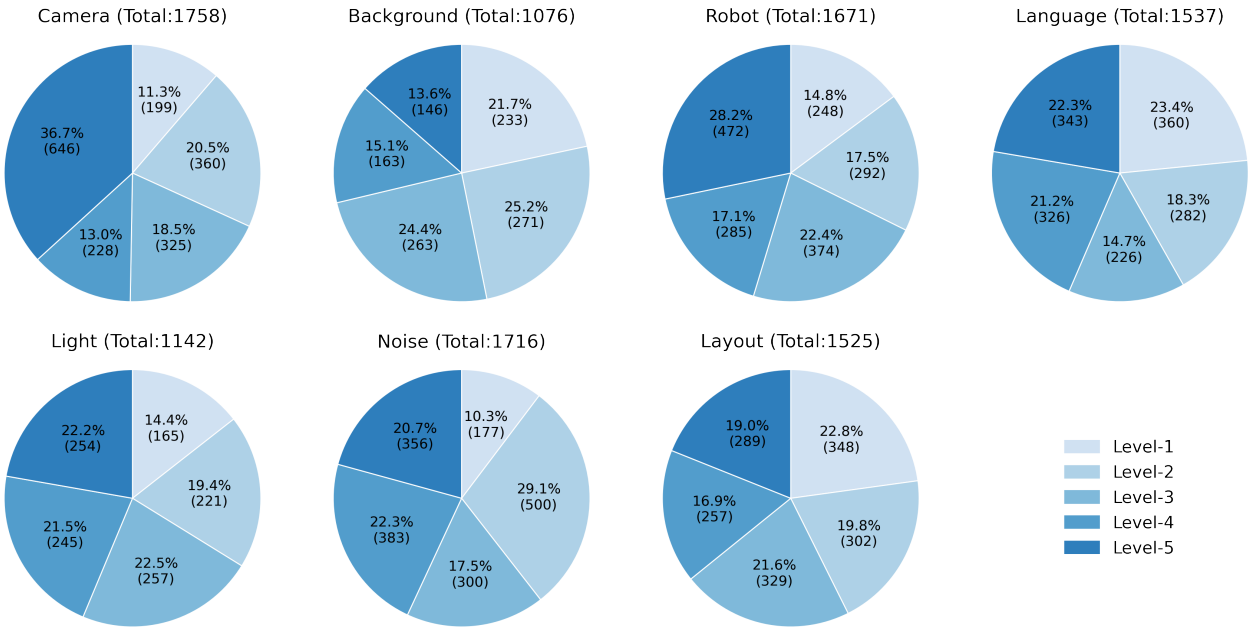


Figure 9. Proportion of tasks per difficulty level across the seven generalization dimensions.

language rewrites, and sensor noise. Among these, the objects spanning variant contains only compounding objects, which are generated by executing existing trajectories and selecting only the successful ones. Variants involving pose changes were not added due to the limited reliability of automatically generated trajectories.

D.2. Data Generation Process

The generalized dataset was constructed using the same automated generalization pipeline, with variations in parameters to produce diverse scenarios:

- **Objects Positioning:** For compounding objects, distractor objects and their poses were varied while ensuring no overlap with the test set.
- **Background Environment Sampling:** Additional textures for tables, walls, and floors were automatically sampled to avoid overlap with the test environments.
- **Light Variations:** Different lighting parameters were applied to the scenes.
- **Camera-view Shifts:** Camera angles differed by 5° on the spherical coordinate system compared to the test set.
- **LLM-based Language Rewrites:** New language instructions were generated to provide additional linguistic diversity.
- **Image Noise:** Sensor noise parameters differed from those listed in Table 4.

D.3. Trajectory Collection

Trajectory collection was performed using the original LIBERO dataset's $(\text{state}, \text{action})$ pairs, executed in the newly generated environments. Only successful trajectories were retained, and any actions corresponding to no-ops were filtered out. Specifically, 2,400 trajectories were collected for the compounding object variant, while 4,000 trajectories were collected for each of the other variants, resulting in a total of 22,400 trajectories. After filtering, over 20,000 high-quality trajectories were retained for training.

D.4. Training Configuration

Using this dataset, we performed mixed fine-tuning based on the official OpenVLA-OFT weights. The training was conducted on $8 \times$ A100 GPUs with a learning rate of 5×10^{-4} for 100,000 steps. The batch size was set to 64 per GPU, resulting in an effective batch size of 512. We employed the AdamW optimizer with weight decay of 0.1 and used a cosine learning rate schedule with warmup. The training results on LIBERO-plus are shown in Table **??**.

D.5. Storage Format

All trajectories are stored in the `rlds` format, consistent with standard practices for robotics datasets and ensuring compatibility with existing training pipelines.

E. Goal Replacement Rollout Cases Analysis

To further probe whether Vision-Language-Action (VLA) models genuinely understand and act upon natural language

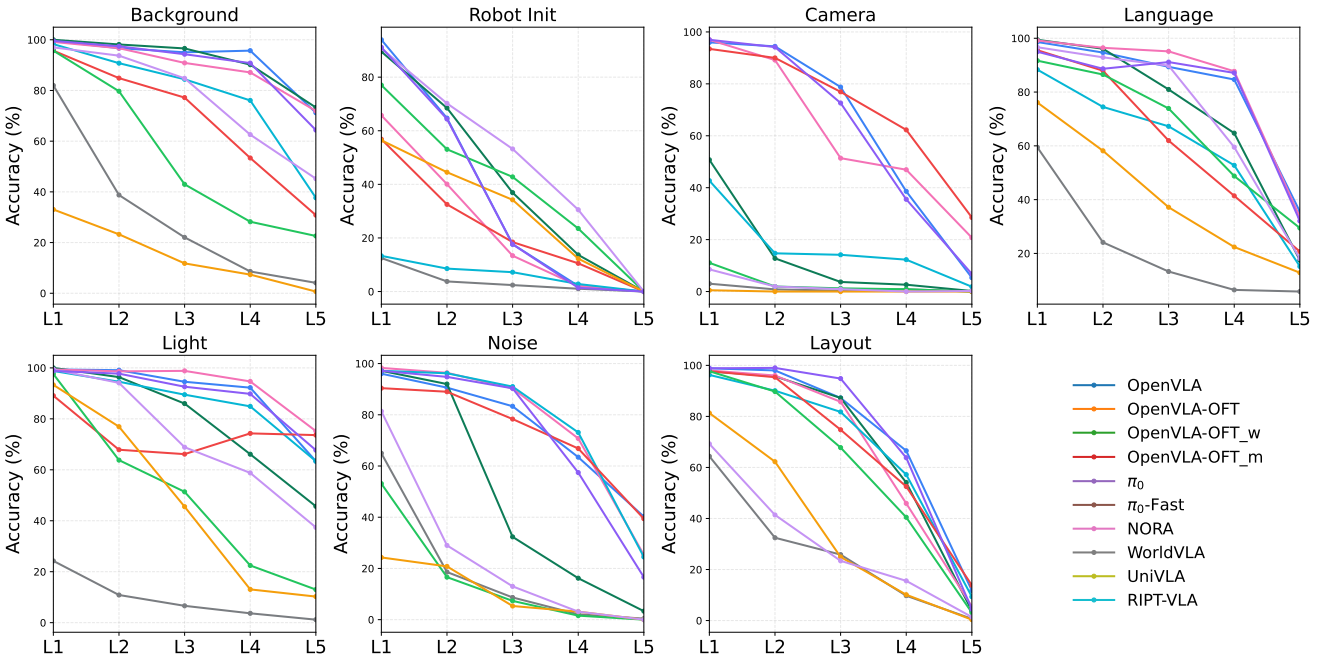


Figure 10. Model performance trends across perturbation difficulty levels. The line plots show the success rate of each model as the intensity of all seven different perturbation dimensions increases.

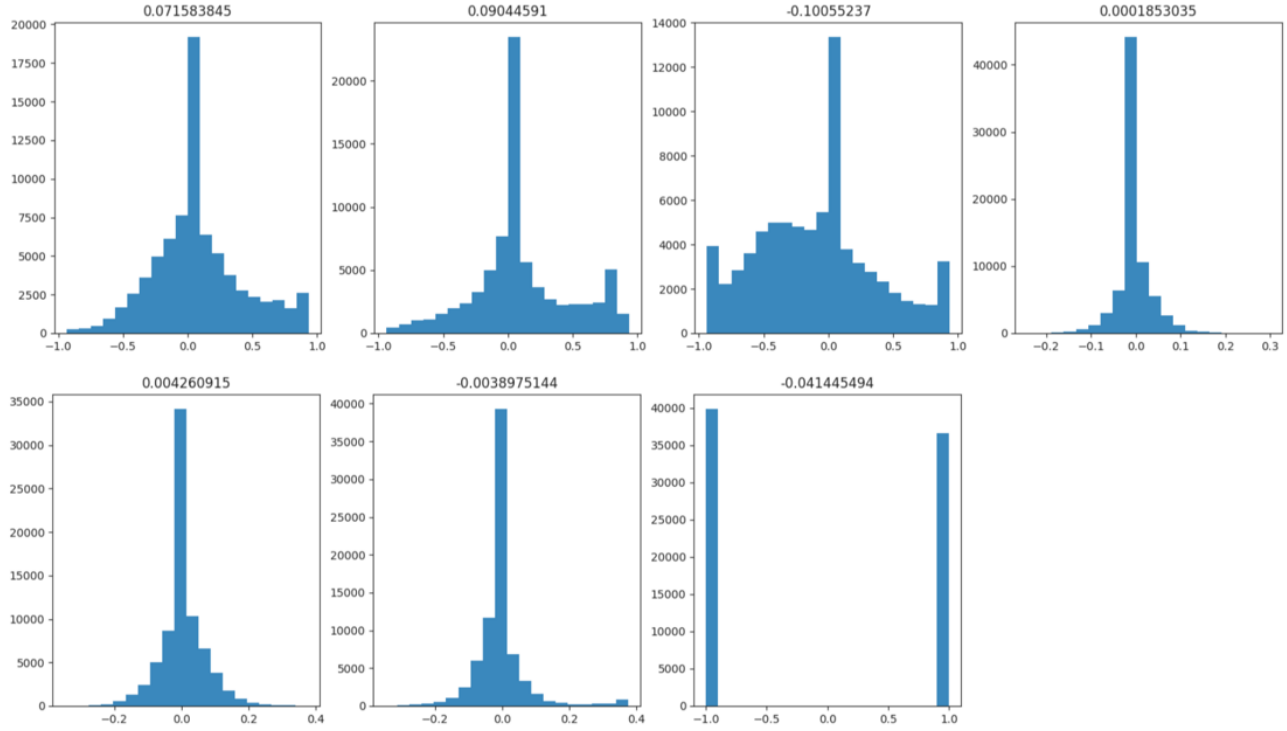


Figure 11. Distribution of the 7-dimensional robot actions in the generalized dataset. Plots are arranged from top to bottom and left to right, corresponding to action dimensions 1–7.

instructions, we designed a **goal replacement** evaluation. In this task, the target object specified in both the instruction and the task goal was replaced with an alternative object from the same scene, while keeping the rest of the environment unchanged. For example, an original instruction such as *"pick up the alphabet soup and place it in the basket"* could be modified to *"pick up the tomato sauce and place it in the basket"*. We performed this manipulation on the *object* suite, where misalignment between model actions and instructions was most pronounced. Figure 3(b) summarizes the performance drop across models, while the rollout cases in Figure 14 reveal how these degradations manifest in execution.

From these results, we observed two key patterns:

1. **Lack of cross-object generalization in instruction following.** Across all tested instances, models failed to adapt to the new target specified in the instruction, with success rates in replaced-target tasks dropping nearly to zero. This drop was particularly dramatic for OpenVLA-OFT, whose accuracy in the modified target setting diminished from high baseline values to almost complete failure. This confirms that the robustness observed in earlier language perturbation experiments did not originate from true linguistic comprehension—the models appear to ignore linguistic signals and rely instead on fixed, learned perception–action associations.
2. **Over-reliance on fixed vision–action mappings rather than dynamic instruction-based planning.** In nearly all rollout cases (Figure 14), the model performed the original action for the original target even when the instruction had explicitly changed. For example:
   - In case (a), the new instruction specified picking up the *butter*, yet the model still picked up the *alphabet soup* as in the original task.
   - In case (c), the model was instructed to pick up *tomato sauce*, but executed the original *butter* action.
   - Similar behavior was observed in (d) and (e), where the model persisted with the original target (e.g., *chocolate pudding*, *cream cheese*) rather than adjusting to the new goal.

These behavioral patterns indicate that the VLA models in our study function more like "visual pattern matchers" mapping scene configurations to predetermined action sequences, rather than integrating task-relevant

F. Details of the Compositional Generalization Experiments

F.1. Success Rate Record

Table 7 reports the pairwise success rates across different perturbation dimensions. Each diagonal entry corresponds to the performance under a single perturbation dimension, while the off-diagonal entries represent joint perturbations of two dimensions. This analysis allows us to examine not only the robustness of models to isolated disturbances, but also the interaction effects between multiple perturbations, which are critical for assessing compositional generalization in realistic robotic scenarios.

F.2. Significance Experiments for Compositional Generalization

To ensure that the observed deviations between the expected product-based success rates and the actual joint success rates are not due to random chance, we conduct significance experiments. Specifically, we aim to statistically validate whether the negative compositional gaps indeed reflect systematic interaction effects between perturbations, rather than sampling noise arising from finite trials. For this purpose, we adopt the classical chi-square test for independence.

Let n_{00} be the number of samples succeeding under neither of the two perturbations, n_{01} the number succeeding under perturbation 2, n_{10} the number succeeding under perturbation 1, and n_{11} the number succeeding under both perturbations.

- **Chi-square test for independence:** To statistically assess whether the deviation is significant, we consider the 2×2 contingency table of success counts under perturbations D_i and D_j:

	$D_j = 0$	$D_j = 1$	Total
$D_i = 0$	n_{00}	n_{01}	$n_{0\cdot}$
$D_i = 1$	n_{10}	n_{11}	$n_{1\cdot}$
Total	$n_{\cdot 0}$	$n_{\cdot 1}$	n

  The chi-square statistic is then given by

$$\chi^2 = \sum_{r,c} \frac{(O_{rc} - E_{rc})^2}{E_{rc}},$$

  where O_{rc} denotes the observed count and $E_{rc} = \frac{(\text{row total}) \times (\text{column total})}{n}$ is the expected count under the independence hypothesis.
- **p-value:** Given the chi-square statistic χ^2 and the corresponding degrees of freedom (here dof $= 1$ for a 2×2 table), the *p-value* is the probability of observing a test statistic at least as extreme as χ^2 under the null hypothesis of independence:

$$p = P\left(\chi^2_{\text{dof}=1} \geq \chi^2\right),$$

  where $\chi^2_{\text{dof}=1}$ denotes a chi-square distribution with 1 degree of freedom. A small p-value (e.g., < 0.05) indicates strong evidence against the independence assumption.

A large χ^2 value (with a small p-value) indicates that the joint success/failure distribution under perturbations d_i and

Table 7. Pairwise evaluation results across different perturbation dimensions.

	Layout	Background	Light	Camera	Robot	Noise
Object	**71.75**	–	–	–	–	–
Background	57.00	**85.75**	–	–	–	–
Light	57.20	67.10	**82.10**	–	–	–
Camera	35.95	37.70	39.65	**57.30**	–	–
Robot	24.40	29.95	29.65	19.05	**39.10**	–
Noise	44.55	51.05	54.00	36.70	22.15	**71.50**

d_j deviates significantly from the independence assumption, implying interaction effects between the two perturbations. Conversely, a small χ^2 (large p-value) suggests no evidence against independence.

Table 8. Chi-square test results for perturbation pairs

Perturbation A	Perturbation B	Chi-square	p-value
Layout	Env	4.09	4.32e-02
Layout	Light	1.23	2.68e-01
Layout	Camera	7.55	6.01e-03
Layout	Robo init	6.13	1.33e-02
Layout	Noise	9.42	2.14e-03
Env	Light	2.37	1.24e-01
Env	Camera	26.1	3.33e-07
Env	Robo init	4.87	2.74e-02
Env	Noise	16.1	6.07e-05
Light	Camera	12.1	4.92e-04
Light	Robo init	2.79	9.48e-02
Light	Noise	4.53	3.34e-02
Camera	Robo init	6.76	9.31e-03
Camera	Noise	5.51	1.90e-02
Robo init	Noise	14.3	1.59e-04

From Table 8, it can be observed that most perturbation pairs yield large χ^2 values, with correspondingly tiny p-values, below conventional significance thresholds (0.05). This indicates that the joint distribution under different perturbations deviates strongly from the independence assumption, implying clear interaction effects between perturbations.

Overall, the results consistently demonstrate that perturbation interactions are significant and cannot be ignored when evaluating compositional generalization.

G. Attention Mask Analysis for Compositional Generalization

To better explore the failure mechanisms under compositional generalization, we analyze the attention masks of the OpenVLA-OFT model. The visualization results are shown in Fig. 12.

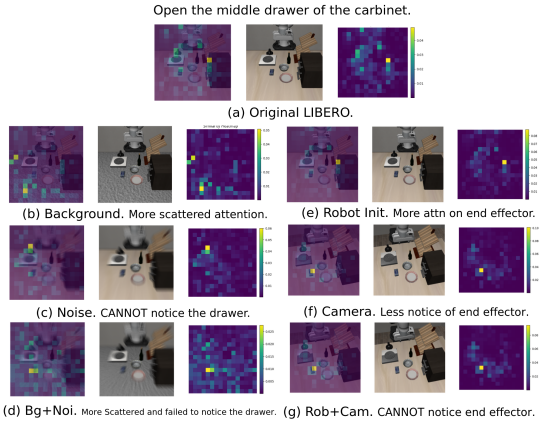


Figure 12. Attention masks under compositional perturbations. (a) Original LIBERO: the model correctly focuses on the target drawer. (b) Background perturbation: attention becomes more scattered. (c) Noise perturbation: the model fails to notice the drawer. (d) Background + Noise: attention is further scattered and the drawer is ignored. (e) Robot initialization perturbation: increased attention on the end effector. (f) Camera perturbation: reduced attention on the end effector. (g) Robot + Camera: the model fails to notice the end effector.

As shown in Fig. 12, the attention patterns reveal critical insights into model failures under different perturbation types. In the original LIBERO setting (a), the model's attention is properly concentrated on the target drawer. However, when background perturbations are introduced (b), the attention distribution becomes more scattered, indicating that the model struggles to maintain focus on the relevant object. Under noise perturbations (c), the model completely fails to notice the drawer. The combined background and noise perturbations (d) exacerbate this issue, leading to even more scattered attention and complete neglect of the target.

For perturbations related to robot and camera configurations, we observe distinct attention patterns. Robot initialization perturbations (e) cause the model to over-focus on the end effector, shifting attention away from the target object. Camera perturbations (f) lead to reduced attention on the end effector. When both robot and camera perturbations are applied simultaneously (g), the model fails to notice the

end effector altogether.

These attention mask analyses demonstrate that the model's performance degradation under compositional perturbations is closely tied to its inability to maintain focused attention on task-relevant regions. The scattering or shifting of attention patterns provides a mechanistic explanation for the failure cases observed in our compositional generalization experiments.

H. Real-World Robustness Evaluation

To further validate the model's robustness under compositional perturbations, we conduct real-world cookie picking experiments. The visualization of different perturbation types is shown in Fig. 13, and the quantitative results are presented in Tab. 9.

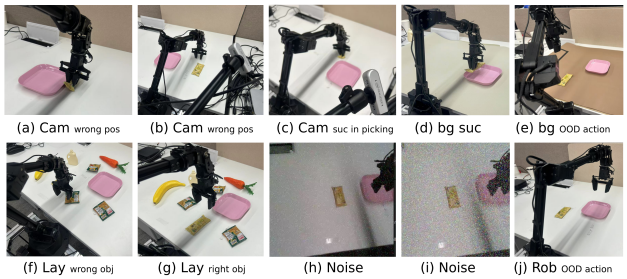


Figure 13. Visualization of real-world perturbations.

Table 9. Real-world robustness evaluation under different perturbation levels.

Suc Rate	Ori	Bg	Lay	Cam	Noi	Lang	Rob
L1	-	60	60	40	60	60	60
L2	-	20	20	40	40	40	40
L3	-	0	20	0	0	40	20
Avg.	**66.7**	26.7	33.3	26.7	33.3	46.7	40.0
Drop	-	40.0	33.4	40.0	33.4	20.0	26.7

The experimental results confirm the model's robustness under six perturbation dimensions, revealing several key insights:

Camera perturbations. Third-person views govern placement accuracy and are orientation-sensitive, while wrist views primarily handle grasping. The model demonstrates complementary use of multi-view inputs to maintain performance.

Visual perturbations. High visual similarity between objects (e.g., different cookie types or layouts) degrades performance, triggering out-of-distribution (OOD) actions. This suggests that the model relies heavily on visual cues for task discrimination.

Other perturbations. Noise perturbations exhibit a linear relationship with success rate degradation, indicating that the model's performance scales monotonically with input quality. Language and robot initialization perturbations show minimal impact due to the diversity of training data, suggesting that the model has learned robust representations for these dimensions.

As shown in Tab. 9, the average success rates and performance drops across perturbation levels provide quantitative evidence of the model's robustness characteristics. Notably, orientation perturbations achieve the highest average success rate (66.7%), while background and camera perturbations exhibit the largest performance drops (40.0% each), highlighting areas for future improvement.

I. Failure Cases Study

To gain deeper insights into the model's failure mechanisms beyond aggregate performance metrics, we conduct a qualitative analysis of characteristic error patterns across different perturbation types. This case study reveals how each perturbation dimension induces distinct failure modes in object localization, task understanding, and action execution, providing explanatory context for the quantitative results presented in previous sections. Typical failure cases can be seen in Figures 21 to 23.

J. Detailed results of LIBERO-Plus

This section presents a comprehensive analysis of generalization performance under diverse perturbations on the LIBERO-Plus benchmark. Table 10 provides detailed success rates across seven perturbation categories (Camera, Robot Initialization, Language Instruction, Lighting, Background, Sensor Noise, and Scene Layout) for various VLA methods, with results further broken down by task suite (Spatial, Object, Goal, and Long). The comparative analysis reveals significant differences in robustness patterns across methods and perturbation types, offering valuable insights for understanding model generalization capabilities.

(a) Pick up the alphabet soup and place it in the basket → pick up the butter and place it in the basket
Actually did the task "Pick up the alphabet soup and place it in the basket"

(b) Pick up the alphabet soup and place it in the basket → pick up the butter and place it in the basket
Actually did the task "Pick up the alphabet soup and place it in the basket"

(c) Pick up the butter and place it in the basket → pick up the tomato sauce and place it in the basket
Actually did the task "Pick up the butter and place it in the basket"

(d) Pick up the chocolate pudding and place it in the basket → pick up the salad dressing and place it in the basket
Actually did the task "Pick up the chocolate pudding and place it in the basket"

(e) Pick up the cream cheese and place it in the basket → pick up the milk and place it in the basket
Actually did the task "Pick up the cream cheese and place it in the basket"

Figure 14. Behavioral Analysis of Goal Replacement Failures. Case studies showing model responses to modified instructions. For each pair: original→new instruction (above); actually executed behavior (below). The consistent execution of original tasks despite changed targets indicates shallow language processing and strong bias toward memorized visual-action associations.

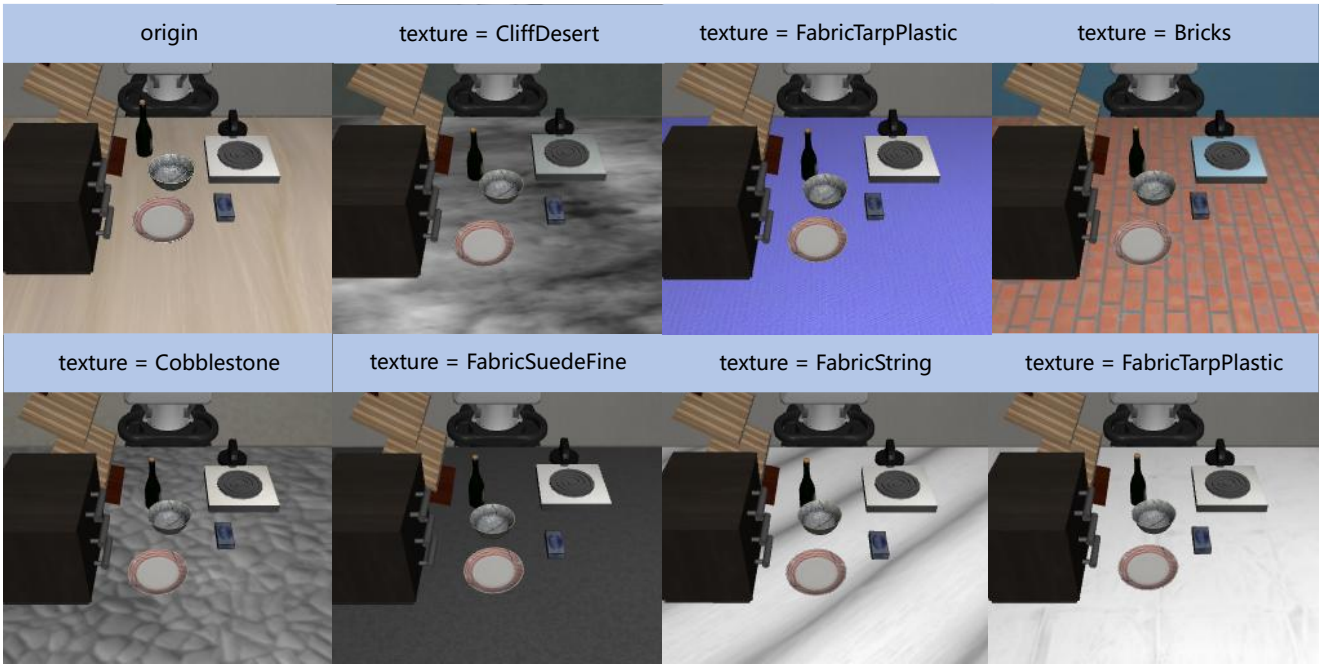


Figure 15. Rendering results with background texture perturbations. The top-left image is the original; the others show results with the textures as labeled.

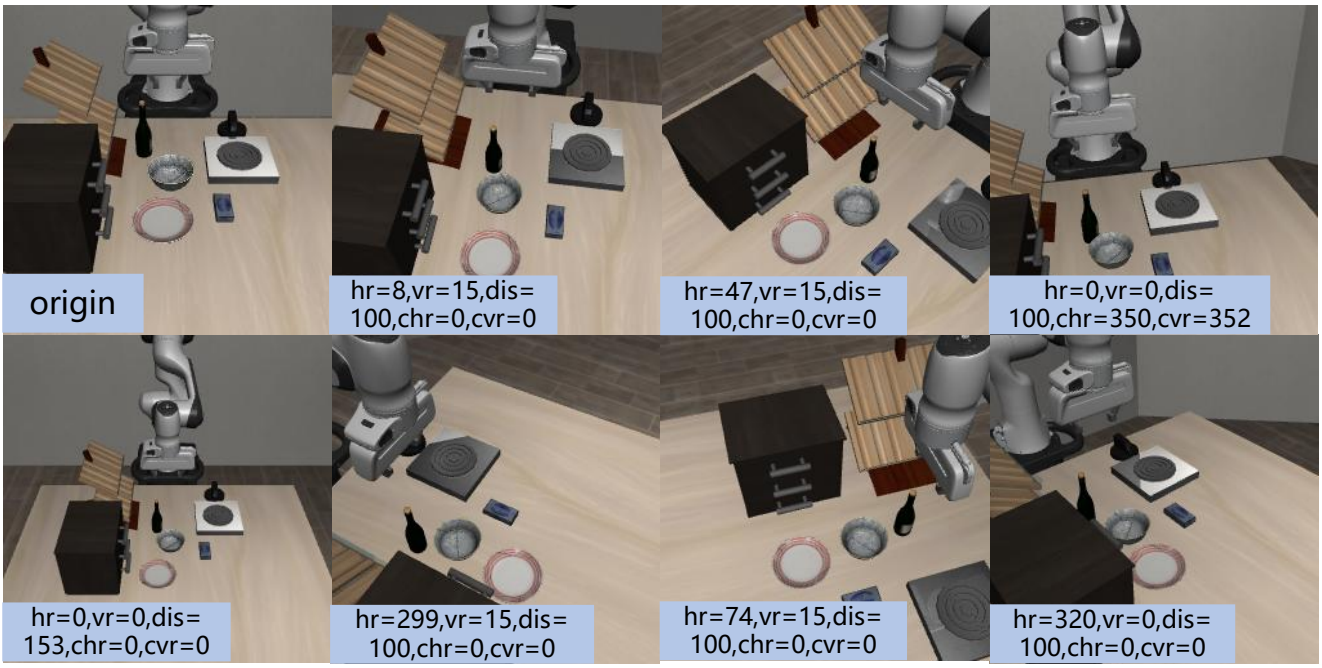


Figure 16. Rendering results under background texture perturbations, comparing the original image (top-left) with transformed versions. The labels denote the following transformation parameters: hr (horizontal rotation angle), vr (vertical rotation angle), dis (distance pulled away), chr (in-place horizontal rotation angle), and cvr (in-place vertical rotation angle).

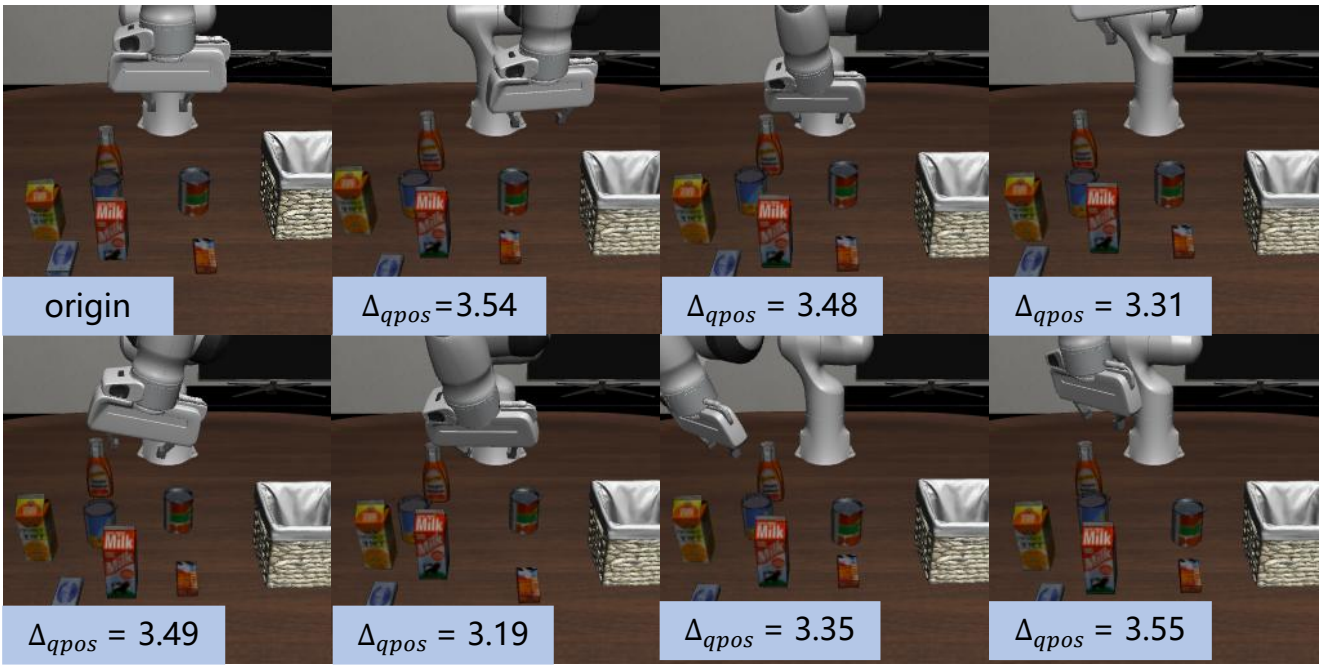


Figure 17. Rendering results with robot initial state perturbations. The top-left image is the original; the others show results with the norm of the change in the robot's joint angles as labeled.

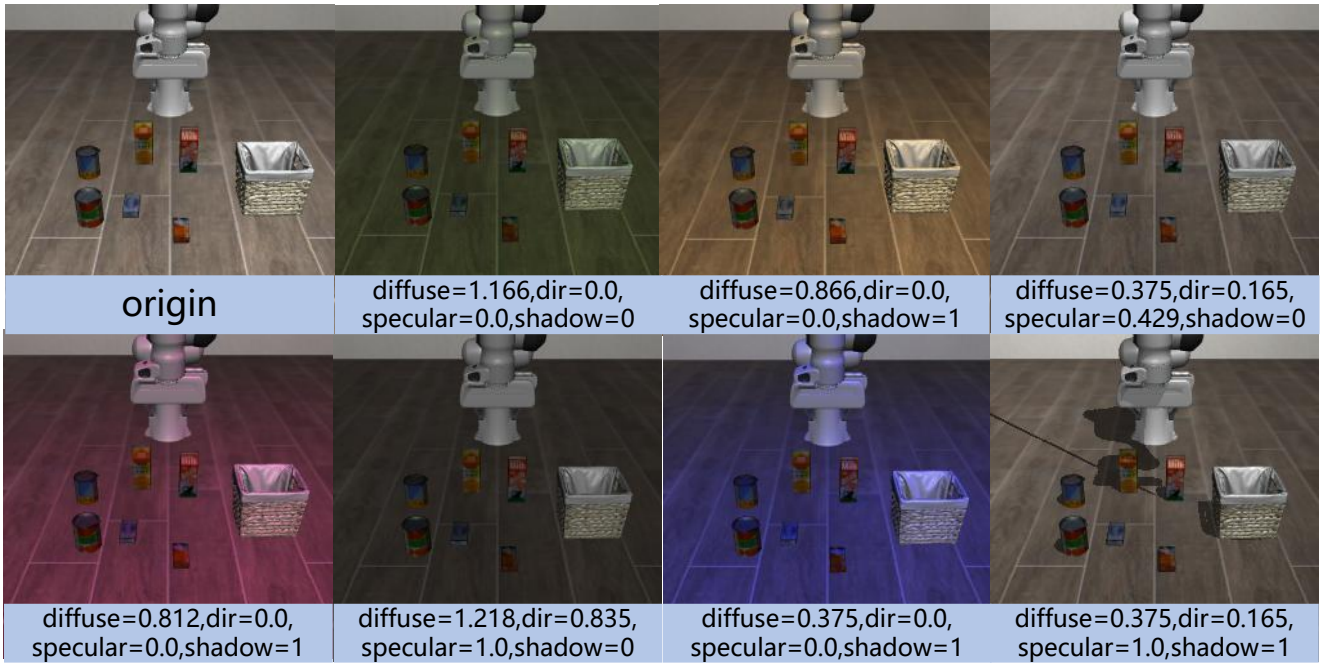


Figure 18. Rendering results with light perturbations. The top-left image is the original; the others show results with the relative change as labeled.

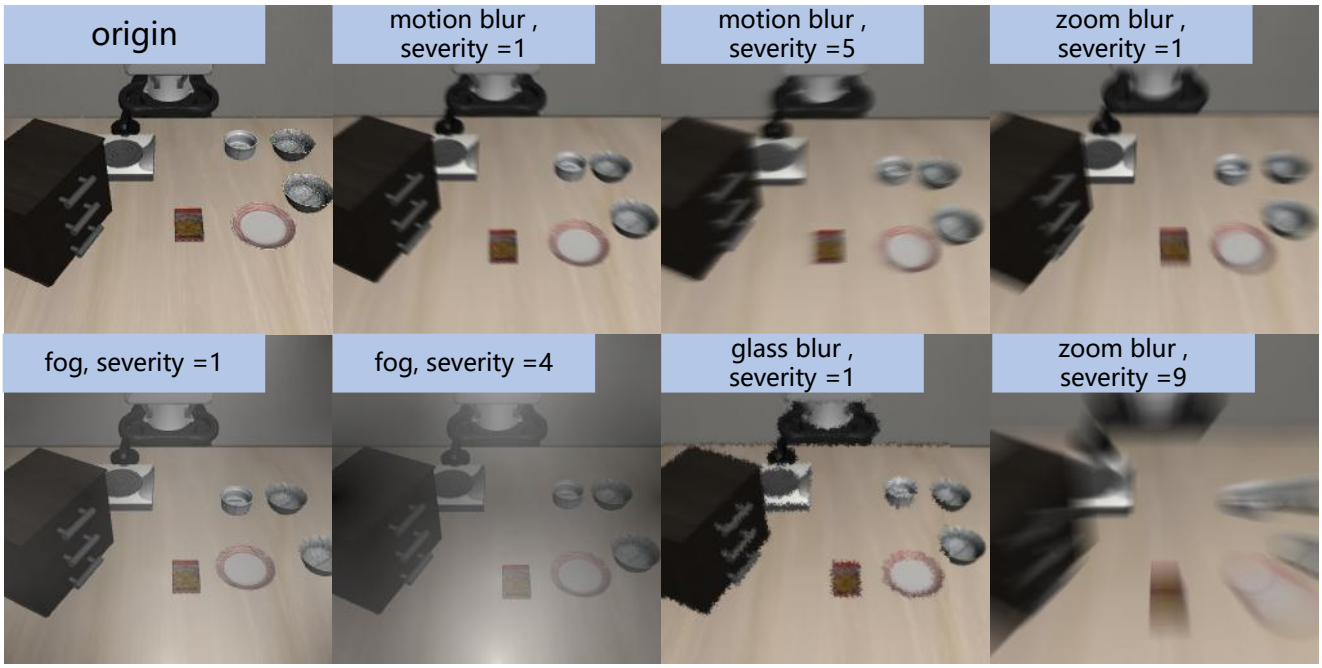
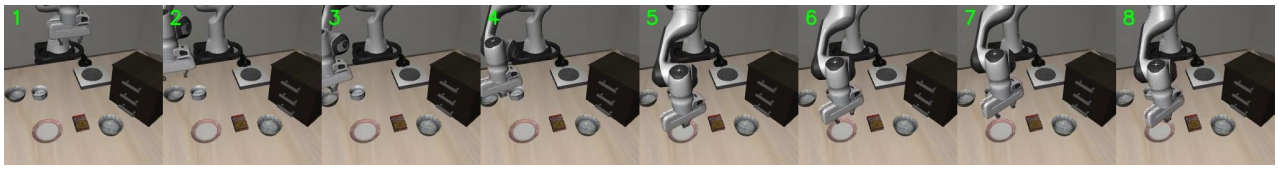
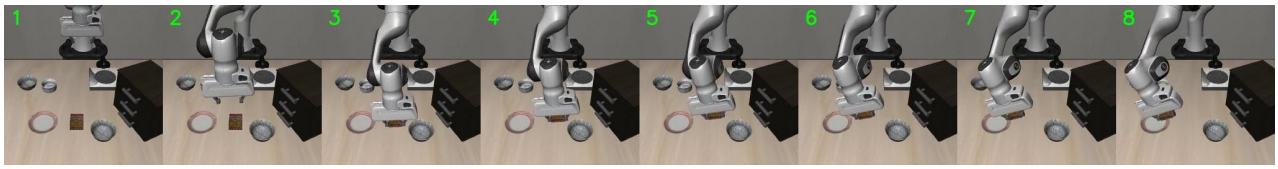


Figure 19. Rendering results with sensor noise perturbations. The top-left image is the original; the others show results corresponding to the type and severity of the applied noise, as indicated by the labels.

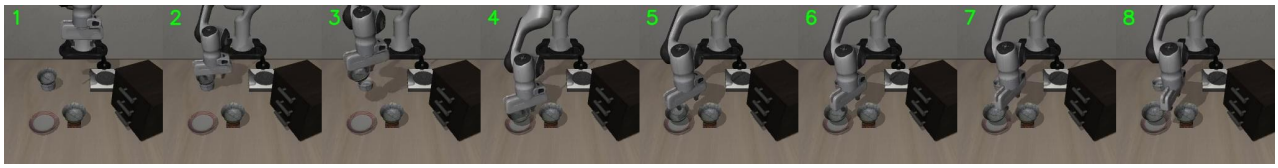
Figure 20. Rendering results with object layout perturbations. The top-left image is the original; the others show results with the number of added objects as labeled.



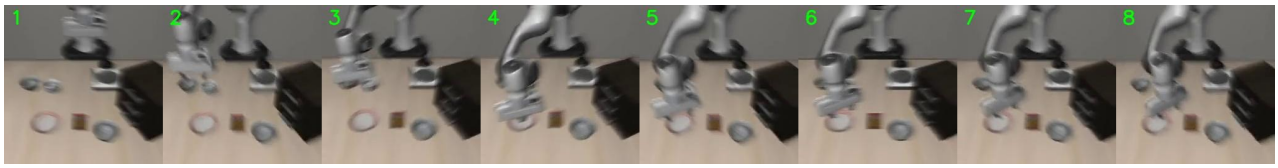
Camera – changes in camera position cause the model to localize the target object inaccurately.



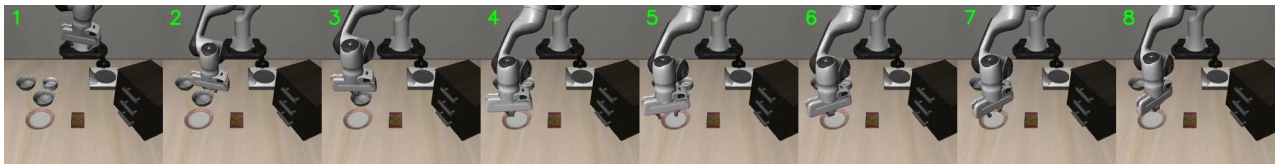
Language - modified language description sets the task object as darkcolored dish, but the model incorrectly localizes cookies



Light - variations in light source position create shadows, leading to biased localization of the target object.



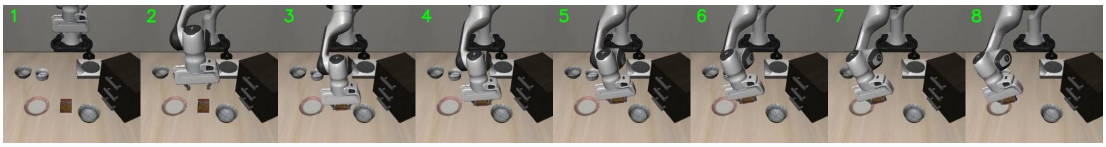
Noise - added noise blurs the image, resulting in inaccurate localization of the target object



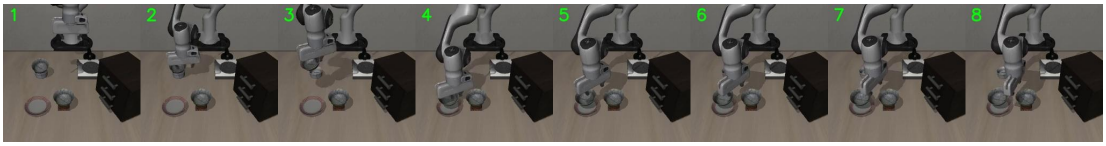
Robot - changes in the robot arm's initial position cause deviations in path planning and final positioning.

Layout - additional distractor objects lead to mislocalization of the target plate, with a nearby object being mistakenly recognized as the plate.

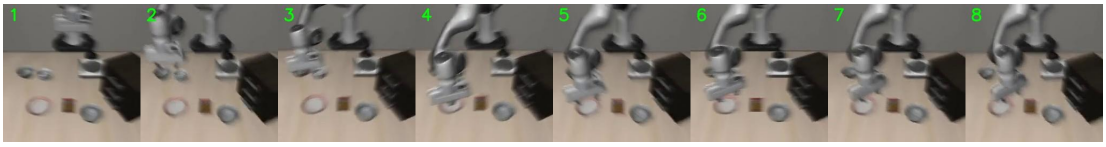
Figure 21. Failure Mode Analysis Across Perturbation Types. Visualization of characteristic failure patterns induced by each perturbation dimension, revealing distinct vulnerability profiles: camera shifts cause viewpoint-dependent localization errors; language modifications lead to semantic misinterpretations; lighting variations introduce shadow artifacts; sensor noise produces feature corruption; initial state changes affect trajectory planning; and object distractors trigger recognition confusion.



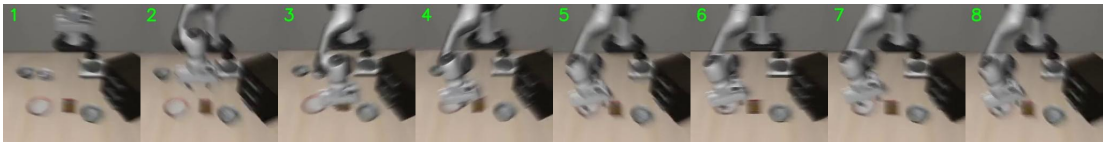
language - modified language description sets the task object as darkcolored dish, but the model incorrectly localizes cookies



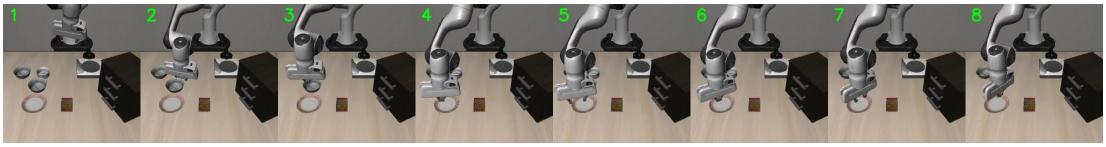
light - variations in light source position create shadows, leading to biased localization of the target object.



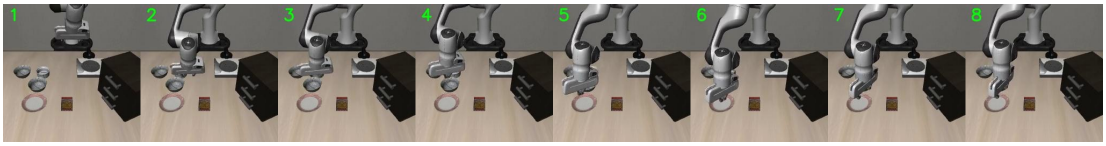
noise - added noise blurs the image, resulting in inaccurate localization of the target object



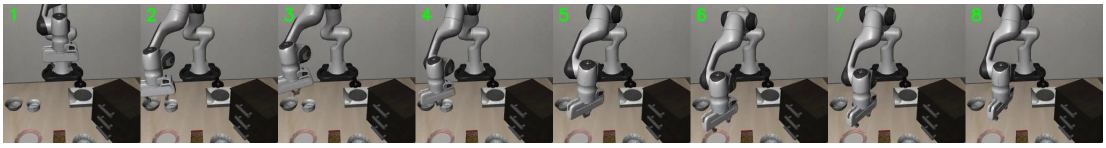
noise - added noise blurs the image, resulting in inaccurate localization of the target object



initstate - changes in the robot arm's initial position cause deviations in path planning and final positioning.

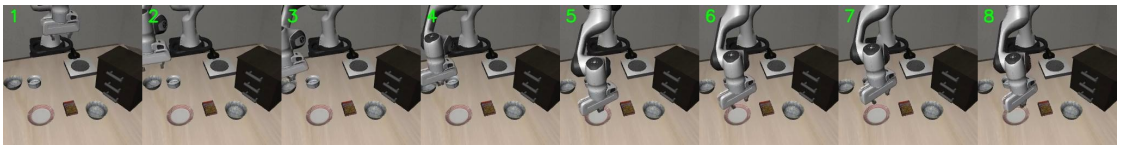


initstate - changes in the robot arm's initial position cause deviations in path planning and final positioning.

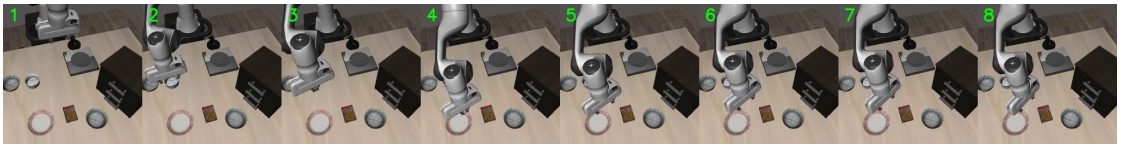


camera – changes in camera position cause the model to localize the target object inaccurately.

Figure 22. Failure Mode Analysis Across Perturbation Types. Visualization of characteristic failure patterns induced by each perturbation dimension, revealing distinct vulnerability profiles: camera shifts cause viewpoint-dependent localization errors; language modifications lead to semantic misinterpretations; lighting variations introduce shadow artifacts; sensor noise produces feature corruption; initial state changes affect trajectory planning; and object distractors trigger recognition confusion.



camera – changes in camera position cause the model to localize the target object inaccurately.



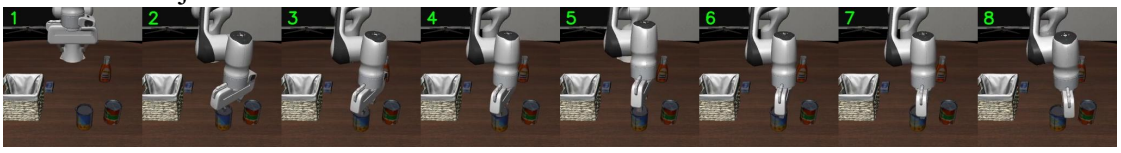
camera – changes in camera position cause the model to localize the target object inaccurately.

object - additional distractor objects lead to mislocalization of the target plate, with a nearby object being mistakenly recognized as the plate.

object - additional distractor objects lead to mislocalization of the target plate, with a nearby object being mistakenly recognized as the plate.

object - additional distractor objects lead to mislocalization of the target plate, with a nearby object being mistakenly recognized as the plate.

object - The model fails to flexibly rotate the robotic arm, resulting in a collision with a distractor object.



object - after the object position is perturbed, the model fails to correctly localize the object.

Figure 23. Failure Mode Analysis Across Perturbation Types. Visualization of characteristic failure patterns induced by each perturbation dimension, revealing distinct vulnerability profiles: camera shifts cause viewpoint-dependent object localization inaccuracy; object distractors provoke recognition confusion and mislocalization of the target, in some cases leading to incorrect collision-prone trajectories when arm motion flexibility is insufficient.

Table 10. Detailed generalization performance comparison across different perturbation types on the LIBERO-Plus benchmark. The table reports success rates (%) for various VLA methods under seven distinct perturbation categories and their average (Total). Results are further broken down by task suite to provide fine-grained insights into each method's robustness capabilities.

	Camera	Robot	Language	Light	Background	Noise	Layout	Total
OpenVLA								
Spatial	0.0	3.7	27.7	12.3	50.4	12.0	40.7	19.4
Object	0.5	4.5	21.0	1.0	45.2	11.4	22.4	14.0
Goal	2.5	2.7	21.5	9.0	27.1	19.5	25.6	15.1
Long	0.0	3.0	22.2	10.6	19.4	17.6	28.3	14.3
Avg	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
OpenVLA-OFT								
Spatial	88.3	40.0	80.5	98.3	97.3	96.3	93.9	84.0
Object	38.9	25.4	99.0	73.7	97.6	72.3	71.8	66.5
Goal	62.0	25.2	53.2	93.9	92.5	75.2	59.1	63.0
Long	38.7	38.2	87.0	89.4	86.8	63.5	76.9	66.4
Avg	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
OpenVLA-OFT_w								
Spatial	8.8	39.7	83.6	88.4	99.2	55.3	82.7	62.5
Object	10.1	31.4	76.4	85.9	96.4	48.3	66.3	56.0
Goal	16.4	39.9	47.1	85.3	89.0	54.9	61.8	53.3
Long	6.2	43.8	77.3	46.0	90.7	43.0	72.0	52.2
Avg	10.4	38.6	70.5	76.8	93.6	49.9	69.9	55.8
OpenVLA-OFT_m								
Spatial	55.3	19.7	92.7	100.0	92.3	85.2	94.5	75.4
Object	70.2	18.1	98.5	100.0	91.9	94.1	77.4	77.1
Goal	56.6	17.1	47.6	87.8	94.7	65.7	46.6	56.2
Long	41.0	31.8	88.3	82.1	85.5	69.9	61.0	63.9
Avg	55.6	21.7	81.0	92.7	91.0	78.6	68.7	67.9
NORA								
Spatial	4.3	50.9	63.8	66.8	65.5	12.5	84.6	47.6
Object	0.5	28.4	76.4	25.3	54.8	5.7	55.8	34.4
Goal	2.9	31.1	56.6	60.6	60.5	18.2	53.9	38.8
Long	1.2	39.4	64.0	30.3	54.0	15.1	59.5	36.3
Avg	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
WorldVLA								
Spatial	0.0	44.3	46.3	65.1	19.8	11.7	46.1	32.5
Object	0.0	26.4	57.2	20.5	17.3	18.0	53.6	28.6
Goal	0.3	30.6	42.2	68.8	30.3	13.5	47.4	31.8
Long	0.0	12.2	20.6	20.4	1.7	1.6	4.4	8.2
Avg	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
UniVLA								
Spatial	1.1	52.6	83.9	96.6	90.7	15.7	69.5	55.5
Object	0.0	42.2	86.9	25.6	81.5	10.4	27.3	36.7
Goal	3.9	37.9	45.6	89.6	78.3	33.5	22.6	40.7
Long	1.9	53.2	64.2	65.7	74.4	25.4	16.4	39.9

Continued on next page

Table 10 (continued)

	Camera	Robot	Language	Light	Background	Noise	Layout	Total
Avg	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
π_0								
Spatial	17.8	6.6	58.8	89.7	90.7	90.9	89.1	60.7
Object	22.2	8.3	70.0	90.9	91.1	87.0	76.2	61.4
Goal	12.3	5.6	39.3	84.2	76.5	76.5	44.7	44.9
Long	3.8	3.6	68.4	74.5	69.5	64.4	69.6	48.4
Avg	13.8	6.0	58.8	85.0	81.4	79.0	68.8	53.6
π_0_Fast								
Spatial	87.2	26.9	84.2	37.0	97.7	93.2	95.5	74.4
Object	72.0	27.6	71.5	71.0	95.2	93.1	84.5	72.7
Goal	70.8	20.5	47.3	95.3	60.9	69.7	51.6	57.5
Long	33.2	12.0	43.6	91.6	44.6	46.1	47.8	43.4
Avg	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
RIPT-VLA								
Spatial	85.4	38.0	99.7	99.7	100.0	92.0	92.3	85.8
Object	37.9	26.4	80.8	85.9	99.2	68.0	70.1	64.3
Goal	65.7	23.2	45.4	74.2	79.7	71.0	59.8	58.0
Long	34.1	38.4	88.3	93.4	89.3	66.4	79.2	67.5
Avg	55.2	31.2	77.5	88.3	91.6	73.5	74.2	68.4
Ours								
Spatial	98.4	31.7	96.0	99.3	98.8	86.3	97.8	86.1
Object	97.0	24.6	100.0	99.7	98.8	97.4	82.8	84.5
Goal	93.9	24.7	55.1	96.8	94.0	93.4	53.9	70.7
Long	82.6	40.7	94.8	83.2	85.1	80.6	80.3	77.7
Avg	92.8	30.3	85.8	94.9	93.9	89.3	77.6	79.5