VISION-LANGUAGE-ACTION MODELS ARE INHERENT VERIFIERS: TEST-TIME SCALING VIA LIKELIHOOD-BASED VISUAL PROMPTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling (TTS) has recently emerged as a powerful paradigm for improving Vision-Language-Action (VLA) models without requiring expensive fine-tuning. However, existing TTS approaches typically rely on training external verifiers or are strictly limited to autoregressive action decoders. To address these limitations, we introduce **L**ikelihood-Based **V**isual **P**rompting (**LVP**), a training-free, verifier-free test-time scaling framework and reveal that generalist VLA policies can serve as their own inherent verifiers. LVP first projects action candidates directly into the 2D image space via perspective-aware action projection, overlaying future trajectory and gripper poses as structured visual prompts. We then reframe action verification as an evaluative Visual Question Answering (VQA) task. By prompting the VLA to evaluate the quality of the visually prompted action via a multiple-choice question (e.g., grading the action from "Poor" to "Excellent"), we leverage the model's internal conditional token likelihoods to compute an expected action quality score for candidate ranking. Furthermore, we introduce an attention-guided token pruning mechanism to alleviate the computational overhead of processing multi-sample visual prompts. By leveraging the visual attention maps generated during the action proposal stage, LVP dynamically discards task-irrelevant visual tokens during the verification phase. In the experiments, LVP improves autoregressive and diffusion/flow-matching based VLA models across diverse simulation and real-world benchmarks.

1 INTRODUCTION

Vision-Language-Action (VLA) models (Brohan et al., 2023; Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2025; Physical Intelligence et al., 2025b; NVIDIA et al., 2025; Gemini Robotics Team et al., 2025b; Physical Intelligence et al., 2025a; Gemini Robotics Team et al., 2025a) have recently emerged as a promising paradigm for general-purpose robotic manipulation, leveraging large-scale vision-language pretraining (Chen et al., 2024b; Driess et al., 2023; Karamcheti et al., 2024; Beyer et al., 2024; Lin et al., 2024; Li et al., 2025; Comanici et al., 2025) and supervised fine-tuning (SFT) on action-labeled robot data (Open X-Embodiment Collaboration et al., 2024; Walke et al., 2023; Khazatsky et al., 2024) to enable instruction-conditioned control across diverse tasks. Despite their impressive performance, pretrained VLA policies often exhibit brittle behaviors when deployed in long-horizon or distribution-shifted scenarios. To overcome these generalization bottlenecks without the prohibitive computational cost of model retraining, recent works (Nakamoto et al., 2024; Kwok et al., 2025; Yang et al., 2025; Jang et al., 2026) have explored **test-time scaling (TTS)** strategies. These methods improve performance at inference time by sampling multiple action candidates and selecting the optimal trajectory using auxiliary evaluation signals.

However, existing TTS methods for VLA models largely suffer from additional training overheads or restrictive architectural dependencies. Specifically, prior works use repeated sampling and Best-of-N selection (Gao et al., 2023) with a specialized value function (Nakamoto et al., 2024), an external VLM-based action verifier (Kwok et al., 2025), or an auxiliary coin-flipping network (Yang et al., 2025). Although effective, these methods require training auxiliary networks, which degrades generalizability on domain shift beyond the verifier's training distribution (Yin et al., 2025; Jang et al., 2026). In contrast, recent verifier-free alternatives attempt to estimate model uncertainty directly.

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.

For instance, MG-Select (Jang et al., 2026) computes self-uncertainty via a condition-masking reference distribution, yet it still requires fine-tuning the base VLA policy. SCALE (Choi et al., 2026) achieves a training-free setup by modulating visual attention and action decoding via token-level self-uncertainty. However, MG-Select and SCALE are strictly restricted to autoregressive action decoders, because the uncertainty estimation is conducted on the categorical distribution. Consequently, they cannot support modern diffusion/flow-matching based VLA policies, significantly limiting their general applicability.

We approach this problem from a fundamentally different perspective and ask: *Can a pretrained VLA model evaluate its own action candidates without additional training or external verifiers?* Our key observation is that action evaluation does not need to occur in the raw robotic action space. Instead, action candidates can be projected directly into the image space by visualizing predicted translations and rotations as spatial overlays directly on the robot's camera observation. Compared to action space trained with action-labeled robotics dataset, image space is highly pre-trained with large-scale vision-language dataset. Once rendered as structured visual prompts, action evaluation is reframed as a multi-modal visual reasoning task. Specifically, we formulate action assessment as an evaluative Visual Question Answering (VQA) problem using a multiple-choice question (MCQ) format (e.g., choosing among "Poor", "Fair", "Good", or "Excellent") and measure the conditional token likelihood of each option under the base VLA. The expected score derived from these token probabilities serves as an implicit reward, enabling a principled selection of the optimal action among the sampled candidates.

Based on this insight, we propose **Likelihood-Based Visual Prompting (LVP)**, a training-free, verifier-free test-time scaling framework for VLA models. LVP operates in three stages: *(1) Action generation,* sampling multiple action candidates from the base policy, *(2) Action projection,* projecting each action candidate into the 2D workspace image as structured visual prompts that visually represent future translation paths and gripper rotations, and *(3) Action verification,* computing the conditional token likelihoods of the quality descriptors to rank and select the best candidate. Crucially, LVP requires **no fine-tuning** and **no additional verifiers**, leveraging only the inherent vision-language pretraining and conditional likelihood estimation capabilities of the frozen VLA policy. To further address the computational bottleneck of evaluating multiple visual prompts, we introduce an attention-guided token pruning mechanism. By utilizing the attention weights generated during the action proposal stage, LVP dynamically discards task-irrelevant visual tokens during the verification phase. Consequently, LVP provides a unified, efficient test-time scaling framework that is natively compatible with both autoregressive and diffusion/flow-matching based VLA policies.

We validate Likelihood-Based Visual Prompting (LVP) on simulated and real-world benchmarks across diverse autoregressive and diffusion/flow-matching based VLA models. Our extensive evaluations demonstrate that LVP consistently yields significant performance gains on diverse state-of-the-art (SOTA) VLAs across diverse environments. In particular, LVP improves task success rates by up to $+\mathbf{17.3}\%$ in Simpler-Fractal environments and $+\mathbf{10.3}\%$ under severe OOD layout perturbations, outperforming existing test-time scaling baselines. In real-world experiments, LVP achieves an $\mathbf{18.8}\%$ improvement in in-distribution environments and $\mathbf{15.6}\%/\mathbf{11.4}\%$ improvements under layout/camera perturbations. Additionally, we show that our attention-guided token pruning strategy decreases verification latency by over $\mathbf{2.3}\times$ for 16 samples with negligible loss in performance, enabling efficient closed-loop deployment.

2 Related Work

Test-time scaling on Vision-Language-Action models. Test-time scaling (TTS) has emerged as a key paradigm to improve the inference-time reasoning and generalization of Vision-Language-Action (VLA) models without parameter retraining. Several methods train the auxiliary verifier to re-rank sampled actions. V-GPS (Nakamoto et al., 2024) trains external value functions, RoboMonkey (Kwok et al., 2025) fine-tunes a VLM-based action verifier, and TACO (Yang et al., 2025) trains a coin-flipping classifier. These verifier-based methods require additional training for verifiers and struggle on generalization to unseen environments (Yin et al., 2025; Jang et al., 2026). On the other hand, verifier-free methods such as MG-Select (Jang et al., 2026) and SCALE (Choi et al., 2026) leverage self-uncertainty to re-rank sampled actions or modulate visual attention and action sampling. However, these are fundamentally incompatible with diffusion or flow-matching policies.

Visual prompting in VLM and VLA. Visual prompting modifies input images with explicit visual markers to improve spatial grounding and visual reasoning in Vision-Language Model (VLM). In general computer vision, methods such as Set-of-Marks (Yang et al., 2023a), Shtedritski et al. (2023), and FGVP (Yang et al., 2023b) overlay bounding boxes, segmentation masks, or colored circles to steer VLM focus on specific target objects. In robotic manipulation, visual overlays have been integrated into both forward generation and candidate evaluation. TraceVLA (Zheng et al., 2025) extracts historical end-effector motion trajectories using tracking models to enhance spatio-temporal awareness. MolmoAct (Lee et al., 2026) and ThinkAct (Huang et al., 2025) predict 2D spatial trajectory points as intermediate visual plans before feeding them into action execution modules. While PIVOT (Nasiriany et al., 2024) renders candidate visual representations onto images to evaluate robot actions using VLM evaluator, LVP unifies both candidate generation and verification within a single VLA model.

VLM/LLM-based reward functions. Using foundation language and multimodal models as automated reward evaluators has been widely adapted across reinforcement learning and robotics. These evaluators can be categorized into black-box, white-box, and likelihood-based frameworks. *Black-box evaluators*, such as DRESS (Chen et al., 2024c) and 3DPhysVideo (Kim et al., 2026), prompt proprietary LLMs or VLMs to generate natural language critique or scalar numerical scores. *White-box evaluators*, including LLaVA-reward (Zhou et al., 2025) and RoboMonkey (Kwok et al., 2025), fine-tune specialized reward heads on human preference or trajectory datasets. More recently, *likelihood-based reward modeling* has gained popularity in NLP and multimodal ranking: Zhuang et al. (2024) and ASRank (Abdallah et al., 2025) convert categorical token log-probabilities into continuous ranking scores, while LLM-as-a-Verifier (Kwok et al., 2026) formalizes token likelihood verification for text and image data. LVP extends likelihood-based reward modeling directly to physical robotic control. By formulating action evaluation as an evaluative VQA task, LVP derives implicit rewards from conditional token likelihoods.

Visual token pruning. Processing high-resolution vision inputs in VLMs and video foundation models introduces significant computational latency, as sequence self-attention scales quadratically with the number of visual tokens. Recent works show that visual token representations contain substantial spatial and temporal redundancy. FastV (Chen et al., 2024a) prunes visual tokens with low-attention in early transformer layers, while SparseVLM (Zhang et al., 2025) uses text-guided attention maps to sparsify vision features. In robotics, VLA-Cache (Xu et al., 2025) adaptively caches and reuses static background visual tokens across frames. While existing VLA token pruning techniques focus exclusively on accelerating the action generation pass, LVP introduces a pruning mechanism for the verification phase. By aggregating self-attention weights from the action proposal pass, LVP can prune visual tokens effectively without performance degradation.

3 Likelihood-Based Visual Prompting

We introduce **Likelihood-Based Visual Prompting (LVP)**, a training-free, verifier-free and decoder-agnostic test-time scaling framework that exploits the inherent self-verification capabilities of generalist Vision-Language-Action (VLA) models. The core insight of LVP is that action verification does not need to rely on separate value functions or reward functions. Instead, we leverage camera projection to map physical action candidates back into the 2D visual-semantic space in which the underlying foundation Vision-Language Model (VLM) was pre-trained. By translating action verification into an evaluative Visual Question Answering (VQA) task, the frozen VLA itself can serve as a zero-shot reward model.

3.1 Overview

Let $I \in \mathcal{I}$ denote the RGB camera observation, and let g be the natural language task instruction. A generalist VLA policy parametrized by θ is formulated as a conditional probability distribution $\pi_\theta(a \mid I, g)$ over the action space $\mathcal{A}$. To enhance the policy's performance and robustness at test-time, LVP implements an inference-time scaling pipeline consisting of three sequential stages:

Stage 1: Multi-sample action proposal. Given the current observation I and task instruction g, we sample N candidates from the learned distribution:

$$\{a_i\}_{i=1}^{N} \sim \pi_\theta(\cdot \mid I, g). \tag{1}$$

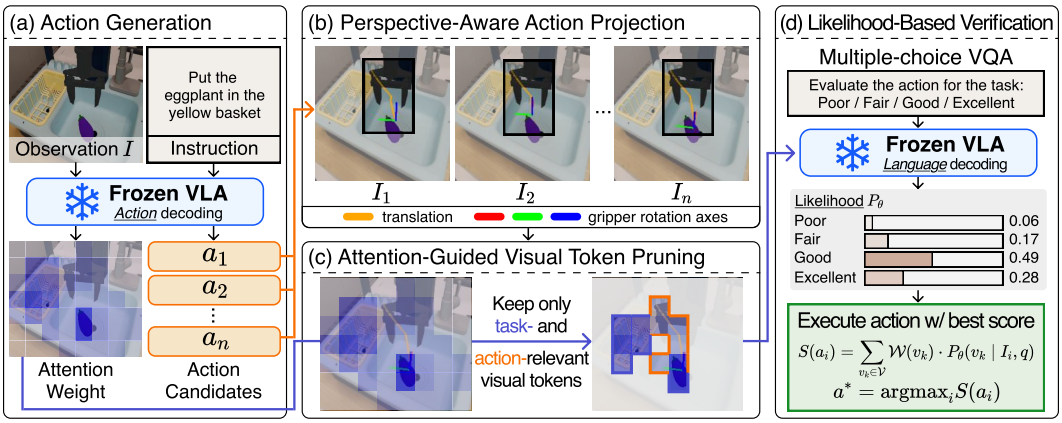


Figure 1: Overview of LVP. **(a)** We sample N candidates using the pre-trained VLA models with action decoding. **(b)** The sampled action candidates are projected into the 2D image space via **perspective-aware action projection** (Sec. 3.2). We visualize the corresponding translation and target gripper's orientation with orange and RGB lines, respectively. **(c)** Based on the visual attention weights in the (a) generation stage, **attention-guided visual token pruning** mechanism discards task-irrelevant visual tokens (Sec. 3.4). **(d)** Given the visually prompted images with token pruning, we rank the action candidates via **token likelihood-based verification** (Sec. 3.3). We query multiple-choice question to the pre-trained VLA models with language decoding, acquire the conditional token likelihoods, and transform into the final score.

Stage 2: Perspective-aware action projection. To make the 3D actions $\{a_i\}$ legible to the visual-reasoning capabilities of the VLM, we project the action parameters directly onto the 2D image. For each candidate a_i, we overlay its translation and the target gripper's orientation (Sec. 3.2):

$$\{I_i\}_{i=1}^N = \Pi(\{a_i\}_{i=1}^N, I), \tag{2}$$

where I_i represents the visually prompted observations and Π represents the projection function.

Stage 3: Token likelihood-based verification. For each visually prompted image I_i, we query the frozen model π_θ with its language-modeling head (Sec. 3.3). We prompt the model with an evaluative multiple-choice question $q(g)$. The final score $\mathcal{S}(a_i)$ for each candidate is the expected utility computed over the normalized conditional token log-likelihoods of the designated ordinal options. Finally, the candidate maximizing this implicit zero-shot reward is executed:

$$a^* = \arg\max_{a_i} \mathcal{S}(a_i). \tag{3}$$

3.2 Perspective-Aware Action Projection

To translate 3D robotic actions into spatial signals legible to the pre-trained VLA model, we project action candidates directly onto the 2D observation I. Given an action candidate $a_i = [\Delta x, \Delta y, \Delta z, \Delta\phi, \Delta\theta, \Delta\psi, s^{\text{grip}}]^T \in \mathcal{A}$ containing 6-DoF end-effector deltas and gripper states, we integrate these local controls through the robot's kinematic chain to compute future 3D poses.

To map these 3D poses onto the image plane, we apply a perspective camera projection pipeline transforming coordinates across frames:

$$\text{Robot Frame } \mathcal{F}_{\text{robot}} \xrightarrow{\mathbf{T}_{\text{camera}}^{\text{robot}}} \text{Camera Frame } \mathcal{F}_{\text{camera}} \xrightarrow{\mathbf{K}} \text{2D Image Plane } \mathcal{U}. \tag{4}$$

For each candidate a_i, we render its translational path as a sequence of connected orange lines and project the target end-effector orientation as a localized tri-axial RGB coordinate frame (X=red, Y=green, Z=blue). This perspective-aware rendering yields N visually prompted observations:

$$\{I_i\}_{i=1}^N = \Pi(\{a_i\}_{i=1}^N, I), \tag{5}$$

where Π denotes the complete 3D-to-2D projection and rendering function. By converting low-dimensional kinematics into visual pixel-space overlays, we exploit the dense spatial-semantic grounding of the underlying VLM. See Appendix A.2 for details.

3.3 TOKEN LIKELIHOOD-BASED VERIFICATION

Once action candidates are projected into the image as visual prompts, LVP transforms action evaluation from a regression problem in unseen parameter space into an evaluative **Visual Question Answering (VQA)** task in visual-semantic space. Rather than relying on auxiliary networks, LVP utilizes the frozen VLA's language decoder to perform zero-shot qualitative assessment. Given the visually prompted image I_i and question $q(g)$, which is constructed with the task instruction g, information of visual overlays, and multiple choices $\mathcal{V}$, we acquire the raw next-token logits and apply a localized softmax over the set $\mathcal{V}$ to obtain a normalized conditional probability distribution:

$$P_\theta(v_k \mid I_i, q) = \frac{\exp\left(\text{logits}(v = v_k \mid I_i, q)\right)}{\sum_{v_j \in \mathcal{V}} \exp\left(\text{logits}(v = v_j \mid I_i, q)\right)}. \tag{6}$$

To rank the action candidates, we convert this discrete categorical distribution into a continuous scalar reward. We define a utility mapping $\mathcal{W} : \mathcal{V} \to \mathbb{R}$, which assigns a reward for each quality option. The continuous implicit reward score $\mathcal{S}(a_i)$ is formulated as the expected utility over the localized probability space:

$$S(a_i) = \sum_{v_k \in \mathcal{V}} \mathcal{W}(v_k) \cdot P_\theta(v_k \mid I_i, q). \tag{7}$$

The action candidate $a^* = \arg\max_{a_i} S(a_i)$ is selected for execution. See Appendix A.3 for details.

3.4 ATTENTION-GUIDED VISUAL TOKEN PRUNING

To eliminate the computational latency of evaluating batches of visually prompted candidate observations, we introduce a training-free visual token pruning strategy. We observe that background visual tokens spatially unrelated to the manipulation task can be safely discarded without degrading verification accuracy. LVP reuses the self-attention weights recorded during the initial action generation pass to extract task-relevant spatial saliency maps.

For a sequence of visual tokens $\mathbf{X}_{\text{vis}} = \{x_j^v\}_{j=1}^M$, we compute a task-attention saliency mask $\mathbf{A}$ by aggregating action-to-vision attention weights across the decoder layers $\mathcal{L}$. This mask identifies the top-M' tokens focused on the robot and workspace objects:

$$\mathbf{X}_{\text{vis}}^{\text{attn}} = \text{Top-}M'(\{x_j^v\}_{j=1}^M, \mathbf{A}), \quad \text{where } M' \ll M. \tag{8}$$

Because visual prompts are rendered *after* action generation, an attention-only mask risks discarding the visual overlays. To preserve these prompt cues, we compute a secondary spatial mask $\mathbf{X}_{\text{vis}}^{\text{prompt}}$ using the ℓ_2-norm difference between visually prompted and unprompted token embeddings:

$$\mathbf{X}_{\text{vis}}^{\text{prompt}} = \text{Top-}M'\left(\{x_j^v\}_{j=1}^M, \|\mathbf{X}_{\text{vis}} - \mathbf{X}_{\text{vis}}^{\text{original}}\|_2\right). \tag{9}$$

The final sparsified token set passed to the VLA evaluator is formed by the union of both pathways: $\mathbf{X}_{\text{vis}}^{\text{pruned}} = \mathbf{X}_{\text{vis}}^{\text{attn}} \cup \mathbf{X}_{\text{vis}}^{\text{prompt}}$. See Appendix A.4 for details.

4 EXPERIMENTS

To evaluate the effectiveness, generality, and computational efficiency of Likelihood-Based Visual Prompting (LVP), we perform simulated and real-world robotic manipulation experiments. We structure our empirical evaluation to answer the following research questions: (1) Does LVP consistently improve the task success rates of diverse generalist VLA backbones, including autoregressive and flow-matching policies? (2) Does LVP exhibit superior robustness under severe out-of-distribution environmental shifts? (3) Can our attention-guided token pruning strategy mitigate the computational latency of multi-sample verification to support closed-loop real-world control?

4.1 SIMULATION EXPERIMENTS

In this section, we conduct extensive simulation experiments to evaluate our proposed LVP framework. These evaluations are structured across different VLA backbones and environments to validate LVP's effectiveness, model-agnostic nature, and robustness under perturbations. We compare more diverse test-time scaling baselines using OpenVLA (Kim et al., 2025) backbone on LIBERO (Liu et al., 2023) benchmark in Appendix B, while we mainly use SOTA VLA models in the main paper. See Appendix A.5 and Appendix A.6 for details of backbones and baselines.

Table 1: Success rate comparisons (%) on the SimplerEnv (Google Robot) using the MolmoAct backbone. We report success rate of LVP using a full token budget ($M = 392$) and our pruned token budget ($M' = 28$) using the zero-shot and fine-tuned base policies. A dagger (†) denotes results obtained following author's implementation.

	Tasks (Success Rate, %)			
Method	Pick coke can	Move near	Open/Close drawer	Average
MolmoAct (Zero-shot)	71.3	73.8	66.5	70.5
+ SCALE†	81.0 (+9.7)	**77.8 (+4.0)**	75.5 (+9.0)	78.1 (+7.6)
+ LVP (Full tokens)	**82.7 (+11.4)**	75.9 (+2.1)	**78.7 (+12.2)**	**79.1 (+8.6)**
+ LVP (Pruned)	**82.7 (+11.4)**	70.4 (-3.4)	75.0 (+8.5)	76.0 (+5.5)
MolmoAct (Fine-tuned)	77.7	77.1	60.0	71.6
+ SCALE†	87.7 (+10.0)	76.9 (-0.2)	70.4 (+10.4)	78.3 (+6.7)
+ LVP (Full tokens)	87.0 (+9.3)	**77.8 (+0.7)**	76.4 (+16.4)	80.4 (+8.8)
+ LVP (Pruned)	**91.3 (+13.6)**	75.9 (-1.2)	**77.3 (+17.3)**	**81.5 (+9.9)**

Table 2: We report success rate of LVP using a full token budget ($M = 64$) and our pruned token budget ($M' = 8$) using GR00T N1.7 backbone. An asterisk (*) denotes results reproduced by utilizing officially released checkpoints.

(a) Success rate comparisons (%) on SimplerEnv (WidowX). A dagger (†) denotes results obtained following author's implementation.

	Tasks (Success Rate, %)				
Method	Spoon	Carrot	Eggplant	Cube	Average
GR00T N1.7*	80.0	65.0	39.5	35.5	55.0
+ RoboMonkey†	56.0 (-24.0)	29.0 (-36.0)	54.0 (+14.5)	**40.0 (+4.5)**	44.8 (-10.2)
+ LVP (Full tokens)	81.0 (+1.0)	**68.0 (+3.0)**	**54.5 (+15.0)**	36.5 (+1.0)	**60.0 (+5.0)**
+ LVP (Pruned)	**83.0 (+3.0)**	66.0 (+1.0)	47.5 (+8.0)	37.0 (+1.5)	58.4 (+3.4)

(b) Success rate comparisons (%) on SimplerEnv (Google Robot).

	Tasks (Success Rate, %)			
Method	Open drawer	Close drawer	Place in closed drawer	Average
GR00T N1.7*	41.0	51.0	15.0	35.7
+ LVP (Full tokens)	**50.0 (+9.0)**	**60.5 (+9.5)**	15.0 (+0.0)	**41.8 (+6.1)**
+ LVP (Pruned)	45.0 (+4.0)	60.0 (+9.0)	**19.0 (+4.0)**	41.3 (+5.6)

SimplerEnv benchmark. We evaluate LVP on the SimplerEnv benchmark (Li et al., 2024) across WidowX and Google Robot platforms. In Tab. 1, we evaluate the performance of LVP on the SimplerEnv (Google Robot) using the autoregressive MolmoAct (Lee et al., 2026) backbone under two training conditions: zero-shot and fine-tuned on the RT-1 (Brohan et al., 2023) dataset. We compare LVP against SCALE (Choi et al., 2026) and the base policy across three representative tasks: *Pick coke can*, *Move near*, and *Open/Close drawer*. For zero-shot MolmoAct, LVP with full tokens achieves a $+8.6\%$ average improvement over the base policy, outperforming SCALE by $+1.0\%$. Under the fine-tuned setup, LVP with token pruning achieves the best overall performance (81.5%) with $+\mathbf{9.9}\%$ **average increase** over the base policy. Most notably, on the highly geometric *Open/Close drawer* task, LVP improves the success rate by a remarkable $+17.3\%$ ($60.0\% \rightarrow 77.3\%$). This indicates that our proposed perspective-aware action projection (Sec. 3.2) directly clarifies spatial trajectory, improving spatial reasoning capability.

Additionally, we validate that LVP is decoder-agnostic. We use the GR00T N1.7 (NVIDIA et al., 2025) policy, which generates continuous action distributions via a **flow-matching action head**. Leveraging uncertainty-based test-time methods such as MG-Select (Jang et al., 2026) and SCALE (Choi et al., 2026) on the GR00T is nontrivial. We compare LVP against base GR00T and RoboMonkey (Kwok et al., 2025) on SimplerEnv (WidowX) in Tab. 2a and SimplerEnv (Google

Table 3: Success rate (%) on the perturbed LIBERO-Plus-Long benchmark under seven domain shifts. We set $M = 392$, $M' = 28$ for MolmoAct2 and $M = 64$, $M' = 8$ for GR00T N1.7.

	Perturbations (Success Rate, %)							
Method	Layout	Camera	Robot	Language	Light	Background	Noise	Average
MolmoAct2	68.6	48.0	60.1	79.6	79.6	77.2	68.4	67.6
+ LVP (Full tokens)	68.9 (+0.3)	46.5 (-1.5)	61.6 (+1.5)	**80.9 (+1.3)**	80.3 (+0.7)	81.0 (+3.8)	**69.7 (+1.3)**	68.6 (+1.0)
+ LVP (Pruned)	**69.6 (+1.0)**	**49.2 (+1.2)**	**63.9 (+3.8)**	79.1 (-0.5)	**81.8 (+2.2)**	**82.7 (+5.5)**	69.0 (+0.6)	**69.5 (+1.9)**
GR00T N1.7	47.4	33.4	46.8	71.5	67.2	**77.9**	49.9	54.7
+ LVP (Full tokens)	51.9 (+4.5)	**39.9 (+6.5)**	**48.9 (+2.1)**	71.0 (-0.5)	**71.5 (+4.3)**	75.1 (-2.8)	48.1 (-1.8)	**56.5 (+1.8)**
+ LVP (Pruned)	**57.7 (+10.3)**	37.5 (+4.1)	46.1 (-0.7)	**72.3 (+0.8)**	67.2 (+0.0)	74.4 (-1.5)	**51.2 (+1.3)**	**56.5 (+1.8)**

Table 4: Success rates (%) on four real-world manipulation tasks using a UR10e robot. We evaluate GR00T N1.7 with and without LVP ($N = 4, M' = 8$) under unperturbed and perturbed settings. *Layout* and *Camera* denote perturbations to object layouts and camera viewpoints, respectively.

		Tasks (Success Rate, %)				
Perturb.	Model	PnPLemonPlate	StackCube	PressStapler	HangTape	Average
–	GR00T N1.7	70.8	33.3	66.7	29.2	50.0
	+ LVP	**83.3 (+12.5)**	**62.5 (+29.2)**	**70.8 (+4.1)**	**58.3 (+29.1)**	**68.8 (+18.8)**
Layout	GR00T N1.7	25.0	20.8	50.0	29.2	31.3
	+ LVP	**45.8 (+20.8)**	**41.7 (+20.9)**	**54.2 (+4.2)**	**45.8 (+16.6)**	**46.9 (+15.6)**
Camera	GR00T N1.7	16.7	8.3	62.5	0.0	21.9
	+ LVP	**33.3 (+16.6)**	**12.5 (+4.2)**	**66.7 (+4.2)**	**20.8 (+20.8)**	**33.3 (+11.4)**

Robot) in Tab. 2b. On SimplerEnv (WidowX), LVP with full tokens achieves a $+5.0\%$ overall success rate increase ($55.0\% \rightarrow 60.0\%$). On SimplerEnv (Google Robot), LVP yields $+6.1\%$ and $+5.6\%$ average success rate boosts under full and pruned visual token budgets, respectively. Conversely, training-based verifiers like RoboMonkey degrade baseline performance ($55.0\% \rightarrow 44.8\%$) due to verifier overfitting, demonstrating the generalization advantages of LVP's training-free self-verification.

Out-of-distribution robustness (LIBERO-Plus-Long). LIBERO-Plus (Fei et al., 2026) is built on top of LIBERO (Liu et al., 2023) to evaluate how VLA models are robust under seven perturbations: *Objects layout*, *Camera viewpoints*, *Robot initial states*, *Language instructions*, *Light conditions*, *Background textures*, and *Sensor noise*. We evaluate LVP on the LIBERO-Plus-Long benchmark using the flow-matching policies, MolmoAct2 (Fang et al., 2026) and GR00T N1.7 (NVIDIA et al., 2025). As shown in Tab. 3, LVP provides robust test-time scaling under distribution shifts. For MolmoAct2, LVP with token pruning achieves an overall average improvement of $+1.9\%$ (reaching 69.5%). Under the GR00T backbone, LVP maintains a consistent $+1.8\%$ average success rate improvement. More specifically, we highlight that LVP yields significant improvement under perturbations that alter spatial alignment: *Objects layout*, *Camera viewpoints*, and *Robot initial states*. Base policies frequently fail under these shifts due to geometric mismatches. By explicitly projecting trajectory candidates into pixel space via perspective-aware action projection (Sec. 3.2), LVP enables the pre-trained VLA to evaluate spatial alignment directly on the image plane. Under the GR00T backbone, LVP achieves notable gains on camera perturbations ($+\mathbf{6.5}\%$ with full tokens and $+\mathbf{4.1}\%$ with token pruning) and yields a $+\mathbf{10.3}\%$ **success rate jump** under layout perturbations. Similarly, for MolmoAct2, LVP with token pruning improves robustness across these perturbations: robot ($+\mathbf{3.8}\%$), camera ($+\mathbf{1.2}\%$), and layout ($+\mathbf{1.0}\%$). These results demonstrate that LVP is robust under distribution shifts, especially spatial perturbations.

4.2 Real-World Experiments

We evaluate LVP on a UR10e robot in the real world under both an unperturbed setting and perturbed object layouts and camera viewpoints. We evaluate four manipulation tasks using a GR00T N1.7 (NVIDIA et al., 2025) trained on 96 demonstrations (24 per task). For each task, we run 24 evaluation rollouts across 12 initial scene configurations. Following (Fei et al., 2026), *Layout* perturbations add a distractor object or shift object positions, while *Camera* perturbations use two different third-person viewpoints unseen during training. See Appendix C for details.

Table 5: Ablation study on SimplerEnv (WidowX) using GR00T N1.7 backbone. We compare our Full LVP to the variants of visual prompting (A-C), and the variants of likelihood-based verification (D-E).

Method	Average (%)
GR00T N1.7	55.0
(A) raw action parameters	55.6 (+0.6)
(B) only translation	55.1 (+0.1)
(C) only rotation	57.0 (+2.0)
(D) binary VQA (yes/no)	54.1 (-0.9)
(E) argmax choice	57.1 (+2.1)
Full LVP	**60.0 (+5.0)**

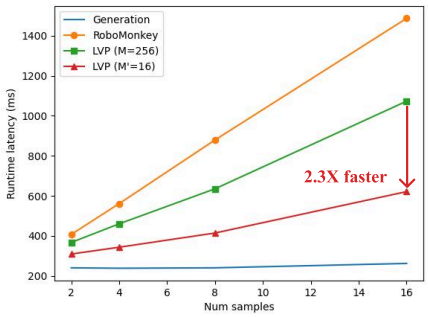


Figure 2: Runtime latency profiling (ms) across varying numbers of action candidates. We report generation latency of OpenVLA with prefix KV-caching and verification latency of LVP and RoboMonkey.

As shown in Tab. 4, LVP improves performance in both the unperturbed and perturbed settings. In the unperturbed setting, LVP yields larger gains over the base GR00T N1.7 model on challenging tasks (i.e., StackCube and HangTape), suggesting that LVP is beneficial for tasks requiring precise manipulation. Moreover, the consistent gains under both perturbations suggest LVP's robustness to changes in scene configuration and viewpoint. Qualitative results in Fig. 3 suggest that LVP selects the most proper action candidate, improving spatial alignment capability of the base policy. See Fig. 8 in Appendix for more qualitative samples.

4.3 Ablation Studies and Analyses

We validate the effectiveness of the proposed components of Likelihood-Based Visual Prompting (LVP) framework: (1) perspective-aware action projection, (2) token-likelihood-based verification, and (3) attention-guided token pruning.

We compare perspective-aware action projection (Sec. 3.2) against alternatives in Tab. 5: (A) query the raw action parameters as text with original image, (B) use only translation overlay, and (C) use only rotation overlay at final location. Providing raw action parameters as text (Variant A) yields only a $+0.6\%$ improvement over the base policy. This highlights that vision-language backbones possess dense spatial reasoning capabilities over images, while they struggle to understand numerical vectors. While rendering only translational trajectory (Variant B) achieves an average success rate of 55.1%, suffering on orientation-sensitive tasks, rendering only rotation axes (Variant C) improves average performance to 57.0%. Combining both translation and rotation achieves the best performance, providing complete spatial clarity for the verification.

Next, we compare token likelihood-based verification (Sec. 3.3) against different scoring functions in Tab. 5: (D) query the binary prompt (*Is this action correct? Choose: "Yes", "No"*) and evaluates $P_\theta(\text{"Yes"} \mid I_i, q) - P_\theta(\text{"No"} \mid I_i, q)$ and (E) query the 4-choice prompt (*Poor, Fair, Good, Excellent*) but selects candidates based solely on the single highest token logit ($\arg\max_{v_k} P_\theta(v_k \mid I_i, q)$). Reframing verification as a coarse binary question (Variant D) degrades performance below the base policy because it cannot differentiate between subtle trajectory qualities. Using a hard argmax choice (Variant E) improves $+2.1\%$ over the base policy, but our token likelihood-based verification outperforms this variant by constructing a smooth, fine-grained rewards.

To evaluate the efficiency gains of attention-guided token pruning (Sec. 3.4), we profile the runtime latency of LVP against RoboMonkey (Kwok et al., 2025) in Fig. 2. First, we reduce generation latency of VLA models by leveraging a prefix KV-cache and batch-wise parallel decoding. The action generation phase maintains a near-constant latency of $\sim$ 240ms to 262ms as the number of samples increases from $N = 2$ to $N = 16$. Our proposed attention-guided token pruning dramatically decreases verification latency from 811ms to 359ms for $N = 16$ samples, achieving a $\mathbf{2.26}\times$ **speedup**. Compared to RoboMonkey, LVP ($M = 256$ and $M' = 16$) does not only operate much faster in inference time, but also does not require any additional training.

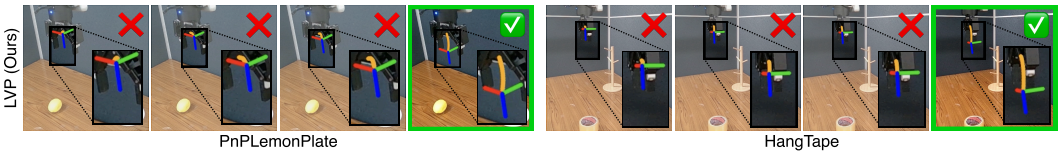


Figure 3: Qualitative results of LVP using GR00T N1.7 backbone on real-world manipulation tasks. For each task, the four panels show the sampled action candidates, each rendered as a visual prompt: the orange line denotes the future translation and the RGB axes denote the target gripper's orientation. Our proposed LVP selects the most spatially aligned action candidate (green box).

4.4 Bridging the Action-Perception Gap: Why Self-Verification Succeeds

A fundamental question arises regarding test-time self-verification: *If a pre-trained VLA policy fails during initial action generation, how can the model itself correctly verify action candidates?*

If policy failures were primarily caused by a lack of spatial reasoning within the underlying VLM backbone, self-verification would fail to yield performance gains. However, Tab. 5 shows that the underlying VLM understands action-related visual prompts and improves the performance by action verification, similar to PIVOT (Nasiriany et al., 2024). Even though VLA backbones are fine-tuned with robotics data, our experimental results demonstrate that the underlying VLM might preserve the original spatial-semantic understanding. Therefore, we assume that base VLA failures stem from an **action-perception representation gap** inherent in current VLA training.

This representation gap manifests from a structural imbalance in training spaces:

1. **VLM Visual-Semantic Space:** The underlying VLM backbone is pre-trained on massive, web-scale multimodal datasets, endowing it with rich, fine-grained 2D spatial reasoning and semantic grounding capabilities over visual inputs.

2. **Action Decoder Parameter Space:** Conversely, the policy's action decoder (e.g., MLP heads or flow-matching networks) is trained on comparatively constrained robotics datasets. The action decoder often struggles to map continuous 3D coordinate trajectories accurately, leading to spatial misalignment during single-sample generation.

LVP directly bridges this representation gap by using 3D-to-2D perspective projection to map action candidates $\{a_i\}_{i=1}^{N}$ back into the VLM's native **2D visual-semantic space**. By transforming candidates into spatial visual prompt overlays $I_i = \Pi(a_i, I)$, the verification task is reframed as an evaluative Visual Question Answering (VQA) task. This allows the underlying VLM backbone to inspect the physical trajectory candidates in its native visual-semantic space. See Appendix D for qualitative analysis of VLM's spatial reasoning capability.

5 Conclusion

We introduce **Likelihood-Based Visual Prompting (LVP)**, a training-free, verifier-free, and decoder-agnostic test-time scaling framework for Vision-Language-Action (VLA) models. LVP demonstrates that generalist VLA policies possess an inherent, zero-shot capacity for self-verification without requiring additional verifier networks, value function training, or policy fine-tuning. By projecting 3D action candidates into 2D image space as spatial visual prompts, LVP converts raw robotic parameters into explicit visual signals legible to the underlying Vision-Language Model (VLM) backbone. Reframing action verification as an evaluative Visual Question Answering (VQA) task, LVP extracts conditional token log-probabilities to derive continuous, well-calibrated expected utility scores for trajectory candidate ranking. Furthermore, to make multi-sample verification computationally efficient for closed-loop real-world execution, we introduce a visual token pruning strategy that combines action-generation attention saliency with prompt-difference masking. Our extensive evaluations across diverse VLA backbones and simulated/real-world benchmarks validate the effectiveness and generality of LVP.

AI USE STATEMENT

In this work, we use generative AI tools for drafting the paper, polishing writing, and code implementation. We have reviewed all AI-assisted work. LLM-generated code and draft was verified. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide source code as supplementary materials.

REFERENCES

Abdelrahman Abdallah, Jamshid Mozafari, Bhawna Piryani, and Adam Jatowt. ASRank: Zero-shot re-ranking with answer scent for document retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision (ECCV)*, 2024a.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Yangyi Chen, Karan Sikka, Michael Cogswell, Heng Ji, and Ajay Divakaran. Dress: Instructing large vision-language models to align and interact with humans via natural language feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024c.

Hyeonbeom Choi, Daechul Ahn, Youhan Lee, Taewook Kang, Seongwon Cho, and Jonghyun Choi. SCALE: Self-uncertainty conditioned adaptive looking and execution for vision-language-action models. In *International Conference on Machine Learning (ICML)*, 2026.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning (ICML)*, 2023.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

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: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 2014.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025a.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025b.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Suhyeok Jang, Dongyoung Kim, Changyeon Kim, Youngsuk Kim, and Jinwoo Shin. Verifier-free test-time sampling for vision language action models. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Hwidong Kim, Yunho Kim, and Tae-Kyun Kim. 3dphysvideo: Consistency-guided flow sde for video generation via 3d scene reconstruction and physical simulation. *arXiv preprint arXiv:2605.16795*, 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)*, 2025.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *Conference on Robot Learning (CoRL)*, 2025.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. Llm-as-a-verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, et al. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arxiv:2501.14818*, 2025.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pretraining for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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 (NeurIPS)*, 2023.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning (CoRL)*, 2024.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

NVIDIA, Alisson Azzolini, Hannah Brandon, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Francesco Ferroni, Rama Govindaraju, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, et al. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, 2025a.

Physical Intelligence, 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. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025b.

Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

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

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023a.

Lingfeng Yang, Yueze Wang, Xiang Li, Xinlong Wang, and Jian Yang. Fine-grained visual prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering vision-language-action models as anti-exploration: A test-time scaling approach. *arXiv preprint arXiv:2512.02834*, 2025.

Zhangyue Yin, Qiushi Sun, Zhiyuan Zeng, Qinyuan Cheng, Xipeng Qiu, and Xuanjing Huang. Dynamic and generalizable process reward modeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning (ICML)*, 2025.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations (ICLR)*, 2025.

Shijie Zhou, Ruiyi Zhang, Huaisheng Zhu, Branislav Kveton, Yufan Zhou, Jiuxiang Gu, Jian Chen, and Changyou Chen. Multimodal llms as customized reward models for text-to-image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

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

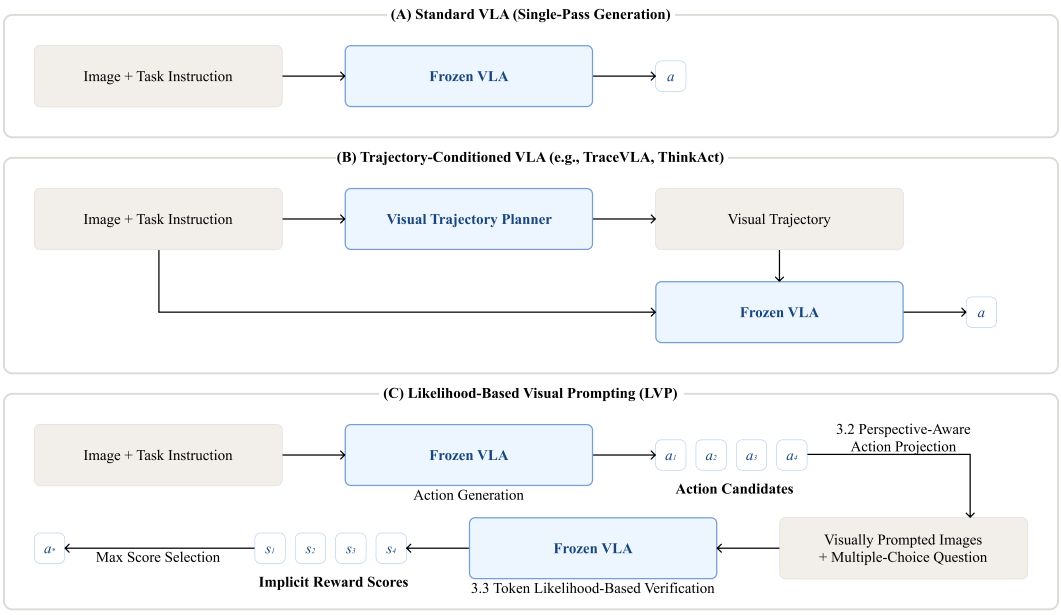


Figure 4: Conceptual comparison of visual prompt utilization across VLA paradigms. **(A) Standard VLA:** Directly maps camera observations to 3D actions in a single pass. **(B) Trajectory-Conditioned VLA (e.g., TraceVLA (Zheng et al., 2025), ThinkAct (Huang et al., 2025)):** Uses an external visual module or latent planner to generate a 2D/3D visual trajectory *before* action generation to guide the policy. **(C) Likelihood-Based Visual Prompting (LVP, Ours):** Reverses this flow by projecting action candidates $\{a_i\}$ into pixel space as visual prompts $\{I_i\}$ *after* action proposal generation. The VLA itself evaluates these spatial overlays zero-shot via VQA token likelihoods, enabling test-time self-verification.

Table 6: Comparison of test-time scaling (TTS) methods for Vision-Language-Action (VLA) models. Existing approaches require training external verifier models (V-GPS (Nakamoto et al., 2024), RoboMonkey (Kwok et al., 2025), TACO (Yang et al., 2025)), fine-tuning the base policy (MG-Select (Jang et al., 2026)), or are restricted solely to autoregressive decoders (SCALE (Choi et al., 2026), MG-Select (Jang et al., 2026)). In contrast, **LVP (Ours)** requires no additional training or fine-tuning, natively supports both autoregressive and non-autoregressive (diffusion/flow-matching) action decoders, and performs zero-shot verification via visual prompts.

Method	Signal	Enhancement	Additional Training	VLA Architecture Support
V-GPS, RoboMonkey, TACO	Auxiliary verifier	Sample & verification	Train verifier model	**Autoregressive & Diffusion/Flow**
MG-Select	Self-uncertainty	Sample & verification	Fine-tune VLA model	Autoregressive
SCALE	Self-uncertainty	Perception & sample	**None**	Autoregressive
LVP (Ours)	Visual prompt	Sample & verification	**None**	**Autoregressive & Diffusion/Flow**

A Implementation Details

A.1 Hardware and Computing Infrastructure

All simulation evaluations are conducted on an NVIDIA RTX 4090 GPU (24 GB VRAM). Candidate sampling and token likelihood verification leverage batch-wise decoding.

A.2 Detailed Perspective-Aware 3D-to-2D Action Projection

In this section, we introduce the camera projection and rendering pipeline that maps 3D action trajectories into 2D visual prompts overlaying the observation.

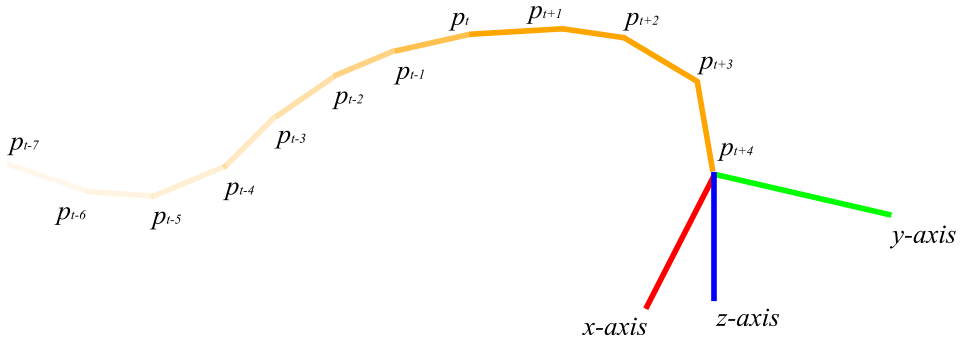


Figure 5: Structure of our visual prompts. We visualize historical translation as gradually transparent orange lines, and future translation as orange lines, supporting arbitrary length of *action chunk*. At the final location, the target gripper's orientation is rendered by a tri-axial RGB coordinate.

LVP Evaluative Prompt Template $q(g)$

Given the task, {`task_instruction`},
Robot action is visualized on the image.
Orange lines are future translation of the robot arms.
The future gripper rotation is overlaid as axis markers: X=red, Y=green, Z=blue.
Please evaluate the quality of robot action.
Choose one option: "Poor", "Fair", "Good", "Excellent".

Figure 6: The ordinal multiple-choice question (MCQ) template used to prompt the VLA model in action verification stage.

3D robotic parameters. The current state of robotic arm can be represented as a homogeneous transformation matrix $\mathbf{T}_t^{\text{robot}} \in \text{SE}(3)$:

$$\mathbf{T}_t^{\text{robot}} = \begin{bmatrix} \mathbf{R}_t & \mathbf{t}_t \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{10}$$

where $\mathbf{R}_t \in \text{SO}(3)$ represents the rotation of end-effector, and $\mathbf{t}_t \in \mathbb{R}^{3\times 1}$ represents the translation of end-effector. An action candidate a_i predicted by the VLA model is defined as:

$$a_i = [\Delta x, \Delta y, \Delta z, \Delta\phi, \Delta\theta, \Delta\psi, s^{\text{grip}}]^T \in \mathcal{A}, \tag{11}$$

where $[\Delta x, \Delta y, \Delta z]^T$ represents the translational displacement, $[\Delta\phi, \Delta\theta, \Delta\psi]^T$ represents the rotational end-effector delta angles (roll, pitch, yaw), and $s^{\text{grip}} \in \{0, 1\}$ represents the binary gripper open/close state. We construct the relative transformation delta matrix $\Delta\mathbf{T}(a_i) \in \text{SE}(3)$ from these action parameters and compute the future pose as:

$$\mathbf{T}_{t+1}^{\text{robot}} = \mathbf{T}_t^{\text{robot}} \cdot \Delta\mathbf{T}(a_i). \tag{12}$$

Camera projection. Next, we transform 3D points $\mathbf{x}$ from the robot coordinate to the camera coordinate using the camera extrinsic parameters, $\mathbf{T}_{\text{robot}}^{\text{camera}} = \begin{bmatrix} \mathbf{R}_{\text{ext}} & \mathbf{t}_{\text{ext}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \text{SE}(3)$:

$$[x_t^{\text{camera}}, y_t^{\text{camera}}, z_t^{\text{camera}}, 1]^T = \mathbf{T}_{\text{robot}}^{\text{camera}} \cdot \mathbf{T}_t^{\text{robot}} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{13}$$

We transform 3D points corresponding to the origin $\mathbf{x}_{origin} = [0, 0, 0]^T$, x-axis $\mathbf{x}_{x-axis} = [0.05, 0, 0]^T$, y-axis $\mathbf{x}_{y-axis} = [0, 0.05, 0]^T$, and z-axis $\mathbf{x}_{z-axis} = [0, 0, 0.05]^T$. Let $\mathbf{p}_t^{\text{camera}} = [x_t^{\text{camera}}, y_t^{\text{camera}}, z_t^{\text{camera}}]^T$. Using the camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times 3}$, we map these points into the 2D projective space:

$$\mathbf{w}_t = [w_{t,1} \quad w_{t,2} \quad w_{t,3}]^T = \mathbf{K}\mathbf{p}_t^{\text{camera}}. \tag{14}$$

The final pixel coordinate $\mathbf{u}_t = [u_t, v_t]^T \in \mathcal{U}$ on the image is obtained via perspective division:

$$u_t = \frac{w_{t,1}}{w_{t,3}}, \quad v_t = \frac{w_{t,2}}{w_{t,3}}. \tag{15}$$

Visual prompt rendering. To effectively encode the current state and action candidate, we design structured visual prompting. The translational trajectory is rendered by a sequence of orange lines and the target gripper's orientation is rendered by a tri-axial RGB coordinate (X=red, Y=green, Z=blue). We can acquire N visually prompted observations for each corresponding candidate:

$$\{I_i\}_{i=1}^N = \Pi(\{a_i\}_{i=1}^N, I), \tag{16}$$

where Π represents the projection function. By rendering the 3D robotic parameters into visual pixel-space overlays, we can fully exploit the VLM's spatial understanding capability. Fig. 5 demonstrates visual prompt overlay examples.

A.3 Detailed Token Likelihood-Based Verification

Verification is structured as a single-turn, multiple-choice Visual Question Answering (VQA) query fed to the frozen VLA model. The complete text prompt template, which consists of the task instruction, information of visual overlays, and multiple choices, is illustrated in Fig. 6. We obtain a normalized conditional probability distribution (Eq. 6) from the conditional next-token probabilities over the vocabulary set $\mathcal{V} = \{v_{\text{Poor}}, v_{\text{Fair}}, v_{\text{Good}}, v_{\text{Excellent}}\}$. The implicit reward score $\mathcal{S}(a_i)$ is formulated as Eq. 7, where the utility mapping is:

$$\mathcal{W}(v_{\text{Poor}}) = -3, \mathcal{W}(v_{\text{Fair}}) = -1, \mathcal{W}(v_{\text{Good}}) = 1, \mathcal{W}(v_{\text{Excellent}}) = 3. \tag{17}$$

A.4 Detailed Attention-Guided Visual Token Pruning

This section details the attention extraction and prompt-difference token selection mechanism.

Action-Generation Attention Extraction. During the **Action Generation Stage**, the input token sequence fed to the transformer decoder is formatted as:

$$\mathbf{T}_{\text{gen}} = [\mathbf{X}_{\text{vis}} \,\|\, \mathbf{X}_{\text{instruct}} \,\|\, \mathbf{X}_{\text{action}}], \tag{18}$$

where $\mathbf{X}_{\text{vis}} = \{x_j^v\}_{j=1}^M$ represents the full sequence of M visual patch tokens (e.g., $M = 256$), $\mathbf{X}_{\text{instruct}}$ contains the tokenized natural language task instruction, and $\mathbf{X}_{\text{action}}$ contains the generated action tokens or embeddings.

During the forward pass of action generation, we extract the attention weights allocated from the predicted action tokens $x^a \in \mathbf{X}_{\text{action}}$ to each visual token $x_j^v \in \mathbf{X}_{\text{vis}}$. For a specific transformer layer l, the attention weight assigned to the j-th visual token is denoted as $\text{Attn}^{(l)}(x^a \to x_j^v)$. To capture high-level task semantics, we define the cumulative attention score A_j by averaging attention weights across the $\mathcal{L}$ decoder layers:

$$A_j = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \text{Attn}^{(l)}(x^a \to x_j^v). \tag{19}$$

The resulting vector $\mathbf{A} = [A_1, A_2, \ldots, A_M]^T \in \mathbb{R}^M$ serves as an implicit spatial saliency mask, highlighting image regions corresponding to the robot arm and target manipulation objects while assigning low values to static background regions.

Task-Salient Visual Token Selection. Using the cumulative attention vector $\mathbf{A}$, we select the M' visual tokens with the highest attention values ($M' \ll M$, e.g., $M' = 16$):

$$\mathbf{X}_{\text{vis}}^{\text{attn}} = \text{Top-}M'(\{x_j^v\}_{j=1}^M, \mathbf{A}). \tag{20}$$

Prompt-Difference Token Selection. Because visual prompt overlays (orange trajectory paths and RGB orientation coordinate frames) are rendered onto the observation image *after* the initial action generation pass, the attention weights in $\mathbf{A}$ do not account for these newly added pixels.

To isolate and retain visual prompt features, we pass both the original unprompted image I and the visually prompted image $I_i = \Pi(a_i, I)$ through the visual encoder. Let $\mathbf{X}_{\text{vis}}^{\text{original}} \in \mathbb{R}^{M \times d}$ and $\mathbf{X}_{\text{vis}} \in \mathbb{R}^{M \times d}$ denote their corresponding d-dimensional visual token embedding matrices. We compute the element-wise ℓ_2-norm difference between these embedding sequences:

$$D_j = \|\mathbf{x}_{j,\text{vis}} - \mathbf{x}_{j,\text{vis}}^{\text{original}}\|_2 \quad \text{for } j \in \{1, \ldots, M\}. \tag{21}$$

The magnitude D_j directly measures the spatial embedding shift introduced by rendering the visual overlay at patch j. We then extract the top-M' prompt-affected visual tokens using the difference vector $\mathbf{D} = [D_1, \ldots, D_M]^T$:

$$\mathbf{X}_{\text{vis}}^{\text{prompt}} = \text{Top-}M'(\{x_j^v\}_{j=1}^{M}, \mathbf{D}). \tag{22}$$

Dual-Selection Union and Verification. Finally, we compute the set union of the task-salient tokens and prompt-salient tokens to form the pruned visual token set:

$$\mathbf{X}_{\text{vis}}^{\text{pruned}} = \mathbf{X}_{\text{vis}}^{\text{attn}} \cup \mathbf{X}_{\text{vis}}^{\text{prompt}}. \tag{23}$$

This union set $\mathbf{X}_{\text{vis}}^{\text{pruned}}$ is fed to the VLA in its verification stage alongside the evaluative VQA question $q(g)$. By eliminating background tokens while guaranteeing the inclusion of both task-relevant objects and spatial prompt overlays, LVP reduces the transformer self-attention sequence length, enabling fast, low-latency test-time verification.

A.5 Model Bacbkones and Pre-trained Checkpoints

All evaluations are performed using officially released pre-trained weights for the respective VLA backbones without fine-tuning:

- **OpenVLA (Kim et al., 2025):** Autoregressive VLA backbone using Prismatic (Karamcheti et al., 2024) VLM. Evaluated using the official OpenVLA checkpoints on the standard LIBERO suites (Spatial, Object, Goal, and Long).
- **MolmoAct (Lee et al., 2026):** Autoregressive VLM backbone using Molmo (Deitke et al., 2025) VLM. Evaluated using the officially released zero-shot and fine-tuned MolmoAct checkpoints on the SimplerEnv (Google Robot) benchmark.
- **MolmoAct2 (Fang et al., 2026):** Flow-matching based VLA backbone using Molmo2-ER (Clark et al., 2026) VLM. Evaluated using the official MolmoAct2 checkpoint on the LIBERO-Plus-Long benchmark.
- **GR00T N1.7 (NVIDIA et al., 2025):** Flow-matching based VLA backbone using Cosmos-Reason2 (NVIDIA et al., 2025) VLM. Evaluated using the official GR00T N1.7 checkpoints on the SimplerEnv (WidowX), SimplerEnv (Google Robot), and LIBERO-Plus-Long benchmarks.

We mainly use shared weights for both action generation and action verification. OpenVLA (Kim et al., 2025) and MolmoAct (Lee et al., 2026), which are autoregressive VLA backbones, extends the original vocabulary of language tokenizer. Therefore, we can natively acquire the next-token logits for text tokens. MolmoAct2 (Fang et al., 2026) backbone predicts depth tokens in autoregressive manner before generating action candidates with flow-matching decoder. We leverage the lm head similar to the autoregressive policies. On the other hand, GR00T N1.7 (NVIDIA et al., 2025) backbone originally loads only the early layers of the VLM backbone and flow-matching decoder generates action candidates conditioned by the intermediate features. We load the full VLM backbone (Cosmos-Reason2-2B) to acquire the next-token logits.

A.6 Baseline Details

In Tab. 1, we evaluate SCALE (Choi et al., 2026) following the official implementation. In Tab. 2a, we evaluate RoboMonkey (Kwok et al., 2025) using the official checkpoint and following the official implementation. The action verifier is trained on synthetic action candidates generated by

Table 7: Success rate comparisons (%) on the LIBERO benchmark using the OpenVLA backbone. We compare LVP using a full token budget ($M = 256$) and our pruned token budget ($M' = 16$) against base OpenVLA and prior test-time methods. An asterisk (*) denotes results reproduced by utilizing officially released checkpoints.

	Task suites (Success Rate, %)				
Method	Spatial	Object	Goal	Long	Average
OpenVLA*	86.2	86.2	77.7	52.7	75.7
+ RoboMonkey	–	–	–	56.5 (+3.8)	–
+ TACO	–	–	–	60.0 (+7.3)	–
+ MG-Select	81.7 (-4.5)	72.5 (-13.7)	73.6 (-4.1)	55.4 (+2.7)	70.8 (-4.9)
+ SCALE	**89.5 (+3.3)**	**91.0 (+4.8)**	**82.3 (+4.6)**	**63.3 (+10.6)**	**81.5 (+5.8)**
+ LVP (Full tokens)	85.0 (-1.2)	89.4 (+3.2)	76.6 (-1.1)	61.2 (+8.5)	78.1 (+2.4)
+ LVP (Pruned)	84.8 (-1.4)	88.6 (+2.4)	81.0 (+3.3)	61.0 (+8.3)	78.4 (+2.7)

OpenVLA (Kim et al., 2025) policy. The performance degradation demonstrates that training-based verifiers are not generalizable beyond the training action distribution.

A.7 Number of Candidates

In Tab. 9, Tab. 10, and Tab. 11, we report success rates of LVP on the simulation benchmark across varying numbers of action candidates. We report the best result for each setting in the main quantitative evaluation. Crucially, scaling test-time compute does not monotonically improve performance because verifier feedback inherently contains noise. We observe a task-complexity trade-off when scaling the candidate sample size N: for simpler tasks, a smaller N prevents noisy candidate evaluations from degrading a reliable baseline policy. Conversely, for challenging or out-of-distribution tasks where the base policy struggles, a larger N expands trajectory exploration, enabling the verifier to identify successful execution paths that the base policy rarely proposes in a single sample.

B Additional Simulation Experiments

LIBERO benchmark. We evaluate LVP using the OpenVLA (Kim et al., 2025) backbone on the LIBERO (Liu et al., 2023) benchmark, which covers four distinct robot capabilities: reasoning about spatial relations (*Spatial*), manipulating different objects (*Object*), completing goal (*Goal*), and reasoning long-horizon (*Long*). We compare LVP against state-of-the-art test-time baselines in Tab. 7. Our framework increases the average success rate from 75.7% of the baseline OpenVLA policy to 78.1% (full tokens) and 78.4% (with 16-token attention-guided pruning). Crucially, LVP outperforms established verifier-based baselines, RoboMonkey (Kwok et al., 2025) and TACO (Yang et al., 2025) on the LIBERO-Long tasks without requiring any downstream verifier training. While SCALE (Choi et al., 2026) achieves a higher average of 81.5%, it is strictly restricted to autoregressive policies due to its self-uncertainty formulation. In contrast, LVP demonstrates competitive performance while achieving decoder-agnostic paradigm.

C Details on Real-world Experiments

Robot and policy setup. We use a 6-DoF UR10e robot equipped with a Robotiq 2F-85 gripper. The VLA policy takes as input the two RGB images, a language instruction, and absolute joint proprioceptive states. For visual observations, we use RealSense D455 cameras for both third-person and wrist views, with images resized to 224×224. The policy predicts action chunks of 30 steps and executes the first 15 steps before replanning. Actions are executed at 30 Hz.

Camera extrinsic calibration. We perform extrinsic calibration of the third-person camera for action trajectory projection in LVP. Specifically, we hold an ArUco (Garrido-Jurado et al., 2014)

Table 8: Success rates (%) on four real-world manipulation tasks using a UR10e robot. We evaluate GR00T N1.7 with and without LVP ($N = 4, M' = 8$) under layout and camera perturbations.

			Tasks (Success Rate, %)				
Perturb.	Sub-dimension	Model	PnPLemonPlate	StackCube	PressStapler	HangTape	Average
Layout	Distractor	GR00T N1.7	33.3	25.0	33.3	16.7	27.1
		+ LVP	**50.0 (+16.7)**	**33.3 (+8.3)**	**41.7 (+8.4)**	**41.7 (+25.0)**	**41.7 (+14.6)**
	Tgt. Obj. Position	GR00T N1.7	16.7	16.7	66.7	41.7	35.4
		+ LVP	**41.7 (+25.0)**	**50.0 (+33.3)**	**66.7 (+0.0)**	**50.0 (+8.3)**	**52.1 (+16.7)**
Camera	View1	GR00T N1.7	16.7	16.7	75.0	0.0	27.1
		+ LVP	**33.3 (+16.6)**	**16.7 (+0.0)**	**83.3 (+8.3)**	**25.0 (+25.0)**	**39.6 (+12.5)**
	View2	GR00T N1.7	16.7	0.0	50.0	0.0	16.7
		+ LVP	**33.3 (+16.6)**	**8.3 (+8.3)**	**50.0 (+0.0)**	**16.7 (+16.7)**	**27.1 (+10.4)**

board with four markers using the gripper and collect 10 calibration poses around the home pose: $\pm 15^\circ$ rotations in roll, pitch, and yaw, and ± 10 cm translations along the x- and y-axes. These poses are used to compute the camera-to-robot transformation matrix.

Task setup. The following four tasks and instructions are used throughout the experiments:

- **PnPLemonPlate**: `put the lemon on the plate`
- **StackCube**: `stack the red cube on the green cube`
- **PressStapler**: `press the stapler`
- **HangTape**: `hang the tape on the {top/bottom} peg`

Each task includes 12 predefined scene configurations defined by different object poses.

Training and evaluation. We collect 96 teleoperated demonstrations (24 per task) using GELLO (Wu et al., 2024), recorded at 30 Hz. We train GR00T N1.7 (NVIDIA et al., 2025) for 20,000 steps using two NVIDIA H100 GPUs. During evaluation, inference is performed on an NVIDIA RTX Pro 5000 GPU. For each task, we run 24 evaluation rollouts across the 12 predefined scene configurations.

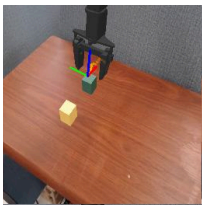
Perturbations. Following (Fei et al., 2026), we consider two types of perturbations: *Layout* and *Camera*. For the *Layout* perturbation, we apply the following:

- **Distractor**: Add a distractor object to the scene: a banana for PnPLemonPlate, a blue cube for StackCube, a mouse for PressStapler, and a cup for HangTape.
- **Target Object Position**: Shift the target object diagonally on the tabletop plane by $5\,\mathrm{cm}$ in each horizontal direction.

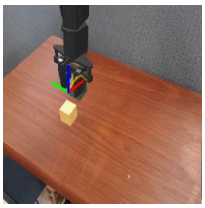
For the *Camera* perturbation, we use two different shifts of the third-person viewpoint, as shown in Fig. 8e and Fig. 8f. For each perturbation setting, we run 12 rollouts across the 12 predefined scene configurations. Tab. 8 reports the detailed results for each perturbation.

D Qualitative Analysis: VLM Understanding of Visual Prompts

To verify that the underlying Vision-Language Model (VLM) backbone understands the spatial-geometric information embedded in our perspective-aware action projection, we conduct a qualitative evaluation in an open-ended Visual Question Answering (VQA) format. Specifically, we feed the visually prompted images ($I_i = \Pi(a_i, I)$) to the VLM and prompt it to evaluate the proposed candidate. The answers of VLM in Fig. 7 show that the underlying VLM backbone can understand the visual prompts well.



The green block is positioned above the yellow block, indicating readiness for stacking. The robot arm is poised to perform the stacking action, but the current state shows no physical contact between the blocks. The gripper appears to be oriented correctly for grasping the green block, and the rotation markers suggest precise alignment. However, since the blocks are not yet stacked, the action has not been completed successfully.



The robot is currently grasping the green block and lifting it slightly off the table, preparing to place it on the yellow block. The green block is being moved vertically upward by the robot's gripper, which is oriented at an angle relative to the yellow block. The orange lines indicate the translation of the robot arms, showing the movement trajectory. The future rotation of the gripper is marked with axes: X=red, Y=green, Z=blue, suggesting the robot will rotate the green block to align it properly with the yellow block before stacking. The current action is focused on lifting the green block, and the next step will likely involve positioning it directly above the yellow block for stacking. The environment appears stable with no obstructions, allowing for precise manipulation.



The robot is currently grasping the middle drawer, but it has not yet begun to open it. The gripper's position and orientation suggest an intent to interact with the drawer, but the drawer remains closed. The robot's action is in progress, but the outcome has not been achieved yet. The middle drawer is the target for this action, and the robot's current position indicates that it is about to initiate the opening motion. The orange lines represent the translation of the robot arms, showing the movement towards the drawer. The axes of rotation (X, Y, Z) indicate the direction of the gripper's movement and the expected rotation needed to open the drawer. Overall, the robot's action is in the early stages of opening the middle drawer, and the visualization provides a clear indication of the current state and future trajectory of the task.



The robot's action has caused the middle drawer to open, fulfilling the task requirement. The robot arm's movements are smooth and precise, with the gripper aligning correctly with the handle. The orange lines trace the translation path of the arm, showing controlled motion. The rotation around the X, Y, and Z axes is clearly marked, indicating a systematic approach to opening the drawer. Overall, the robot's performance is efficient and accurate.

Figure 7: Qualitative textual critiques generated by the VLM given visual prompt overlays. On the Simpler (WidowX) and Simpler (Google Robot), we generate action candidate using GR00T N1.7 (NVIDIA et al., 2025) and evaluate the visual prompt using Cosmos-Reason2-2B (NVIDIA et al., 2025).

E LIMITATIONS

While LVP provides significant performance and generalization gains without additional training, it has a few practical limitations: (1) **Calibration dependencies:** Perspective-aware projection depends on camera intrinsic and extrinsic parameters ($\mathbf{K}$, $\mathbf{T}_{\text{robot}}^{\text{camera}}$). (2) **VLA model boundedness:** As a test-time sampling method, LVP selects from candidates drawn from the base policy π_θ; if the base policy fails to sample any viable trajectory, LVP cannot synthesize a successful plan. (3) **Inference latency**: Despite our visual token pruning delivering a $2.26\times$ speedup, evaluating multi-sample visual prompts introduces additional computational latency over single-sample execution.

Table 9: Success rate comparisons (%) on the LIBERO benchmark using the OpenVLA backbone. We compare LVP using a full token budget ($M = 256$) and our pruned token budget ($M' = 16$) across varying numbers of action candidates (N).

		Task suites (Success Rate, %)				
N	Token pruning	Spatial	Object	Goal	Long	Average
4	×	84.8	89.6	77.4	59.6	77.9
8	×	85.0	89.4	76.6	61.2	78.1
16	×	87.8	89.8	78.4	62.4	79.6
4	✓	85.0	88.6	81.2	58.2	78.3
8	✓	84.8	86.6	81.0	61.0	78.4
16	✓	85.6	86.8	78.4	56.8	76.9

Table 10: We report success rate of LVP using a full token budget ($M = 64$) and our pruned token budget ($M' = 8$) using GR00T N1.7 backbone across varying numbers of action candidates (N).

(a) Success rate comparisons (%) on SimplerEnv (WidowX).

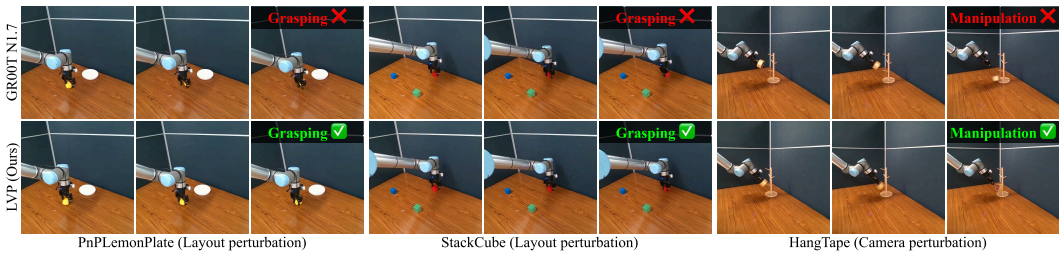
		Tasks (Success Rate, %)				
N	Token pruning	Spoon	Carrot	Eggplant	Cube	Average
4	×	81.0	68.0	54.5	36.5	60.0
8	×	77.5	68.0	47.5	38.5	57.9
16	×	84.0	59.0	40.5	40.5	56.0
4	✓	83.0	66.0	47.5	37.0	58.4
8	✓	74.5	75.5	45.0	35.0	57.5
16	✓	76.5	62.5	39.0	38.0	54.0

(b) Success rate comparisons (%) on SimplerEnv (Google Robot).

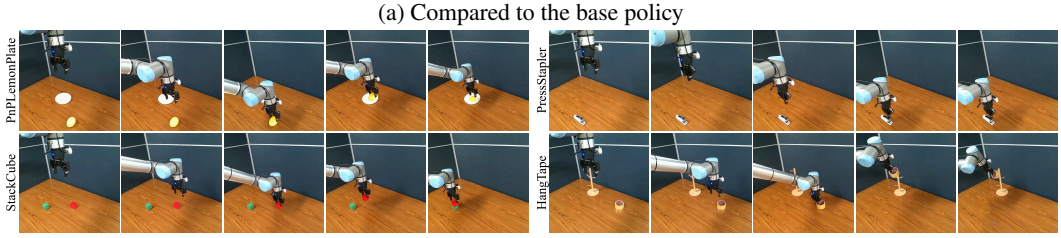
		Tasks (Success Rate, %)			
N	Token pruning	Open drawer	Close drawer	Place in closed drawer	Average
4	×	45.5	56.0	12.5	38.0
8	×	40.5	55.5	16.0	37.3
16	×	50.0	60.5	15.0	41.8
4	✓	45.0	56.5	16.5	39.3
8	✓	45.0	58.0	18.0	40.3
16	✓	45.0	60.0	19.0	41.3

Table 11: Success rate (%) on the perturbed LIBERO-Plus-Long benchmark under seven domain shifts. We set $M = 392, M' = 28$ for MolmoAct2 and $M = 64, M' = 8$ for GR00T N1.7.

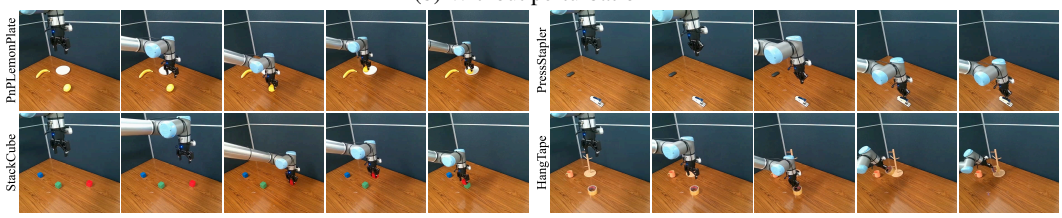
	Perturbations (Success Rate, %)							
N	Layout	Camera	Robot	Language	Light	Background	Noise	Average
MolmoAct2 + LVP w/o token pruning								
4	68.9	46.5	61.6	80.9	80.3	81.0	69.7	68.6
8	72.1	47.9	60.1	78.9	78.8	79.9	69.3	68.3
16	70.2	50.8	61.3	77.0	77.7	79.2	69.5	68.4
MolmoAct2 + LVP w/ token pruning								
4	69.6	49.2	63.9	79.1	81.8	82.7	69.0	69.5
8	69.9	49.4	62.1	80.2	81.0	80.3	69.3	69.1
16	70.8	47.3	63.6	82.5	81.8	78.9	70.2	69.6
GR00T N1.7 + LVP w/o token pruning								
4	54.2	35.3	47.1	70.2	71.5	76.1	47.7	55.6
8	51.9	39.9	48.9	71.0	71.5	75.1	48.1	56.5
16	52.6	32.9	46.8	70.2	67.2	73.7	49.7	54.6
GR00T N1.7 + LVP w/ token pruning								
4	52.9	36.0	47.6	73.4	66.1	75.4	48.1	55.5
8	57.7	37.5	46.1	72.3	67.2	74.4	51.2	56.5
16	51.0	37.2	43.8	71.3	69.3	75.1	52.1	55.6



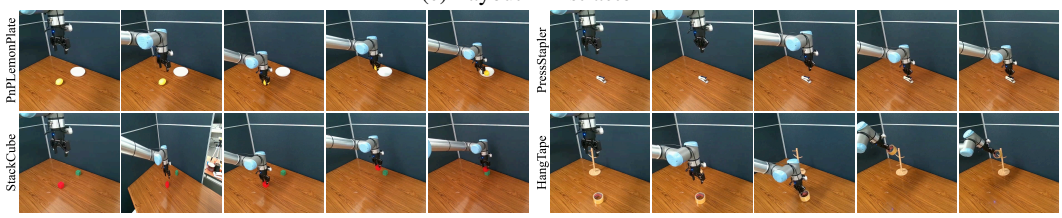
(a) Compared to the base policy



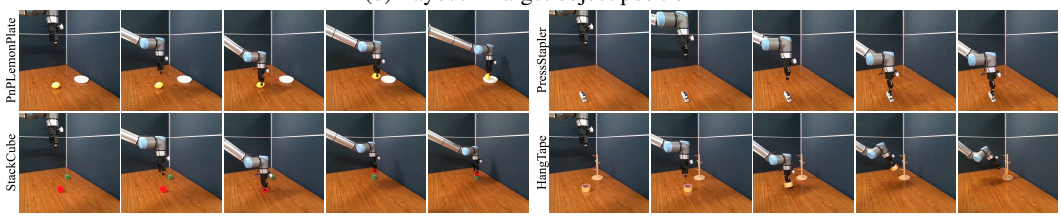
(b) Without perturbation



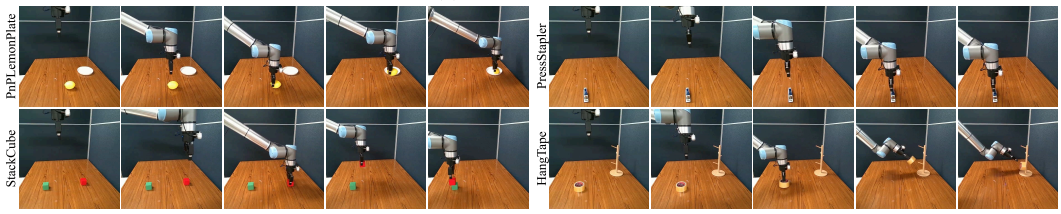
(c) Layout – Distractor



(d) Layout – Target object position



(e) Camera – View1



(f) Camera – View2

Figure 8: Qualitative results of LVP using GR00T N1.7 backbone on real-world manipulation tasks.