RETHINKING ADVERSARIAL PATCH ATTACKS ON VLA: FROM PIXEL SPACE TO PHYSICAL REALITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models are increasingly deployed in safety-critical manipulation tasks, making adversarial robustness a security-critical concern. Recent studies report that adversarial patches can reduce task success rate to near zero. But how much of this attack strength survives when a patch is fixed to a physical surface? We revisit this question by separating patch generation from patch evaluation. Pixel-space compositing places a patch directly in the input image, whereas a fixed physical texture must produce all of its appearances through camera projection. We cross these two settings in a 2×2 study on OpenVLA, OpenVLA-OFT, and π_0. Switching from pixel-space to geometry-consistent evaluation raises task success rate in all 24 combinations, by 3.0-69.2 percentage points. Re-optimizing under geometric constraints then recovers attack strength in 23 of these combinations. For example, OpenVLA's success rate on LIBERO-Spatial under UADA attack changes from 0.2% to 52.0% when evaluation changes, and to 17.4% after re-optimization. Printed patches also disrupt a physical robot task. Finally, we examine how the same constraints affect adversarial training, finding substantial gains on VLA models. Together, these results show why physical patch risk should be evaluated under an explicit deployment model, with attack generation and evaluation considered separately.[1]

1 INTRODUCTION

Vision-Language-Action (VLA) models (Black et al., 2024; Ghosh et al., 2024; Kim et al., 2025; 2024; Brohan et al., 2023; Zitkovich et al., 2023; Driess et al., 2023; Team, 2025) are increasingly deployed in robotic manipulation tasks (Jang et al., 2021; Liu et al., 2023a) such as robotic grasping and object rearrangement. Because these models act in physical embodied settings, errors in visual perception can propagate to downstream manipulation behavior and lead to unsafe actions. Recent studies (Li et al., 2025; Wang et al., 2025; Xu et al., 2025) report that adversarial patches can reduce task success rate to near zero, leading to unexpected robotic actions and thus raising serious safety concerns for VLA models.

These studies (Wang et al., 2025; Xu et al., 2025) directly composite patches into observed RGB images. But a patch fixed in image coordinates and a patch fixed in the workspace follow different deployment models. Consider a printed texture attached to a tabletop: its location on the table stays fixed, but its image location, scale, and perspective change with the camera. A main camera and a wrist camera must also observe the same underlying texture. Pixel-space compositing does not enforce these relationships, even when the texture and image-space location are held fixed throughout a rollout. This raises a concrete question: how much do reported attack results change when evaluation enforces a single fixed planar patch? In VLA manipulation, camera geometry and robot behavior are coupled through the observation-action loop: an altered observation changes the action, which shapes subsequent viewpoints and patch appearances. Evaluating a fixed patch along the executed trajectory captures this interaction and connects the attack to its deployment.

We separate this question into two comparisons. First, we keep the generated texture and change its evaluation from pixel-space compositing to geometry-consistent projection. Second, we keep

[1]Code is available at `https://anonymous.4open.science/r/geometry_consistent_patch-1E9B`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

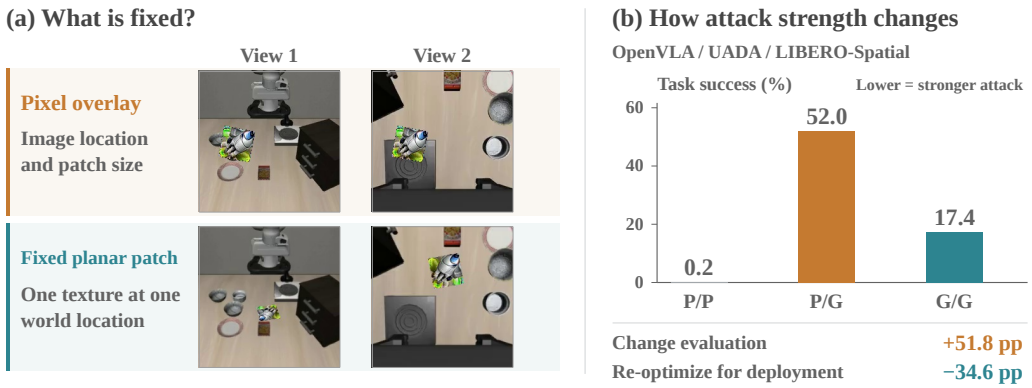

Figure 1: **The deployment changes the measured attack strength.** Left: schematic comparison of a patch fixed in image coordinates and a texture fixed on a physical plane. Right: task success rate on OpenVLA under UADA attack on LIBERO-Spatial. P and G denote pixel-space and geometry-consistent settings, in Generation/Evaluation order.

geometry-consistent evaluation and change how the texture is generated. This gives a 2×2 Generation/Evaluation design. We enforce camera projection and placement constraints while retaining the existing attack objectives. Patch generation remains an offline optimization procedure; evaluation runs the policy in closed loop.

Across three VLA models and four LIBERO suites, task success rate rises in every tested combination when we change the evaluation. Yet the models are still vulnerable: optimizing the patch under the same geometric constraints usually recovers attack strength. Figure 1 gives one example, where task success rate changes from 0.2% to 52.0%, then to 17.4%. This gap exposes a mismatch between pixel-space attack scores and the risk of deploying the same texture in reality. We also test printed patches on a physical robot and examine the implications for adversarial training. Both the original EDPA (Xu et al., 2025) patch and its geometry-consistent variant disrupt the action. Geometry-consistent training improves robustness in this setup and in most simulation settings.

Overall, our contributions are as follows:

- We identify and quantify a deployment mismatch in VLA patch evaluation. The same texture produces substantially different attack results under pixel-space compositing and fixed planar projection.
- We show that re-optimization strengthens attacks under the same geometry-consistent evaluation in 23 of 24 combinations, and demonstrate the benefit of joint optimization across camera views.
- We provide a geometry-consistent evaluation implementation and physical robot evidence, and study how matching the training threat model affects robustness under adaptive attacks in simulation.

2 Related Work

2.1 VLA Models for Embodied Manipulation

VLA models aim to bridge perception and control by aligning visual observations and language instructions with robot actions (Ma et al., 2024; Nair et al., 2021; Jiang et al., 2022). Built upon large vision language models (Liu et al., 2023b; Team, 2024; Radford et al., 2021), recent VLA models such as RT-2 (Zitkovich et al., 2023), Octo (Ghosh et al., 2024), OpenVLA (Kim et al., 2024), and π_0 (Black et al., 2024) demonstrate strong generalization across diverse tasks and environments (Cui et al., 2023; Huang et al., 2024). By aligning visual observations and language inputs with action

outputs, these models can execute tasks (Jang et al., 2021; Liu et al., 2023a; O'Neill et al., 2024) such as robotic grasping and object rearrangement.

Existing VLA approaches adopt different action decoding strategies. One line of work (Zitkovich et al., 2023; Kim et al., 2024) discretizes robot actions into tokens and performs autoregressive prediction over action sequences, following the next-token modeling strategy (Brown et al., 2020; Touvron et al., 2023) of large language models. Another line of work (Ghosh et al., 2024; Black et al., 2024; Kim et al., 2025) directly generates continuous control trajectories, either through regression-based action heads or diffusion-based formulations that model multimodal action distributions. Despite differences in action modeling, VLA architectures rely on high-dimensional visual observations as the primary input, making visual perception central to downstream action execution.

2.2 Adversarial Robustness of VLA Models

Adversarial attacks have been extensively studied in image classifiers (Madry et al., 2018; Goodfellow et al., 2015; Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2016) and object detection (Dong et al., 2022; Croce & Hein, 2020; Chen et al., 2021; Du et al., 2022). Early work (Szegedy et al., 2014) demonstrated that deep neural networks for image classifiers are vulnerable to imperceptible perturbations added across the entire image. Such global perturbations (Madry et al., 2018; Goodfellow et al., 2015; Carlini & Wagner, 2017) are typically constrained by l_p norms and optimized to induce misclassification with minimal change. Subsequently, adversarial patch (Thys et al., 2019; Brown et al., 2017; Song et al., 2018; Karmon et al., 2018; Liu et al., 2019) was proposed to restrict perturbations to localized regions of the image but do not constrain their magnitude which allowed the patch to be physically realizable. To improve patch robustness under viewpoint and environmental changes, Athalye et al. (2018) proposed EOT, which optimizes adversarial patches over a distribution of geometric and photometric transformations such as rotation, scaling, and lighting. While EOT reduces sensitivity to predefined perturbations, the question here is whether VLA patch evaluation enforces a common physical texture and placement across timesteps and camera views.

Recent works (Wang et al., 2025; Xu et al., 2025; Jiao et al., 2025; Zhou et al., 2025; Jones et al., 2025) extend adversarial attacks to VLA models. Unlike image classifiers (Krizhevsky et al., 2012; He et al., 2016; Dosovitskiy et al., 2021), VLA models operate in embodied or simulated environments. To adapt patch attacks to VLA models, existing approaches typically adopt a two-stage pipeline: patch generation on datasets, followed by patch evaluation in simulation environments. During evaluation, adversarial patches are applied via pixel-space compositing at fixed pixel positions, without enforcing that observations across timesteps and views arise from a single physical texture.

Xu et al. (2025) begins to explore adversarial training for VLA models by conducting adversarial training on the visual encoder. By incorporating adversarial examples during training, this approach improves the robustness of the visual encoder. However, both attacks and defenses are evaluated under pixel-space compositing, which may exaggerate their harm in reality, while existing adversarial training may strengthen VLAs in the wrong direction, leaving them vulnerable to real-world attacks.

3 Threat Model and Evaluation Design

3.1 From an Image Overlay to a Fixed Planar Patch

We consider a VLA model f_θ that maps multi-view RGB observations $\{\mathbf{x}^i\}_{i=1}^N$ and a language instruction $\mathbf{t}$ to an action $\mathbf{a}$:

$$\mathbf{a} = f_\theta(\{\mathbf{x}^i\}_{i=1}^N, \mathbf{t}). \tag{1}$$

Existing VLA patch attacks optimize a patch δ by directly compositing it onto each input view, implicitly assuming pixel-level control over every input frame:

$$\max_{\delta} \mathbb{E}_{(\{\mathbf{x}^i\},\mathbf{t})} \mathcal{L}_{\text{train}}\Big(f_\theta(\{\mathbf{x}^i \oplus \delta\}_{i=1}^N, \mathbf{t})\Big), \tag{2}$$

where $\oplus$ denotes pixel-space compositing and N is the number of camera views.

In physical embodied settings, however, the attacker controls a fixed planar texture, not the final pixel tensor. A fixed planar texture $\delta \in [0,1]^{H\times W\times 3}$ is affixed to a surface, and the adversarial

Table 1: **Generation and evaluation answer different questions.** P/P versus P/G measures the change in a generated patch's effect when the deployment model changes. P/G versus G/G measures the benefit of re-optimization under the same G evaluation.

	Evaluation	
Generation	Pixel-space (P)	Geometry-consistent (G)
Pixel-space (P)	P/P: digital baseline	P/G: deploy P-generated texture
Geometry-consistent (G)	G/P: cross-evaluation	G/G: generate for deployment

observation is generated by full camera projection with perspective and placement constraints.

$$\mathbf{x}_{\text{adv}}^{i} = \Pi(\mathbf{S}, \delta, \mathbf{c}^{i}), \tag{3}$$

where $\mathbf{S}$ denotes scene geometry, $\mathbf{c}^{i}$ denotes the parameters of camera i, and $\Pi(\cdot)$ enforces that the patch lies flat on an unoccupied surface region and induces multi-view consistency, because all views are determined by the same physical texture under camera projection. Geometry-consistent patch generation is then:

$$\max_{\delta} \mathbb{E}_{(\mathbf{S},\{\mathbf{c}^{i}\},\mathbf{t})} \mathcal{L}_{\text{train}}\Big(f_{\theta}(\{\Pi(\mathbf{S}, \delta, \mathbf{c}^{i})\}_{i=1}^{N}, \mathbf{t})\Big). \tag{4}$$

Patch evaluation follows the same geometry-consistent threat model, reporting task success rate under full camera projection with perspective and placement constraints.

3.2 Crossing Generation and Evaluation

We apply P or G independently at patch generation and evaluation, giving the four configurations in Table 1. P/P reproduces the pixel-space protocol. P/G evaluates a P-generated texture under the fixed planar-patch model. G/G optimizes and evaluates under that model, while G/P completes the crossed comparison.

For both generation settings, we optimize textures offline on a dataset using the original attack objective. In G, gradients pass through the patch projection. During evaluation, the policy runs in closed loop, so the current camera pose determines the patch appearance at each timestep. We report robot task success rate; a lower value means a more effective attack.

The P patch occupies 50×50 pixels in a 224×224 image. The G patch measures $0.12\,\text{m} \times 0.12\,\text{m}$. P/P versus P/G measures the combined deployment effect of projection, footprint, and visibility. P/G versus G/G holds the G deployment constraints fixed and measures the benefit of re-optimization.

3.3 Implementation of the Geometric Constraints

We use calibrated camera intrinsics and extrinsics to project a rectangular patch onto each view. Since the patch lies on the plane $Z = Z_0$, the projection reduces to a homography. To choose a valid location, we represent free tabletop regions with an occupancy grid and sample a position that accommodates the patch footprint. These operations supply the common rendering operator for generation, evaluation, and training. Full projection and placement details are given in Appendix A; the main study changes the observation construction while retaining the existing attack objectives.

4 How the Deployment Model Changes Attack Results

4.1 Experiment Setup

Models and attacks. We evaluate OpenVLA (Kim et al., 2024), OpenVLA-OFT (Kim et al., 2025), and π_0 (Black et al., 2024). OpenVLA uses a single camera view, while the other two models use a static main view and a wrist view. We test UADA, TMA, UPA (Wang et al., 2025) and EDPA (Xu et al., 2025). The first three attacks rely on discrete action tokens and are not applicable to OpenVLA-OFT and π_0. Across the four task suites, this gives 24 combinations.

Benchmark and Tasks. Experiments were conducted on the LIBERO benchmark (Liu et al., 2023a), which contains four task suites targeting distinct axes of generalization: LIBERO-Spatial,

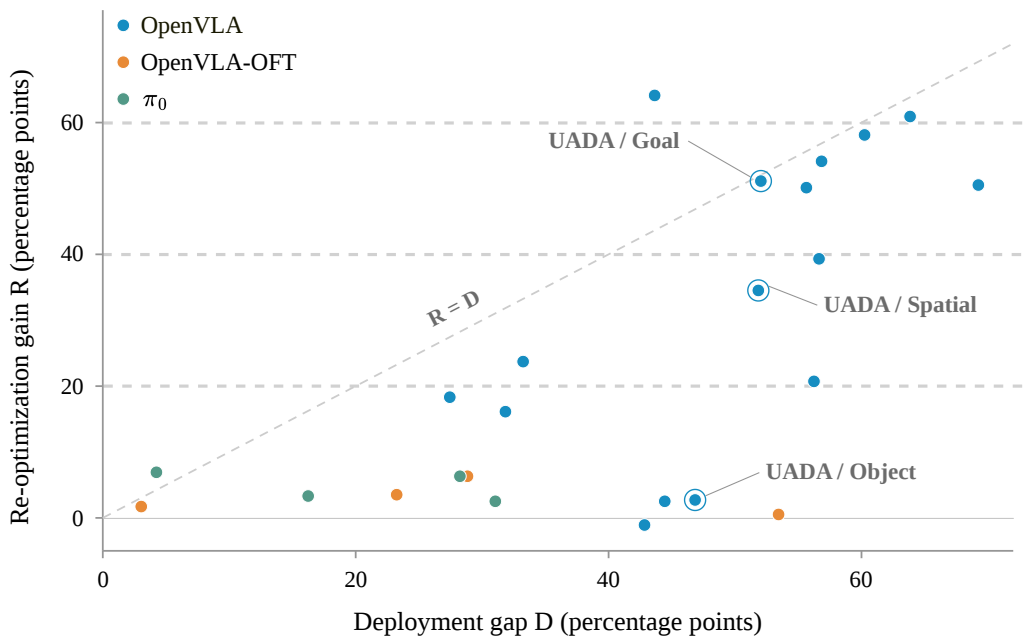

Figure 2: **Deployment weakens transferred patches, while re-optimization recovers attack strength.** Each point represents one of 24 combinations; colors identify models. All points have a positive deployment gap, and 23 have a positive re-optimization gain. The dashed line marks $R = D$, where G/G matches P/P task success; above it, G/G is more effective than P/P. Rings highlight OpenVLA under UADA on Spatial, Object, and Goal.

LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each suite comprises 10 manipulation tasks spanning spatial arrangement, object identity, goal specification, and long-horizon planning. Following the standard evaluation protocol, we executed 50 rollouts per task, resulting in 500 rollouts per suite and 2,000 rollouts in total across all four suites.

Generation and evaluation. We follow the patch-generation settings of Wang et al. (2025); Xu et al. (2025), including EOT (Athalye et al., 2018), and replace compositing with differentiable projection for G generation. Patch position is sampled once per rollout and then held fixed in the corresponding image or world coordinates. All four configurations use task success rate as the main metric. We additionally compare attacked trajectories with clean rollouts and evaluate printed patches on a physical robot. Compute resources are reported in Appendix B. Additional tests of patch rotation, size, and occlusion are reported in Appendix C.

4.2 Deployment Gap and Re-optimization Gain

We examine how deployment changes attack strength and how much re-optimization recovers (Figure 2). We define the deployment gap as $D = S_{P/G} - S_{P/P}$ and the re-optimization gain as $R = S_{P/G} - S_{G/G}$, where S is task success rate and both differences are in percentage points. D captures the full deployment change, including footprint and visibility; R measures the gain under fixed G evaluation. The full crossed results are reported in Appendix D (Tables A2 to A4).

Across all 24 combinations, switching from P/P to P/G raises task success by 3.0–69.2 points: the same textures become less effective under fixed planar deployment. Re-optimization strengthens the attack in 23 combinations. On OpenVLA-Spatial, UADA yields 0.2% task success under P/P, 52.0% under P/G, and 17.4% under G/G. On Goal, G/G reaches 2.0%, close to the 1.2% P/P result. The recovery varies across tasks, but the implication is consistent: weak transfer of a pixel-generated patch does not establish robustness against an attack optimized for deployment. Physical-patch risk

Table 2: **Task success rate (%) on OpenVLA-OFT and π_0 under multi-view physical patch constraints on LIBERO-Spatial.** Higher values indicate stronger model robustness. EDPA (per-view patches) optimizes independent patches δ_1 and δ_2 for each camera view, implicitly assuming pixel-level control over every input frame. The main-view and wrist-view patch variants each take a single patch generated by EDPA on one view and evaluate it under our geometry-consistent pipeline. Our geometry-consistent pipeline optimizes a single patch texture across both views via differentiable geometric projection.

Method	Patch Type	OpenVLA-OFT	π_0
EDPA (per-view patches)	δ_1, δ_2	2.8	74.8
EDPA (main-view patch)	δ_1 generated on view 1	31.6	86.4
EDPA (wrist-view patch)	δ_2 generated on view 2	31.6	85.6
Geometry-Consistent	single patch	25.2	72.0

should therefore be evaluated with both the deployment constraints and an attack optimized under those constraints.

4.3 Multi-View Consistency of Physical Patches

A single tabletop patch must produce both the main-view and wrist-view observations through camera projection. The original multi-view EDPA setting instead optimizes independent textures δ_1 and δ_2. In Table 2, we compare this per-view baseline with transferring either single-view texture to G evaluation and with jointly optimizing one texture through both projections. The per-view baseline uses pixel-space evaluation; the other three rows use geometry-consistent evaluation.

On OpenVLA-OFT, independent per-view patches give 2.8% task success, while transferring either single-view patch gives 31.6%. Joint geometry-consistent generation reduces success to 25.2%. On π_0, joint generation reaches 72.0%, compared with 85.6–86.4% for single-view transfer and 74.8% for the per-view baseline. Joint optimization produces a more effective shared patch than transferring a single-view texture on both models. On π_0, it also outperforms the independent per-view baseline.

4.4 Closed-Loop Trajectory Analysis

Task success rate captures the final outcome, but not how an attack changes the robot's motion. We compare each attacked rollout with clean execution from the same initial state and simulator seed, using 500 episodes across the 10 LIBERO-Spatial tasks. For the numerical metrics, we measure end-effector position differences at synchronous timesteps over the common valid prefix of each pair. We then average the mean and maximum deviations across episodes. Divergence onset is the first timestep with a deviation above $2\,\mathrm{cm}$, expressed as a fraction of the common valid prefix.

Figure 3(a) shows how the deviation changes over normalized rollout progress. Geometry-consistent EDPA produces larger average deviations through most of the execution, with a visible gap in the middle of the rollout. This pattern is consistent with the synchronous metrics: mean position deviation increases from $5.74\,\mathrm{cm}$ to $6.56\,\mathrm{cm}$, and the mean episode-level maximum increases from $11.84\,\mathrm{cm}$ to $13.41\,\mathrm{cm}$. Among the 497 episodes in which both attacks exceed $2\,\mathrm{cm}$, the mean divergence onset is 17.03% of the common valid prefix for original EDPA and 15.24% for geometry-consistent EDPA. Together, these results show larger trajectory deviations and earlier divergence under geometry-consistent patch generation.

The paired comparison in Figure 3(b) shows larger mean position deviations under geometry-consistent EDPA in 285 of 500 episodes (57%). The mean paired increase is $0.82\,\mathrm{cm}$, with a 95% paired bootstrap confidence interval of $[0.44, 1.19]\,\mathrm{cm}$; the median increase is $0.28\,\mathrm{cm}$. The paired measurements and progress curves show complementary effects: larger average trajectory deviations across episodes and a sustained gap during execution.

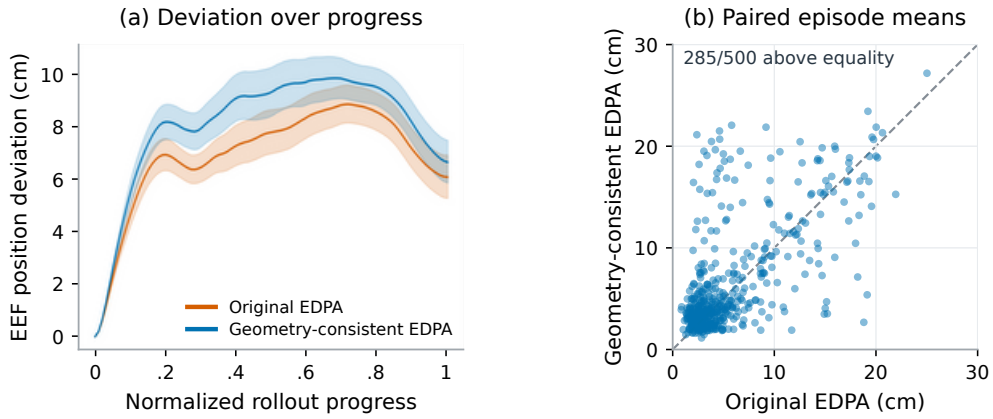

Figure 3: **Paired closed-loop trajectory analysis of π_0 on LIBERO-Spatial.** (a) Each clean and attacked trajectory is independently interpolated to 100 progress points before computing position differences. Lines show means over 500 paired episodes; shading shows pointwise 95% bootstrap confidence intervals. (b) Each point compares the two attacks' mean position deviations for the same episode, measured at synchronous timesteps over the common valid prefix with clean execution. Points above the dashed equality line have larger deviations under geometry-consistent EDPA.

Table 3: **Task success rate (%) under UADA on OpenVLA with comparable patch footprints.** We compare UADA re-optimized for a 30×30 pixel budget with the P/G baseline from Table A2. P/G projects the original pixel-generated texture as a $12 \times 12\,\mathrm{cm}$ patch. Lower success indicates a stronger attack.

Evaluation	Patch budget	Spatial	Object	Goal	Long
Pixel-space	30×30 pixels	22.8	28.8	33.0	18.6
P/G	$12 \times 12\,\mathrm{cm}$	52.0	47.0	53.2	27.6

Table 4: **Task success rate (%) of π_0 in physical robot attack and defense evaluation.** The original EDPA and Geometry-Consistent EDPA patches are printed and physically placed in the robot workspace. Lower values indicate stronger attack effectiveness for Vanilla, while higher values indicate stronger robustness for Geometry adversarial training.

Model	Clean	Original EDPA	Geometry-Consistent EDPA
Vanilla	90.0	23.3	16.7
Geometry AT	86.7	**76.7**	**80.0**

4.5 Effect of Patch Area and Visibility

We compare pixel-space and geometry-consistent evaluation at similar patch areas. On OpenVLA, the projected $12 \times 12, \mathrm{cm}$ patch occupies approximately 1.834% of the input image. We therefore use a 30×30-pixel patch, which occupies 1.79% of a 224×224 image. We re-optimize UADA for this pixel budget and evaluate the resulting patch in pixel space.

With the 30×30 patch, OpenVLA achieves 22.8%, 28.8%, 33.0%, and 18.6% task success on Spatial, Object, Goal, and Long (Table 3). These rates remain 29.2, 18.2, 20.2, and 9.0 percentage points below P/G. Thus, the pixel-space attack remains stronger on all four completed suites even with a comparable footprint. The deployment gap persists in this area-calibrated comparison.

4.6 Physical Robot Evaluation

We evaluate printed patches on an ALOHA AgileX platform with a fixed camera and a wrist camera. The robot performs a block-stacking task: picking up the red cube and placing it on the blue cube.

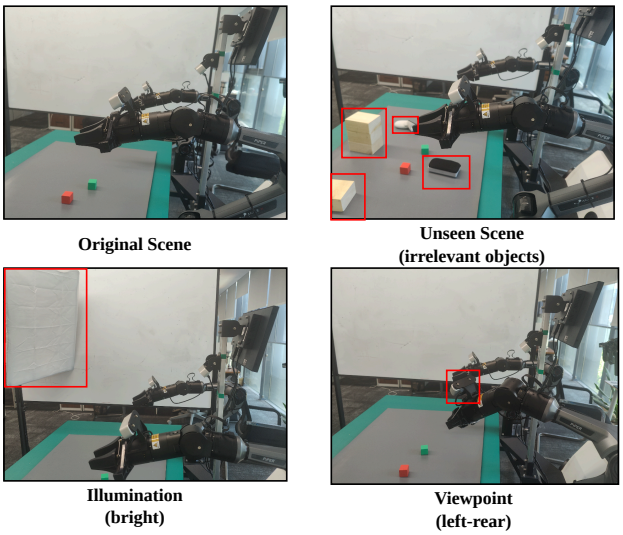

Figure 4: **Physical robot setup and physical variations.** The unseen scene introduces additional irrelevant objects, while the illumination setting changes the lighting condition. For the viewpoint variation, the camera mounted on the left arm is used as the fixed camera, since only the right arm is used for task execution. Red boxes highlight the corresponding changes.

Table 5: **Task success rate (%) of π_0 under different physical variations.** We vary illumination, patch placement, and fixed-camera viewpoint while keeping the remaining conditions unchanged. Lower values indicate stronger attack effectiveness.

	Illumination		Patch Placement			Camera Viewpoint	
Method	Normal	Bright	Left	Right	Front	Default	Left-rear
Original EDPA	23.3	16.7	20.0	23.3	30.0	23.3	23.3
Geometry-Consistent EDPA	**10.0**	**16.7**	**20.0**	**10.0**	**10.0**	**10.0**	**3.3**

Both EDPA variants are optimized offline using data from this setup, printed, and placed on the tabletop. Each condition is evaluated for 30 trials. Table 4 shows that the original EDPA patch reduces vanilla-model success from 90.0% to 23.3%, while geometry-consistent EDPA reduces it to 16.7%. These results demonstrate that a fixed printed patch can severely disrupt closed-loop manipulation.

We also vary illumination, patch placement, and camera viewpoint (Figure 4 and Table 5). Bright illumination adds a light source in front of the setup; the three placements position the patch to the left, right, or in front of the cubes. For the viewpoint change, the camera on the unused left arm supplies the fixed view. Geometry-consistent EDPA has an equal or lower observed success rate than original EDPA in every tested condition. Under the left-rear viewpoint, for example, the rates are 3.3% and 23.3%. The attacks remain effective across changes in lighting, placement, and camera viewpoint, demonstrating physical transfer beyond a single deployment condition.

4.7 Implications for Adversarial Training

We compare Pixel AT and Geometry AT under geometry-consistent evaluation. Both use a persistent adversarial patch and parameter-efficient training; the training procedure and hyperparameters are given in Appendix E. We examine the effect of LoRA rank in Appendix F. In simulation, all attacks are adaptive: a new patch is generated against each trained model. Table 6 shows that both strategies improve OpenVLA success across all 16 attack–suite combinations. Geometry AT is higher in 15 of them, with large gains on Spatial and Long. The exception is Goal under TMA, where Pixel AT reaches 32.4% and Geometry AT 15.8%. Geometry AT retains similar overall clean success to the vanilla model (71.1% versus 70.4%; Appendix G).

Table 6: **Task success rate (%) on OpenVLA under different adversarial training strategies.** They were evaluated against geometry-consistent adversarial patches across four attacks on four LIBERO task suites. **Vanilla**: no adversarial training. **Pixel AT**: adversarial training with patches generated via pixel-space compositing. **Geometry AT**: adversarial training with patches generated via our geometry-consistent pipeline.

LIBERO-Spatial				
Training	UADA	TMA	UPA	EDPA
Vanilla	17.4	5.6	23.6	18.2
Pixel AT	33.4	82.2	54.8	55.0
Geometry AT	**85.6**	**85.8**	**86.2**	**72.4**

LIBERO-Object				
Training	UADA	TMA	UPA	EDPA
Vanilla	44.2	47.8	50.0	36.4
Pixel AT	55.6	55.4	54.0	52.6
Geometry AT	**68.0**	**64.2**	**64.0**	**61.0**

LIBERO-Goal				
Training	UADA	TMA	UPA	EDPA
Vanilla	2.0	3.2	5.0	3.8
Pixel AT	68.2	**32.4**	69.2	20.0
Geometry AT	**78.6**	15.8	**76.4**	**30.0**

LIBERO-Long				
Training	UADA	TMA	UPA	EDPA
Vanilla	9.2	10.6	20.0	10.8
Pixel AT	13.6	16.0	30.8	17.0
Geometry AT	**46.4**	**52.0**	**46.6**	**41.6**

Table 7: **Task success rate (%) of π_0 in the original and unseen physical robot scenes.** The unseen scene contains additional irrelevant objects. Patches are randomly placed in both scenes. Higher values indicate stronger model robustness against the attack.

Method	Clean	Original Scene	Unseen Scene
Vanilla	90.0	16.7	13.3
Pixel AT	73.3	43.3	40.0
Geometry AT	86.7	**80.0**	**50.0**

On OpenVLA-OFT and π_0, Geometry AT improves on Pixel AT on Goal and Long, while Pixel AT leads on Spatial and Object. For example, Pixel AT reaches 95.6% on OpenVLA-OFT-Object versus 87.2% for Geometry AT. The full results are in Appendix H. Together with the OpenVLA results, these comparisons show that training geometry materially affects robustness under physical-patch deployment.

Physical experiments show gains from both training strategies. Geometry AT reaches 76.7% and 80.0% success under the two printed attacks, with 86.7% clean success (Table 4). Under random placement, Pixel AT and Geometry AT reach 43.3% and 80.0% in the original scene, compared with 16.7% for the vanilla model. In the unseen scene with additional objects, the rates are 40.0% and 50.0%, compared with 13.3% for vanilla (Table 7). The smaller gap in the unseen scene shows how changes in deployment conditions can affect the relative benefit of the training strategies.

5 Limitations and Conclusion

Our study models camera projection and planar placement and evaluates one physical task and platform with 30 trials per condition. The P/P-P/G comparison includes footprint and visibility changes alongside projection. Extending the analysis to additional tasks, platforms, and photometric effects, including shadows and reflections, is a natural next step.

The central finding is that the deployment model changes the measured strength of VLA patch attacks. Across all 24 combinations, pixel-space evaluation overstates the effectiveness of the same texture under fixed planar deployment. Re-optimization recovers attack strength in 23 combinations, and printed patches substantially disrupt physical task execution. Geometry-consistent adversarial training further improves robustness across a broad range of settings. These findings establish deployment-aware evaluation as a foundation for measuring and improving VLA robustness to physical patch attacks.

AI USE STATEMENT

In this work, we used generative AI tools for translating the manuscript. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, or interpreting results, and generating synthetic datasets, proposing or refining hypotheses, and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code and editing the manuscript to improve readability. We have reviewed all AI-assisted work. We independently reviewed and tested all code, and carefully checked AI-assisted translations and edited to ensure technical accuracy. 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 the source code for reproducing our experiments at `https://anonymous.4open.science/r/geometry_consistent_patch-1E9B`.

REFERENCES

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Int. Conf. Mach. Learn. (ICML)*, pp. 284–293, 2018.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint*, arXiv:2410.24164, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023.

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint*, arXiv:1712.09665, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2020.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Pin-Chun Chen, Bo-Han Kung, and Jun-Cheng Chen. Class-aware robust adversarial training for object detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 10420–10429, 2021.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *Int. Conf. Mach. Learn. (ICML)*, volume 119, pp. 2196–2205, 2020.

Yiming Cui, Cheng Han, and Dongfang Liu. CML-MOTS: collaborative multi-task learning for multi-object tracking and segmentation. *arXiv preprint*, arXiv:2311.00987, 2023.

Ziyi Dong, Pengxu Wei, and Liang Lin. Adversarially-aware robust object detector. In *Eur. Conf. Comput. Vis. (ECCV)*, pp. 297–313, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent. (ICLR)*, 2021.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *Int. Conf. Mach. Learn. (ICML)*, pp. 8469–8488, 2023.

Andrew Du, Bo Chen, Tat-Jun Chin, Yee Wei Law, Michele Sasdelli, Ramesh Rajasegaran, and Dillon Campbell. Physical adversarial attacks on an aerial imagery object detector. In *WACV*, pp. 3798–3808. IEEE, 2022.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Int. Conf. Learn. Represent. (ICLR)*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 770–778, 2016.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Int. Conf. Learn. Represent. (ICLR)*, 2022.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *Int. Conf. Mach. Learn. (ICML)*, pp. 20413–20451, 2024.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: zero-shot task generalization with robotic imitation learning. In *CoRL*, Proceedings of Machine Learning Research, pp. 991–1002. PMLR, 2021.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: general robot manipulation with multimodal prompts. *arXiv preprint*, arXiv:2210.03094, 2022.

Ruochen Jiao, Shaoyuan Xie, Justin Yue, Takami Sato, Lixu Wang, Yixuan Wang, Qi Alfred Chen, and Qi Zhu. Can we trust embodied agents? exploring backdoor attacks against embodied llm-based decision-making systems. In *Int. Conf. Learn. Represent. (ICLR)*, 2025.

Eliot Krzysztof Jones, Alexander Robey, Andy Zou, Zachary Ravichandran, George J. Pappas, Hamed Hassani, Matt Fredrikson, and J. Zico Kolter. Adversarial attacks on robotic vision language action models. *arXiv preprint*, arXiv:2506.03350, 2025.

Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *Int. Conf. Mach. Learn. (ICML)*, pp. 2512–2520, 2018.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint*, arXiv:2502.19645, 2025.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, pp. 1106–1114, 2012.

Jiayu Li, Yunhan Zhao, Xiang Zheng, Zonghuan Xu, Yige Li, Xingjun Ma, and Yu-Gang Jiang. Attackvla: Benchmarking adversarial and backdoor attacks on vision-language-action models. *arXiv preprint*, arXiv:2511.12149, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2023b.

Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Yiran Chen, and Hai Li. DPATCH: an adversarial patch attack on object detectors. In *SafeAI@AAAI*, CEUR Workshop Proceedings. CEUR-WS.org, 2019.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *arXiv preprint*, arXiv:2405.14093, 2024.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conf. Learn. Represent. (ICLR)*, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint*, arXiv:1610.08401, 2016.

Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *CoRL*, Proceedings of Machine Learning Research, pp. 1303–1315. PMLR, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew E. Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Paul Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Haoshu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey T. Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding,

Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Sünderhauf, Ning Liu, Norman Di Palo, Nur Muhammad (Mahi) Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R. Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante-Gomez, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham D. Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Liangwei Xu, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration. In *ICRA*, pp. 6892–6903. IEEE, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Int. Conf. Mach. Learn. (ICML)*, pp. 8748–8763, 2021.

Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *ECCV Workshops (5)*, Lecture Notes in Computer Science, pp. 429–448. Springer, 2020.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, pp. 3353–3364, 2019.

Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *WOOT @ USENIX Security Symposium*. USENIX Association, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Int. Conf. Learn. Represent. (ICLR)*, 2014.

Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *arXiv preprint*, arXiv:2503.20020, 2025.

Gemma Team. Gemma: Open models based on gemini research and technology. *arXiv preprint*, arXiv:2403.08295, 2024.

Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 49–55, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint*, arXiv:2302.13971, 2023.

Taowen Wang, Cheng Han, James Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *Int. Conf. Comput. Vis. (ICCV)*, 2025.

Haochuan Xu, Yun Sing Koh, Shuhuai Huang, Zirun Zhou, Di Wang, Jun Sakuma, and Jingfeng Zhang. Model-agnostic adversarial attack and defense for vision-language-action models. *arXiv preprint*, arXiv:2510.13237, 2025.

Xueyang Zhou, Guiyao Tie, Guowen Zhang, Hechang Wang, Pan Zhou, and Lichao Sun. Badvla: Towards backdoor attacks on vision-language-action models via objective-decoupled optimization. *arXiv preprint*, arXiv:2505.16640, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In *CoRL*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

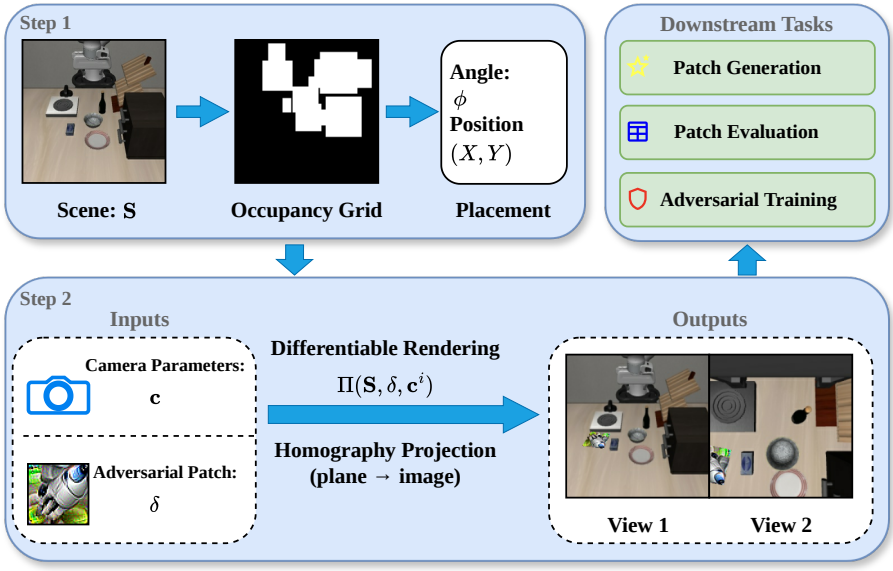

Figure A1: **Overview of the geometry-consistent adversarial patch pipeline.** Step 1: Given scene $\mathbf{S}$, an occupancy grid is constructed to enforce placement constraints, from which a valid placement is sampled, yielding a patch angle ϕ and position (X, Y) on the manipulation surface. Step 2: The adversarial patch δ is projected into each camera view via differentiable geometric projection $\Pi(\mathbf{S}, \delta, \mathbf{c}^i)$, producing geometry-consistent observations across multiple views. Across patch generation, patch evaluation, and adversarial training, we use the same geometry-consistent rendering operator.

A Geometry-Consistent Rendering and Placement

Here we give the implementation details summarized in Section 3.3.

A.1 Geometry-Consistent Patch Rendering

We implement the rendering operator $\Pi(\mathbf{S}, \delta, \mathbf{c}^i)$ under a geometry-consistent camera model, as illustrated in Figure A1. The adversarial patch is restricted to lie on a horizontal plane in the scene, corresponding to the manipulation surface in VLA tasks. We denote the height of this plane in the world coordinate system by Z_0. The patch δ is a planar rectangle with center (X_c, Y_c, Z_0), width w, height h, and yaw angle ϕ. Given scene geometry $\mathbf{S}$ and camera parameters $\mathbf{c}^i$, we project the physical patch into each camera view according to the camera projection model.

For a 3D point in space, we denote its world coordinates as $\mathbf{P}_w = [X, Y, Z]^\top$. Its homogeneous form is written as $\tilde{\mathbf{P}}_w = [X, Y, Z, 1]^\top$. For camera i, the projection onto the image plane follows:

$$\lambda \mathbf{p} = \mathbf{K}^i \begin{bmatrix} \mathbf{R}^i & \mathbf{t}^i \end{bmatrix} \tilde{\mathbf{P}}_w, \tag{A1}$$

where $\mathbf{K}^i$ denotes the intrinsic matrix and $\mathbf{R}^i, \mathbf{t}^i$ are the extrinsic parameters. Together, $\mathbf{K}^i$, $\mathbf{R}^i$, and $\mathbf{t}^i$ define the camera parameters $\mathbf{c}^i$. $\mathbf{p} = [u, v, 1]^\top$ denotes the homogeneous image coordinate and λ is a scale factor.

Since all patch points satisfy $Z = Z_0$, they are parameterized by their (X, Y) coordinates in the world coordinate system, with Z fixed to Z_0. The 3D to 2D projection therefore reduces to a planar homography:

$$\lambda \mathbf{p} = \mathbf{H}^i \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{A2}$$

where $\mathbf{H}^i$ is the homography from the plane to the image for camera i. Although the homography can be derived from the plane equation and camera parameters, we adopt the direct linear transfor-

mation (DLT) for simplicity. We project the four corner points of the patch onto the image using Equation (A1) to obtain point correspondences, from which $\mathbf{H}^i$ is estimated via DLT. The detailed derivation is provided in Appendix A. With $\mathbf{H}^i$, the patch is mapped from planar coordinates to the image plane via a homography transformation.

A.2 Placement Constraints for Planar Patch Attachment

To ensure geometric consistency, the patch must lie on the horizontal plane and satisfy placement constraints on unoccupied surface regions. Without this constraint, an attack may exploit invalid placements that are impossible to reproduce in simulation or physical embodied settings. We formulate placement as a 2D collision problem. The plane has width L_x and length L_y, centered at $\mathbf{q} = (X_0, Y_0, Z_0)^\top$ in world coordinates.

We represent the manipulation surface as a 2D map. Each grid cell records whether the corresponding physical region is available for patch placement. We divide the plane into grid cells of size r, producing an occupancy grid $\mathbf{M} \in \{0,1\}^{N_x \times N_y}$, where $N_x = \lceil \frac{L_x}{r} \rceil$ and $N_y = \lceil \frac{L_y}{r} \rceil$ denote the grid dimensions along the x and y axes. 1 denotes free regions and 0 denotes occupied regions.

Given patch width w, height h, and rotation angle ϕ, we define a binary convolution kernel $\mathbf{K}_\phi$ where a value of 1 indicates patch coverage. Candidate patch placements are determined via

$$\mathbf{C} = \mathbf{M} * \mathbf{K}_\phi, \tag{A3}$$

where $*$ denotes 2D discrete convolution. The convolution operation can be regarded as sliding the rotated patch over the occupancy grid. A placement at location $\mathbf{C}(i,j)$ is valid if the patch lies entirely within an unoccupied region, i.e., $\mathbf{C}(i,j) = \sum \mathbf{K}_\phi$. The valid placement set is therefore $\Omega = \{(i,j) \mid \mathbf{C}(i,j) = \sum \mathbf{K}_\phi\}$.

If $\Omega \neq \emptyset$, we randomly sample $(i^*, j^*) \in \Omega$ and convert it back to world coordinates:

$$X = X_0 - \frac{L_x}{2} + i^* \cdot r, \quad Y = Y_0 - \frac{L_y}{2} + j^* \cdot r, \quad Z = Z_0. \tag{A4}$$

This yields a valid patch center that satisfies the placement constraints.

A.3 Derivation of the Geometry-Consistent Rendering Operator

Recall the camera projection model in Section A.1:

$$\lambda \mathbf{p} = \mathbf{K}^i \begin{bmatrix} \mathbf{R}^i & \mathbf{t}^i \end{bmatrix} \tilde{\mathbf{P}}_w. \tag{A5}$$

For points on the horizontal plane $Z = Z_0$, we have $\tilde{\mathbf{P}}_w = [X, Y, Z_0, 1]^\top$. Substituting into the projection equation Equation (A5) gives

$$\lambda \mathbf{p} = \mathbf{K}^i \left(\mathbf{R}^i \begin{bmatrix} X \\ Y \\ Z_0 \end{bmatrix} + \mathbf{t}^i \right). \tag{A6}$$

We write the rotation matrix in column form: $\mathbf{R}^i = [\mathbf{r}_1^i \ \mathbf{r}_2^i \ \mathbf{r}_3^i]$. Then

$$\mathbf{R}^i \begin{bmatrix} X \\ Y \\ Z_0 \end{bmatrix} + \mathbf{t}^i = X\mathbf{r}_1^i + Y\mathbf{r}_2^i + Z_0\mathbf{r}_3^i + \mathbf{t}^i. \tag{A7}$$

Define the modified translation vector $\mathbf{t'}^i := Z_0\mathbf{r}_3^i + \mathbf{t}^i$. Then the projection Equation (A5) becomes

$$\lambda \mathbf{p} = \mathbf{K}^i \begin{bmatrix} \mathbf{r}_1^i & \mathbf{r}_2^i & \mathbf{t'}^i \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{A8}$$

We therefore define the plane-to-image homography $\mathbf{H}^i := \mathbf{K}^i \begin{bmatrix} \mathbf{r}_1^i & \mathbf{r}_2^i & \mathbf{t'}^i \end{bmatrix}$, which yields the planar projection form

$$\lambda \mathbf{p} = \mathbf{H}^i \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{A9}$$

B Experiment Compute Resources

Our experiments were conducted on NVIDIA A100, RTX 5090, and B200 GPUs. For OpenVLA, each geometry-consistent adversarial training run takes approximately 35 hours on either an A100 GPU or an RTX 5090 GPU. For OpenVLA-OFT, adversarial training requires more GPU memory than available on the RTX 5090 in our setup. Each OpenVLA-OFT adversarial training run takes approximately 40 hours on an A100 GPU and approximately 20 hours on a B200 GPU. For π_0, each adversarial training run takes approximately 14 hours on an A100 GPU.

For adversarial patch generation, generating an EDPA patch takes approximately 4 hours on either an A100 GPU or an RTX 5090 GPU. Generating a TMA, UPA, or UADA patch takes approximately 14 hours on either an A100 GPU or an RTX 5090 GPU. For simulation evaluation on LIBERO, evaluating one task takes approximately 14 hours for OpenVLA, 2 hours for OpenVLA-OFT, and 2 hours for π_0.

C Sensitivity to Patch Conditions

We compare the two EDPA patches on LIBERO using the vanilla π_0 model under the same geometry-consistent evaluation. The occlusion ratio is the fraction of patch area hidden from the camera.

Table A1: **Task success rate (%) under controlled variations in patch rotation, physical size, and occlusion.** Both patches are evaluated under the same geometry-consistent pipeline. Lower values indicate stronger attack effectiveness.

	Patch Rotation			Physical Patch Size			Patch Occlusion Ratio		
Method	$0°$	$30°$	$45°$	$10\,\mathrm{cm}$	$12\,\mathrm{cm}$	$15\,\mathrm{cm}$	0%	10%	20%
Original EDPA	85.4	76.8	74.4	86.6	83.6	74.2	79.0	80.8	81.8
Geometry-Consistent EDPA	**77.8**	**70.6**	**70.0**	**79.8**	**75.8**	**57.6**	**72.0**	**74.4**	**80.2**

Table A1 shows lower task success under geometry-consistent EDPA in all nine conditions. Larger patches strengthen both attacks, with the largest gap at $15\,\mathrm{cm}$: 74.2% versus 57.6%. Occlusion weakens both attacks and narrows their gap from 7.0 to 1.6 points. Re-optimization remains effective across these conditions, while its benefit depends on deployment.

D Crossed Generation-Evaluation Results

We report the full crossed results for OpenVLA, OpenVLA-OFT, and π_0 in Tables A2 to A4, respectively.

UADA, TMA, and UPA are formulated to attack discrete action tokens and rely on the bin-based action head of OpenVLA. OpenVLA-OFT and π_0 adopt continuous action representations, with a parallel action head and a flow-matching-based diffusion head respectively, and are therefore incompatible with these three attacks. Accordingly, results for these two models are reported under EDPA only, which is model-agnostic and applicable across action head architectures.

The results in Table A3 and Table A4 show that the gap between pixel-space compositing and geometry-consistent projection also holds for OpenVLA-OFT and π_0. On OpenVLA-OFT, replacing pixel-space evaluation with geometry-consistent evaluation substantially increases the task success rate, e.g., from 2.8% to 31.6% on LIBERO-Spatial and from 7.8% to 61.2% on LIBERO-Goal. This indicates that the fully pixel-space pipeline does not faithfully reflect physical patch deployment. Meanwhile, G/G consistently yields lower success rates than P/G across all suites, suggesting that geometry-consistent patch generation is more effective than directly transferring pixel-space patches under physically constrained evaluation.

For π_0, the same trend is observed. P/G improves over P/P on all suites, confirming that geometric consistency matters for both patch evaluation and patch generation. In addition, G/G consistently reduces success rates compared with P/G, showing that geometry-consistent generation remains beneficial even for the more robust π_0 model. Overall, these results support the main conclusion that

Table A2: **Task success rate (%) on OpenVLA under different generation-evaluation pipelines.** P and G denote pixel-space compositing and geometry-consistent projection, respectively; notation follows Generation/Evaluation order. Higher values indicate stronger model robustness against the attack. Colored delta values share the color of the row from which they are computed.

LIBERO-Spatial

Pipeline	UADA	TMA	UPA	EDPA
P/P	0.2	0.2	5.0	1.0
P/G	52.0 (↑ 51.8)	55.8 (↑ 55.6)	74.2 (↑ 69.2)	57.6 (↑ 56.6)
G/P	4.0	2.2	19.0	5.4
G/G	17.4 (↓ 34.6)	5.6 (↓ 50.2)	23.6 (↓ 50.6)	18.2 (↓ 39.4)

LIBERO-Object

Pipeline	UADA	TMA	UPA	EDPA
P/P	0.2	6.0	6.2	1.0
P/G	47.0 (↑ 46.8)	50.4 (↑ 44.4)	49.0 (↑ 42.8)	57.2 (↑ 56.2)
G/P	21.8	30.2	24.8	17.6
G/G	44.2 (↓ 2.8)	47.8 (↓ 2.6)	50.0 (↑ 1.0)	36.4 (↓ 20.8)

LIBERO-Goal

Pipeline	UADA	TMA	UPA	EDPA
P/P	1.2	1.2	25.6	1.2
P/G	53.2 (↑ 52.0)	61.4 (↑ 60.2)	69.2 (↑ 43.6)	58.0 (↑ 56.8)
G/P	5.8	7.0	3.4	13.8
G/G	2.0 (↓ 51.2)	3.2 (↓ 58.2)	5.0 (↓ 64.2)	3.8 (↓ 54.2)

LIBERO-Long

Pipeline	UADA	TMA	UPA	EDPA
P/P	0.2	1.2	4.4	8.0
P/G	27.6 (↑ 27.4)	34.4 (↑ 33.2)	36.2 (↑ 31.8)	71.8 (↑ 63.8)
G/P	3.8	2.8	4.4	4.8
G/G	9.2 (↓ 18.4)	10.6 (↓ 23.8)	20.0 (↓ 16.2)	10.8 (↓ 61.0)

Table A3: **Task success rate (%) on OpenVLA-OFT under different generation-evaluation pipelines.** P and G denote pixel-space compositing and geometry-consistent projection, respectively; notation follows Generation/Evaluation order. Higher values indicate stronger model robustness against the attack. Colored delta values share the color of the row from which they are computed.

Pipeline	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long
P/P	2.8	6.6	7.8	0.2
P/G	31.6 (↑ 28.8)	29.8 (↑ 23.2)	61.2 (↑ 53.4)	3.2 (↑ 3.0)
G/P	8.6	6.6	18.6	0.2
G/G	25.2 (↓ 6.4)	26.2 (↓ 3.6)	60.6 (↓ 0.6)	1.4 (↓ 1.8)

Table A4: **Task success rate (%) on π_0 under different generation-evaluation pipelines.** P and G denote pixel-space compositing and geometry-consistent projection, respectively; notation follows Generation/Evaluation order. Higher values indicate stronger model robustness against the attack. Colored delta values share the color of the row from which they are computed.

Pipeline	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long
P/P	74.8	80.2	64.8	39.6
P/G	79.0 (↑ 4.2)	96.4 (↑ 16.2)	93.0 (↑ 28.2)	70.6 (↑ 31.0)
G/P	83.0	84.0	72.2	55.0
G/G	72.0 (↓ 7.0)	93.0 (↓ 3.4)	86.6 (↓ 6.4)	68.0 (↓ 2.6)

geometry-consistent evaluation provides a more realistic estimate of physical patch attack effectiveness, while geometry-consistent generation is necessary for stronger physically plausible attacks.

E Adversarial Training Details

E.1 Patch-Based Adversarial Training

Adversarial training against patch attacks is formulated under the same geometry-consistent threat model by replacing pixel-space compositing with $\Pi(\cdot)$ in the standard min-max objective:

$$\min_{\theta} \mathbb{E}_{(\mathbf{S},\{\mathbf{c}^i\},\mathbf{t})} \max_{\delta} \mathcal{L}_{\text{train}}\Big(f_\theta(\{\Pi(\mathbf{S},\delta,\mathbf{c}^i)\}_{i=1}^N,\mathbf{t})\Big). \tag{A10}$$

The inner maximization finds the worst-case planar texture under camera geometry, while the outer minimization trains the model to be robust against this worst-case texture.

E.2 LIGHTWEIGHT ADVERSARIAL TRAINING

Building upon the formulation in Equation (A10), Adversarial training can be formulated as a min-max optimization problem (Rao et al., 2020). Solving the inner maximization via multi-step PGD (Madry et al., 2018) is computationally expensive for VLA models. This cost is particularly substantial in our setting because each iteration requires rendering the patch through the geometry-consistent projection operator. Instead, we adopt a lightweight approximation inspired by FreeAT (Shafahi et al., 2019), which maintains a global adversarial patch that is iteratively updated across mini-batches.

Instead of optimizing the patch from scratch for each mini-batch, we maintain a single global patch that persists across training iterations. At each iteration, we first render the current patch onto the input images using the geometry-consistent rendering operator and placement constraints in Sections A.1 and A.2. The patch is then updated using a single gradient ascent step to maximize the task loss. After updating the patch, we finally render it onto the inputs and update model parameters.

To further reduce computational overhead, we perform adversarial training using LoRA (Hu et al., 2022). Only LoRA adapter parameters are updated during training. In practice, fewer than 1% of model parameters are optimized. The detailed training procedure is presented in Appendix E.

For OpenVLA and OpenVLA-OFT, LoRA is applied to all linear layers with rank $r = 8$, using 100,000 training steps and a learning rate of 5×10^{-4}. For π_0, training uses 30,000 steps and a learning rate of 5×10^{-5}. Rank and compute details are given in Appendices B and F.

The pseudo-code of adversarial training is shown in Algorithm 1.

Algorithm 1 Lightweight Adversarial Training

Require: Training dataset $\mathcal{D}$, pretrained model f_θ, adversarial patch δ, patch learning rate η_δ, model learning rate η_m, rendering operator Π (Sections A.1 and A.2)

Initialize θ, δ
Convert model to PEFT form
for gradient step $= 1, \ldots, N_{\text{steps}}$ **do**
  Sample mini-batch $(\mathbf{x}, \mathbf{t}, \mathbf{S}, \mathbf{c}) \sim \mathcal{D}$
  Patch update:
    $\text{loss}_\delta = 1/\mathcal{L}_{\text{train}}\Big(f_\theta(\{\Pi(\mathbf{x}^i, S^i, \delta, c^i)\}_{i=1}^N, \mathbf{t})\Big)$.
    Backpropagate loss_δ and update δ
  Model update:
    $\text{loss}_\theta = \mathcal{L}_{\text{train}}\Big(f_\theta(\{\Pi(\mathbf{x}^i, S^i, \delta, c^i)\}_{i=1}^N, \mathbf{t})\Big)$.
    Backpropagate loss_θ and update θ
end for

F EFFECTIVENESS OF DIFFERENT LORA RANKS

The choice of LoRA rank controls the capacity of the adaptation introduced during adversarial training, and directly affects the trade-off between robustness and clean task performance. A rank that is too low may limit the model's ability to learn robust representations, while an unnecessarily high rank increases computational cost without meaningful gains. To identify a suitable operating point, we train OpenVLA on LIBERO-Spatial using LoRA ranks of 4, 8, 16, and 32, and evaluate each model under our geometry-consistent pipeline across four attacks.

Table A5 shows that the LoRA rank slightly affects the robustness achieved by geometry-consistent adversarial training. Among the evaluated ranks, rank 8 provides the strongest overall robustness, achieving 85.6%, 85.8%, 86.2%, and 72.4% task success rates against UADA, TMA, UPA, and EDPA, respectively. The results also show that the benefit of geometry-consistent adversarial training is not specific to a single LoRA rank. Although lower or higher ranks lead to some performance variation, several alternative ranks still outperform Pixel adversarial training. For example, rank 4 achieves strong robustness against UADA and UPA, while rank 16 and rank 32 also improve robustness over Pixel adversarial training on several attacks.

Table A5: **Task success rate (%) on OpenVLA with different LoRA ranks.** They were evaluated using our geometry-consistent pipeline on LIBERO-Spatial tasks.

Rank	UADA	TMA	UPA	EDPA
4	**86.2**	77.2	81.6	26.8
8	85.6	**85.8**	**86.2**	**72.4**
16	78.8	53.8	78.4	46.8
32	60.0	31.4	74.6	37.0

Table A6: **Clean task success rate (%) of the vanilla and Geometry adversarial training models on four LIBERO task suites.** Higher values indicate stronger clean task performance.

Method	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long	Average
Vanilla	83.4	66.2	**78.4**	**53.6**	70.4
Geometry AT	**86.4**	**67.6**	77.0	53.2	**71.1**

Table A7: **Task success rate (%) of OpenVLA-OFT under different adversarial training strategies.** Models are evaluated against geometry-consistent EDPA patches. **Vanilla**: no adversarial training. **Pixel AT**: adversarial training with pixel-composited patches. **Geometry AT**: adversarial training with geometry-consistent patches from our pipeline.

Training	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long
Vanilla	25.2	26.2	60.6	1.4
Pixel AT	**89.6**	**95.6**	89.0	0
Geometry AT	88.4	87.2	**93.0**	**7.4**

Table A8: **Task success rate (%) of π_0 under different adversarial training strategies.** Models are evaluated against geometry-consistent EDPA patches. **Vanilla**: no adversarial training. **Pixel AT**: adversarial training with pixel-composited patches. **Geometry AT**: adversarial training with geometry-consistent patches from our pipeline.

Training	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long
Vanilla	72.0	93.0	86.6	68.0
Pixel AT	**94.4**	**96.4**	91.8	75.2
Geometry AT	91.6	96.0	**92.0**	**76.6**

Overall, we select rank 8 because it provides consistently strong robustness across all four attacks while maintaining the lightweight nature of LoRA-based adversarial training. Larger ranks do not bring consistent additional gains, suggesting that increasing the adaptation capacity beyond this point is not necessary for improving robustness in our setting.

G Clean Performance of Geometry AT

To examine whether adversarial training improves robustness at the cost of clean task performance, we evaluate the vanilla and Geometry adversarial training models under clean observations across all four LIBERO task suites.

The results in Table A6 show that Geometry adversarial training does not reduce overall clean task performance. Compared with the vanilla model, Geometry adversarial training improves the average task success rate from 70.4% to 71.1%. Although the task success rates decrease by 1.4 and 0.4 percentage points on LIBERO-Goal and LIBERO-Long, respectively, the differences are small. These results demonstrate that Geometry adversarial training achieves a favorable trade-off between adversarial robustness and clean task performance.

H GEOMETRY-CONSISTENT ADVERSARIAL TRAINING ON OPENVLA-OFT AND π_0

Tables A7 and A8 report EDPA results after training. Geometry AT is higher than Pixel AT on Goal and Long for both models, while Pixel AT is higher on Spatial and Object. Most trained-model results improve over vanilla, but Pixel AT on OpenVLA-OFT-Long decreases from 1.4% to 0.0%. These results support task-dependent benefits, as discussed in Section 4.7.