arXiv:submit/8402527 [cs.RO] 20 Nov 2026

GENPA: IMAGE GENERATORS AS EFFICIENT ROBOT POLICIES

Zian Deng[1,2] **Tianyi Zhang**[3] **Zilong Zheng**[4] **Ziyu Liu**[5]
Tao Zhang[5] **Yao Lyu**[3,†] **Shengbo Eben Li**[3,6,†]

[1]College of Computer Science and Artificial Intelligence, Fudan University
[2]Beijing Zhongguancun Academy [3]School of Vehicle and Mobility, Tsinghua University
[4]Faculty of Mechanical Engineering, Delft University of Technology [5]SunRising AI Ltd.
[6]College of Artificial Intelligence, Tsinghua University
[†]Corresponding authors

Project page https://deng-zian.github.io/GenPA-website/

ABSTRACT

Pretrained visual generation models have shown great promise in robotic manipulation. However, a representative line of work still treats future visual prediction as the central link between visual generative pretraining and action learning, retaining it as a training objective even when future scenes are no longer generated at inference time. This paradigm organizes policy learning around the proxy task of predicting future visual states. We argue that the instruction-following, spatial understanding, and visual imagination capabilities acquired through image generation pretraining can support control without future visual prediction: image generation models themselves can be directly adapted into powerful action learners. Building on this perspective, we propose GenPA, which fine-tunes an image generation backbone with a lightweight 19M-parameter action regression adapter and reuses its native generative objective for perception co-training. The native image-editing interface accommodates different visual targets without task-specific perception heads, enabling comparisons of current-scene perception, future prediction, and their combinations within a common policy architecture. At deployment, GenPA requires only a single backbone forward pass and action regression, without visual generation or iterative action denoising. Without additional large-scale robot-data pretraining, GenPA demonstrates strong control performance and robustness to perturbations across simulation benchmarks (99.10% success on LIBERO and 83.79% on LIBERO-Plus) and real-world manipulation tasks. Under compiled inference, GenPA takes 20.9 ms per action chunk on a single H100 GPU, achieving 1.7–3.4$\times$ speedups over strong VLA and WAM baselines. Our latency analysis further highlights architectural advantages of visual generation backbones for efficient robot control. Ablations across visual tasks and their combinations identify current-depth supervision as a simple, effective training recipe. Together, these results show that image generators can be directly adapted into strong, efficient robot policies without requiring future visual prediction.

1 INTRODUCTION

Recent advances in image generation pretraining have yielded strong instruction-following and visual understanding capabilities. Beyond synthesizing realistic images, these models can identify objects referred to in language, follow spatial constraints, and visualize instructed changes to a scene (Xiao et al., 2025; Black Forest Labs et al., 2025; Zhao et al., 2026). As illustrated in Figure 1(a), given a robot manipulation instruction and a corresponding editing request, an image generation system can depict the scene after task completion or sketch end-effector motion paths on the current image. These behaviors suggest that the spatial understanding and visual imagination capabilities acquired through generative pretraining may provide valuable priors for robot action learning.

Pretrained image and video generation models have both been adapted for robot control. A representative line of work treats future visual prediction as the central link between generative pretraining

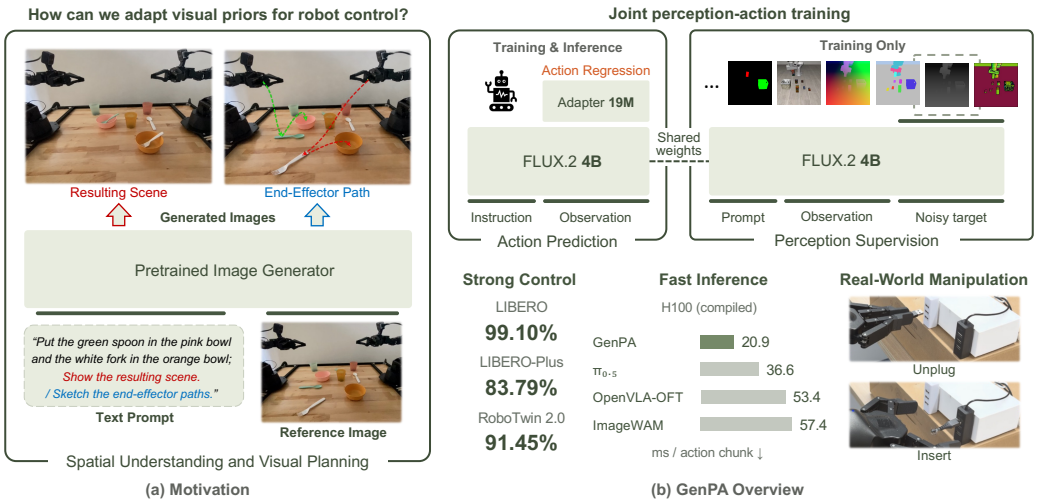


Figure 1: **Motivation and overview of GenPA. (a)** Given a scene image, a manipulation instruction, and an editing request, GPT Image 2 visualizes a goal state or sketches end-effector paths in image space, illustrating the potential of image generation models for spatial understanding and visual planning (generation details in Appendix F). **(b)** GenPA jointly trains action prediction and perception with shared backbone weights and uses only the action path at deployment. The action regression adapter has 19M parameters in the LIBERO configuration.

and action learning, using generated visual subgoals to guide goal-conditioned policies (Black et al., 2024) or decoding actions from predicted visual transitions through inverse dynamics (Li et al., 2026). Several methods jointly predict future observations and actions within a shared generative model (Shen et al., 2025; Kim et al., 2026; Ye et al., 2026b). Recent approaches further reduce or bypass future visual generation at deployment while retaining future prediction as a co-training objective (Yuan et al., 2026b; Zhang et al., 2026a; Ye et al., 2026a). In the broader context of predictive policy learning, future supervision has also been extended to depth, 3D pointmaps, and semantic features (Zhang et al., 2025; Yan et al., 2026). Despite differences in architecture and deployment, these methods continue to organize policy learning around the proxy task of predicting the future.

We argue that the control value of image generation pretraining need not be realized through the intermediate objective of future visual prediction: image generation models themselves can be directly adapted into powerful action learners. Prior work demonstrates the value of features from frozen image generators for control (Shin et al., 2026). We instead adapt the backbone itself through action learning, with a lightweight adapter mapping its features to actions. Meanwhile, direct action learning need not discard the generator's visual output capabilities. Research on generative perception shows that depth estimation, surface normal prediction, and segmentation can be learned as image generation tasks (Gabeur et al., 2026; Han et al., 2026). Together with evidence that auxiliary depth and spatial supervision benefit policy learning (Li et al., 2025b; Tu et al., 2026), these findings motivate reusing the generator's native interface for current-scene perception co-training. Representing different targets through this common interface also enables a systematic investigation of which visual supervision benefits control, including comparisons with future prediction and combinations of visual tasks.

Building on this perspective, we introduce **GenPA** (**Gen**erative **P**erception for **A**ction), an end-to-end policy that fine-tunes the pretrained image generation backbone FLUX.2 klein Base 4B (Black Forest Labs, 2026) with a 19M-parameter action regression adapter by default. During training, action and perception objectives jointly update the backbone, with perception supervision reusing its native image-editing interface and generative objective. Deployment requires only a single backbone forward pass and action regression, without visual generation or iterative action denoising. Figures 1(b) and 2 provide an overview and the detailed architecture, respectively.

Without additional large-scale robot-data pretraining, GenPA achieves strong control performance on LIBERO, LIBERO-Plus, and RoboTwin 2.0. On real-world contact-rich manipulation, GenPA

outperforms $\pi_{0.5}$, Fast-WAM, and ImageWAM while demonstrating strong robustness to visual perturbations. Under compiled inference on a single H100 GPU, GenPA takes 20.9 ms per action chunk, achieving 1.7–$3.4\times$ speedups over the evaluated VLA and WAM baselines. Our ablations further show that direct action regression achieves better closed-loop control performance with lower inference latency than a flow-based action head in our main evaluations. Compared with direct depth regression, native flow-based depth supervision improves control robustness at a higher training cost.

Our contributions are twofold. **(i)** We introduce GenPA, demonstrating that an image generation backbone can be directly adapted into a strong, efficient robot policy without a future visual prediction objective. Its native visual co-training interface supports diverse supervision targets without task-specific perception heads, enabling systematic comparisons of visual tasks and their combinations within a common policy architecture. **(ii)** Our controlled studies show that strong closed-loop control can be achieved with direct action regression, without iterative action denoising, and identify current-depth supervision as a simple, effective training recipe. Our latency analysis further reveals an architectural advantage of visual generation backbones for robot control: lightweight observation encoding and independent language encoding align computation with the different update frequencies of observations and instructions.

2 GENPA

At robot time step t, given current RGB observations $\mathbf{o}_t$, a language instruction ℓ, and proprioception $\mathbf{s}_t$, GenPA predicts an action chunk $\mathbf{A}_t = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1})$ of length H. As shown in Figure 2, GenPA predicts actions from the image generation backbone's final reference-token features and reuses its native image-editing interface for perception supervision. Action and perception training use separate forward passes with shared backbone parameters.

2.1 PRELIMINARIES: IMAGE-CONDITIONED GENERATION

We instantiate GenPA with FLUX.2 klein Base 4B, a pretrained rectified-flow transformer supporting image generation and editing (Black Forest Labs, 2026). Its editing interface processes three groups of tokens: text conditions, clean reference-image latents, and noised target-image latents. The reference image supplies visual context; the target represents the image to be generated. Let E_{vae} denote the VAE encoder. For a target image $\mathbf{y}$ with latent $\mathbf{z} = E_{\text{vae}}(\mathbf{y})$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, the linear flow path is (Liu et al., 2023b)

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z} + \tau\boldsymbol{\epsilon}, \qquad \tau \in [0,1]. \tag{1}$$

Here τ is the flow time: $\tau = 0$ denotes clean data, and $\tau = 1$ denotes noise. The generator predicts the velocity $\boldsymbol{\epsilon} - \mathbf{z}$ along this path; image synthesis requires iterative integration followed by VAE decoding.

2.2 DIRECT ACTION ADAPTATION

Adapting the backbone. We encode the current camera observations as reference latents $\mathbf{z}_t^R = E_{\text{vae}}(\mathbf{o}_t)$ and the manipulation instruction as text embeddings $E_{\text{text}}(\ell)$, where E_{text} is the text encoder. A learned Action Prefix $\mathbf{P}^A$ is prepended to the text embeddings to provide action-specific conditioning. The action forward pass contains only text and reference-image tokens, with the flow-time condition fixed at $\tau_A = 0$. GenPA uses bidirectional attention among valid text tokens, including the prefix, and reference tokens, allowing the reference features to incorporate the manipulation instruction. Let θ denote the trainable generator parameters, and F_θ^R the backbone computation that returns the final reference-token features. The action path is

$$\begin{aligned} \mathbf{h}_t^R &= F_\theta^R\big([\mathbf{P}^A; E_{\text{text}}(\ell)], \mathbf{z}_t^R; \tau_A = 0\big), \\ \hat{\mathbf{A}}_t &= g_\psi(\mathbf{h}_t^R, \mathbf{s}_t). \end{aligned} \tag{2}$$

Lightweight action readout. The action adapter g_ψ, with trainable parameters ψ, is a four-layer Transformer decoder that uses one learned query per action step to read the final reference features through cross-attention. The queries are conditioned on proprioception, and a linear head predicts

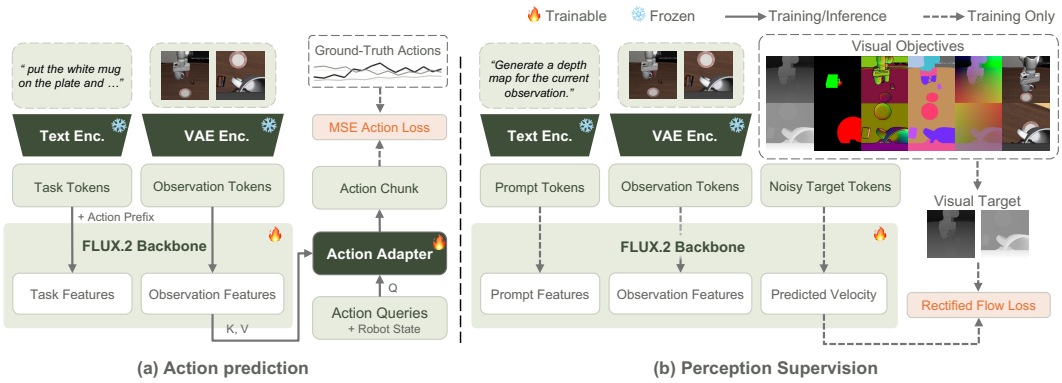


Figure 2: **Architecture of GenPA. (a)** The action path extracts instruction-conditioned observation features from the image generation backbone. A lightweight regression adapter reads these features and predicts an action chunk conditioned on proprioception. **(b)** The perception path reuses the generator's native image-editing interface to learn image-form perceptual targets with a rectified-flow objective. The two paths use separate forward passes with shared backbone weights. Only the action path is used at inference.

the action chunk in parallel. The default adapter contains 19M parameters, counted for the LIBERO configuration. Over a demonstration dataset $\mathcal{D}$, we optimize a mean-squared action loss

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\mathcal{D}}\left[\left\|\hat{\mathbf{A}}_t - \mathbf{A}_t\right\|^2_{\text{valid}}\right], \tag{3}$$

where the error is averaged over valid action elements in each demonstration sample.

2.3 Native Perception Supervision

For perception co-training, we reuse FLUX's image-conditioned flow formulation with a modified attention pattern. For each perception task k, we pair the current observation with a task prompt ℓ_k^V and an image-form perceptual target $\mathbf{y}_t^{(k)}$, such as a depth map, surface normals, or segmentation. These targets are encoded by the same VAE and represented as target-image tokens, without introducing task-specific perceptual heads.

We encode each target as $\mathbf{z}^{(k)} = E_{\text{vae}}(\mathbf{y}_t^{(k)})$ and construct $\mathbf{z}_\tau^{(k)} = (1-\tau)\mathbf{z}^{(k)} + \tau\boldsymbol{\epsilon}$ following Equation 1. The generator predicts the target velocity conditioned on the reference observation and perception-task prompt. In this forward pass, text and reference tokens attend to each other but cannot attend to the noised target; target tokens can attend to all three groups. Text and target tokens receive the sampled time condition τ, while the reference tokens receive a zero time condition. Let v_θ denote the generator's velocity prediction for target tokens. The perception objective is

$$\mathcal{L}_{\text{vis}}^{(k)} = \mathbb{E}_{\mathcal{D}_k,\tau\sim p(\tau),\boldsymbol{\epsilon}}\left[w(\tau)\left\|v_\theta\left(\mathbf{z}_\tau^{(k)}; \mathbf{z}_t^R, E_{\text{text}}(\ell_k^V), \tau\right) - (\boldsymbol{\epsilon} - \mathbf{z}^{(k)})\right\|^2_{\text{vis}}\right]. \tag{4}$$

Here $\mathcal{D}_k$ contains observation–target pairs for task k, $p(\tau)$ and $w(\tau)$ denote the noise-level sampling distribution and loss weight, and $\|\cdot\|^2_{\text{vis}}$ denotes the mean squared velocity-prediction error. Prompts, noise-level sampling, and loss weighting are specified in Appendix B.3.

2.4 Joint Training and Inference

By default, each action forward pass is paired with one perception forward pass on the same observations. For each sample, a perception task k is drawn from $p_{\mathcal{K}}$, the sampling distribution over the enabled task set $\mathcal{K}$. Both losses contribute to the same parameter update through the joint objective

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{vis}}\,\mathbb{E}_{k\sim p_{\mathcal{K}}}\left[\mathcal{L}_{\text{vis}}^{(k)}\right], \tag{5}$$

where $\lambda_{\text{vis}} \geq 0$ weights perception supervision. Both objectives update the shared backbone; action supervision also trains the prefix and adapter. The text encoder and VAE remain frozen.

At inference, only the action path in Equation 2 is executed. GenPA encodes the current observation and predicts an action chunk with one backbone forward pass and the regression adapter, without visual generation or iterative action denoising. The text encoder's instruction embeddings are computed once and reused while the instruction remains unchanged. VAE encoding and the backbone forward pass are repeated for each new observation. Architecture and training details are provided in Appendix A; target construction and sampling are detailed in Appendices B.1 and B.4.

3 EXPERIMENTS

We evaluate whether a pretrained image generator can be directly adapted into a strong, efficient robot policy without a future visual prediction objective. Simulation and real-world experiments assess control performance and robustness, while latency profiling and visual-supervision ablations examine the sources of inference cost and the effects of perception supervision.

3.1 EXPERIMENTAL SETUP

We train separate multitask policies on LIBERO and RoboTwin 2.0 without additional large-scale robot-data pretraining. Unless otherwise stated, GenPA uses current-depth supervision during training. Training and evaluation details are provided in Appendices A and C.

LIBERO. We train jointly on the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023a), comprising 40 tasks. We replay the 2,000 released demonstrations to construct visual supervision labels, retaining 1,808 successfully replayed demonstrations for training (Appendix B.2). The LIBERO policy configuration uses 224×224 views with training-time image augmentation, whereas the visual-target ablation configuration retains native 256×256 views without augmentation. Our policies in the external benchmark comparisons and visual-target ablations use training seed 42. Internal comparisons under the LIBERO policy configuration report mean $\pm$ sample standard deviation across three training seeds. LIBERO results marked *independent* use the same statistics across three evaluation seeds with new initial-state sets for fixed policies trained with seed 42. We evaluate 50 trials per task and report suite success rates and their arithmetic mean. The GenPA score in Table 1 is from checkpoint selection; selection and independent-evaluation protocols are specified in Appendix C.1.

LIBERO-Plus. We evaluate LIBERO-Plus using the same checkpoint reported on standard LIBERO, without fine-tuning or separate checkpoint selection. We run one trial on each of the 10,030 released instances across seven perturbation categories (Fei et al., 2026). Unless otherwise specified, we use *Clean-Instruction*, which removes extraneous perturbation metadata (Appendix C.2). Overall and category success rates pool outcomes over all instances and within each category, respectively.

RoboTwin 2.0. We train one policy jointly across 50 bimanual tasks using 2,500 clean and 25,000 randomized demonstrations with depth annotations (Chen et al., 2025; Yan et al., 2026). We use a 99%/1% episode-level training/validation split. We evaluate 100 trials per task under each of the Clean and Randomized conditions and report mean task success for each condition and their arithmetic mean. For RoboTwin, we use a wider action adapter for coordinated bimanual manipulation.

Baselines. Our comparisons cover representative VLA and WAM policies (Kim et al., 2025; Physical Intelligence et al., 2025; Wang et al., 2026b; Luo et al., 2026; Wu et al., 2026b; Yuan et al., 2026a; Chen et al., 2026; Kim et al., 2026; Yuan et al., 2026b; AMAP CV Lab, 2026; Zhang et al., 2026a; Bi et al., 2025). Tables indicate additional robot-data pretraining. LIBERO and RoboTwin baseline scores come from published reports. For LIBERO-Plus, we re-evaluate six released baselines under Clean-Instruction using their native inference configurations.

3.2 SIMULATION PERFORMANCE AND ROBUSTNESS

LIBERO. GenPA achieves 99.10% mean success on LIBERO (Table 1), the highest average among the listed methods without additional robot-data pretraining. It comes within 0.30 percentage points of ABot-M0.5, which additionally undergoes large-scale pretraining on diverse robot

Table 1: Performance on LIBERO and model characteristics. Robot PT indicates additional robot-data pretraining. Bold and underline mark the highest and second-highest scores in each column, respectively, including ties.

Method	Robot PT	Parameters		Action decoding	LIBERO success (%)				
		Backbone	Head		Spatial	Object	Goal	Long	Avg.
VISION-LANGUAGE BACKBONES									
OpenVLA-OFT[j]	✓	7B	151M	Regr.	97.7	98.0	96.1	95.3	96.80
$\pi_{0.5}$	✓	3B	300M	Flow	98.8	98.2	98.0	92.4	96.85
VLA-Adapter-Pro	✗	1.25B	218M	Regr.	<u>99.6</u>	99.6	98.2	96.4	98.50
SimVLA	✗	0.5B	∼300M	Flow	<u>99.6</u>	<u>99.8</u>	98.6	96.4	98.60
VLANeXt	✗	2B	612M	Flow	99.0	99.2	96.6	94.8	97.40
Qwen-RobotManip	✓	4B	—	Flow	—	—	—	—	<u>99.10</u>
LaWAM[†]	✓	1.52B	306M	Flow	99.4	99.6	98.4	97.0	98.60
VIDEO-GENERATION BACKBONES									
Cosmos Policy	✗	2B	Shared	Diffusion	98.1	**100.0**	98.2	<u>97.6</u>	98.50
Fast-WAM	✗	5B	1B	Flow	98.2	**100.0**	97.0	95.2	97.60
ABot-M0.5	✓	5B	—	Flow	**100.0**	<u>99.8</u>	**99.4**	**98.4**	**99.40**
IMAGE-GENERATION BACKBONES									
ImageWAM-4B	✗	4B	642M	Flow	97.2	99.2	<u>98.8</u>	**98.4**	98.40
GenPA	✗	4B	19M	Regr.	99.4	<u>99.8</u>	<u>98.8</u>	**98.4**	<u>99.10</u>

j: joint four-suite policy. Regr.: regression. Shared: no separate action expert. —: unavailable. †: LaWAM additionally uses a latent world decoder and a DINO encoder. Generative backbone sizes exclude separate text encoders and VAEs. Sources and parameter counts: Appendix C.1.

datasets. Relative to ImageWAM-4B, which uses the same image generation backbone family, GenPA achieves 99.10% versus 98.40% mean success while using a 19M-parameter action adapter rather than a 642M-parameter action expert. GenPA achieves these results with current-depth supervision rather than a future visual prediction objective. Appendix C.3 provides complete checkpoint scans and independent LIBERO evaluations of fixed policies on new initial states.

Table 2: LIBERO-Plus success under Clean-Instruction evaluation (%). All policies are evaluated by us. Bold and underline mark the highest and second-highest scores.

Method	Robot PT	Camera	Robot Init.	Lang.	Light	Backgr.	Noise	Layout	Overall
OpenVLA-OFT	✓	64.73	28.32	87.12	95.36	<u>89.59</u>	76.83	72.20	71.75
$\pi_{0.5}$	✓	72.23	**79.61**	86.21	96.67	**96.47**	89.76	<u>86.75</u>	**85.90**
VLANeXt	✗	**89.49**	62.77	79.18	95.10	89.22	**95.57**	77.90	83.60
Cosmos Policy	✗	80.24	64.97	<u>88.29</u>	**99.12**	88.85	95.07	**87.15**	<u>85.60</u>
Fast-WAM	✗	46.40	<u>72.39</u>	67.86	97.11	66.91	69.52	78.30	70.22
ImageWAM	✗	79.36	50.19	**91.28**	97.55	88.48	<u>95.19</u>	78.56	82.13
GenPA	✗	<u>81.80</u>	67.94	84.97	<u>97.99</u>	82.43	92.88	81.57	83.79

Robot PT follows Table 1. Released checkpoints and native settings: Appendix C.2.

LIBERO-Plus. GenPA achieves perturbation robustness competitive with the strongest evaluated policies, without additional robot-data pretraining or a future visual prediction objective. It reaches 83.79% success without fine-tuning (Table 2), exceeding the released ImageWAM policy by 1.66 percentage points and remaining within 2.11 percentage points of both $\pi_{0.5}$ and Cosmos Policy. GenPA ranks second under camera and lighting perturbations, while robot initial-state perturbations remain more challenging. Together with the latency results in Section 3.4, these findings support direct adaptation of image generators as an effective approach to robust and efficient robot control.

RoboTwin 2.0. GenPA achieves strong performance across 50 bimanual tasks, reaching 92.08% and 90.82% success under Clean and Randomized conditions, respectively, and averaging 91.45% (Table 3). Its average success is close to Fast-WAM (91.83%) and 1.93 percentage points below ImageWAM, the strongest baseline in this comparison. These results support the applicability of direct image-generator adaptation to coordinated bimanual manipulation.

Table 3: RoboTwin 2.0 success rates (%).

Method	Robot PT	Clean	Rand.	Avg.
$\pi_{0.5}$	✓	82.74	76.76	79.75
Qwen-RobotManip	✓	**93.4**	92.5	93.0
Motus	✓	88.66	87.02	87.84
Fast-WAM	✗	91.88	91.78	91.83
ImageWAM	✗	93.20	**93.56**	**93.38**
GenPA	✗	92.08	90.82	91.45

Sources and settings: Appendix C.4.

3.3 Real-world Manipulation

We evaluate contact-rich USB transfer: one arm unplugs a connector and inserts it into a fixed target port. We compare GenPA with $\pi_{0.5}$, Fast-WAM, and ImageWAM, all fine-tuned on the same 600 demonstrations. GenPA uses current-depth co-training. In all three conditions, initial arm pose and source port vary. *Standard* uses the nominal scene; *Background* adds visual distractors; and *Lighting* replaces room lighting with a phone LED. Each policy receives 50 trials per condition (150 total). Success requires task completion and gripper release within 50 seconds without dropping the connector. We report per-condition success and their arithmetic mean; setup and execution examples appear in Appendix E.

Table 4: USB task success rates (%).

Method	Standard	Bg.	Light	Avg.
$\pi_{0.5}$	74.0	76.0	24.0	58.0
Fast-WAM	68.0	64.0	18.0	50.0
ImageWAM	**92.0**	80.0	38.0	70.0
GenPA	**92.0**	**90.0**	**52.0**	**78.0**

Bg./Light: background/lighting perturbations. Avg.: mean across the three conditions.

GenPA achieves the highest mean success (78.0%), matching ImageWAM under Standard and exceeding it by 10 and 14 percentage points under Background and Lighting, respectively (Table 4). These results support direct image-generator adaptation for contact-rich real-world manipulation under visual perturbations, although lighting remains challenging.

3.4 Action Readout and Efficiency

Action readout. We compare GenPA's regression adapter with a comparable flow-based adapter using the LIBERO policy configuration (Table 5; architectural details in Appendix A.2).

Table 5: Action readout: success and latency.

Head	LIBERO (%)	Plus (%)	Latency (ms)↓	
			Uncompiled	Compiled
Flow	98.27 ± 0.12	81.49 ± 1.18	53.89	29.22
Regression	98.87 ± 0.25	82.33 ± 1.27	**27.86**	**20.94**

Success rates: mean $\pm$ SD across three training runs. Latency: H100, cached instructions. Success rates use uncompiled inference; results after compilation are reported in Appendix D.7.

Unless otherwise stated, GenPA latency excludes Qwen text encoding, since instruction embeddings can be reused while the instruction remains unchanged. In this comparison, regression achieves higher mean success rates on both LIBERO and LIBERO-Plus while reducing H100 latency under both uncompiled and compiled inference. Full comparisons and uncertainty estimates are provided in Appendix D.1.

Latency across policy architectures. Figure 3 decomposes latency into visual encoding (*Vision*), backbone computation (*Backbone*), action decoding (*Action*), and remaining overhead (*Other*); Appendix D provides timing protocols, component measurements, and additional latency results.

Like GenPA, Fast-WAM and ImageWAM reuse independently encoded instructions, so their timings also exclude text encoding. The three policies have similar VAE-based visual encoding costs, but iterative action decoding dominates the uncompiled latency of Fast-WAM and ImageWAM and remains their largest component after compilation. GenPA takes 27.9 ms without compilation and 20.9 ms with compilation. The smaller action expert of $\pi_{0.5}$ benefits substantially from compilation, bringing total latency to 36.6 ms, below OpenVLA-OFT's 53.4 ms despite retaining iterative decoding. OpenVLA-OFT uses regression, but incurs higher visual-encoding and backbone costs. GenPA thus combines strong manipulation performance with 20.9 ms action-chunk inference while using a 4B-parameter backbone. Deployment memory is reported in Appendix D.6.

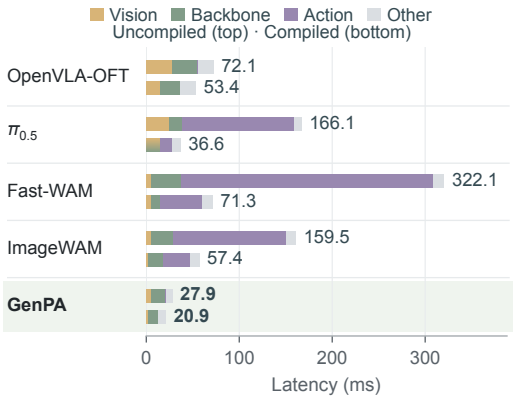


Figure 3: **H100 action-chunk latency.** Totals are timed separately; Vision and Backbone are jointly profiled for compiled $\pi_{0.5}$.

Why visual generation backbones suit efficient control. A more revealing difference lies in the computation required for each new observation. Visual encoding is substantially more expensive in the two evaluated VLM-based policies than VAE encoding in policies built on visual generation backbones. This reflects different divisions of computation: VLMs first extract high-level features with a vision encoder and then perform fusion and reasoning in a multimodal backbone; latent visual generators use a lightweight VAE to compress observations into reconstructable latents, perform conditional visual modeling in the visual generation backbone, and encode language independently (Rombach et al., 2022). Comparing backbone sizes alone therefore overlooks the different visual-encoding costs these architectures incur at each observation update. The latter organization matches the inputs of robotic manipulation: observations change continuously, whereas instructions typically remain fixed within a task. Language encoding can thus be amortized across policy calls, while each new observation passes through a lightweight VAE and the visual generation backbone. GenPA combines this organization with a lightweight regression adapter, producing actions through a single backbone forward pass and direct action readout. These results suggest that efficient policy design should not only simplify action generation but also align module boundaries with input update frequencies; visual generation pretraining provides a favorable architectural basis for this organization.

3.5 Native Depth Supervision versus Direct Depth Regression

To assess the benefit of the native perception interface, we compare native flow-based depth supervision (Native flow) with direct depth regression (DPT regression) using a DPT-style head (Ranftl et al., 2021) under the same backbone, depth labels, and action-update budget. For DPT regression, the depth head and action adapter read the same final reference-token features and jointly train the backbone in a single forward pass without target-image tokens. The two approaches perform similarly on LIBERO, while native depth supervision achieves higher LIBERO-Plus success (Table 6). Training takes 1.294 s per update for native depth supervision and 0.517 s for DPT regression on eight H100 GPUs. Implementation details and full results are provided in Appendices A.6 and B.6; training resource costs are summarized in Appendix A.5.

Table 6: Native depth supervision vs. direct depth regression (%).

Supervision	LIBERO	Plus
Native flow	98.87 ± 0.25	82.33 ± 1.27
DPT regression	98.85 ± 0.10	80.61 ± 0.35

Mean $\pm$ SD across three training runs.

3.6 Visual Supervision and Design Choices

We examine whether current-scene perception can provide effective supervision without future prediction, and how target choice and combination affect robustness. To limit preprocessing-related

confounds when comparing supervision targets, Table 7 uses the visual-target ablation configuration. All configurations in this table share the same model architecture, training data, and maximum action-update budget. Target encodings and sampling details are given in Appendices B.1 and B.4.

Table 7: Effect of visual supervision on closed-loop success (%). LIBERO and Plus give absolute rates; perturbations give None rates and changes from None (percentage points).

Target	LIBERO (independent)	Plus	Camera	Robot Init.	Lang.	Light	Bg.	Noise	Layout
None	97.02 ± 0.15	70.87	66.42	59.10	79.38	93.08	43.40	73.52	78.89
D	97.87 ± 0.65	76.69	+7.6	+7.7	+8.0	+2.9	+1.8	+8.2	+2.5
N	96.52 ± 0.53	71.70	−1.6	+6.6	+2.4	−4.7	+8.8	−2.9	−1.6
S	97.17 ± 0.39	76.08	+1.2	+5.6	+3.8	+2.9	+15.0	+10.5	−0.3
P	97.52 ± 0.60	72.19	−4.7	+0.4	+5.5	+2.6	+13.5	−2.9	−0.7
G	96.62 ± 0.16	71.80	−7.3	+0.0	+6.8	+1.1	+3.9	+2.3	+0.9
F	96.98 ± 0.20	77.10	+8.1	+4.6	+1.7	+1.4	+17.9	+12.4	−0.6
FD	98.02 ± 0.20	74.46	−4.4	+7.5	+5.1	−3.0	+19.5	+1.7	+2.2
D+S	97.23 ± 0.53	75.09	+6.8	+6.4	+0.6	+2.4	+9.9	+2.8	+1.9
D+P	97.78 ± 0.28	75.53	+10.9	+3.6	+5.1	+1.6	+3.0	+6.5	+0.4
D+F	97.37 ± 0.47	77.42	+6.1	+9.2	+4.9	+0.3	+12.3	+11.7	+1.2
D+N+S+P†	97.77 ± 0.33	75.45	+9.4	+0.8	+0.8	+1.4	+4.3	+12.4	+1.6

Each row uses one fixed checkpoint for both benchmarks; LIBERO reports mean $\pm$ SD across three evaluation seeds. D: current depth; N: normals; S: segmentation; P: pointmaps; G: task-conditioned grounding; F: future RGB; FD: future depth. †Two perception passes per action pass, with summed losses. All other configurations use one, except None, which omits perception supervision.

Current perception provides effective control supervision. Current depth (D) raises LIBERO-Plus success from 70.87% to 76.69%. Future depth (FD), using the same depth encoding for a target 16 retained action steps into the future, reaches a similar independent LIBERO score (98.02% versus 97.87%) but lower LIBERO-Plus success (74.46%). Future RGB (F) remains effective at 77.10%, despite lower independent LIBERO success (96.98%). Standard-task success alone is therefore insufficient for choosing an auxiliary objective. These results suggest that future prediction is not a prerequisite for improving closed-loop robustness through visual supervision. Under the LIBERO policy configuration, mean LIBERO-Plus success increases from $79.49 \pm 0.85\%$ without visual supervision to $82.33 \pm 1.27\%$ with current depth and $81.80 \pm 2.05\%$ with future RGB (Appendix B.5).

More targets do not consistently improve control. Although D and S each improve robustness, D+S scores below either alone. D+P improves camera robustness over D but lowers overall success, showing that category-specific gains need not improve robustness overall. The four-target configuration also scores below D despite using more perception computation. D+F achieves the highest overall score, but exceeds F by only 0.32 percentage points. These results suggest that informative visual targets are not automatically complementary for control: their joint benefit must be established through closed-loop evaluation. Current-depth supervision remains a simple, effective default.

Why is depth supervision effective? Depth improves success over None in all seven perturbation categories, whereas other geometric targets do not yield equally broad gains. We hypothesize that depth benefits from combining control-relevant geometry with a simple supervision format. It provides metric distance constraints on visible surfaces, encouraging the backbone to learn spatial structure relevant to reaching, grasping, and placement. This information is expressed as a dense scalar field aligned with the current observation, whose definition does not depend on task roles or execution stages. In contrast, robot-frame pointmaps additionally require learning a mapping from the current view to the robot frame; normals primarily constrain surface orientation rather than metric distance; and segmentation emphasizes categories and boundaries. These differences suggest that the value of auxiliary supervision depends not only on its information content, but also on what it asks the backbone to learn and how these constraints support control.

4 Related Work

Visual generative pretraining for robot control. Pretrained image and video generators provide visual subgoals for goal-conditioned policies (Black et al., 2024) or predicted transitions for inverse

dynamics (Li et al., 2026). Other methods jointly generate future observations and actions (Shen et al., 2025; Kim et al., 2026; Ye et al., 2026b), or retain future prediction for co-training while enabling action-only inference (Yuan et al., 2026b; Ye et al., 2026a). The closely related ImageWAM combines image-editing pretraining, future-image co-training, and a flow-based action expert, supporting inference without image generation while retaining iterative action decoding (Zhang et al., 2026a). Beyond future prediction, generators can represent actions through joint-localization targets, pixel-motion fields, or multiview action images (Shridhar et al., 2024; Nguyen et al., 2026; Zhen et al., 2026). ORCA instead extracts task-adaptive features from a frozen diffusion model through learned conditioning (Shin et al., 2026). Whereas Fast-WAM and ImageWAM retain future-prediction co-training, GenPA fine-tunes the image generation backbone for direct action prediction through a lightweight regression adapter, coupled with current-scene perception co-training.

Perceptual supervision for policy adaptation. Auxiliary perception objectives have been studied across policy architectures. They range from future visual prediction (Wu et al., 2024; Tian et al., 2025; Zhang et al., 2025) to current-scene depth supervision (Li et al., 2025b; Yuan et al., 2025) and localization, segmentation, or spatial-relation tasks (Tu et al., 2026; Li et al., 2025a). Geometric pretraining and supervision further support policy adaptation (Song et al., 2026; Peng et al., 2026), while representation analyses link state decodability to control performance (Dong et al., 2026). LingBot-VLA 2.0 distills depth and video representations at both current and future timesteps (Wu et al., 2026a). Generative perception research demonstrates that visual generators can learn dense geometry (Ke et al., 2024; He et al., 2025) and diverse instruction-conditioned perception tasks (Gabeur et al., 2026; Han et al., 2026; Wang et al., 2026a). GenPA uses the generator's native image-editing interface for perception co-training without task-specific perception heads. Perceptual objectives shape the shared backbone during training; predicted annotations are not inputs to the deployed policy. Our ablations compare current-scene and future targets, as well as target combinations, within a common policy architecture to assess their utility for closed-loop control. We further compare native depth supervision with a direct depth-regression head to evaluate their control performance and training cost.

Action prediction and inference efficiency. Action chunking is established in visuomotor imitation (Zhao et al., 2023), with actions produced through diffusion or flow matching (Chi et al., 2023; Black et al., 2025), autoregressive token decoding (Pertsch et al., 2025), or parallel regression (Kim et al., 2025). Lightweight action adapters (Wang et al., 2026b), fewer generative decoding steps (Guo et al., 2026), and decoupling action-module depth from the visual backbone (Ma et al., 2026) offer different routes to efficient inference. GenPA combines lightweight regression with an adapted image generation backbone and compares regression and flow-based action readout under a common training framework to examine whether strong closed-loop control requires iterative action denoising. Beyond action decoding, our component-level analysis examines visual encoding and backbone computation, relating architectural differences to the recurring costs of changing observations and the reusable computation associated with fixed instructions.

5 Conclusion

We presented GenPA, which directly adapts a pretrained image generation model into a robot policy through lightweight action regression and native perception co-training, without requiring future visual prediction as an intermediate training objective. At deployment, GenPA requires only a single backbone forward pass and action regression. GenPA combines strong control performance and perturbation robustness in simulation and real-world manipulation with low measured inference latency. Ablations of visual supervision show that depth supervision on current observations offers a simple, effective training recipe, while adding targets does not always improve control. Our latency analysis further suggests that efficient policy design should not only simplify action decoding but also match computation to input update frequencies. Visual generation models provide a favorable architectural basis for this organization: a lightweight VAE encodes changing observations, while independently encoded language features can be reused across observations.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with writing, translation, and language polishing; literature retrieval and discovery; and research execution, including experimental design, implementation and debugging, data processing, and result interpretation. GPT Image 2 was used to produce the illustrative image edits in Figure 1(a); the source image and complete prompts are documented in Appendix F. The authors reviewed and revised the AI-assisted material and take full responsibility for the final manuscript, methodology, and reported results.

REPRODUCIBILITY STATEMENT

Appendix A provides the model architecture, input and output interfaces, preprocessing, and training configurations. Appendix B describes visual-target construction and the visual-supervision ablation protocol. Appendix C documents simulation evaluation, checkpoint selection, baseline configurations, and LIBERO-Plus instruction processing. Appendix D provides timing protocols, detailed latency measurements, and closed-loop evaluation after compilation. The real-world setup and evaluation protocol are described in Appendix E. For code, model checkpoints, and data, see the GenPA project page.

REFERENCES

AMAP CV Lab. ABot-M0.5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026. URL https://arxiv.org/abs/2607.00678v2. Version 2.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025. URL https://arxiv.org/abs/2512.13030.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ea50bf458f6070548b11babbe0bf89b-Abstract-Conference.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Black Forest Labs. FLUX.2 [klein] Base 4B. Model card, 2026. URL https://huggingface.co/black-forest-labs/FLUX.2-klein-base-4B.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025. URL https://arxiv.org/abs/2506.15742.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026. URL https://arxiv.org/abs/2606.15768.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Jiahua Dong, Yunze Man, Pavel Tokmakov, and Yu-Xiong Wang. Capturing visual environment structure correlates with control performance. *arXiv preprint arXiv:2602.04880*, 2026. URL https://arxiv.org/abs/2602.04880.

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*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.html.

Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T. Barron, Kyle Genova, Nithish Kannen, Sherry Ben, Yandong Li, Mandy Guo, Suhas Yogin, Yiming Gu, Huizhong Chen, Oliver Wang, Saining Xie, Howard Zhou, Kaiming He, Thomas Funkhouser, Jean-Baptiste Alayrac, and Radu Soricut. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026. URL https://arxiv.org/abs/2604.20329.

Yanzhao Guo, Wenkai Chen, and Jianwei Zhang. ReactVLA: Fast and lightweight reactive robot manipulation via improved mean flow action generation. *arXiv preprint arXiv:2606.14255*, 2026. URL https://arxiv.org/abs/2606.14255.

Xiaoyang Han, Jianhua Li, Kewang Deng, Zukai Chen, Xuanke Shi, Sihan Wang, Boxuan Li, Linyan Wang, Siyi Xie, Xin You, Jinsheng Quan, Zhongang Cai, Haiwen Diao, Ziwei Liu, Lei Yang, Dahua Lin, and Quan Wang. Vision as unified multimodal generation. *arXiv preprint arXiv:2607.06560*, 2026. URL https://arxiv.org/abs/2607.06560.

Jing He, Haodong Li, Wei Yin, Yixun Liang, Leheng Li, Kaiqiang Zhou, Hongbo Zhang, Bingbing Liu, and Ying-Cong Chen. Lotus: Diffusion-based visual foundation model for high-quality dense prediction. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/dea9b4b6f55ae611c54065d6fc750755-Abstract-Conference.html.

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9492–9502, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Ke_Repurposing_Diffusion-Based_Image_Generators_for_Monocular_Depth_Estimation_CVPR_2024_paper.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/748becc400a57c0e31cfe6a2e7951467-Abstract-Conference.html.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Xiang Li, Cristina Mata, Jongwoo Park, Kumara Kahatapitiya, Yoo Sung Jang, Jinghuan Shang, Kanchana Ranasinghe, Ryan Burgert, Mu Cai, Yong Jae Lee, and Michael S. Ryoo. LLaRA: Supercharging robot learning data for vision-language policy. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/909f526db5127f8bd8158af32d9e313a-Abstract-Conference.html.

Yixuan Li, Yuhui Chen, Mingcai Zhou, Haoran Li, Zhengtao Zhang, and Dongbin Zhao. QDepth-VLA: Quantized depth prediction as auxiliary supervision for vision-language-action models. *arXiv preprint arXiv:2510.14836*, 2025b. URL https://arxiv.org/abs/2510.14836.

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*, volume 36, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

Yuankai Luo, Woping Chen, Tong Liang, Baiqiao Wang, and Zhenguo Li. SimVLA: A simple VLA baseline for robotic manipulation. *arXiv preprint arXiv:2602.18224*, 2026. URL https://arxiv.org/abs/2602.18224v1.

Yao Lyu, Xiangteng Zhang, Shengbo Eben Li, Jingliang Duan, Letian Tao, Qing Xu, Lei He, and Keqiang Li. Conformal symplectic optimization for stable reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36:11049–11063, 2025. doi: 10.1109/TNNLS.2024.3511670. URL https://arxiv.org/abs/2412.02291.

Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-WAM: Do world action models need deep action modules? *arXiv preprint arXiv:2608.02365*, 2026. URL https://arxiv.org/abs/2608.02365.

E-Ro Nguyen, Yichi Zhang, Kanchana Ranasinghe, Xiang Li, and Michael S. Ryoo. Pixel motion diffusion is what we need for robot control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2509.22652.

Yue Peng, Yongzhe Zhao, Artur Habuda, Khuyen Pham, Yanheng Zhu, Tran Nguyen Le, Fares Abu-Dakka, and Li Guo. G^3VLA: Geometric inductive bias for vision-language-action models. *arXiv preprint arXiv:2606.24472*, 2026. URL https://arxiv.org/abs/2606.24472.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

Physical Intelligence. openpi: LIBERO benchmark and released $\pi_{0.5}$ checkpoint. Official repository documentation, 2026. URL https://github.com/Physical-Intelligence/openpi/blob/main/examples/libero/README.md. Accessed September 7, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin

LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2103.13413.

Zhixin Ren, Yao Lyu, Congrong Li, Liping Zhang, and Shengbo Eben Li. Momentum as residual-driven multiplier correction for deep learning optimization. *arXiv preprint arXiv:2608.12925*, 2026. URL https://arxiv.org/abs/2608.12925.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2112.10752.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/89a3b655a8b68ae1c76b768152c9c19d-Abstract-Conference.html.

Heeseong Shin, Byeongho Heo, Dongyoon Han, Seungryong Kim, and Taekyung Kim. Exploring conditions for diffusion models in robotic control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2510.15510.

Mohit Shridhar, Yat Long Lo, and Stephen James. Generative image as action models. *arXiv preprint arXiv:2407.07875*, 2024. URL https://arxiv.org/abs/2407.07875.

Zijian Song, Qichang Li, Jiawei Zhou, Zhenlong Yuan, Tianshui Chen, Liang Lin, and Guangrun Wang. Robotic manipulation is vision-to-geometry mapping: Vision-geometry backbones over language and video models. In *ACM International Conference on Multimedia*, 2026. URL https://arxiv.org/abs/2604.12908. Accepted; arXiv:2604.12908.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e5b5c402bb7bd5e60bede6961d6fe39e-Abstract-Conference.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Ruisen Tu, Arth Shukla, Sohyun Yoo, Xuanlin Li, Junxi Li, Jianwen Xie, Hao Su, and Zhuowen Tu. SG-VLA: Learning spatially-grounded vision-language-action models for mobile manipulation. *arXiv preprint arXiv:2603.22760*, 2026. URL https://arxiv.org/abs/2603.22760.

Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, Andrei Zanfir, and Cristian Sminchisescu. Video generation models are general-purpose vision learners. In *European Conference on Computer Vision*, 2026a. URL https://arxiv.org/abs/2607.09024. Accepted; arXiv:2607.09024.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp.

18638–18646, 2026b. doi: 10.1609/aaai.v40i22.38931. URL https://ojs.aaai.org/index.php/AAAI/article/view/38931. Pro results are reported in the extended version, arXiv:2509.09372v2.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/2c37c5bcef24b9541550261dcd63261b-Abstract-Conference.html.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, Yibin Liu, Shuai Yang, Tianxiang Zhou, Kejia Zhang, Lei Zhou, Cheng Su, Nan Xue, Bin Tan, Han Zhang, Youchao Zhang, Fei Liao, Xing Zhu, Yujun Shen, and Kecheng Zheng. From foundation to application: Improving VLA models in practice. *arXiv preprint arXiv:2607.06403*, 2026a. URL https://arxiv.org/abs/2607.06403.

Xiao-Ming Wu, Bin Fan, Kang Liao, Jian-Jian Jiang, Runze Yang, Yihang Luo, Zhonghua Wu, Wei-Shi Zheng, and Chen Change Loy. VLANeXt: Recipes for building strong VLA models. In *International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2602.18532v3. Version 3.

Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. OmniGen: Unified image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Xiao_OmniGen_Unified_Image_Generation_CVPR_2025_paper.pdf.

Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026. URL https://arxiv.org/abs/2608.10860.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b. URL https://arxiv.org/abs/2602.15922.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-RobotManip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a. URL https://arxiv.org/abs/2606.17846v2.

Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. DepthVLA: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025. URL https://arxiv.org/abs/2510.13375.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b. URL https://arxiv.org/abs/2603.16666.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and

Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/22d4f952efa13970f0b1ffb22170d416-Abstract-Conference.html.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. ImageWAM: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026a. URL https://arxiv.org/abs/2606.19531.

Zhanguang Zhang, Zhiyuan Li, Behnam Rahmati, Rui Heng Yang, Yintao Ma, Amir Rasouli, Sajjad Pakdamansavoji, Yangzheng Wu, Lingfeng Zhang, Tongtong Cao, Feng Wen, Xinyu Wang, Xingyue Quan, and Yingxue Zhang. Do world action models generalize better than VLAs? a robustness study. *arXiv preprint arXiv:2603.22078*, 2026b. URL https://arxiv.org/abs/2603.22078.

Bing Zhao, Chenfei Wu, Deqing Li, Hao Meng, Jiahao Li, Jie Zhang, Jingren Zhou, Junyang Lin, et al. Qwen-Image-2.0 technical report. *arXiv preprint arXiv:2605.10730*, 2026. URL https://arxiv.org/abs/2605.10730.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.html.

Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Seyed Kamyar Seyed Ghasemipour, Chelsea Finn, and Ayzaan Wahid. ALOHA Unleashed: A simple recipe for robot dexterity. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1910–1924, 2025. URL https://proceedings.mlr.press/v270/zhao25b.html.

Haoyu Zhen, Zixian Gao, Qiao Sun, Yilin Zhao, Yuncong Yang, Yilun Du, Pengsheng Guo, Tsun-Hsuan Wang, Yi-Ling Qiao, and Chuang Gan. Action images: End-to-end policy learning via multiview video generation. *arXiv preprint arXiv:2604.06168*, 2026. URL https://arxiv.org/abs/2604.06168.

This appendix provides implementation and training details (Appendix A), visual supervision and ablation details (Appendix B), simulation evaluation protocols and validation (Appendix C), action-readout comparisons and measurements of inference latency and memory (Appendix D), real-world experimental details (Appendix E), and generation details for Figure 1 (Appendix F).

A Implementation and Training Details

This appendix specifies GenPA's model architecture, tensor interfaces, image preprocessing, and training configuration. Perception-target construction and auxiliary-objective details are provided in Appendix B; simulation protocols and checkpoint selection in Appendix C; and platform-specific real-world implementation in Appendix E.

A.1 Model Architecture and Trainable Components

GenPA is initialized from FLUX.2 klein Base 4B (Black Forest Labs, 2026). Table 8 summarizes the architecture and training status of each component. Following the native FLUX.2 conditioning pipeline, we concatenate the frozen Qwen3-4B encoder's hidden states from layers 9, 18, and 27 at each text position to produce 7,680-dimensional conditioning features.

The frozen VAE encodes observations and perception targets into 32-channel latent feature maps. Packing each 2×2 latent neighborhood produces 128-dimensional image tokens. The VAE decoder is not used in either the action forward pass or the perception forward pass.

All DiT parameters are fine-tuned. The action forward pass contains no target-image tokens and uses the final reference-token features for action prediction. In the perception forward pass, the native output layer maps the final target-token features to latent velocities and receives gradients only from perception supervision. The Action Prefix and action adapter are randomly initialized.

With instruction embeddings cached and Qwen unloaded, the LIBERO policy contains approximately 3.98B parameters, including the frozen VAE; keeping Qwen resident increases this to 8.00B. The 19M figure refers only to the action adapter.

Table 8: GenPA components and their training status.

Component	Architecture	Role	Status
Qwen3-4B	36 layers, width 2,560; 32 attention heads, 8 key–value heads	Text conditioning for both forward passes	Frozen
FLUX.2 VAE	4 stages, widths 128/256/512/512; 2 residual blocks per stage; $8\times$ spatial compression	Encode observations and perception targets	Frozen
Visual generation DiT	5 dual-stream + 20 single-stream blocks; width 3,072; 24 heads of dimension 128; MLP ratio 3	Shared action–perception backbone	Trainable
Native output layer	Non-affine LayerNorm, timestep modulation, and bias-free $3{,}072 \to 128$ projection inside the DiT	Perception-flow prediction	Trainable
Action Prefix	16 learned 7,680-dimensional tokens	Action-specific backbone conditioning	Trainable
Action adapter	Four-layer Transformer decoder; see Appendix A.2	Action-chunk prediction	Trainable

The action and perception forward passes use the token groups, attention patterns, and time conditioning defined in Sections 2.2 and 2.3, respectively.

A.2 Action Adapters

Regression action adapter. The default adapter layer-normalizes the final reference-token states and projects them from 3,072 to 512 dimensions. We add two-dimensional sinusoidal position features derived from the image-token coordinates and apply a second layer normalization to form the

visual memory. Proprioception is encoded by LayerNorm–Linear–GELU–Linear, with both linear outputs of width 512, and added to each of 16 learned action queries in the simulation configurations. Four pre-normalized decoder blocks apply bidirectional self-attention among the queries, cross-attention to the visual memory, and a feed-forward network. Each attention layer has eight heads; the feed-forward width is 2,048, with GELU activation and zero dropout. A final LayerNorm and linear projection predict the complete action chunk in parallel. The adapter reads the backbone output without writing states back into the backbone.

For RoboTwin 2.0, we retain the four-layer, eight-head architecture and 16 action queries, but increase the adapter width to 1,024 and the feed-forward width to 4,096. The visual-memory projection, proprioception encoder, learned queries, and output projection use this wider representation.

Flow-based action adapter. The flow-based adapter matches the default regression adapter in backbone, Action Prefix, visual-memory projection, decoder depth, hidden width, and attention-head count. Each noised seven-dimensional action is linearly projected to width 512, replacing the learned action queries. Proprioception is encoded by the same MLP as in the regression adapter, followed by LayerNorm, and appended as one token to the visual memory. Self-attention uses one-dimensional RoPE with base 10,000 over the 16 action positions. Flow time is represented by a 256-dimensional sinusoidal embedding of 1000τ, followed by a two-layer width-512 MLP with SiLU. Each decoder block uses the embedding to modulate its three pre-normalized sublayers; the modulation projections are zero initialized. A final LayerNorm and linear projection predict the seven-dimensional action velocity.

For a normalized demonstration chunk $\mathbf{A}_t$, we sample one time $\tau \sim \mathcal{U}(0, 1)$ per chunk and Gaussian noise $\boldsymbol{\epsilon}$ of the same shape. Following the flow convention in Equation 1, the noised action chunk is

$$\mathbf{A}_{t,\tau} = (1 - \tau)\mathbf{A}_t + \tau\boldsymbol{\epsilon}. \tag{6}$$

The adapter predicts the velocity $\boldsymbol{\epsilon} - \mathbf{A}_t$ and minimizes mean-squared error over valid action elements within each chunk, averaged across the batch. Key padding masks prevent attention to padded action steps, which are also excluded from the loss. At inference, eight uniform Euler steps integrate from $\tau = 1$ to zero. The VAE and DiT execute once per observation, and the decoder's cross-attention keys and values are cached throughout the action loop. The action-flow time is independent of the perception-flow time; the backbone retains the fixed condition $\tau_A = 0$ during the action forward pass.

Table 9: Action-adapter configurations. Parameter counts include the complete adapter, including visual-memory and proprioception projections, but exclude the Action Prefix.

Configuration	Width	FFN	Layers	Heads	Action dim.	Parameters
LIBERO regression	512	2,048	4	8	7	18.68M
LIBERO flow-based	512	2,048	4	8	7	25.38M
RoboTwin 2.0 regression	1,024	4,096	4	8	14	71.45M

A.3 Input and Output Interfaces

Text conditioning. The text-conditioning sequence contains 128 positions in both forward passes. In the action forward pass, 16 learned Action Prefix tokens occupy the first positions, leaving up to 112 positions for instruction embeddings. In the perception forward pass, no Action Prefix is inserted, so all 128 positions are available for the perception prompt. Unused positions are right padded and masked.

Visual inputs and perception outputs. For an image canvas of height H_I and width W_I, the VAE produces a latent tensor in $\mathbb{R}^{32\times H_I/8\times W_I/8}$. Packing 2×2 neighborhoods gives $(H_I/16)(W_I/16)$ image tokens of dimension 128. Reference tokens encode the current observations. Target tokens have the same layout but are instantiated only in the perception forward pass. The perception output is a velocity tensor aligned with these target tokens.

Proprioception and actions. The simulation action output is a chunk in $\mathbb{R}^{16\times d_a}$, where d_a is the action dimension. For LIBERO, we retain the dataset's seven-dimensional control action and eight-dimensional proprioception. Both use dimension-wise min–max normalization to $[-1, 1]$.

For RoboTwin 2.0, the policy predicts 12 absolute target joint angles and two continuous gripper commands. Proprioception contains the corresponding 14-dimensional current state. Both joint and gripper commands use dimension-wise z-score normalization, while proprioception uses min–max normalization.

Table 10 lists the simulation interfaces. The flow-based action comparison uses the same interface as the LIBERO policy. Image sizes are height $\times$ width, and the execution horizon is the number of predicted actions applied before the next observation.

Table 10: Simulation input and output interfaces. The "Image tokens" column reports the number of reference tokens. Perception passes use the same number of target tokens. Pred./exec. denotes predicted/executed actions per chunk.

Configuration	Views	RGB canvas	Image tokens	State dim.	Action dim.	Pred./exec.
LIBERO policy	2	224×448	392	8	7	16/12
RoboTwin 2.0 policy	3	288×256	288	14	14	16/16
Visual-target ablations	2	256×512	512	8	7	16/12

A.4 Image Preprocessing and Augmentation

Both the action and perception forward passes use the same preprocessed RGB observations; the perception forward pass additionally uses the corresponding perception targets. For depth supervision, observations and targets share spatial preprocessing and geometric augmentation, while photometric augmentation is applied only to RGB observations.

Augmentation. Augmentation is applied only during training. For the LIBERO and RoboTwin 2.0 policies, each training sample is augmented with probability 0.65. When enabled, each camera independently selects additive noise, color adjustments, or both with probabilities $1/4$, $1/4$, and $1/2$, respectively, followed by geometric transforms. Table 11 lists the operations and parameters in each group. Evaluation uses deterministic preprocessing only.

Table 11: Training-time image augmentation.

Group	Operation	Value / range	Applied to
Color adjustments	Brightness factor	$[0.7, 1.3]$	RGB observations
	Contrast factor	$[0.7, 1.3]$	
	Saturation factor	$[0.75, 1.25]$	
	Hue shift	$[-0.04, 0.04]$	
	Gamma	$[0.8, 1.25]$	
	Exposure	$[-0.25, 0.25]$ EV	
Additive noise	Gaussian noise (std.)	0.015	
Geometric transforms	Crop (side-length ratio)	$[0.92, 1]$	RGB observations and depth targets
	Rotation	$[-8^\circ, 8^\circ]$	

Parameters given as intervals are sampled uniformly. Noise standard deviation is specified on a $[0, 1]$ intensity scale. Crops preserve aspect ratio and are resized back to the pre-crop resolution. Resizing uses bilinear interpolation with antialiasing. Rotation uses bilinear interpolation, with mean RGB intensity as fill for observations and zero-valued pixels for depth targets. No horizontal flips are applied.

LIBERO policy. Each 256×256 camera image is first resized to 224×224, augmented during training, and then concatenated horizontally with the other view. The regression and flow-based policies in the action-readout comparison use the same processing.

RoboTwin 2.0 policy. Each of the three camera images is first resized to 240×320. After augmentation during training, the main view is resized to 192×256 and the two wrist views to 96×128 each. The main view is placed above the side-by-side wrist views, producing a 288×256 canvas.

Visual-target ablations. These experiments retain both native 256×256 views, concatenate them into a 256×512 canvas, and apply no stochastic augmentation or spatial resizing. All perception targets and corresponding RGB observations follow the same image preprocessing.

A.5 Training and Optimization

We train all GenPA policies reported in this paper with RADAR, an optimizer from the RAD family (Lyu et al., 2025; Ren et al., 2026). We set $(\beta_1, \beta_2) = (0.9, 0.95)$, enable decoupled weight decay with coefficient 0.01, and use $\gamma = 0$, $l = 0$, $\delta = 1$, and $\zeta = 10^{-16}$.

During native perception co-training, action and perception losses are backpropagated separately, with gradients accumulated before each optimizer update. Configurations that sample multiple perception targets are specified in Appendix B.4. We train in BF16 and clip the gradient norm at 1.0. Seed settings are specified in Appendix C.1. Learning rates warm up linearly over the first 5% of updates and then follow cosine decay to an absolute minimum of 10^{-6} for each parameter group. The action-interface learning rate applies to both the Action Prefix and adapter. The training configuration uses eight NVIDIA H100 GPUs with 80 GB of memory each.

The LIBERO policy configuration in the first column also applies to the regression–flow action-readout comparison and the native-depth–DPT supervision comparison.

Table 12: Simulation training configurations for GenPA.

Hyperparameter	LIBERO policy and comparisons	LIBERO visual-target ablations	RoboTwin 2.0 policy
Peak backbone LR	10^{-5}	10^{-5}	10^{-4}
Peak action-interface LR	10^{-4}	10^{-4}	10^{-4}
Global batch size	80	80	384
Maximum updates	36,750	36,750	70,847
Training epochs	10	10	5
Perception coefficient	0.1	0.1 per sampled target	0.1

Training cost. Table 13 reports resource costs under the LIBERO policy configuration, using BF16 and ZeRO-1 with global batch 80. Update times are averaged over three runs; memory is the maximum per-GPU PyTorch allocation during the training loop. GPU-hours are estimated from measured update times for 36,750 updates, excluding initialization and checkpoint saving.

Table 13: Training resources on eight H100 80GB GPUs.

Training	Update time (s)	Peak memory (GiB/GPU)	Estimated GPU-hours
Action-only	0.501	35.43	40.9
Native depth	1.294	54.11	105.7

A.6 Auxiliary Depth-Regression Head

For the direct-depth comparison in Section 3.5, we use a randomly initialized DPT-style decoder (Ranftl et al., 2021) with 11.064M parameters, branch widths of [128, 256, 512, 512], and a fusion width of 128. All four branches read the same final reference-token features. The two camera views are decoded separately with shared weights to produce two 224×224 depth maps. We add two-dimensional sinusoidal positional embeddings after branch projection and final upsampling, and use an unconstrained linear output.

The depth head and action adapter share a single backbone forward pass without target-image tokens. We jointly optimize the action loss and a pixel-wise depth MSE loss with coefficient 0.1, using

encoded depth targets in $[0, 1]$. Both losses update the shared backbone. The depth head is omitted at inference.

B Visual Supervision and Ablation Details

This appendix specifies the image-form targets in Section 2.3 and the protocol behind Table 7. Model interfaces and image processing are specified in Appendix A; reported-policy and baseline simulation protocols are specified in Appendix C. The visual targets supervise the perception path during training and are not inputs to the deployed action path.

B.1 Visual Tasks as Image Targets

Image representations. Each visual target is represented as a three-channel image paired with the current observation, with values in $[0, 1]$. LIBERO D and G use floating-point image values, whereas N, P, and S use fixed 8-bit RGB encodings. F uses the future RGB image, and FD follows D's floating-point encoding. Figure 4 shows the current two-view RGB reference alongside its current- and future-state targets.

Current metric depth (D). D represents optical-axis depth d in meters, not distance to the camera center. For positive depth, the fixed log-inverse mapping is

$$q_D(d; d_{\min}, d_{\max}) = \frac{\log d_{\max} - \log(\mathrm{clip}(d, d_{\min}, d_{\max}))}{\log d_{\max} - \log d_{\min}}, \qquad d_{\max} = 3.5 \text{ m}. \tag{7}$$

The scalar is copied into all three channels without per-image normalization. We use $d_{\min} = 0.04$ m for LIBERO and $d_{\min} = 0.01$ m for all three RoboTwin 2.0 cameras. Nonpositive RoboTwin depths are encoded as black $(0, 0, 0)$.

Depth backprojection. To construct normals (N) and pointmaps (P), we first backproject LIBERO depth maps into 3D points in camera coordinates. With intrinsics K and homogeneous pixel $\tilde{\mathbf{p}}$ in the OpenCV camera convention, $\mathbf{p}_C = dK^{-1}\tilde{\mathbf{p}}$; the ray is not unit-normalized. The saved images' orientation and renderer sampling positions are accounted for before backprojection: a stored pixel (u_s, v_s) maps to $\tilde{\mathbf{p}} = (256 - 0.625 - u_s,\ v_s + 0.875,\ 1)^\mathsf{T}$. The camera axes are $+x$ right, $+y$ down, and $+z$ forward. Encoded targets return to the stored RGB pixel layout.

Camera-frame surface normals (N). Using the backprojected points $\mathbf{p}_C$, we compute unit normals in the camera frame. At each pixel, horizontal and vertical central differences of $\mathbf{p}_C$ give a cross product that is normalized and oriented toward the camera, $\mathbf{n}^\mathsf{T}(-\mathbf{p}_C) \geq 0$. The three image channels contain $\mathrm{round}[255(\mathbf{n} + 1)/2]$. We exclude neighborhoods with invalid depth or degenerate local geometry. Valid neighborhoods require center-to-neighbor depth differences below $0.05\,\mathrm{m}$ and pairwise normal angles of at most 30° among the four center-neighbor triangles. Borders and rejected neighborhoods are encoded as black.

Robot-frame pointmaps (P). P encodes the robot-base coordinates of visible surface points. Camera-frame points $\mathbf{p}_C$ are transformed using camera extrinsics and the current robot-base pose; X, Y, and Z occupy the R, G, and B channels. Their fixed central intervals are $[-0.25, 1.00]$, $[-0.45, 0.50]$, and $[-0.10, 0.50]$ m. For one coordinate x with interval $[l, h]$, let $\alpha = 224/(h - l)$ and encode

$$c(x) = \begin{cases} 1 + \dfrac{15}{1 + \alpha(l - x)/15}, & x < l, \\ 16 + \alpha(x - l), & l \leq x \leq h, \\ 255 - \dfrac{15}{1 + \alpha(x - h)/15}, & x > h. \end{cases} \tag{8}$$

Values are clipped to $[1, 255]$, rounded to the nearest integer, and converted to image channels. The central interval maps linearly to codes $[16, 240]$; smooth tails preserve out-of-range surface points. The intervals were calibrated on training trajectories and remain fixed across frames and cameras. Pixels without a valid 3D point are encoded as black.

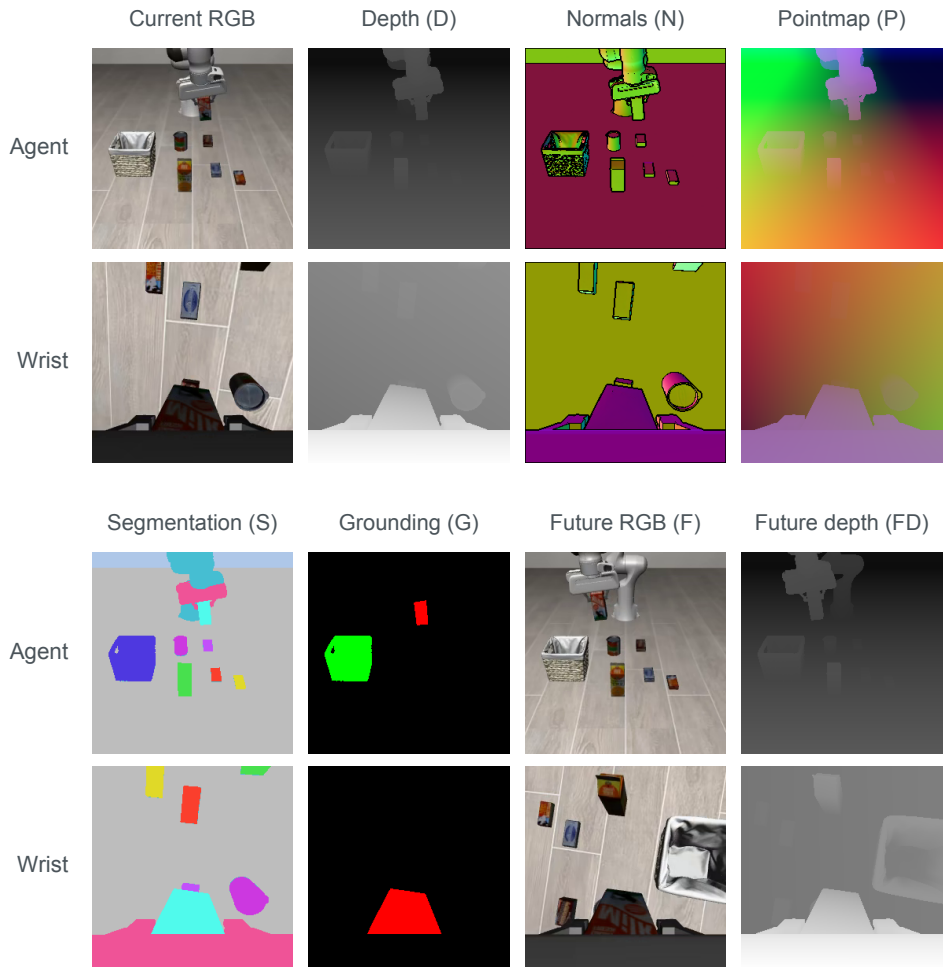


Figure 4: Aligned LIBERO observations and supervision targets in agent and wrist views. D, N, P, S, and G denote current depth, normals, pointmaps, segmentation, and grounding. F and FD use RGB and depth from the same future frame. All panels show training data, not model predictions.

Semantic parts (S). Simulator geometry IDs are mapped through an episode-specific lookup to a fixed 37-class semantic-part ontology and RGB palette shared by the four LIBERO suites and both cameras. Classes include void, static scene surfaces, robot parts, movable objects, and articulated parts such as drawers, doors, and buttons (Figure 5). Same-class instances share a color. S is task independent and covers the whole scene; it is not a per-scene instance coloring.

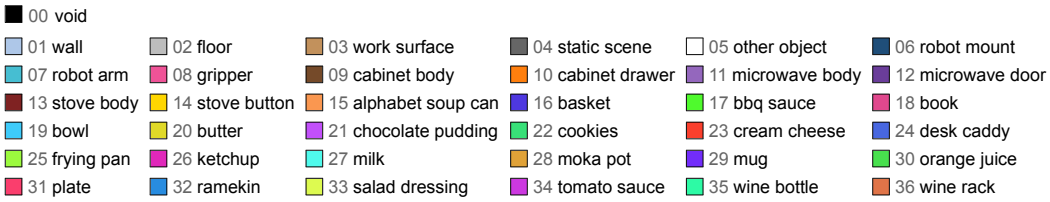


Figure 5: The fixed 37-class LIBERO semantic-part ontology and RGB palette. Colors are shared across all four suites and both cameras.

Task-conditioned grounding (G). G encodes three independent binary roles in R/G/B: the operation object, goal or anchor, and active articulated part. A pixel can occupy more than one role. Roles are assigned to task-relevant instances using task predicates and the demonstration's execution

stage. The active stage follows the order in which predicates become persistently satisfied in that demonstration. The prerequisite drawer-opening stage requires three consecutive satisfied frames to be marked complete. Grounding is defined only before full task completion and only for a camera with at least 64 visible role pixels.

Future RGB (F) and future depth (FD). For current timestep j in a filtered demonstration with L retained timesteps, the target index is $j_f = \min(j + 16, L - 1)$. F is the two-view RGB image at j_f; FD is the depth image from the same future state, encoded and validated exactly as LIBERO D. Both use the *current* RGB canvas as their reference. When $j + 16$ exceeds the final retained timestep, we use the final frame. The offset counts retained action steps after no-op filtering, not necessarily 16 original simulator frames. FD is a single future-depth task; D+F combines current depth with future RGB.

B.2 Label Sources

LIBERO. To supplement the released LIBERO demonstrations (Liu et al., 2023a) with visual supervision, we restore their recorded states in MuJoCo (Todorov et al., 2012) and construct labels for the agent and wrist views. D comes from simulator-rendered depth. N is computed from depth and camera intrinsics; P additionally uses camera extrinsics and the robot-base pose. S maps simulator geometry IDs to semantic categories. G uses task predicates, object identities, and demonstration stages to assign task roles. F and FD use RGB and depth from later states within the same demonstration. All LIBERO target configurations use the same 1,808 retained training demonstrations.

RoboTwin 2.0. Depth supervision comes from the RoboTwin 2.0 RGB-D demonstrations released with Flex-π (Yan et al., 2026), which provide depth maps paired with RGB observations from all three cameras. Dataset splits and evaluation protocols are specified in Appendix C.4.

B.3 Prompts and Perception Loss

Table 14 gives the exact perception prompts. The placeholder {`task`} is replaced by the manipulation instruction. D, N, S, and P use fixed text; G, F, and FD incorporate task language. RoboTwin D uses the same D prompt.

Table 14: Perception prompts.

Target	Prompt
D	Generate a full-scene metric depth map for the current observation.
N	Generate a camera-coordinate surface normal map for the current observation.
S	Generate a semantic part segmentation map for the current observation using the fixed LIBERO part palette.
P	Generate a robot-base-coordinate XYZ point map for the current observation.
G	For the robot task: {`task`}. Generate a current-stage task grounding map: red for the operation object, green for the goal or anchor, and blue for the active part.
F	A video recorded from a robot's point of view executing the following instruction: {`task`}
FD	For the robot task: {`task`}. Generate a full-scene metric depth map for the future observation 16 action steps ahead.

Noise-level sampling and weighting. For the perception objective in Equation 4, we independently sample $u \sim \mathcal{U}(0, 1)$ for each sample and set $\tau = 5u/(1 + 4u)$. The loss weights follow

$$w(\tau) \propto \exp[-2(\tau - \tfrac{1}{2})^2] - \exp(-\tfrac{1}{2}), \tag{9}$$

with numerical normalization to unit mean under this sampling schedule.

Loss averaging. For LIBERO, perception loss is averaged over target tokens within each view, equally over valid views within each sample, and then over the batch. G excludes views without valid grounding targets; samples with no valid views receive zero perception loss. Black-coded

regions in N and P remain included. RoboTwin depth supervision averages over the full three-view canvas, including zero-depth regions, giving the main and two wrist views weights proportional to their areas $(4 : 1 : 1)$.

B.4 VISUAL-TARGET ABLATION PROTOCOL

The experiments in Table 7 use the visual-target ablation configuration in Table 12.

Task sampling. Table 15 summarizes task selection and loss allocation. In multi-target configurations, tasks are sampled independently for each observation in the training batch. Each selected task receives one perception forward pass with loss coefficient 0.1 in Equation 5; when two tasks are selected, their weighted losses are summed.

Table 15: Perception-task sampling and loss weights.

Configuration	Task selection	Perception passes per action pass	Total perception loss coefficient
None	No perception task	0	0
Single target	Fixed task	1	0.1
D+S / D+P / D+F	Uniformly sample one task	1	0.1
D+N+S+P	Uniformly sample two distinct tasks	2	0.2

The action-update budget is fixed across configurations, while perception computation and the total perception-loss coefficient vary as listed in the table.

Additional evaluation results. Table 16 gives the fixed checkpoints and suite-level independent evaluations underlying Table 7.

On LIBERO-Plus, the D–F difference is -0.41 percentage points (95% confidence interval: $[-1.97, 1.19]$; Appendix C.2).

Table 16: Fixed checkpoints and independent LIBERO suite results for visual-target ablations (%).

Target	Step	Spatial	Object	Goal	Long
None	32,000	95.33 ± 0.70	99.40 ± 0.53	97.67 ± 0.42	95.67 ± 0.46
D	36,750	97.60 ± 0.35	99.87 ± 0.23	97.87 ± 0.76	96.13 ± 1.40
N	30,000	97.60 ± 0.53	98.00 ± 0.69	95.53 ± 0.12	94.93 ± 1.01
S	36,000	97.13 ± 0.42	99.67 ± 0.31	96.33 ± 0.76	95.53 ± 1.01
P	36,750	97.60 ± 0.20	98.93 ± 0.31	98.33 ± 0.61	95.20 ± 1.64
G	32,000	97.07 ± 0.23	99.60 ± 0.35	93.80 ± 0.53	96.00 ± 0.60
F	30,000	97.13 ± 0.31	99.33 ± 0.31	96.60 ± 0.72	94.87 ± 0.83
FD	34,000	97.13 ± 0.31	99.87 ± 0.12	97.87 ± 0.50	97.20 ± 0.87
D+S	34,000	97.40 ± 0.35	99.53 ± 0.12	96.33 ± 0.42	95.67 ± 1.70
D+P	32,000	97.60 ± 0.60	99.27 ± 0.42	97.80 ± 0.35	96.47 ± 1.03
D+F	28,000	97.07 ± 0.42	99.60 ± 0.53	97.27 ± 1.14	95.53 ± 1.50
D+N+S+P	34,000	96.53 ± 0.31	99.80 ± 0.20	98.07 ± 0.81	96.67 ± 0.64

Mean $\pm$ SD across three evaluation seeds for each fixed policy.

B.5 PERCEPTION SUPERVISION UNDER THE LIBERO POLICY CONFIGURATION

We additionally compare action-only training (None), current-depth supervision (D), and future RGB supervision (F) under the LIBERO policy configuration. The policies share the architecture, pretrained initialization, and training demonstrations, following the LIBERO training recipe in Appendix A.5. None omits the perception forward pass; D and F each add one perception pass per action update with loss weight 0.1. This comparison complements the visual-target ablations in Table 7.

Table 17: Visual supervision under the LIBERO policy configuration (%).

Metric	None	D	F
Top LIBERO	98.80 ± 0.22	98.87 ± 0.25	98.68 ± 0.10
Final LIBERO	98.28 ± 0.16	98.53 ± 0.08	98.18 ± 0.25
Top Plus Clean	79.49 ± 0.85	82.33 ± 1.27	81.80 ± 2.05
Final Plus Clean	79.45 ± 0.39	82.35 ± 0.85	82.15 ± 2.15
Top Plus Official	74.63 ± 4.03	78.67 ± 3.97	77.74 ± 6.57
Final Plus Official	74.32 ± 3.48	79.27 ± 3.62	78.64 ± 5.82

Mean $\pm$ SD across three training runs.

Both D and F improve mean LIBERO-Plus success over None at Top and Final checkpoints under both instruction protocols. For Top checkpoints under Clean-Instruction, the D–F difference is $+0.53$ percentage points (95% confidence interval: $[-0.43, 1.54]$; Appendix C.2).

B.6 Direct Depth-Regression Control

We compare native flow-based depth supervision (Native flow) with direct depth regression (DPT regression) using the DPT-style head described in Appendix A.6. Both follow the LIBERO policy configuration in Appendix A.5, with the same depth labels and aligned geometric augmentation.

Training costs follow the setup in Appendix A.5, using 50 warm-up and 150 timed updates per run.

Table 18 extends Table 6 to Final checkpoints and Official instructions.

Table 18: Native depth supervision and DPT regression (%).

Metric	Native flow	DPT regression
Top LIBERO	98.87 ± 0.25	98.85 ± 0.10
Final LIBERO	98.53 ± 0.08	98.67 ± 0.24
Top Plus Clean	82.33 ± 1.27	80.61 ± 0.35
Final Plus Clean	82.35 ± 0.85	80.27 ± 0.59
Top Plus Official	78.67 ± 3.97	76.72 ± 2.95
Final Plus Official	79.27 ± 3.62	76.39 ± 1.96

Mean $\pm$ SD across three training runs.

C Simulation Evaluation and Baseline Details

C.1 Standard LIBERO

The repeated-training comparisons use seeds 42, 142, and 242 for None, current depth, future RGB, the flow-based action adapter, and DPT regression under the LIBERO policy configuration. Other GenPA training runs retain the original seed 42.

Evaluation protocol. We evaluate each of the 40 tasks for 50 trials, using the benchmark's released initial states in their provided order and the simulator's task-success signal. Episodes allow up to 400 action steps for Spatial, Object, and Goal and 700 for Long, following 30 initial no-op steps. The policy predicts 16 actions per call and executes the first 12 before replanning, with binary gripper postprocessing and no temporal action ensemble. Unless otherwise stated, GenPA benchmark success rates are measured using uncompiled inference.

Checkpoint selection. We apply one selection rule to all our LIBERO-trained policies, including both action heads and all perception-supervision configurations. The candidate set consists of ten checkpoints at 20k, 22k, 24k, 26k, 28k, 30k, 32k, 34k, 36k, and 36.75k action updates. We select the checkpoint with the most successes across the 2,000 standard LIBERO trials, breaking ties in favor of the earlier checkpoint. The same checkpoint is used for LIBERO-Plus, without further

selection or fine-tuning. We denote the selected checkpoint as *Top*, the default for reported results, and the fixed 36,750-step checkpoint as *Final*. Unless marked independent, our standard LIBERO results use the same initial states as checkpoint selection. Complete checkpoint scans are provided in Appendix C.3.

Independent LIBERO evaluation. Since standard LIBERO scores are used for checkpoint selection, we re-evaluate the Top checkpoints from training seed 42 using three evaluation seeds (1042, 1142, and 1242). Each seed provides 50 new initial states per task, yielding 6,000 episodes per policy. The states satisfy the original task constraints and are disjoint across sets and from those used for checkpoint selection. All policies use the same three sets.

Baseline sources. Standard LIBERO baseline scores come from published reports; Table 19 identifies their sources and model variants. Their training and checkpoint-selection procedures may differ from ours. Robot PT denotes robot-data pretraining before benchmark adaptation, excluding generic image, video, or vision-language pretraining.

Table 19: Sources and selected configurations for the LIBERO comparison.

Method	Score source	Configuration used
OpenVLA-OFT	Kim et al. (2025), Table XIV	Joint four-suite policy initialized from robot-pretrained OpenVLA.
$\pi_{0.5}$	Physical Intelligence (2026), official LIBERO release; model described by Physical Intelligence et al. (2025)	Released 30k-step flow policy.
VLA-Adapter-Pro	Wang et al. (2026b), arXiv v2, Table 5	Pro variant; four suite-specific checkpoints (official release).
SimVLA	Luo et al. (2026), Table 2	Large action-head configuration with generic VLM initialization.
VLANeXt	Wu et al. (2026b), arXiv v3, Table 2	Base LIBERO configuration. Its separate DROID-pretrained real-world variant is not used.
Qwen-RobotManip	Yuan et al. (2026a), Table 3	Base model, not the Context variant; only overall success is reported.
LaWAM	Chen et al. (2026), Table 1	Robot-video-pretrained policy with a truncated Qwen3-VL backbone and a latent world model.
Cosmos Policy	Kim et al. (2026), Table 1	Author-reported mean over 6,000 evaluation episodes.
Fast-WAM	Yuan et al. (2026b), arXiv v2, Table 2	Fast-WAM, not its Joint or IDM variant.
ABot-M0.5	AMAP CV Lab (2026), Table 5	Robot-pretrained model with Wan2.2-5B initialization.
ImageWAM-4B	Zhang et al. (2026a), Table 2	FLUX.2 4B configuration.
GenPA	This work	Depth-supervised policy trained jointly on all four suites.

Parameter counts. Unless specified below, counts in Table 1 follow the papers or official model releases listed in Table 19. Backbone sizes use reported model sizes; separate text encoders and VAEs are excluded for visual generation backbones. Head counts include input/output and conditioning projections, but exclude separately implemented proprioception projectors and backbone query tokens. GenPA's adapter count follows Appendix A.2.

We count OpenVLA-OFT's 151M-parameter head from its official implementation, confirming it against the loaded model. Counts from public weights use VLA-Adapter-Pro's Spatial-Pro checkpoint (1.25B VLM, 218M head) and LaWAM's LIBERO checkpoint (1.52B VLM, 306M head). VLANeXt's 612M head is computed from its official base configuration and implementation. SimVLA's approximately 300M head follows the authors' estimate.

VLA-Adapter-Pro's VLM count includes its vision encoders, language model, and projector; its quoted 0.5B refers only to the language model. LaWAM's checkpoint additionally contains a 228M

latent world decoder and an 86M DINO encoder, identified separately from its action head; training-only modules are excluded. Cosmos Policy uses its diffusion backbone for action generation, so its head entry is *Shared*. Qwen-RobotManip and ABot-M0.5 have unverified head counts because complete configurations or accessible weights for the corresponding LIBERO policies were unavailable at the time of inspection; their entries remain blank.

C.2 LIBERO-PLUS: INSTRUCTION PROCESSING, RE-EVALUATION, AND PUBLISHED RESULTS

Instruction processing. We refer to evaluation with released instructions as *Official* and with the processing below as *Clean*. For 8,493 non-language instances, the released instructions contain appended perturbation metadata. We remove only the exact perturbation suffix and preserve the canonical task instruction. All 1,537 Language-category instructions remain unchanged, including any numbers that are part of the intended instruction. Table 20 shows an example from each perturbation category before and after processing. Flex-π also identifies this instruction issue (Yan et al., 2026).

Table 20: Representative LIBERO-Plus instructions before and after processing. The Language instruction is preserved; in the other categories only appended perturbation identifiers are removed.

Perturbation	Released instruction	Clean-Instruction
Camera	put both the alphabet soup and the tomato sauce in the basket view 0 0 100 2 10 initstate 0	put both the alphabet soup and the tomato sauce in the basket
Robot initialization	put both the alphabet soup and the tomato sauce in the basket view 0 0 100 0 0 initstate 51	put both the alphabet soup and the tomato sauce in the basket
Language	before we start cooking dont forget to put both the alphabet soup and the tomato sauce in the basket	before we start cooking dont forget to put both the alphabet soup and the tomato sauce in the basket
Lighting	turn on the stove and put the moka pot on it light 19	turn on the stove and put the moka pot on it
Background	turn on the stove and put the moka pot on it table 19	turn on the stove and put the moka pot on it
Sensor noise	put both the alphabet soup and the tomato sauce in the basket view 0 0 100 0 0 initstate 0 noise 5	put both the alphabet soup and the tomato sauce in the basket
Object layout	turn on the stove and put the moka pot on it add 7	turn on the stove and put the moka pot on it

Evaluation protocol. Across our LIBERO-Plus evaluations, we retain the released environments and run one trial per task instance for each policy, using environment seed 42 and the first released initial state. Rendering uses two 256×256 views, followed by each policy's native preprocessing. We use 30 initial no-op steps and limits of 400 action steps for Spatial, Object, and Goal and 700 for Long. The seven categories contain 1,599 Camera, 1,550 Robot, 1,537 Language, 1,142 Light, 1,076 Background, 1,601 Noise, and 1,525 Layout instances. Overall success divides total successes by 10,030; category scores use their respective instance counts. All evaluations cover all 10,030 instances. Unless otherwise stated, success rates use uncompiled inference. The external policies use released LIBERO checkpoints; we do not fine-tune them on LIBERO-Plus.

We evaluate all policies on the same environment instances with the same processed task instructions. Policies differ in pretraining and retain their native image transforms, action normalization, prompt templates, action horizons, and replanning frequencies. Fast-WAM uses a noise-schedule shift of 5.0. Cosmos Policy retains its native parallel future/value outputs without additional autoregressive planning or candidate search. VLANeXt uses five denoising steps, following its public LIBERO-Plus entry point rather than the checkpoint default of ten. ImageWAM's released LIBERO evaluator uses Prefix-Only inference by default, which we follow for re-evaluation. This path omits the image-denoising step used in its paper's main results.

Table 21: Checkpoints and inference settings for the LIBERO-Plus comparison in Table 2. H is the predicted action horizon; R is the number of executed actions before replanning; K is the number of action-denoising steps.

Policy	Checkpoint and inference path	H	R	K
OpenVLA-OFT	Official joint four-suite 300k checkpoint; regression (Kim et al., 2025).	8	8	—
$\pi_{0.5}$	Official `pi05_libero`; released PyTorch conversion and native transforms (Physical Intelligence et al., 2025).	10	5	10
VLANeXt	Released suite-specific checkpoints; official LIBERO-Plus entry point (Wu et al., 2026b).	8	8	5
Cosmos Policy	Released 2B LIBERO policy, retaining parallel future/value generation (Kim et al., 2026).	16	16	5
Fast-WAM	Released two-view 224-pixel LIBERO checkpoint; action-only inference (Yuan et al., 2026b).	32	10	10
ImageWAM	Released FLUX.2 4B LIBERO checkpoint; default Prefix-Only inference (Zhang et al., 2026a).	16	12	10
GenPA	Depth-supervised 32k checkpoint; direct regression.	16	12	—

At the time of evaluation, Qwen-RobotManip did not provide a public model checkpoint, and we could not obtain an accessible ABot-M0.5 checkpoint fine-tuned for LIBERO. We exclude these two methods from the common-protocol re-evaluation while retaining their published results in the standard LIBERO comparison and Table 22.

Published LIBERO-Plus scores. Table 22 lists published baseline scores alongside GenPA's evaluation with the released instructions. These results do not constitute a common-protocol comparison: instruction processing and aggregation are not uniformly documented across sources. Table 23 records each source, configuration, and aggregation method. Differences from our Clean-Instruction re-evaluations cannot be attributed solely to instruction cleaning, since checkpoints, inference paths, and evaluation settings may also differ.

Table 22: Published LIBERO-Plus baseline scores and GenPA evaluation with released instructions (%).

Method	Robot PT	Camera	Robot Init.	Lang.	Light	Backgr.	Noise	Layout	Overall
OpenVLA-OFT^{j}	✓	55.6	21.7	81.0	92.7	91.0	78.6	68.7	67.9
$\pi_{0.5}$	✓	75.4	77.5	85.6	96.9	94.6	89.7	85.7	85.7
VLANeXt	✗	90.1	65.2	81.0	96.0	82.1	93.6	80.1	83.7
Qwen-RobotManip	✓	87.2	75.5	85.6	96.6	97.7	97.7	87.3	89.0
Cosmos Policy	✗	75.8	63.3	81.7	96.5	88.9	92.7	82.2	82.2
Fast-WAM	✗	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
ABot-M0.5†	✓	70.5	87.4	88.6	94.0	89.7	75.5	85.2	83.4
ImageWAM-4B†	✗	80.8	50.3	91.4	98.1	85.5	93.8	80.5	83.1
GenPA	✗	81.18	67.29	84.97	97.64	80.11	91.69	80.92	83.01

Robot PT is defined in Table 1; j denotes joint four-suite training. †: The published overall aggregation cannot be reconstructed from the reported breakdowns.

Uncertainty in LIBERO-Plus comparisons. For the internal regression–flow and depth–future-RGB comparisons on LIBERO-Plus, we report pointwise 95% percentile confidence intervals from 20,000 paired task-cluster bootstrap replicates. Within each of the four suites, we resample its ten original tasks with replacement, retaining all perturbation instances and paired policy outcomes and preserving instance-pooled scoring. For three-run comparisons, we average the three paired differences within each replicate. These intervals quantify task-level uncertainty for fixed policies; training-run variability is reported separately.

Table 23: Sources and aggregation of the published LIBERO-Plus scores in Table 22.

Method	Evaluation source	Selection and aggregation
OpenVLA-OFT	Fei et al. (2026), Table 2 and official leaderboard	Joint four-suite policy (OpenVLA-OFT_m).
$\pi_{0.5}$	Zhang et al. (2026b), Table 4	Third-party evaluation.
VLANeXt	Wu et al. (2026b), Table 3	Approximate overall score, reweighted from rounded suite scores using released instance counts.
Qwen-RobotManip	Yuan et al. (2026a), Table 4	Base configuration.
Cosmos Policy	Zhang et al. (2026b), Table 4	Third-party evaluation.
Fast-WAM	Zhang et al. (2026b), Table 4	Third-party evaluation; overall is the mean across four LIBERO suites.
ABot-M0.5	AMAP CV Lab (2026), Table 6	The reported overall weighting is not established as instance-pooled.
ImageWAM-4B	Zhang et al. (2026a), Table 3	The 4B result's overall weighting is not established as instance-pooled.
GenPA	This work	Released instructions; 8,326 successes over 10,030 instances from the same 32k-step checkpoint.

C.3 Checkpoint Sensitivity and Independent LIBERO Results

Table 24 gives complete checkpoint scans for the supervision comparison in Appendix B.5. Table 25 covers Flow (Appendix D.1) and DPT (Appendix B.6), with their shared D baseline in Table 24.

Table 24: Standard-LIBERO checkpoint scans for None, D, and F (%). Column suffixes denote training seeds.

Step	None 42	None 142	None 242	D 42	D 142	D 242	F 42	F 142	F 242
20,000	98.25	97.60	97.80	97.60	98.30	98.30	98.25	97.30	98.50
22,000	97.90	97.30	97.00	97.20	97.95	97.85	97.85	98.05	97.10
24,000	97.65	97.40	98.15	98.45	**98.90**	98.45	97.15	98.55	97.90
26,000	98.25	98.10	98.30	98.65	98.65	98.20	98.25	98.00	**98.60**
28,000	**98.95**	98.30	**98.55**	99.00	98.05	**98.60**	98.65	98.10	98.60
30,000	98.10	98.35	97.80	98.65	98.40	98.40	98.10	98.30	97.70
32,000	98.95	98.30	98.55	**99.10**	98.25	98.20	98.75	**98.65**	98.30
34,000	98.75	**98.90**	98.40	98.75	98.30	98.50	98.20	98.60	97.90
36,000	98.50	98.60	98.55	98.60	98.55	98.40	**98.80**	98.55	98.25
36,750	98.40	98.10	98.35	98.45	98.60	98.55	98.35	98.30	97.90

Bold marks the selected checkpoint.

Table 25: Standard-LIBERO checkpoint scans for Flow and DPT (%). Column suffixes denote training seeds.

Step	Flow 42	Flow 142	Flow 242	DPT 42	DPT 142	DPT 242
20,000	93.15	94.75	83.60	97.65	97.00	97.80
22,000	94.20	93.40	93.20	**98.95**	97.90	97.75
24,000	95.35	93.70	93.95	97.55	98.15	96.85
26,000	94.90	95.55	96.45	98.10	98.00	98.35
28,000	96.95	97.25	97.45	98.25	97.90	98.55
30,000	97.85	98.00	97.25	98.30	**98.75**	98.45
32,000	97.80	**98.20**	**98.20**	98.95	98.30	98.40
34,000	97.95	97.90	97.90	98.90	98.25	98.75
36,000	97.50	97.90	98.00	98.15	98.60	98.25
36,750	**98.40**	97.60	98.00	98.75	98.40	**98.85**

Bold marks the selected checkpoint. Flow denotes the flow-based action head.

Table 26 reports independent LIBERO results for the selected checkpoints from training seed 42, following the protocol in Appendix C.1.

Table 26: Independent evaluation under the LIBERO policy configuration (%).

Policy	Step	Spatial	Object	Goal	Long	Overall
None	28,000	96.40 ± 0.35	99.67 ± 0.31	97.60 ± 0.53	96.87 ± 0.42	97.63 ± 0.24
D (regression action head)	32,000	97.60 ± 0.20	99.73 ± 0.31	98.20 ± 0.40	97.80 ± 1.11	98.33 ± 0.29
F	36,000	97.13 ± 0.64	99.27 ± 0.12	97.73 ± 0.70	97.07 ± 0.12	97.80 ± 0.05
D (flow action head)	36,750	96.60 ± 0.35	99.00 ± 0.35	97.07 ± 0.90	95.47 ± 0.31	97.03 ± 0.33
DPT regression	22,000	97.47 ± 0.31	99.73 ± 0.12	98.47 ± 0.83	96.80 ± 0.40	98.12 ± 0.33

Mean $\pm$ SD across three evaluation seeds for each fixed policy.

C.4 RoboTwin 2.0: Evaluation, Baselines, and Adapter Width

Training data and evaluation. Table 3 compares policies trained jointly across 50 tasks using both clean and randomized demonstrations. We use the RGB-D-annotated RoboTwin dataset released with Flex-π (Yan et al., 2026), containing 2,500 clean and 25,000 randomized demonstrations. We randomly split the demonstrations at the episode level with seed 42 into 27,225 training episodes and 275 validation episodes, without stratification by task or scene condition. We evaluate 100 trials per task under each of the Clean and Randomized conditions.

Published baseline results. The $\pi_{0.5}$ and Fast-WAM scores are from Yuan et al. (2026b), arXiv v2, Table 1; the latter is the action-only Fast-WAM variant. ImageWAM scores are from Zhang et al. (2026a), Table 1, using the FLUX.2 4B configuration described in Section 4.1 of that paper. Qwen-RobotManip uses the Base configuration from Yuan et al. (2026a), arXiv v2, Table 3, where Easy/Hard correspond to Clean/Randomized. Motus scores are from Bi et al. (2025), Table 2, for the robot-pretrained policy fine-tuned on 2,500 clean and 25,000 randomized demonstrations. Clean and Randomized scores are taken from published reports. Where a source does not report the Avg. at the displayed precision, we compute the arithmetic mean of its two condition scores and round to the displayed precision. These are not matched re-evaluations.

Action-adapter width. The RoboTwin results in Table 3 use an action adapter with a hidden width of 1,024, compared with 512 for the LIBERO policy. To assess the effect of this increase, we compare both widths on RoboTwin in Table 27. Both adapters follow the design in Appendix A.2 and use the same backbone, data split, depth supervision, and training recipe, with training seed 42. For each width, we save and screen 12 checkpoints at 28k, 32k, 36k, 42k, 46k, 50k, 54k, 56k, 60k, 64k, 68k, and 70,847 updates. We select the checkpoint with the highest average success across Clean and Randomized conditions, using 25 trials per task per condition, then evaluate the selected checkpoint with the 100-trial protocol above.

Table 27: RoboTwin action-adapter width comparison (success rates in %). The 1,024-width configuration is used in Table 3.

Adapter width	Checkpoint	Parameters	Clean	Rand.	Avg.
512	64k	18.69M	91.04	89.84	90.44
1,024	64k	71.45M	92.08	90.82	91.45

D Action Readout and Inference Efficiency

This appendix supplements Section 3.4 with action-readout comparisons, timing protocols, latency and memory measurements, and compiled-policy evaluation. The timed policies follow the LIBERO inference settings in Table 21.

D.1 ACTION-READOUT COMPARISON

We compare the regression and flow-based adapters in Appendix A.2, retaining current-depth supervision and the LIBERO policy configuration in Appendix A.5. Table 28 extends Table 5 to Final checkpoints and Official instructions. Regression has higher mean standard LIBERO success, whereas the Plus ranking changes with checkpoint and instruction protocol. For Top checkpoints under Clean-Instruction, regression exceeds flow by 0.84 percentage points on LIBERO-Plus (95% confidence interval: $[-0.49, 2.24]$; Appendix C.2).

Table 28: Action-readout success across checkpoint and instruction settings (%).

Metric	Regression	Flow
Top LIBERO	98.87 ± 0.25	98.27 ± 0.12
Final LIBERO	98.53 ± 0.08	98.00 ± 0.40
Top Plus Clean	82.33 ± 1.27	81.49 ± 1.18
Final Plus Clean	82.35 ± 0.85	82.46 ± 0.43
Top Plus Official	78.67 ± 3.97	79.47 ± 2.42
Final Plus Official	79.27 ± 3.62	79.98 ± 2.60

Mean $\pm$ SD across three training runs.

D.2 MEASUREMENT PROTOCOL

We measure batch-one policy calls on an NVIDIA H100 80GB HBM3 (700 W power limit) and an NVIDIA L40S (350 W power limit). Both systems use PyTorch 2.7.1 and CUDA 11.8. Models use BF16 for their main computations while retaining implementation-specific higher-precision operations. Each configuration is benchmarked in three independent runs, each in a fresh process with exclusive GPU access and four CPU inference threads. After 50 warmup calls, each process measures the same 200 observations from 40 LIBERO tasks (five observations per task). We report the mean and the sample standard deviation of the three process means. Latencies are per native action chunk, whose horizon differs across policies. *Uncompiled* denotes PyTorch eager execution; *compiled* denotes execution with `torch.compile` and TorchInductor, with model-specific compilation scope described below.

For the main comparison, synchronized wall-clock timing begins with decoded CPU RGB images, raw proprioception, and either a raw instruction or cached text features, as specified below. It ends with the CPU action chunk after native postprocessing. Image preprocessing, CPU–GPU transfers, model execution, and action denormalization are included. Image-file decoding, model loading, compilation, and environment or robot communication are excluded. Only observation-independent text features may be reused across calls; visual features and observation-conditioned representations are recomputed for every observation. These offline measurements do not measure robot request–response time or physical control frequency.

The total labels in Figure 3 and the L40S totals below come from complete calls without internal instrumentation. The colored component intervals in that figure come from a separate profiling pass with the same inputs and warmup procedure. They are not rescaled to match the independently timed totals.

D.3 COMPONENT PROFILING AND COMPILATION

Table 29 maps each policy's measured stages to the *Vision*, *Backbone*, and *Action* labels in Figure 3. GenPA, Fast-WAM, and ImageWAM reuse independently encoded text features across observations; OpenVLA-OFT and $\pi_{0.5}$ do not.

Profiling boundaries. Component latencies are measured with nonoverlapping CUDA-event intervals within full policy calls rather than by running each module in isolation. *Other* is the profiled complete-call latency minus the sum of the timed components. It includes remaining preprocessing, transfers, and postprocessing.

Table 29: Model-specific component boundaries for Figure 3.

Policy	Vision	Backbone	Action
OpenVLA-OFT	Vision encoder and projector	Multimodal Llama forward, including parallel action slots	Proprioception projector and regression head
$\pi_{0.5}$	SigLIP encoding and visual projection	VLM prefix forward	Ten-step action expert
Fast-WAM	VAE encoding and visual projection	Video input preparation and backbone prefill	Ten-step action loop
ImageWAM	VAE encoding and visual projection	FLUX Prefix-Only prefill	Ten-step action loop
GenPA	VAE encoding and visual projection	One FLUX forward	Complete regression adapter

Model-specific boundaries. The official PyTorch $\pi_{0.5}$ configuration encodes two observed images and a masked zero-image slot. Its *Action* interval includes ten expert calls; Euler updates and loop control fall in *Other* without compilation, whereas compiled *Action* also includes Euler updates. Compiled *Vision* and *Backbone* are timed jointly because they share a captured graph.

Compilation. GenPA and ImageWAM compile VAE encoding, backbone conditioning, and action decoding, including the complete eight-step loop for GenPA's flow variant. OpenVLA-OFT compiles its vision encoder, projectors, Llama backbone, and regression head. These policies use the `default` compilation mode. Fast-WAM uses `reduce-overhead` for backbone prefill and individual action steps, leaving its VAE uncompiled. $\pi_{0.5}$ uses `max-autotune` for its full action-inference procedure. Native preprocessing and attention implementations are retained. These cross-policy comparisons characterize the measured implementations rather than isolate the causal effect of a single module.

D.4 H100 Component and Total Latency

Table 30 gives the component and independently timed total latencies underlying Figure 3.

Table 30: H100 latency breakdown and independently timed totals (ms per native action chunk).

Policy	Execution	Vision	Backbone	Action	Other	Total
OpenVLA-OFT	Uncompiled	27.57	26.75	1.05	16.97	72.12 ± 1.29
	Compiled	14.91	21.64	0.32	16.70	53.36 ± 0.29
$\pi_{0.5}$	Uncompiled	24.61	13.58	121.09	7.76	166.10 ± 0.96
	Compiled	14.09 (joint)		14.20	8.75	36.59 ± 0.06
Fast-WAM	Uncompiled	5.05	32.34	271.42	11.81	322.07 ± 9.53
	Compiled	4.96	10.22	45.25	11.08	71.34 ± 0.23
ImageWAM	Uncompiled	4.82	24.30	121.49	10.30	159.51 ± 1.69
	Compiled	2.37	15.78	28.73	10.20	57.39 ± 1.27
GenPA	Uncompiled	4.71	16.04	0.62	6.62	27.86 ± 0.32
	Compiled	2.29	10.46	0.46	7.83	20.94 ± 0.47

Components come from separate profiling passes; Total reports independently timed complete calls (mean $\pm$ SD across three process means), so components need not sum to Total. Vision and Backbone are timed jointly for compiled $\pi_{0.5}$.

D.5 Complete Policy-Call Latency on L40S

Table 31 reports the L40S complete-call results for the five policies in Figure 3 and the GenPA flow readout in Table 5.

Table 31: L40S action-chunk latency (ms; mean ± SD).

Policy	Uncompiled	Compiled
GenPA	42.73 ± 0.29	36.36 ± 0.14
GenPA (flow)	78.43 ± 0.44	47.96 ± 0.56
OpenVLA-OFT	107.98 ± 0.85	89.06 ± 1.08
$\pi_{0.5}$	249.58 ± 0.88	62.02 ± 0.76
Fast-WAM	462.32 ± 8.23	96.56 ± 0.67
ImageWAM	226.89 ± 1.25	96.86 ± 0.24

D.6 Latency and Memory with Instruction Encoding

When an instruction requires encoding, GenPA's uncompiled complete-call latency is 54.90 ± 0.47 ms on H100 and 75.62 ± 0.48 ms on L40S. These measurements use the same batch-one LIBERO inputs with Qwen3-4B already loaded and warmed, and include prompt preparation, tokenization, and text encoding in addition to the policy computation timed above.

For batch-one compiled inference on H100, peak PyTorch allocated memory is 7.63 GiB with cached instructions and Qwen unloaded, and 15.18 GiB with Qwen resident when encoding a new instruction. Action-only and depth-co-trained policies have the same deployment architecture and measured memory footprint. These peaks cover post-warm-up policy calls, excluding loading, compilation, and driver allocations.

D.7 Closed-Loop Performance after Compilation

We re-evaluate the original training-seed42 regression and flow policies on LIBERO and all 10,030 LIBERO-Plus Clean-Instruction instances after compilation, using the same checkpoints, the evaluation protocols in Appendices C.1 and C.2, and the compilation settings used for latency measurement. Table 32 compares their success rates before and after compilation.

Table 32: GenPA closed-loop success before and after compilation (%). LIBERO-Plus uses Clean-Instruction.

Action adapter	Execution	LIBERO	LIBERO-Plus
Regression	Uncompiled	99.10	83.79
	Compiled	99.00	83.58
Flow	Uncompiled	98.40	82.52
	Compiled	98.20	82.19

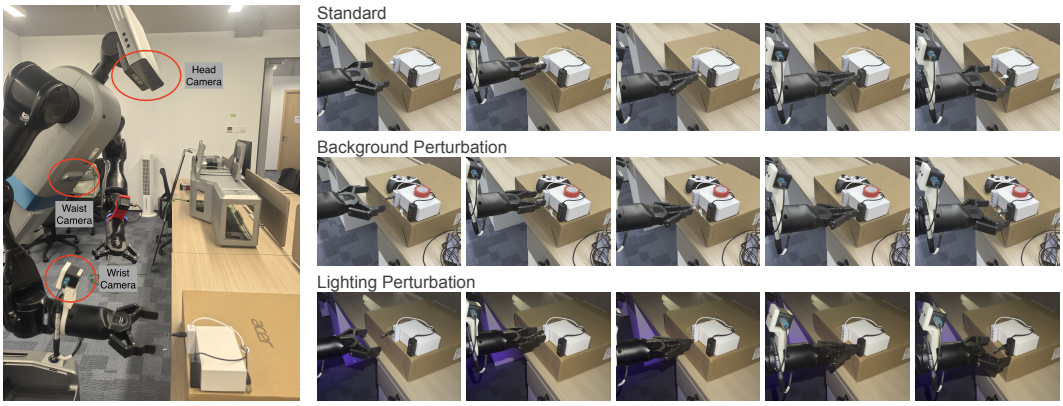
E Real-world Experimental Details

This appendix supplements the real-world comparison in Section 3.3 with the robot interface, training and deployment protocol, and evaluation conditions.

E.1 Robot, Data, and Policy Deployment

Robot and observations. The experiments use the dual-arm robot shown in Figure 6. Only the right arm and gripper are controlled; the left arm, waist, and head remain fixed. All policies receive the same three RGB views from the head, waist, and right wrist, together with a language instruction and a 15-dimensional state comprising seven right-arm joint positions, seven joint efforts, and the gripper state. No policy receives depth at deployment.

Demonstrations and training. We collect 600 demonstrations through teleoperation, with 200 demonstrations for each of three source ports on the left hub. GenPA, $\pi_{0.5}$, Fast-WAM, and ImageWAM use this shared dataset; the baselines use their official implementations. GenPA additionally



(a) Experimental Setup (b) USB Manipulation

Figure 6: **Real-world setup and USB manipulation.** (a) The robot platform and camera locations. (b) Example execution sequences under standard conditions, background perturbation, and lighting perturbation (top to bottom), progressing from left to right. The task is to unplug a USB connector from the left hub and insert it into the right hub.

uses current-depth supervision from depth cameras. All four policies are fine-tuned on these demonstrations for 10 epochs and evaluated at their final checkpoints. Fast-WAM and ImageWAM are initialized from `Wan2.2-TI2V-5B` and `FLUX.2-klein-base-4B`, respectively, with their action branches initialized following the official procedures. Neither loads a policy checkpoint trained on other robot tasks, and both retain joint supervision on actions and future RGB observations. $\pi_{0.5}$ is initialized from the official robot-pretrained `pi05_base` checkpoint. For real-world training, GenPA uses the LIBERO augmentation recipe specified in Appendix A.4. ImageWAM and $\pi_{0.5}$ retain their native image augmentations, while Fast-WAM uses deterministic preprocessing without stochastic image augmentation, as in its publicly released implementation. The real-world results therefore compare policies trained with each method's own image augmentation.

Policy deployment. All policies are deployed on the same NVIDIA RTX PRO 6000 Blackwell GPU and use a 50 Hz controller. For all policies, arm targets are represented as offsets from the joint positions observed at the beginning of each chunk and converted to joint targets for execution. Gripper commands are absolute and binary at deployment. Table 33 summarizes the prediction and execution horizons and flow integration steps. ImageWAM uses its released Prefix-Only inference configuration, without generating future images. Each policy executes the number of actions specified in the table, discards any unused predictions, and replans from fresh observations. For the baselines, inference and execution do not overlap: the controller holds the last commanded joint target during inference, without temporal action averaging or smoothing across chunks.

Table 33: Real-world policy deployment settings.

Policy	Predicted actions	Executed actions	Flow integration steps
GenPA	16	16	—
Fast-WAM	32	24	10
ImageWAM	16	16	10
$\pi_{0.5}$	50	25	10

— denotes direct action regression.

E.2 Evaluation Protocol

We evaluate task execution under three visual conditions. In all conditions, the initial right-arm pose and the source port on the left hub vary randomly across trials, while the destination remains the first port on the right hub. At reset, the end-effector position is perturbed by $\Delta x \in [-0.05, 0]$ m and

$\Delta y, \Delta z \in [-0.05, 0.05]$ m, while its orientation remains fixed. Each policy is evaluated for 50 trials per condition, totaling 150 trials.

- **Standard:** the nominal scene, with the initial-condition variations described above.
- **Background:** visual distractors, such as a game controller and tape, are added to the workspace.
- **Lighting:** room lights are switched off, and the scene is illuminated by the recording phone's LED.

A trial succeeds only if the robot unplugs the connector, inserts it into the target port, and releases the gripper within 50 seconds without dropping the connector. This timeout is used to terminate stalled trials; in our evaluations, it did not interrupt any trial still progressing toward task completion. We report success rates for each condition and their arithmetic mean in Table 4. Figure 6 shows example executions under each condition.

F Generation Details for Figure 1

F.1 Source Image and Illustrative Edits

The input is a frame from the *Random Kitchen* source video of ALOHA Unleashed (Zhao et al., 2025). Both outputs in Figure 7 were generated from the same source image using GPT Image 2 with separate prompts. These are illustrative image edits, not GenPA predictions or robot execution results. Figure 1 uses abbreviated prompts for layout; the complete prompts follow.



(a) Original scene (b) Generated goal state (c) Generated motion sketch

Figure 7: Original scene and generated goal-state and motion-sketch images used in Figure 1(a).

F.2 Complete Prompts

Goal-state editing.

> Edit this photograph to show the scene after the robot has put the green spoon into the pink bowl and the white fork into the yellow-orange bowl. Keep the other utensils, cups, bowls, background, and camera view unchanged. Each moved object must appear exactly once. Return only the edited scene.

Image-space motion sketch.

> On this photograph, sketch how the robot would pick up the green spoon and place it into the pink bowl, then pick up the white fork and place it into the yellow-orange bowl. Use dashed, one-way arrows that follow each movement from the gripper to the utensil and then to its destination. Keep the photograph unchanged.