GEOMETRIC ACTION MODEL: REPURPOSING A GEOMETRIC FOUNDATION MODEL FOR ROBOT POLICY LEARNING

Anonymous authors
Paper under double-blind review

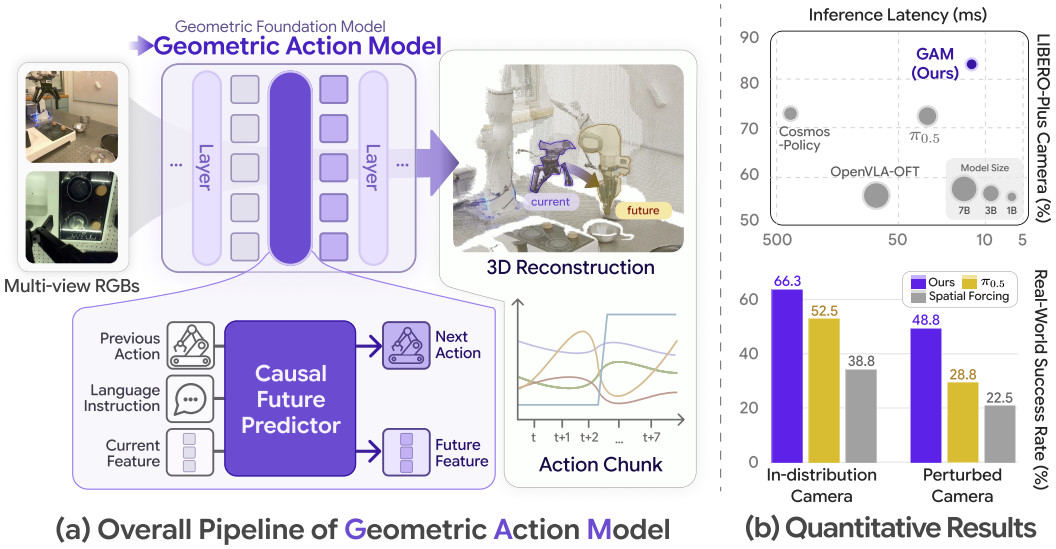


Figure 1: **Overview of the Geometric Action Model (GAM).** (a) GAM jointly predicts action chunks and future 3D geometry within a shared geometric backbone. (b) By directly leveraging 3D geometric priors from geometric foundation models, GAM improves robustness and real-world performance while reducing latency and model size compared to existing baselines.

ABSTRACT

Language-conditioned robot manipulation policies must follow user instructions while reasoning jointly about objects, camera viewpoints, and robot actions in the 3D physical world. Recent vision-language-action models (VLAs) and video world-action models (WAMs) inherit strong semantic and temporal priors from large-scale foundation models. However, these models operate primarily in 2D image space, leaving the 3D geometry essential for contact-rich manipulation only implicitly represented. In this paper, we introduce the **Geometric Action Model (GAM)**, which directly repurposes a pretrained geometric foundation model (GFM) for policy learning. We show that the GFM's rich visual representations and geometric priors provide an effective foundation for manipulation. However, a GFM alone does not natively support future prediction or action generation. GAM bridges this gap by splitting the GFM at an intermediate layer and inserting a causal future predictor between its shallow and deep blocks. The shallow blocks encode observations into geometric features, while the predictor integrates language instructions, proprioception, and action history to forecast future geometric features and action representations. These predictions are then processed by the remaining pretrained GFM blocks to reconstruct future geometry and decode robot actions, with future-feature and future-depth losses explicitly supervising geometric forecasting. This design unifies geometric perception, future prediction, and action decoding within a shared pretrained backbone. Ablations show complementary benefits from geometric initialization and future-prediction supervision for out-of-distribution robustness. Across diverse simulation and real-robot benchmarks, GAM achieves competitive success

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

rates while requiring smaller model size and lower inference latency than representative foundation-model-scale baselines.

1 INTRODUCTION

A long-standing goal in robotics is to build broadly capable robot manipulation policies that follow natural-language instructions across diverse objects and scenes (Kim et al., 2024; 2025; 2026; Black et al., 2024). Such policies must not only recognize objects and parse instructions, but also reason about how the physical world evolves under their actions. This requires a unified understanding of language, visual appearance, scene geometry, robot state, and physical dynamics.

Recent progress has increasingly relied on large-scale foundation models as pretrained policy substrates. Vision-language-action models (VLAs) build on vision-language models and map visual and linguistic tokens to robot actions (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Video world-action models (WAMs) instead leverage pretrained video generation models to jointly predict future frames and actions (Kim et al., 2026; Pai et al., 2025). Despite their impressive language-conditioned manipulation capabilities, these models operate primarily on image-space representations: depth, scale, and other 3D structure remain implicit in visual cues and must be inferred by the policy. This can limit generalization under spatial distribution shifts, particularly changes in camera viewpoint (Fei et al., 2025).

Recent work addresses this limitation by incorporating 3D geometry into robot policy learning. One line learns directly from explicit 3D observations such as point clouds (Ze et al., 2024; Huang et al., 2026), demonstrating the value of geometry for generalization but often requiring task-specific encoders trained from scratch. With the emergence of geometric foundation models (GFMs) (Lin et al., 2025; Wang et al., 2025a; 2026), other approaches transfer pretrained geometric priors into robot policies by aligning selected GFM features with VLA backbones (Li et al., 2025; Sun et al., 2026), following representation alignment (Yu et al., 2024), or attaching lightweight action heads to final GFM features (Qian et al., 2025; Ge et al., 2025). However, these methods primarily use the GFM as a perceptual front end, leaving future prediction and action generation to separate policy modules.

In this work, we propose the **Geometric Action Model (GAM)**, which repurposes a pretrained GFM as a shared policy substrate coupling geometric perception, future prediction, and action decoding. Rather than extracting GFM features and passing them to a separate policy module, GAM performs future prediction within the GFM representation space and continues computation through its pretrained deep layers. Specifically, we split the GFM at an intermediate layer: the shallow layers encode observations, while an inserted causal future predictor integrates language instructions, proprioception, and action history to forecast future geometric features and action representations. These predictions are processed by the remaining pretrained GFM blocks to reconstruct future geometry and decode robot actions, allowing the geometric backbone to participate directly in control. Future prediction losses explicitly supervise this predictive pathway, coupling policy learning with geometric forecasting. Figure 2 illustrates the distinction between existing paradigms and GAM.

Across diverse simulation (Liu et al., 2023; Fei et al., 2025; Nasiriany et al., 2024) and real-robot benchmarks, GAM matches or exceeds the success rates of representative foundation-model-scale VLA and WAM baselines with a substantially smaller model, lower inference latency, and higher robustness to camera viewpoint changes. Our contributions are as follows:

- We introduce the **Geometric Action Model (GAM)**, a language-conditioned manipulation policy that uses a pretrained geometric foundation model as its primary policy backbone rather than as an auxiliary perceptual encoder.
- We combine intermediate-layer causal future prediction with future-prediction supervision, routing predicted geometric and action representations through the remaining pretrained GFM layers to decode future geometry and robot actions.
- We demonstrate strong manipulation performance and robustness across simulation and real-robot benchmarks with smaller model size and lower inference latency than representative foundation-model-scale baselines.

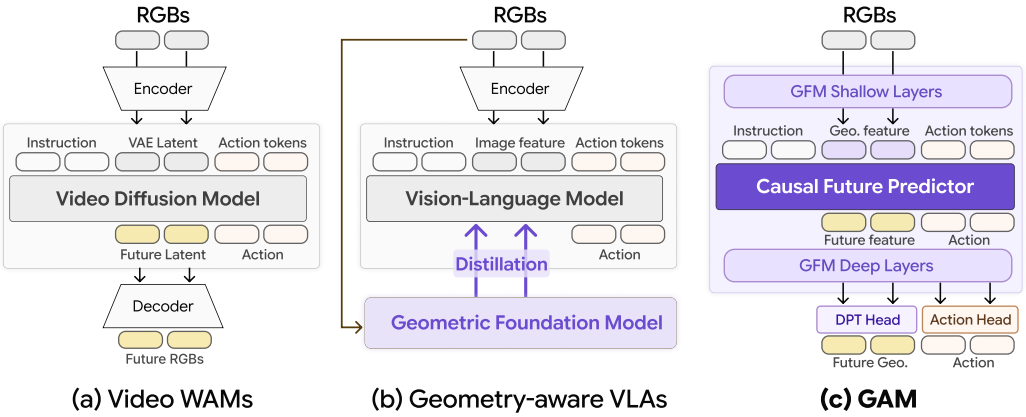


Figure 2: **(a) Video WAMs** (Kim et al., 2026; Pai et al., 2025; Ye et al., 2026b) operate in video latent space, predicting future latents and actions via video diffusion. **(b) Geometry-aware VLAs** (Li et al., 2025; Sun et al., 2026) predict actions using a VLA with passive feature distillation from an external GFM. **(c) GAM (ours)** unifies perception, geometry prediction, and action decoding by inserting a causal future predictor inside a single GFM.

2 Related Work

Vision-Language-Action Models. Vision-language-action models (VLAs) adapt pretrained vision-language models (VLMs) into robot policies. Early methods autoregressively decode discrete action tokens from fine-tuned VLMs (Zitkovich et al., 2023; Kim et al., 2024). Subsequent work explores parallel decoding (Kim et al., 2025), flow-matching action experts (Black et al., 2024; Physical Intelligence et al., 2025; Shi et al., 2025), diffusion-based action heads (Liu et al., 2025), frequency-domain action tokenization (Pertsch et al., 2025), and compact open-source backbones (Hung et al., 2025; Bai et al., 2025). The paradigm has also expanded to generalist models for humanoid and embodied control (Bjorck et al., 2025; Team et al., 2025; Cheang et al., 2025; Yang et al., 2025), spatially grounded VLA variants (Qu et al., 2025), and improved training and post-training procedures (Bu et al., 2025; Tan et al., 2025; Cen et al., 2025). Despite their strong semantic and open-vocabulary priors, most VLAs operate primarily on image-space observations and 2D-derived latent representations, leaving metric geometry implicit. Recent methods inject geometric priors through GFM-based representation alignment or geometry-aware encoders (Li et al., 2025; Sun et al., 2026; Abouzeid et al., 2025). These approaches use the GFM as a source of perceptual features or supervision, rather than processing predicted future representations through its pretrained layers.

World Action Models for Robot Manipulation. World action models (WAMs) couple robot control with the prediction of future observations or latent states. One line fine-tunes large pretrained video generation models (Agarwal et al., 2025) to jointly predict future frames and actions (Kim et al., 2026; Pai et al., 2025; Ye et al., 2026b). Recent studies suggest that much of the benefit may arise from future prediction as a training-time supervisory signal rather than from explicitly generating futures at deployment (Ye et al., 2026b; Yuan et al., 2026). A second line keeps the visual backbone frozen and learns a temporal predictor in its feature space for model-predictive control (Zhou et al., 2024; Assran et al., 2025), while related work uses implicit future-latent alignment as an auxiliary objective (Zheng et al., 2025). These approaches provide strong temporal priors but remain grounded in image-space representations that do not explicitly preserve depth, metric scale, or viewpoint-consistent 3D structure. GAM similarly couples future prediction with policy learning, but performs prediction within a GFM representation space and routes predicted geometric and action representations through the remaining pretrained geometric layers.

Geometric Foundation Models for Manipulation. GFMs (Wang et al., 2025a; Lin et al., 2025; Wang et al., 2025b; Keetha et al., 2025) infer dense 3D structure from images and have recently been adopted as perceptual backbones for robot policies. Early approaches incorporate GFMs through frozen feature alignment (Li et al., 2025; Sun et al., 2026), encoder replacement or lightweight action heads over final GFM features (Abouzeid et al., 2025; Qian et al., 2025; Ge et al., 2025), and point-cloud fusion (Sun et al., 2025). More recent concurrent work couples GFMs more tightly with control. Song et al. (2026) uses VGGT as a multimodal backbone and predicts action chunks with

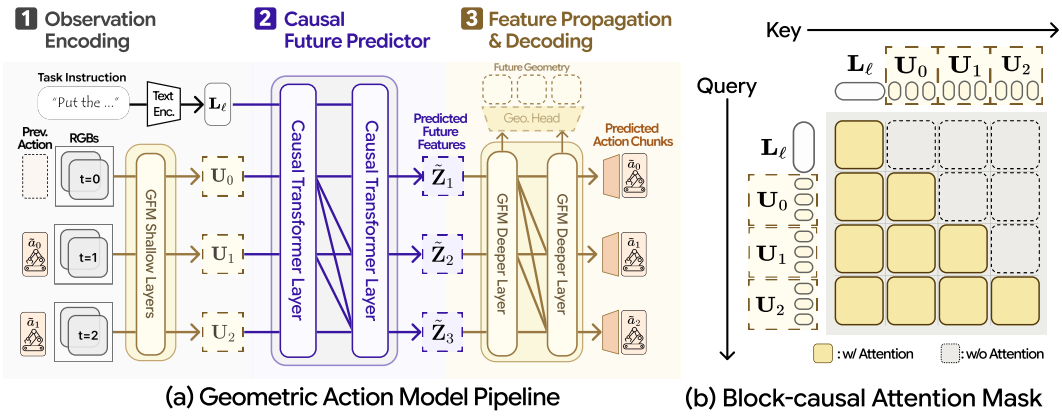


Figure 3: **Main architecture of GAM.**

a separate action decoder progressively modulated by hierarchical VGGT features, while auxiliary heads supervise current-frame depth and camera parameters. Xu et al. (2026) extracts GFM features and fuses them with semantic and proprioceptive features to condition a separate diffusion decoder that jointly denoises future action chunks and a future 3D latent. GAM instead performs temporal prediction within the GFM: a causal predictor inserted at an intermediate layer jointly forecasts future geometric and action representations, which are then processed by the remaining pretrained GFM blocks for future prediction and action decoding.

3 Preliminaries: Geometric Foundation Models

A geometric foundation model (Lin et al., 2025) (GFM) is a feed-forward transformer that maps one or more RGB images to dense 3D geometry. Given a sequence of V views $\mathcal{I} = \{I_v\}_{v=1}^{V}$ with $I_v \in \mathbb{R}^{3\times h\times w}$, a GFM produces per-pixel depth $D_v \in \mathbb{R}^{h\times w}$ or 3D point maps $P_v \in \mathbb{R}^{3\times h\times w}$ in a shared world frame, and per-view camera intrinsics and extrinsics $(K_v, \xi_v) \in \mathbb{R}^{3\times 3} \times SE(3)$ via auxiliary heads. Specifically, each view I_v is partitioned into P non-overlapping patches of size $p \times p$ and projected by a patch embedding into a per-view token sequence:

$$\mathbf{z}_v^{(0)} = \left[\mathbf{c}_v,\ \mathbf{x}_v^1, \ldots, \mathbf{x}_v^P\right] \in \mathbb{R}^{(1+P)\times d}, \tag{1}$$

where $\mathbf{c}_v$ is a per-view camera token, $\{\mathbf{x}_v^j\}_{j=1}^{P}$ are patch tokens, and d is the hidden dimension. The full input sequence concatenated across views is $\mathbf{Z}^{(0)} = [\mathbf{z}_1^{(0)}, \ldots, \mathbf{z}_V^{(0)}] \in \mathbb{R}^{V(1+P)\times d}$.

These tokens are processed by a stack of M transformer blocks $\{f^{(m)}\}_{m=1}^{M}$ employing one of two attention modes. Frame-wise attention $f_{\text{frame}}^{(m)}$ operates on each view's tokens independently, attending over the $(1+P)$ tokens of a single image. Global attention $f_{\text{global}}^{(m)}$ operates jointly over all $V(1+P)$ tokens, fusing information across viewpoints. The hidden state at the m-th transformer block evolves as follows:

$$\mathbf{Z}^{(m)} = f^{(m)}\big(\mathbf{Z}^{(m-1)}\big), \quad f^{(m)} \in \{f_{\text{frame}}^{(m)}, f_{\text{global}}^{(m)}\}. \tag{2}$$

After the transformer blocks, dense geometry is decoded from multiple intermediate hidden states $\mathbf{Z}^{(m^*)}$, where m^* is one of the selected layers in $\mathcal{S} = \{m_1, m_2, m_3, m_4\}$. The extracted multi-layer hidden states are then fed into the DPT (Ranftl et al., 2021) head to estimate per-pixel geometry.

4 GAM: Geometric Action Model

Problem Formulation. We consider language-conditioned robot manipulation. At each timestep t, the robot receives a multi-view RGB observation $o_t = \{I_{v,t}\}_{v=1}^{V}$ from V cameras, a proprioceptive state $s_t \in \mathbb{R}^{d_s}$ describing the robot's joint configuration and end-effector pose, and a natural-language task instruction ℓ that is held constant throughout an episode. The policy π_θ must produce an action chunk $\hat{a}_t \in \mathbb{R}^{C\times d_a}$ of length C, encoding the next C delta-pose or joint commands to be

executed open-loop before the next observation is acquired. We learn a policy:

$$\pi_\theta\colon \big(\{o_{t-H+1},\dots,o_t\},\{s_{t-H+1},\dots,s_t\},\{a_{t-H},\dots,a_{t-1}\},\ell\big) \mapsto \hat{a}_t \tag{3}$$

from a dataset of N expert demonstrations $\mathcal{D} = \{(\tau_i,\ell_i)\}_{i=1}^N$, where each trajectory $\tau_i = (o_t,s_t,a_t)_{t=1}^{T_i}$ pairs a sequence of observations, states, and executed action chunks with a fixed instruction ℓ_i. The policy conditions on a context window of H recent timesteps.

Overview. We transform a pretrained geometric foundation model (GFM) into a language-conditioned manipulation policy by splitting it at an intermediate layer and inserting a causal temporal predictor between its shallow and deep blocks. Rather than extracting GFM features and passing them to a separate policy module, GAM performs temporal prediction within the GFM representation space and routes the predicted representations through its remaining pretrained layers.

Concretely, GAM operates in three sequential stages. First, the **observation encoder** (§4.1) repurposes the shallow GFM layers to encode multi-view observations into latent geometric features. Next, the **causal future predictor** (§4.2), inserted at the split layer, integrates these features with language instructions, proprioception, and action history to predict future geometric and action representations. Finally, during **feature propagation and decoding** (§4.3), the predicted representations are processed by the remaining pretrained GFM blocks to decode future geometry and the action chunk $\hat{a}_t$. Figure 3(a) illustrates the overall architecture.

4.1 Observation Encoder

We first reuse the shallow layers of the pretrained GFM as the observation encoder. Let L_s denote the split layer where the causal future predictor is inserted. The original GFM transformer stack is then decomposed into an encoder and a decoder:

$$E_{\le L_s} = f^{(L_s)}\circ\cdots\circ f^{(1)}, \qquad D_{>L_s} = f^{(M)}\circ\cdots\circ f^{(L_s+1)}. \tag{4}$$

Here, the choice of L_s is important because L_s must be deep enough to extract sufficiently rich visual features from the raw observations, yet shallower than the earliest layer used in the DPT head $L_s < m_1$, so that predicted future states can be decoded into future geometries by the DPT head.

After defining this split layer L_s, for each timestep t' in the context window, we tokenize the multi-view RGB observation $o_{t'} = \{I_{v,t'}\}_{v=1}^V$ using the original GFM patch embedding. This produces the initial multi-view token sequence:

$$\mathbf{Z}_{t'}^{(0)} = \big[\mathbf{z}_{1,t'}^{(0)},\dots,\mathbf{z}_{V,t'}^{(0)}\big] \in \mathbb{R}^{V(1+P)\times d}, \tag{5}$$

where each view contributes one camera token and P patch tokens. The observation encoder maps these tokens to the split-layer representation $\mathbf{Z}_{t'}^{(L_s)}$. By applying this encoding independently to each timestep in the context window, the output of this stage is a sequence of per-timestep geometric latent states $\{\mathbf{Z}_{t-H+1}^{(L_s)},\dots,\mathbf{Z}_t^{(L_s)}\}$.

4.2 Causal Future Predictor

After the observation encoder, GAM performs temporal prediction directly at the split layer L_s, forecasting the next latent geometric state from current and past observations while conditioning on the task instruction, proprioception, and action history. To this end, we insert a causal predictor g_ϕ between the shallow encoder $E_{\le L_s}$ and the deep decoder $D_{>L_s}$. For each timestep t' in the context window, the encoder provides latent tokens $\mathbf{Z}_{t'}^{(L_s)}$, and we embed the proprioceptive state $s_{t'}$ and previous action $a_{t'-1}$ as tokens:

$$\mathbf{p}_{t'} = \psi_s(s_{t'}), \qquad \mathbf{q}_{t'} = \psi_a(a_{t'-1}), \tag{6}$$

with ψ_s,ψ_a are lightweight projection layers, and we encode the instruction ℓ into language tokens $\mathbf{L}_\ell$ with a pretrained text encoder (Raffel et al., 2020). We then form a per-timestep token block by concatenating the encoded GFM tokens with the proprioception and action-history tokens $\mathbf{U}_{t'} = [\mathbf{p}_{t'};\mathbf{q}_{t'};\mathbf{Z}_{t'}^{(L_s)}]$. The full input to the causal future predictor is $\mathbf{X} = [\mathbf{L}_\ell;\mathbf{U}_{t-H+1};\dots;\mathbf{U}_t]$.

The combined sequence $\mathbf{X}$ is then processed through block-causal self-attention (Wang et al., 2025b), ensuring the model incorporates past and present contexts without future leakage, as illustrated in Figure 3 (b). At the final layer of the predictor g_ϕ, we read off the predictions from

their respective sequence slots. Specifically, the hidden states corresponding to the geometry slots forecast the latent geometric tokens of the future frame, denoted as $\tilde{\mathbf{Z}}^{(L_s)}_{t'+1}$. Concurrently, the hidden state of the designated previous-action slot is projected to produce a predicted next action token $\tilde{\mathbf{a}}_{t'} \in \mathbb{R}^d$, in direct analogy to next-token prediction in a causal language model. By jointly forecasting action and geometric latents in this layer, we ensure that action tightly interacts with spatial representations.

This design of introducing a causal transformer predictor g_ϕ allows the pretrained GFM to acquire language-conditioned future prediction with *minimal* architectural modification. The inserted g_ϕ is the only new module; it learns to fuse language, proprioception, and action history with GFM latent features. The resulting predictions, $\widetilde{\mathbf{Z}}^{(L_s)}_{t'+1}$ and $\tilde{\mathbf{a}}_{t'}$, are then passed to the remaining GFM blocks for joint geometry and action decoding.

4.3 Feature Propagation and Action Decoding

Following the causal future predictor, the single action token $\tilde{\mathbf{a}}_{t'}$ is replicated V times to form a set of per-view action tokens $\{\tilde{\mathbf{a}}_{v,t'}\}_{v=1}^{V}$, where $\tilde{\mathbf{a}}_{v,t'} = \tilde{\mathbf{a}}_{t'}$. Concatenated with the geometry tokens, they are fed through the remaining GFM blocks $D_{>L_s}$. We perform this *feature propagation* by appending each view's corresponding action token $\tilde{\mathbf{a}}_{v,t'}$ directly to its geometry token sequence for each timestep:

$$\tilde{\mathbf{Z}}^{(M)}_{t'+1} = \big(f^{(M)} \circ \cdots \circ f^{(L_s+1)}\big)\Big(\Big[\big[\tilde{\mathbf{Z}}^{(L_s)}_{1,t'+1};\, \tilde{\mathbf{a}}_{1,t'}\big], \ldots, \big[\tilde{\mathbf{Z}}^{(L_s)}_{V,t'+1};\, \tilde{\mathbf{a}}_{V,t'}\big]\Big]\Big). \tag{7}$$

To prevent future leakage, we extend the predictor's causal mask strategy to the GFM's remaining global attention layers ($f^{(m)}_{\text{global}}$).

Finally, the propagated features are decoded by two heads. The lightweight action head h_{act} applies mean pooling to the action tokens over the viewpoints for each timestep and maps the pooled representation to the executable action chunk $\hat{a}_{t'}$, while the original GFM depth head h_{depth} decodes geometry tokens into action-aligned future depth maps (examples in Appendix B.2). The GFM's deep blocks, originally pretrained to decode shallow features into 3D geometry, are thus repurposed here as the decoder of predicted future features.

4.4 Training and Inference

The policy is trained end-to-end by minimizing a multi-task objective over action execution, future feature prediction, and geometric decoding:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}}, \tag{8}$$

where the λ factors balance each term and $\mathcal{H} = \{t - H + 1, \ldots, t\}$ is the context window. The *action* loss $\mathcal{L}_{\text{act}}$ is an ℓ_1 regression (Kim et al., 2025) between the decoded action chunk $\hat{a}_{t'}$ and the expert action $a_{t'}$ over all $t' \in \mathcal{H}$. The *future-feature* loss $\mathcal{L}_{\text{feat}}$ anchors the predictor g_ϕ to temporal geometric transitions by aligning predicted future tokens $\tilde{\mathbf{Z}}^{(L_s)}_{t'+1}$ with the actual next frame $\mathbf{Z}^{(L_s)}_{t'+1}$ extracted from frozen GFM:

$$\mathcal{L}_{\text{feat}} = \sum_{t' \in \mathcal{H}} \left\| \tilde{\mathbf{Z}}^{(L_s)}_{t'+1} - \mathbf{Z}^{(L_s)}_{t'+1} \right\|_1. \tag{9}$$

The *future-depth* loss $\mathcal{L}_{\text{depth}}$ grounds the predicted future in valid 3D structure by supervising the decoded depth $\tilde{D}_{t'+1} = h_{\text{depth}}(\tilde{\mathbf{Z}}^{(m^*)}_{t'+1})$ against ground-truth future depth $D_{t'+1}$, adopting the scale-invariant and gradient-matching penalties of the GFM (Lin et al., 2025; Wang et al., 2026).

5 Experiments

5.1 Implementation Details

We use DA3-Giant (Lin et al., 2025; Lu et al., 2026) as our GFM. We insert a 12-layer causal predictor with width $d_g = 1024$ at layer $L_s = 12$, where alternating attention begins. For the task instruction, we extract language tokens using a frozen T5-Base encoder (Raffel et al., 2020).

Table 1: **Evaluation results on LIBERO and LIBERO-Plus.** Success rates are reported in % with absolute performance drops from LIBERO to LIBERO-Plus shown in parentheses. Color highlights denote the top three performing methods within each column: first , second , and third .

Method	Size	Orig.	Plus	Cam.	Robot	Lang.	Light	BG	Noise	Layout
VLAs										
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	3.3B	96.9	84.6 (↓12.3)	72.0	76.6	86.5	96.1	95.2	86.7	86.0
OpenVLA-OFT (Kim et al., 2025)	7B	97.1	69.6 (↓27.5)	56.4	31.9	79.5	88.7	93.3	75.8	74.3
π_0 (Black et al., 2024)	3.3B	91.3	69.3 (↓22.0)	61.0	40.8	63.7	89.3	84.1	80.1	75.9
OpenVLA (Kim et al., 2024)	7B	76.5	15.6 (↓60.9)	0.8	3.5	23.0	8.1	34.8	15.2	28.5
WAMs										
Cosmos-Policy (Kim et al., 2026)	2B	98.5	82.4 (↓16.1)	73.4	63.3	89.3	98.9	83.5	89.3	84.0
Fast-WAM (Yuan et al., 2026)	6B	97.6	50.0 (↓47.5)	16.4	44.5	68.9	78.2	53.7	37.7	60.7
WorldVLA (Cen et al., 2025)	7B	79.1	25.0 (↓54.1)	0.1	27.9	41.6	43.7	17.1	11.0	38.0
Geometry-aware VLAs										
Spatial Forcing (Li et al., 2025)	7B	95.1	69.2 (↓25.9)	57.1	36.6	82.5	92.0	87.3	64.8	76.5
ROCKET (Sun et al., 2026)	7B	96.3	69.1 (↓27.2)	49.7	39.6	81.8	94.0	91.4	62.6	78.8
GWM-VLA (Zhao et al., 2026)	>2B	97.1	76.9 (↓20.2)	57.9	54.5	89.8	95.4	90.8	72.5	77.1
GAM										
GAM (Ours)	1.4B	97.4	85.5 (↓11.9)	83.1	70.0	84.8	97.2	94.3	95.3	79.1

The policy uses a context horizon of $H = 4$ for pre-training and $H = 1$ for post-training and predicts $C = 8$ step action chunks in a $d_a = 7$ end-effector action space from $d_s = 7$ proprioceptive states. We pretrain GAM on 784K single-arm robot trajectories from RoboCasa365 (Nasiriany et al., 2026), MimicGen (Mandlekar et al., 2023), and Open-X Embodiment (Collaboration et al., 2023), then post-train it on each benchmark. We optimize with AdamW using a constant learning rate, freeze layers before L_s and the depth head, and supervise depth with simulator ground truth and teacher DA3-generated pseudo-GT depth. We set $\lambda_{\text{act}} = 3$, $\lambda_{\text{feat}} = 1$, and $\lambda_{\text{depth}} = 3$. Further details are provided in the appendix.

5.2 Experimental Setup

Simulation Benchmarks. We evaluate policy performance and robustness across three complementary simulation benchmarks. We first train and evaluate on **LIBERO** (Liu et al., 2023), a lifelong single-arm manipulation benchmark spanning diverse spatial layouts, object identities, and task goals. Because performance on standard LIBERO is increasingly saturated, we further assess out-of-distribution robustness by evaluating the same trained policies zero-shot on **LIBERO-Plus** (Fei et al., 2025), which introduces controlled perturbations to camera viewpoint, robot initialization, lighting, background, and other environmental factors. Finally, we separately train and evaluate on **RoboCasa-Kitchen** (Nasiriany et al., 2024), a 24-task benchmark set in realistic and diverse kitchen environments. Following the benchmark protocol of Nasiriany et al. (2024), we evaluate each task over 50 episodes across five scenes, with ten episodes per scene. We use 300 training demonstrations per task. Its tasks span pick-and-place, articulated-object interaction, appliance control, and coffee-related manipulation, providing a complementary test of multi-view and longer-horizon manipulation.

Real-Robot Setup. We train on four manipulation tasks (169–284 demonstrations each) using wrist-mounted and third-person cameras, adhering to the simulation protocol. Since ground-truth geometry is unavailable in the real world, target future depth maps are obtained as pseudo-labels directly from the pretrained backbone GFM. We evaluate robustness via 40 trials per task, which adds up to 160 trials in total, divided equally between nominal setups and perturbed environments, specifically varying external camera positions. Unlike simple single-step tasks, each task consists of a complex compound action sequence with an increased number of atomic actions. See the appendix for robot environment with full task and evaluation details.

Baselines. We compare GAM against representative baselines from three families discussed in §2: **VLAs** (Kim et al., 2024; 2025; Black et al., 2024; Physical Intelligence et al., 2025), **WAMs** (Kim et al., 2026; Cen et al., 2025; Yuan et al., 2026), and **geometry-aware VLAs** (Li et al., 2025; Sun et al., 2026). For the real-robot setup, we compare against $\pi_{0.5}$ (Physical Intelligence et al., 2025) and Spatial Forcing (Li et al., 2025).

Table 2: **Pretraining ablation.** Average SR (%) on LIBERO. All variants are pretrained with the same optimization budget. We define $\mathcal{L}_{\text{future}} = \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{depth}}$. The last row supervises current-frame depth.

Backbone	Causal Future Predictor	$\mathcal{L}_{\text{future}}$	Orig. SR (%)	Plus SR (%)
DINO-Initialized	✓	✓	94.9	79.3
	✓	✗	85.2	68.1
DA3-Pretrained	✓	✓	**97.4**	**85.5**
	✓	✗	89.8	72.8
	✗	✗	91.8	65.8

Table 3: **Average success rates across 24 manipulation tasks from RoboCasa-Kitchen .**

Method	Avg. SR (%)
DreamGen (Jang et al., 2025)	57.6
π_0 (Black et al., 2024)	62.5
GR00T-N1.5 (Bjorck et al., 2025)	64.1
HAMLET (Koo et al., 2026)	66.4
FLARE (Zheng et al., 2025)	66.4
Cosmos Policy (Kim et al., 2026)	67.1
GAM (Ours)	**69.4**

5.3 Main Results

LIBERO and RoboCasa-Kitchen Results. As shown in Table 1 and Table 3, GAM achieves competitive success rates on standard LIBERO and RoboCasa-Kitchen despite having the smallest model. Because standard LIBERO is nearly saturated, we focus on LIBERO-Plus, where GAM achieves the highest overall success rate among the compared methods. It is also noteworthy that the largest improvement is observed in the **camera-perturbation setting** (83.1% vs. 73.4% for the strongest baseline), highlighting one of the advantages of the integration of geometric priors from GFM, and of learning future geometric representations through our causal future predictor and future losses. While existing geometry-aware VLAs only partially exploit GFM representations, GAM embeds the GFM throughout its entire predictive pathway to yield a deeply geometry-aware policy.

Real-world Results. To examine whether the gains observed in simulation transfer to physical execution, we additionally evaluate GAM in a real-world setting. Figure 4 shows that GAM achieves the highest average success in both settings. In particular, our model remains robust under out-of-domain conditions (the camera-perturbation setting) where other baselines struggle. These results demonstrate that GAM generalizes to the real-world domain and is robust under perturbations.

Table 4: **Component ablation on LIBERO-Object.**

Pretrain	$\mathcal{L}_{\text{depth}}$	$\mathcal{L}_{\text{feat}}$	Orig. SR (%)	Plus SR (%)
✓	✓	✓	99.0	**90.6**
✓	✗	✓	98.4	89.0
✓	✓	✗	**99.6**	89.7
✓	✗	✗	98.6	89.5
✗	✓	✓	98.4	73.4
✗	✗	✓	95.2	66.5
✗	✓	✗	96.4	80.0
✗	✗	✗	93.6	50.0

Table 5: **Matched and deployment latency.**

Method	Size	Matched	Deploy.
OpenVLA-OFT Kim et al. (2025)	7B	70.1 ms	–
$\pi_{0.5}$ Physical Intelligence et al. (2025)	3.3B	29.2 ms	–
Cosmos Policy Kim et al. (2026)	2B	382.4 ms	–
GAM (Ours)	**1.4B**	**17.5 ms**	**6.9 ms**

Table 6: **Horizon ablation.**

H	Orig. SR (%)	Plus SR (%)
1	**99.0**	**90.6**
2	97.2	84.4
4	98.2	85.1

Table 7: **Split-layer ablation.**

Split layer L_s	Orig. (%)	Plus (%)
0	5.4	1.8
12	**99.6**	**70.1**
19	95.6	63.4
27	1.2	1.6
33	0.0	0.0
39	0.0	0.0

5.4 Ablation Study

Pretraining Ablation. Table 2 compares the backbone prior, future-prediction supervision, and causal future predictor under the same pretraining budget and downstream protocol. We adopt the same DINOv2 backbone as DA3 for DINO-initialized. Implementation details of the DINO-initialized control are provided in the appendix. DA3 variants consistently outperform DINO variants with and without $\mathcal{L}_{\text{future}}$, with larger gains on LIBERO-Plus, confirming the benefit of pretrained GFM priors. Future supervision also improves both backbones, indicating that the geometric prior and predictive objective provide complementary benefits. Without the causal future predictor and future supervision, relying exclusively on current depth supervision results in the steepest performance drop on LIBERO-Plus, highlighting the importance of our design choices, causal future predictor and future losses.

Post-training Component Analysis. Table 4 summarizes ablation study of key post-training components on Object suite of LIBERO and LIBERO-Plus; further ablations are in Appendix C. Without robot pretraining, future-prediction losses provide strong geometric supervision and substantially

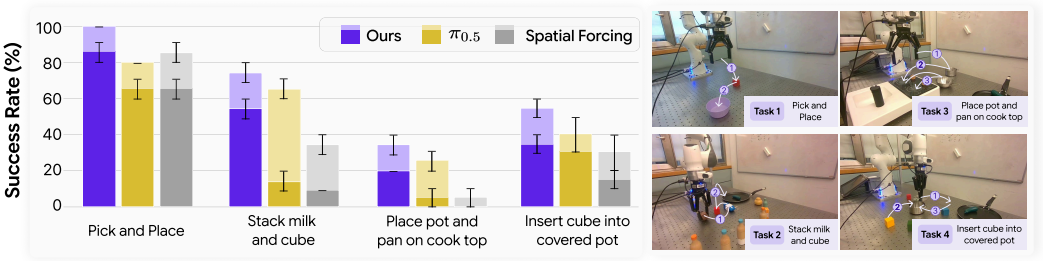


Figure 4: **Real-world robot tasks and results.** Each task is evaluated under both in-domain (Light bar) and out-of-domain (Dark bar) settings. Error bars indicate standard deviation. The illustration of each task is shown on the right.

improve robustness on LIBERO-Plus, although adding $\mathcal{L}_{\text{feat}}$ on top of $\mathcal{L}_{\text{depth}}$ lowers it from 80.0% to 73.4%. After robot pretraining, however, removing the same losses has little effect, indicating that future prediction contributes mainly during robot pretraining and serves as auxiliary supervision during downstream adaptation. As shown in Table 6, a single frame of observation history ($H = 1$) performs best; longer histories lower LIBERO-Plus success, consistent with prior findings on LIBERO (Ye et al., 2026a).

Split Layer L_s Selection. Table 7 evaluates where to insert the predictor by shifting L_s; unlike the other ablations, the predictor is re-initialized at every split, discarding its pretraining (details in the appendix). We exclude future-depth loss in this experiment because it is not equally applicable to all split layers, and to isolate the effect of $\mathcal{L}_{\text{feat}}$ itself. Our default choice of $L_s = 12$, where DA3 switches from frame-wise to alternating cross-view attention, achieves the best performance, and layer 19 remains competitive. Inserting the predictor too early ($L_s = 0$) or late ($L_s \in \{27, 33, 39\}$) fails in this setting. Since all variants start from the checkpoint pretrained with $L_s = 12$, this comparison favors the default split, and we treat it as a sensitivity check rather than as evidence that $L_s = 12$ is optimal.

5.5 Analysis

Inference Speed and Model Size. As shown in Table 5, GAM requires 17.5 ms per action-chunk prediction under the matched runtime setting, achieving a 21.9× speedup over Cosmos Policy. Our optimized deployment setting further reduces GAM's model-only latency to 6.9 ms. All methods are benchmarked under the same setup, with further details provided in the appendix. By utilizing single-pass prediction, GAM avoids the multi-step denoising of diffusion policies, achieving low latency while matching prior accuracy and robustness with only 1.4B parameters.

Robustness to Viewpoint and Scene Variation. Figure 5 further breaks down the camera-perturbation results by difficulty level of LIBERO-Plus (Fei et al., 2025). GAM matches the best baseline at the mildest level and outperforms all baselines from L2 to L5.

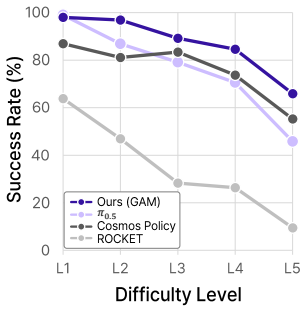


Figure 5: **Success rate vs. camera perturbation difficulty.** Other perturbation types are shown in Figure 9.

6 Conclusion and Limitation

We introduced the Geometric Action Model, which unifies future geometry and action prediction inside a single shared GFM. By inserting a causal transformer between the GFM's shallow and deep layers, GAM decodes actions and future geometries. Across extensive simulation and real-world benchmarks, GAM achieves competitive accuracy, faster inference, and strong out-of-distribution robustness to environmental perturbations. The framework also has limitations. Its language reasoning and commonsense capabilities are bounded by the frozen text encoder; integrating a large language model or an external reasoning module is a natural next step.

REFERENCES

Ali Abouzeid, Malak Mansour, Qinbo Sun, Zezhou Sun, and Dezhen Song. Geoaware-vla: Implicit geometry aware vision-language-action model. *arXiv preprint arXiv:2509.14117*, 2025.

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, et al. Gr-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

OX-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 1(2), 2023.

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

Shijia Ge, Yinxin Zhang, Shuzhao Xie, Weixiang Zhang, Mingcai Zhou, and Zhi Wang. Vggt-dp: Generalizable robot control via vision foundation models. *arXiv preprint arXiv:2509.18778*, 2025.

Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, volume 2026, pp. 101537–101558, 2026.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Jiahao Lu, Jiayi Xu, Wenbo Hu, Ruijie Zhu, Chengfeng Zhao, Sai-Kit Yeung, Ying Shan, and Yuan Liu. Track4world: Feedforward world-centric dense 3d tracking of all pixels. *arXiv preprint arXiv:2603.02573*, 2026.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. *arXiv preprint arXiv:2603.04356*, 2026.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.

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. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Quanhao Qian, Guoyang Zhao, Gongjie Zhang, Jiuniu Wang, Ran Xu, Junlong Gao, and Deli Zhao. Gp3: A 3d geometry-aware policy with multi-view images for robotic manipulation. *arXiv preprint arXiv:2509.15733*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12179–12188, 2021.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Zijian Song, Qichang Li, Jiawei Zhou, Zhenlong Yuan, Tianshui Chen, Liang Lin, and Guangrun Wang. Robotic manipulation is vision-to-geometry mapping ($f(v) \rightarrow g$): Vision-geometry backbones over language and video models. *arXiv preprint arXiv:2604.12908*, 2026.

Guoheng Sun, Tingting Du, Kaixi Feng, Chenxiang Luo, Xingguo Ding, Zheyu Shen, Ziyao Wang, Yexiao He, and Ang Li. Rocket: Residual-oriented multi-layer alignment for spatially-aware vision-language-action models. *arXiv preprint arXiv:2602.17951*, 2026.

Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

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

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025a.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Scalable permutation-equivariant visual geometry learning. *arXiv e-prints*, pp. arXiv–2507, 2025b.

Chongyang Xu, Haipeng Li, Shen Cheng, Jingyu Hu, Haoqiang Fan, Ziliang Feng, and Shuaicheng Liu. Action-geometry prediction with 3d geometric prior for bimanual manipulation. *arXiv preprint arXiv:2602.23814*, 2026.

Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, et al. Magma: A foundation model for multimodal ai agents. In *Proceedings of the computer vision and pattern recognition conference*, pp. 14203–14214, 2025.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026a.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

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*, 2026.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Yanping Zhao, Hang Yu, Yiwei Wang, Chen Ye, Siyu Tian, Di Zhang, Qingjun Wang, Qian Chen, Junqiao Zhao, and Guang Chen. Gwm-vla: Geometry-aware latent world modeling for vision-language-action learning. *arXiv preprint arXiv:2608.07619*, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

APPENDIX

This appendix provides experimental details, results, and analyses that complement the main paper.

- Section A describes the training data, implementation details, simulation and real-world evaluation settings, baseline settings, and inference benchmark protocol.
- Section B reports per-suite and per-difficulty LIBERO-Plus results, per-task LIBERO results, and future depth predictions.
- Section C provides additional ablations on where actions are decoded, the action head, and the attention visualizations.

A EXPERIMENTAL SETTINGS AND REPRODUCIBILITY DETAILS

GAM is trained in two stages, following standard practice for generalist robot policies (Kim et al., 2025; Black et al., 2024). The first stage jointly trains the predictor, action head, and the GFM backbone on a large mixture of single-arm robot data. The model was trained using 64 NVIDIA GH200 GPUs with a batch size of 1024, which takes approximately 96 hours. The second stage fine-tunes the same parameters on each benchmark's official training set before evaluation; in both stages, the GFM blocks before the split layer and the depth head stay frozen. The second stage on simulation benchmark was trained using 8 NVIDIA GH200 GPUs with batch size of 160, which takes $\sim$ 48 hours.

A.1 PRE-TRAINING DETAILS

We pre-train GAM on a weighted mixture of Open-X Embodiment (Collaboration et al., 2023) (OXE), MimicGen (Mandlekar et al., 2023), and RoboCasa365 (Nasiriany et al., 2026). We exclude the 24 RoboCasa-Kitchen evaluation tasks from the RoboCasa365 subset used for robot pretraining. OXE provides broad real-robot coverage across multiple embodiments and manipulation domains, while MimicGen and RoboCasa365 provide simulation demonstrations with clean geometric supervision. The sampling ratios are 72%, 18%, and 10% for OXE, MimicGen, and RoboCasa365, respectively. For future-depth supervision, we use teacher pseudo-depth for OXE and re-rendered simulator depth for MimicGen and RoboCasa365. Figure 6 shows the dataset mixture used for pre-training.

For OXE, we use the subset whose actions can be mapped to our common control interface and exclude datasets that are incompatible with our action space. For RoboCasa365, we use only the manipulation-task subset. Across all three sources, we keep the original task language provided by the datasets and do not synthesize additional instructions. The language encoder is kept frozen during pre-training.

All datasets are converted to a common observation and action format before training. Actions are represented as end-effector commands in the robot base frame and normalized using the 1st and 99th percentiles. Images are resized to 224. We use external and wrist-camera observations when both are available; for datasets without wrist-camera observations, we use only the external camera image.

A.2 SIMULATION EXPERIMENTS DETAILS

We adopt the LIBERO evaluation protocol established by OpenVLA (Kim et al., 2024) and OpenVLA-OFT (Kim et al., 2025). Specifically, we evaluate on the four standard LIBERO task suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, each consisting of 10 tasks. Following OpenVLA-OFT, we train on filtered LIBERO demonstrations by removing unsuccessful episodes and filtering idle/no-op frames, i.e., training samples with near-zero actions.

We report task execution success rate (SR, %) as our primary evaluation metric. For the original LIBERO benchmark, each task is evaluated over 50 randomized trials, resulting in 500 rollouts per suite. For LIBERO-Plus, we follow the official evaluation setting and use one rollout per perturbed task instance. All models are trained with a global batch size of 160 for up to 110k training steps, until convergence is achieved.

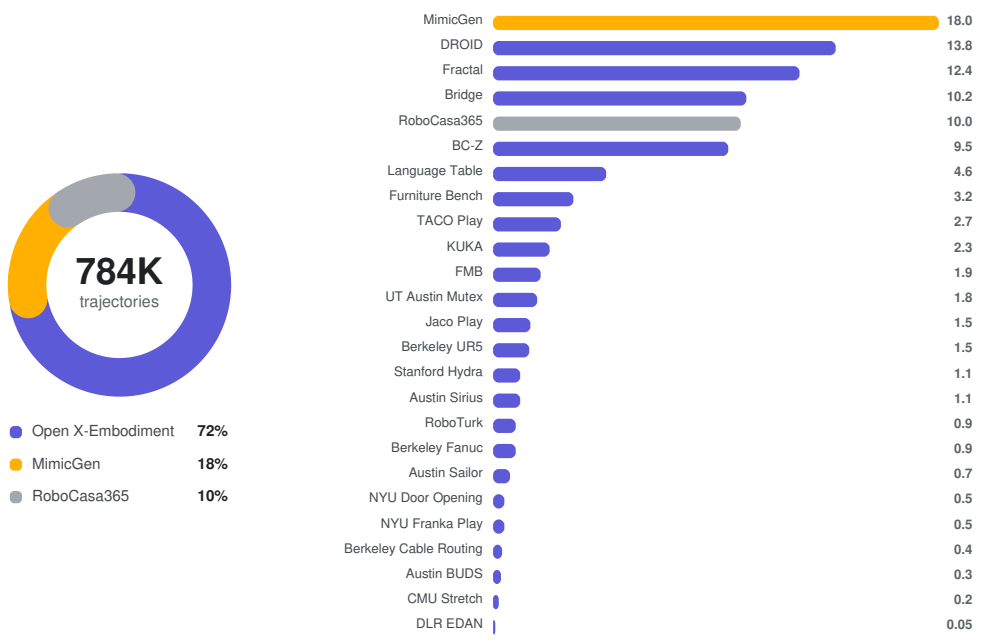


Figure 6: **Training Dataset Mixture.** We illustrate the dataset mixture utilized during pretraining, detailing the relative proportions of each constituent dataset. The pie chart shows the high-level source mixture, and the bar chart shows the percentage of each constituent dataset relative to the entire training corpus.

Table 8: **Hyperparameters for LIBERO, LIBERO-Plus and real-world experiments.**

hyperparameter	value
# GPUs	8 × NVIDIA GH200
learning rate (LR)	5.16e-5 backbone; 5.16e-4 action head and predictor
total batch size	160
input images	1 external camera image, 1 wrist-mounted camera image
input image size	224 x 224 px
use observation history	no (use single-step inputs)
action prediction horizon	8 steps
action execution horizon	8 steps (open-loop)
use proprio (robot state)	yes
# trainable parameters	983.2M total

A.3 Real-World Experiments Details

Figure 7 illustrates the experimental environment used for our real-world evaluations. Our hardware setup includes a wrist-mounted ZED Camera and an external RealSense camera providing a third-person perspective.

For training and evaluation, we defined four distinct tasks: Pick and place, Stack milk and cube, Place pot and pan on cooktop, and Insert cube into covered pot. We collected teleoperated demonstrations for each task: 284, 202, 184, and 169 demonstrations, respectively. Figure 8 shows the text instructions and corresponding visual illustrations for each task. All tasks were jointly trained within a unified dataset. The hardware specifications and hyperparameters used to train GAM, alongside the baselines ($\pi_{0.5}$ (Physical Intelligence et al., 2025) and Spatial Forcing (Li et al., 2025)), are de-

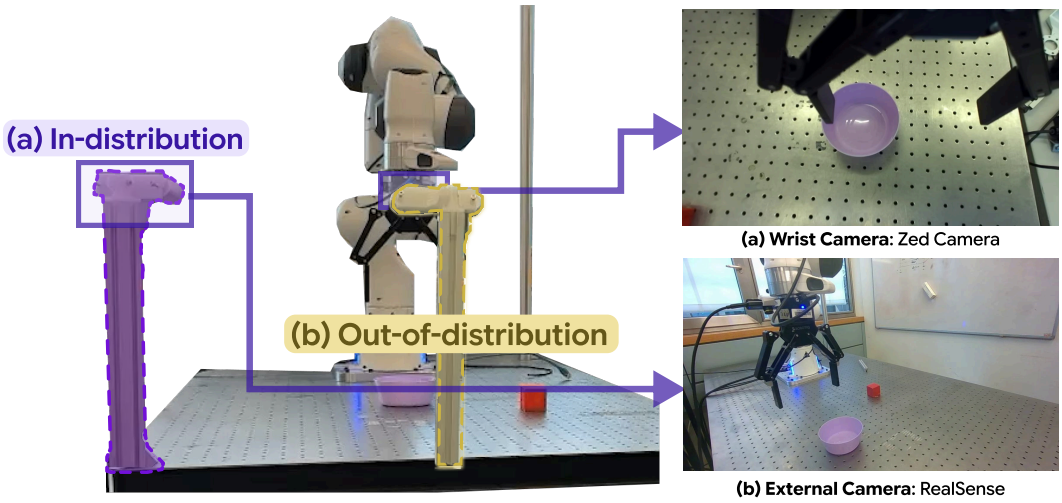


Figure 7: **Real-world experiments environment setup and ID vs. OOD Camera setup.**

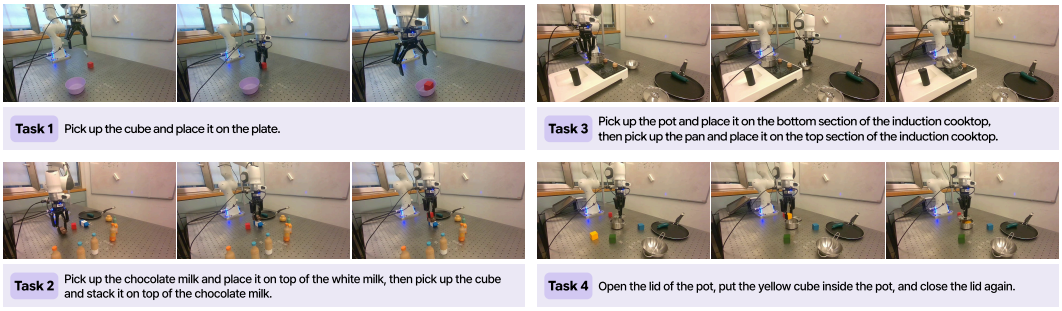


Figure 8: **Illustration of four real-world manipulation tasks.**

tailed in Tables 8, 9, and 10, respectively. All baseline models were trained until convergence using the default hyperparameters from the paper.

During evaluation, we measured the success rate of each task across 40 trials. Specifically, 20 trials were conducted under a normal setup (ID), while the remaining 20 trials evaluated robustness under an out-of-distribution (OOD) setup with camera perturbation. The camera perturbation was introduced by applying a translation of 85 cm and a rotation of 45° to the external camera shown in Figure 7. Figure 7 (left) compares the ID and OOD camera placements. This perturbation setup was kept consistent across all evaluations.

A.4 Ablation Study Details

DINOv2-initialized Control. We use the same DINOv2 backbone as DA3, matching DA3-Giant in size and initialize the depth head from scratch. We retain the same DA3 teacher for geometric supervision. During robot pretraining, we retain the standard $0.1\times$ learning-rate multiplier for the backbone and apply the same multiplier to the depth head, as training the newly initialized depth head at the base learning rate produced NaN losses. All other settings, including the backbone-freezing configuration, follow GAM.

Split-layer Selection. All variants in Table 7 start from the same robot-pretrained GAM checkpoint, with the causal future predictor re-initialized for each split. We freeze the GFM blocks preceding the selected split layer L_s and train the remaining blocks and predictor under the same post-training budget. We omit $\mathcal{L}_{\text{depth}}$ across all variants because depth supervision is not equally applicable at every insertion point, while retaining $\mathcal{L}_{\text{feat}}$ for future supervision. Evaluation is conducted on LIBERO-Object and LIBERO-Plus-Object.

Table 9: $\pi_{0.5}$ **hyperparameters.**

hyperparameter	value
# GPUs	8 x NVIDIA H100 GPU
learning rate (LR)	5e-5
total batch size	16
input images	1 external camera image, 1 wrist-mounted camera image
input image size	224 x 224 px
use observation history	no (use single-step inputs)
action prediction horizon	10 steps
action execution horizon	10 steps (open-loop)
use proprio (robot state)	yes
# trainable parameters	3.3B total
diffusion sampling algorithm	flow matching
number of integration steps	10

Table 10: **Spatial Forcing hyperparameters.**

hyperparameter	value
# GPUs	8 x NVIDIA H100 GPU
learning rate (LR)	2.5e-5
total batch size	16
input images	1 external camera image, 1 wrist-mounted camera image
input image size	224 x 224 px
use observation history	no (use single-step inputs)
action prediction horizon	10 steps
action execution horizon	10 steps (open-loop)
use proprio (robot state)	yes
# trainable parameters	853M total

Table 11: **Model-only inference latency on a single GH200 GPU.**

Policy	**Official PyTorch**	**bf16 precision**	**Torch Compile**	**CUDA Graphs**	**Model-only latency**
GAM (Ours)	✓	✓	✓	✗	17.5 ms
GAM (Ours)	✓	✓	✓	✓	**6.9 ms**
$\pi_{0.5}$	✓	✓	✓	✗	29.2 ms
OpenVLA-OFT	✓	✓	✓	✗	70.1 ms
Cosmos Policy	✓	✓	✓	✗	382.4 ms

A.5 Real-World Experiments Baseline Training Details

For the baseline methods of real-world experiments, we follow the training recipes and default hyperparameters from the corresponding papers (Li et al., 2025; Physical Intelligence et al., 2025), changing only the task data and camera streams to match our evaluation setup (Table 9 and 10). Both $\pi_{0.5}$ (Physical Intelligence et al., 2025) and Spatial Forcing (Li et al., 2025) use the same two RGB inputs as GAM, one external camera and one wrist-mounted camera, resized to 224×224. We keep each baseline's original inference protocol to preserve its intended deployment behavior. $\pi_{0.5}$ uses flow-matching action decoding with 10 integration steps, while Spatial Forcing adopts its feature alignment loss recipe.

A.6 Inference Latency Comparison Details

Table 11 reports the runtime configuration used for the inference-speed comparison in the main paper. All policies are evaluated on a single GH200 GPU with the same canonical observation input, bf16 precision, warmup and measurement protocol, and model-only latency metric, excluding model loading and input preprocessing.

We use the official PyTorch inference path for each baseline. For $\pi_{0.5}$, whose original implementation is based on JAX, we use the official PyTorch implementation. To separate common compiler and runtime effects from deployment-specific execution, we report a matched setting in which all policies use Torch Compile and none uses CUDA Graphs.

Under this setting, GAM requires 17.5 ms for a single feed-forward action prediction, compared to 29.2 ms for $\pi_{0.5}$, 70.1 ms for OpenVLA-OFT, and 382.4 ms for Cosmos Policy. GAM's deployment setting further uses CUDA Graphs over its static single-pass inference path, reducing latency to 6.9 ms. We report this number separately because CUDA Graphs were not applied to the baselines.

A.7 Model Size Breakdown

Table 12 details the parameter breakdown of the DA3-based GAM architecture. As reported, GAM uses a 1.4B-parameter model, making it substantially smaller than VLM-based and video-diffusion-based baselines such as $\pi_{0.5}$, OpenVLA-OFT, and Cosmos-Policy. These counts exclude the frozen T5-Base text encoder. This compact size comes from repurposing a pretrained geometric backbone (Lin et al., 2025) as the shared substrate for perception, future-geometry prediction, and action decoding, rather than attaching a large language model or video-generation model as the policy backbone.

Table 12: **Parameter breakdown of the DA3-based GAM model.**

Module	Parameters	Trainable?	Trainable parameters
backbone (ViT-Giant, 40 blocks)	1136.5M	blocks 13–39 trainable; blocks 0–12 frozen	≈765M
DPT head	50.1M	frozen	0
Causal Future Predictor	210.2M	trainable	210.2M
action head	8.0M	trainable	8.0M
total	≈1404.8M	–	≈983.2M

Of the full model, approximately 983.2M parameters are trainable. Most of these trainable parameters come from the later blocks of the ViT-Giant backbone, while the initial geometric layers and the DPT depth head remain frozen to preserve pretrained geometric structure. The Causal Future Predictor and lightweight action head are fully trainable and account for the remaining trainable parameters. This design allows GAM to adapt the geometric representation for control while keeping the overall model size below the larger foundation-model baselines in the main comparison.

B Additional Benchmark Results

B.1 Additional LIBERO and LIBERO-Plus Results

Table 13: **LIBERO-Plus robustness results across four task suites.** Success rates are reported in %. "Orig." denotes the results of LIBERO benchmark without any perturbation. Color highlights indicate the three highest distinct success rates within each suite and column, with ties sharing the same color: first , second , and third .

(a) Spatial

Method	Orig.	Cam.	Robot	Lang.	Light	BG	Noise	Layout	Total
GAM (Ours)	98.6	91.5	79.4	95.6	100.0	98.8	96.6	94.3	93.4
$\pi_{0.5}$	98.8	76.6	86.3	96.4	97.9	99.6	92.0	98.2	92.0
Cosmos-policy	98.1	83.5	59.7	96.4	99.7	85.3	91.7	95.1	87.3
Fast-WAM	98.2	14.4	44.0	69.5	87.3	69.8	35.0	60.5	54.4
OpenVLA-OFT	97.6	88.3	40.0	80.5	98.3	97.3	96.3	93.9	84.0
RIPT-VLA	97.5	85.4	38.0	99.7	99.7	100.0	92.0	92.3	85.8
π_0	96.8	70.7	49.1	67.9	92.8	95.0	87.7	94.0	78.6
π_0-Fast	96.4	87.2	26.9	84.2	37.0	97.7	93.2	95.5	74.4
UniVLA	96.5	1.1	52.6	83.9	96.6	90.7	15.7	69.5	55.5
NORA	92.2	4.3	50.9	63.8	66.8	65.5	12.5	84.6	47.6
WorldVLA	85.6	0.0	44.3	46.3	65.1	19.8	11.7	46.1	32.5
OpenVLA	84.7	0.0	3.7	27.7	12.3	50.4	12.0	40.7	19.4
OpenVLA + Spatial Forcing	90.8	71.8	42.3	81.0	88.4	85.7	80.9	87.5	76.4
OpenVLA + ROCKET	92.4	71.8	43.7	88.5	89.7	90.3	77.2	96.4	79.3

(b) Object

Method	Orig.	Cam.	Robot	Lang.	Light	BG	Noise	Layout	Total
GAM (Ours)	99.0	91.4	71.4	100.0	100.0	99.6	98.8	79.7	90.6
$\pi_{0.5}$	98.2	86.4	71.9	91.0	99.0	99.2	96.2	91.1	89.9
Cosmos-policy	100.0	88.6	61.6	94.1	99.7	96.8	97.4	86.4	88.3
Fast-WAM	100.0	25.3	64.1	96.3	97.9	77.8	63.7	73.0	71.2
OpenVLA-OFT	98.4	38.9	25.4	99.0	73.7	97.6	72.3	71.8	66.5
RIPT-VLA	97.5	37.9	26.4	80.8	85.9	99.2	68.0	70.1	64.3
π_0	98.8	80.1	31.9	75.4	94.3	85.9	87.9	76.2	74.7
π_0-Fast	96.8	72.0	27.6	71.5	71.0	95.2	93.1	84.5	72.7
UniVLA	96.8	0.0	42.2	86.9	25.6	81.5	10.4	27.3	36.7
NORA	95.4	0.5	28.4	76.4	25.3	54.8	5.7	55.8	34.4
WorldVLA	89.0	0.0	26.4	57.2	20.5	17.3	18.0	53.6	28.6
OpenVLA	88.4	0.5	4.5	21.0	1.0	45.2	11.4	22.4	14.0
OpenVLA + Spatial Forcing	99.6	52.0	39.2	98.6	99.0	94.8	43.4	75.2	68.5
OpenVLA + ROCKET	100.0	52.5	29.6	90.7	99.0	92.3	74.9	80.9	72.0

(c) Goal

Method	Orig.	Cam.	Robot	Lang.	Light	BG	Noise	Layout	Total
GAM (Ours)	97.4	93.1	67.5	62.7	96.8	91.8	97.1	64.5	80.4
$\pi_{0.5}$	98.0	77.2	73.6	70.5	93.2	92.5	87.3	70.4	79.3
Cosmos-policy	98.2	64.0	64.1	73.4	100.0	85.1	75.2	65.2	73.5
Fast-WAM	97.0	8.1	24.7	49.8	75.3	44.5	23.7	48.0	39.2
OpenVLA-OFT	97.9	62.0	25.2	53.2	93.9	92.5	75.2	59.1	63.0
RIPT-VLA	97.5	65.7	23.2	45.4	74.2	79.7	71.0	59.8	58.0
π_0	95.8	56.6	43.3	43.2	90.3	84.7	82.8	59.8	63.4
π_0-Fast	88.6	70.8	20.5	47.3	95.3	60.9	69.7	51.6	57.5
UniVLA	95.6	3.9	37.9	45.6	89.6	78.3	33.5	22.6	40.7
NORA	89.4	2.9	31.1	56.6	60.6	60.5	18.2	53.9	38.8
WorldVLA	82.6	0.3	30.6	42.2	68.8	30.3	13.5	47.4	31.8
OpenVLA	79.2	2.5	2.7	21.5	9.0	27.1	19.5	25.6	15.1
OpenVLA + Spatial Forcing	97.0	70.8	30.6	62.4	95.7	86.8	82.1	63.1	67.9
OpenVLA + ROCKET	98.2	44.4	40.1	57.1	99.6	96.8	56.5	67.1	62.8

(d) Long

Method	Orig.	Cam.	Robot	Lang.	Light	BG	Noise	Layout	Total
GAM (Ours)	94.6	58.0	62.8	83.3	91.6	88.2	89.5	79.5	78.0
$\pi_{0.5}$	92.4	49.2	76.1	89.6	93.8	90.3	73.1	85.9	77.9
Cosmos-policy	97.6	58.9	67.4	94.8	96.4	68.9	91.8	92.9	81.0
Fast-WAM	95.2	17.7	45.3	60.1	52.2	22.8	28.5	61.2	41.1
OpenVLA-OFT	94.5	38.7	38.2	87.0	89.4	86.8	63.5	76.9	66.4
RIPT-VLA	97.5	34.1	38.4	88.3	93.4	89.3	66.4	79.2	67.5
π_0	73.8	38.7	39.9	69.7	79.2	72.3	64.6	77.6	61.3
π_0-Fast	60.2	33.2	12.0	43.6	91.6	44.6	46.1	47.8	43.4
UniVLA	92.0	1.9	53.2	64.2	65.7	74.4	25.4	16.4	39.9
NORA	74.6	1.2	39.4	64.0	30.3	54.0	15.1	59.5	36.3
WorldVLA	59.0	0.0	12.2	20.6	20.4	1.7	1.6	4.4	8.2
OpenVLA	53.7	0.0	3.0	22.2	10.6	19.4	17.6	28.3	14.3
OpenVLA + Spatial Forcing	93.0	35.3	35.1	90.6	84.7	82.7	57.9	83.0	64.4
OpenVLA + ROCKET	94.6	32.2	45.5	93.5	87.2	86.5	45.0	70.2	62.8

Table 13 provides the suite-by-perturbation breakdown for LIBERO-Plus (Fei et al., 2025), expanding the aggregate LIBERO and LIBERO-Plus results reported in the main paper. Figure 9 further breaks down LIBERO-Plus performance by perturbation difficulty level, showing how GAM behaves as each perturbation becomes more severe. Table 14 reports task-wise success rates on the original LIBERO (Liu et al., 2023) benchmark.

Overall, these breakdowns show that GAM preserves strong performance on the original LIBERO tasks while improving robustness on LIBERO-Plus, especially under perturbations that require stable geometric understanding such as camera-viewpoint changes.

B.2 Generated Future Depth Maps

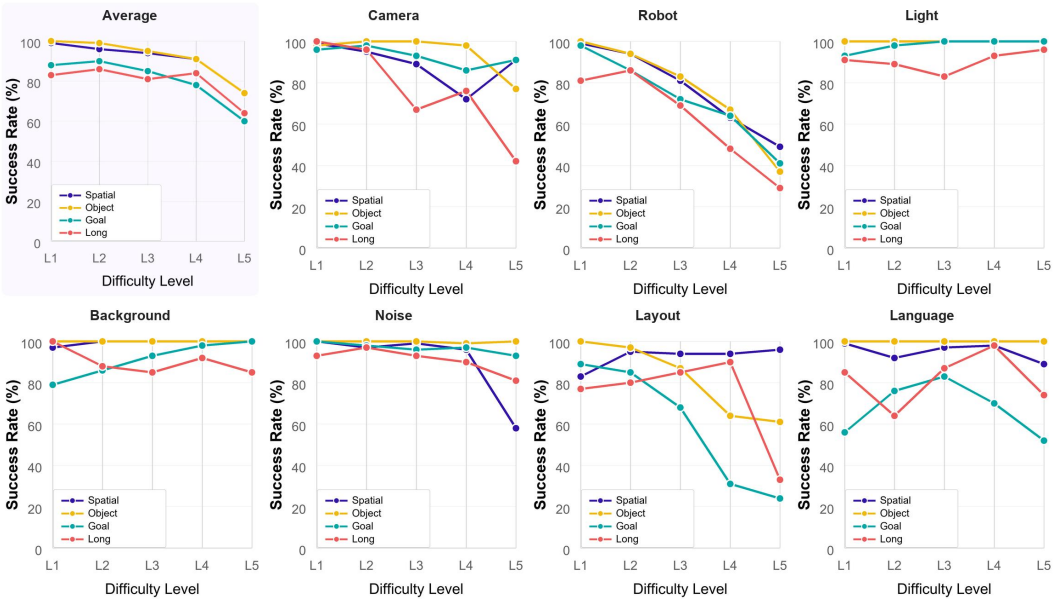


Figure 9: **Detailed zero-shot robustness on LIBERO-Plus.** We report success rates across difficulty levels L1–L5 for each perturbation category in the LIBERO-PLUS benchmark. The Average panel summarizes performance across all perturbation categories.

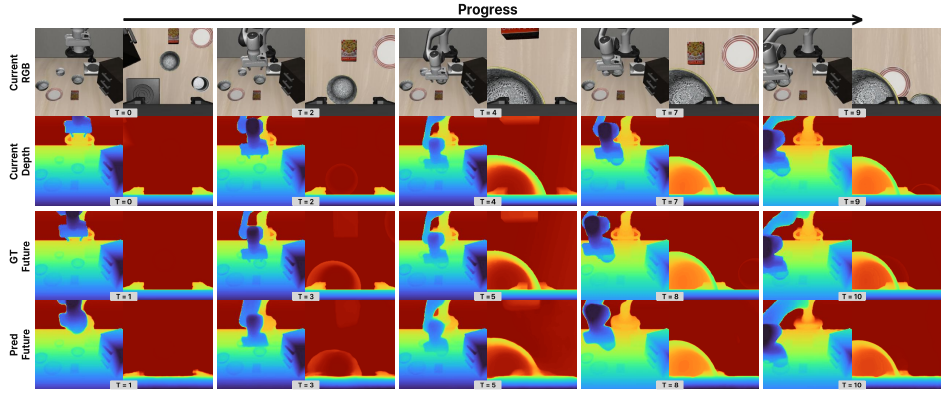
Table 14: **Per-task success rates on LIBERO Original.** Task names are abbreviated by removing suite-level repeated context.

Spatial 98.6%		**Object 99.0%**		**Goal 97.4%**		**Long 94.6%**	
Task	SR	Task	SR	Task	SR	Task	SR
Between plate & ramekin	100%	Alphabet soup	100%	Open middle drawer	100%	Soup + tomato sauce to basket	96%
Next to ramekin	100%	Cream cheese	98%	Bowl on stove	100%	Cream cheese + butter to basket	100%
From table center	98%	Salad dressing	100%	Wine bottle on cabinet	96%	Stove on + moka pot on stove	98%
On cookie box	100%	BBQ sauce	100%	Top drawer open + bowl inside	90%	Bowl in bottom drawer + close	100%
In top drawer	98%	Ketchup	98%	Bowl on cabinet	98%	Mugs to left/right plates	80%
On ramekin	96%	Tomato sauce	94%	Plate to front of stove	98%	Book to caddy back compartment	100%
Next to cookie box	100%	Butter	100%	Cream cheese in bowl	100%	Mug to plate + pudding to plate	96%
On stove	100%	Milk	100%	Turn on stove	100%	Soup + cream cheese box to basket	96%
Next to plate	94%	Chocolate pudding	100%	Bowl on plate	92%	Both moka pots on stove	88%
On wooden cabinet	100%	Orange juice	100%	Wine bottle on rack	100%	Mug to microwave + close	92%

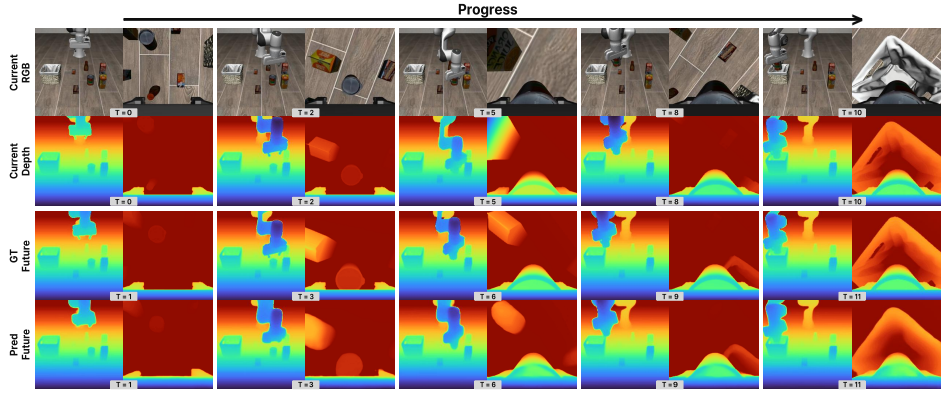
Figure 10 visualizes GAM's future depth predictions across the LIBERO task suites. Given current RGB observations, GAM jointly predicts an action chunk and the future depth expected after its execution. Table 15 evaluates these predictions against the resulting simulator depth. On GAM's rollouts, the predicted future depth achieves lower errors than the current-depth baseline evaluated against the same targets, indicating that GAM captures future changes in scene geometry.

Table 15: **Future Depth (D_{t+1}) prediction on LIBERO.** We compare each estimate with the simulator depth after executing a predicted action chunk.

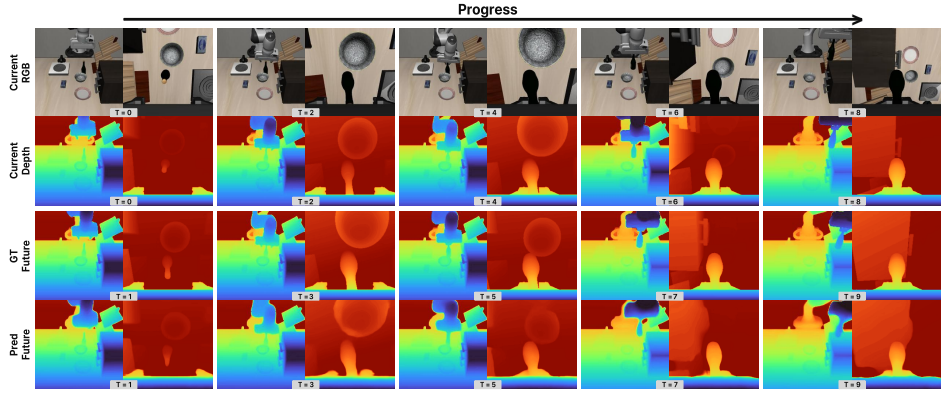
Prediction	AbsRel ↓	RMSE ↓
Current depth (D_t)	0.1056	0.1540
GAM w/o $\mathcal{L}_{\text{future}}$	0.0695	0.1063
GAM (Ours)	**0.0546**	**0.0842**



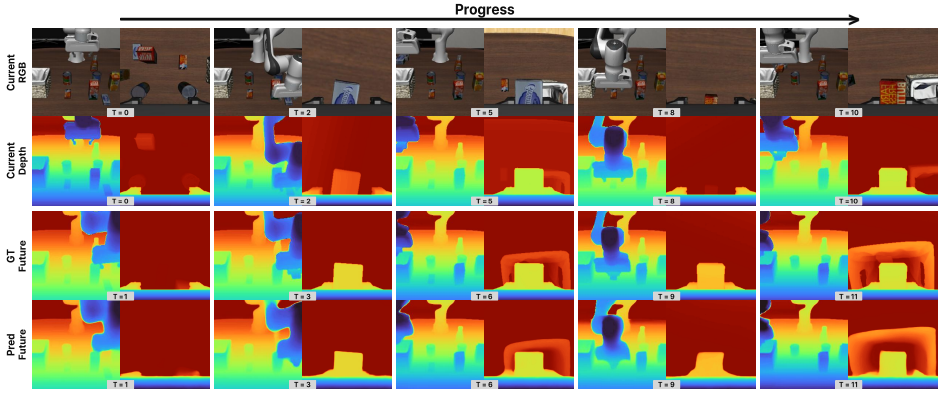
(a) LIBERO-Spatial: bowl from table center to plate.



(b) LIBERO-Object: tomato sauce to basket.



(c) LIBERO-Long: cream cheese and butter to basket.



(d) LIBERO-Goal: wine bottle on cabinet.

Figure 10: **Future depth visualizations predicted by our model on representative LIBERO tasks.**

C Ablation and Diagnostic Analyses

C.1 When to Decode Actions?

In Table 16, we additionally evaluate a direct-action supervision variant that applies the action loss directly to the output action token of the causal future predictor, without passing the action token through the remaining DA3 blocks. This ablation uses the same setting as the main component ablation on LIBERO-Object.

Table 16: **Direct action-token ablation.**

Variant	Orig.	Plus
Direct-action supervision	98.4	84.1
GAM (Ours)	**99.0**	**90.6**

Both variants start from the same robot-pretrained GAM checkpoint. The direct-action variant changes the action readout only during LIBERO-Object post-training, applying action supervision to the predictor output before the remaining GFM blocks.

Passing the action token through the deep geometric decoder provides an additional improvement, particularly on LIBERO-Plus-Object, suggesting that the remaining GFM layers help refine the action representation, especially under camera perturbations.

C.2 Adaptation to a Flow-Matching Action Decoder

We replace GAM's pretrained regression head with a newly initialized flow-matching head and post-train on LIBERO-Object. The flow-matching head is a 4-layer Transformer with 7.90M parameters.

Table 17: **Flow-matching head adaptation.**

Variant	Orig.	Plus
GAM + flow matching	95.8	83.7
GAM (default)	**99.0**	**90.6**

Table 17 shows that the flow-matching head variant recovers strong performance on both LIBERO and LIBERO-Plus. While generative decoders can model multimodal action distributions (Chi et al., 2025), prior studies report that direct regression also achieves competitive task performance (Ye et al., 2026a; Kim et al., 2025). We therefore adopt regression as our default to prioritize a simple training objective and single-pass action generation, while retaining the flexibility to use a flow-matching head.

C.3 Attention Analysis

In Figure 11, we visualize the attention maps of action tokens across GFM layers to inspect which visual regions contribute to action decoding. As shown in the attention maps, several intermediate layers attend to task-relevant regions, with clear saliency around manipulated objects and nearby contact regions.

D AI Use Statement

In this work, we used generative AI tools (LLMs) to aid and polish the writing to improve readability and clarity. We have not used generative AI tools for generating synthetic datasets, developing theoretical models, formulating or proving mathematical claims, proposing hypotheses, designing experiments, or interpreting results. All AI-assisted text was thoroughly reviewed, verified, and edited by the authors to ensure accuracy and fidelity to the original research. The authors take full responsibility for the final content and claims presented in this paper.

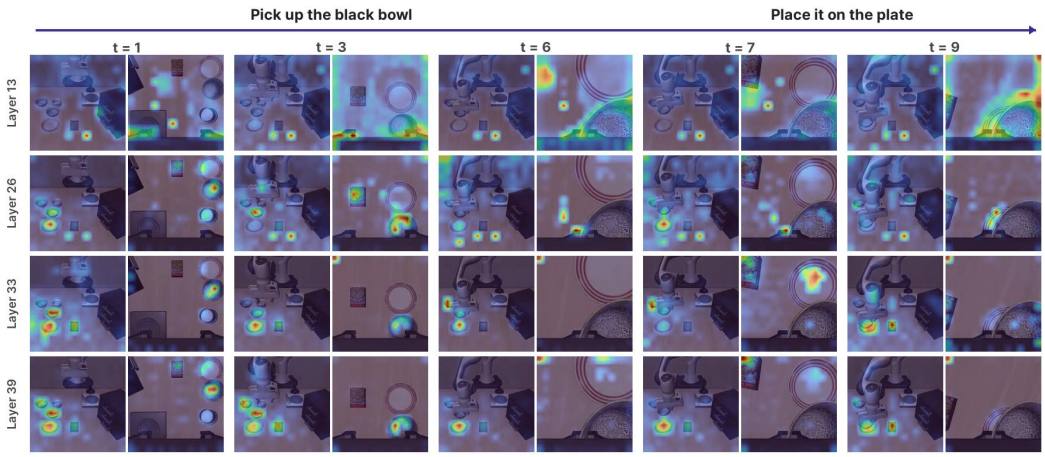


Figure 11: **Attention visualizations of action tokens.**