PLATO: A UNIFIED WORLD-STATE REPRESENTATION FOR VLA PERCEPTION AND PREDICTION

Anonymous authors
Paper under double-blind review

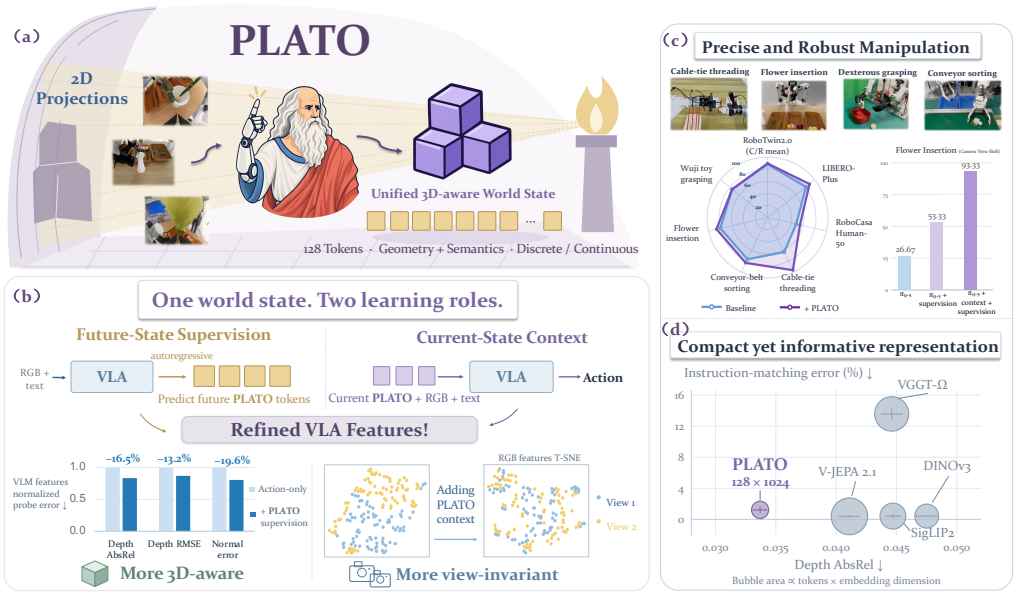


Figure 1: **PLATO: a unified world-state representation for VLA perception and prediction.** **(a)** Multi-view RGB is compressed into 128 tokens encoding geometry, semantics, and interaction-relevant information. **(b)** Supervision and context enhance geometric encoding and cross-view consistency in VLM features. **(c)** Simulation and real-world tests show improved manipulation success and robustness. **(d)** PLATO balances geometric fidelity, semantic alignment, and compactness.

ABSTRACT

As in Plato's allegory of the cave, vision-language-action (VLA) models learn from 2D visual projections, while the underlying physical world state remains implicit. We introduce PLATO (**P**hysical **L**anguage for **A**ction through **T**okenized **O**bservations), a unified world-state representation for VLA perception and prediction. Learned from multi-view observations with geometric, semantic, and forward–inverse dynamics supervision, PLATO preserves 3D structure, semantics, and interaction-relevant information in a compact form. Continuous features or embeddings of discrete codes provide current-state context, and 128 discrete state tokens serve as autoregressive future-state targets alongside action supervision. Both roles leverage the VLM's native conditioning and next-token prediction paradigm, without specialized dense prediction branches. Across VLA pretraining and post-training on three simulation benchmarks and four real-world tasks, PLATO improves manipulation success and robustness. Future-state supervision raises average real-world success by 11.7 and 25.4% under nominal and randomized conditions, respectively, and current-state context further improves viewpoint robustness, with a 40% gain over supervision alone in flower insertion. Beyond performance, our analyses show that PLATO reshapes VLA representations: geometry becomes more linearly accessible and RGB embeddings more consistent across viewpoints, revealing that explicit world-state inputs and targets substantially alter the physical structure learned by the VLA from robot data.

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.

1 INTRODUCTION

Robot manipulation translates human intent into controlled changes in the physical world. To this end, vision-language-action (VLA) models learn the conditional action distribution $p(A \mid V, L)$ from large-scale robot data (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). However, this formulation leaves a central abstraction implicit: the physical world state underlying 2D observations. Without this representation, the connections between perception, intent, control and world dynamics remain implicit in the mapping from vision and language to action.

Recent work approaches this problem through richer perception and broader supervision. Geometry-aware VLAs introduce spatial information into policy inputs (Zhen et al., 2024; Qu et al., 2025; Li et al., 2026a), while predictive approaches train policies to anticipate visual or geometric outcomes alongside actions (Cen et al., 2025; Ye et al., 2026; Bi et al., 2026; Wu et al., 2026b). However, incorporating multiple heterogeneous geometric and semantic signals through separate inputs and prediction targets can complicate their alignment with vision, language, and action. We instead seek a compact shared representation that unifies these signals into a fourth modality for VLA learning.

We introduce PLATO: **P**hysical **L**anguage for **A**ction through **T**okenized **O**bservations (Figure 1). Learned scene queries compress multi-view VGGT-Ω features (Wang et al., 2026) into 128 scene slots, represented continuously or quantized into discrete state codes. Depth, pointmap, and dense semantic reconstruction preserve spatial structure and semantic content, while forward and inverse dynamics encourage the representation to retain interaction-relevant information. Together, these objectives produce a compact state abstraction tailored to perception, prediction, and control.

Once learned, the tokenizer is frozen and reused directly within the VLM's native sequence-modeling framework. Discrete state codes provide autoregressive future-state supervision alongside action learning, while continuous features or discrete code embeddings serve as current-state context together with RGB and language. Thus, the same learned state can be both conditioned and predicted without specialized dense prediction branches.

Across three simulation benchmarks and four real-world tasks, PLATO improves manipulation accuracy and robustness. Future-state supervision raises average real-world success by 11.7 percentage points under nominal conditions and 25.4 under randomization. In cable-tie threading, which requires 1–2 mm alignment accuracy, a $\pi_{0.5}$ model with PLATO supervision incurs only one failure during a 30-minute continuous run. Current-state context further improves viewpoint robustness, with a 40-point gain over supervision alone in flower insertion. Ablations examine key design choices, while comparisons with VGGT-Ω (Wang et al., 2026), DINOv3 (Siméoni et al., 2025), SigLIP2 (Tschannen et al., 2025), and V-JEPA 2.1 (Mur-Labadia et al., 2026) demonstrate a favorable balance of compactness, geometric fidelity, and semantic alignment.

Beyond success rates, PLATO induces marked changes in VLA representations. Matched probes on frozen checkpoints show that PLATO supervision makes depth and surface orientation more decodable from VLM features, while context-augmented policies exhibit reduced view-dependent clustering in their RGB features. Together, these results show that introducing world state as an explicit VLA modality through context or supervision reshapes the representations learned from robot data: PLATO drives VLA features toward stronger 3D awareness and cross-view consistency, supporting more precise and robust manipulation.

Our contributions are threefold: **(1) A compact world-state representation for manipulation.** PLATO encodes multi-view observations into continuous features and 128 discrete state codes using geometric, semantic, and dynamics objectives; **(2) A shared interface for VLA perception and prediction.** A frozen tokenizer provides a shared representation for current-state conditioning and autoregressive future-state supervision; **(3) Evidence of improved control and altered VLM representations.** Simulation and real-world experiments establish downstream gains, while representation analyses reveal improved geometric decodability and greater cross-view consistency.

2 RELATED WORK

Geometry-aware VLA models. Spatial grounding is pursued through both input augmentation and feature supervision. SpatialVLA adds 3D position encodings to visual tokens (Qu et al., 2025), while PointVLA injects point-cloud features into the action expert (Li et al., 2026a). Spatial Forcing

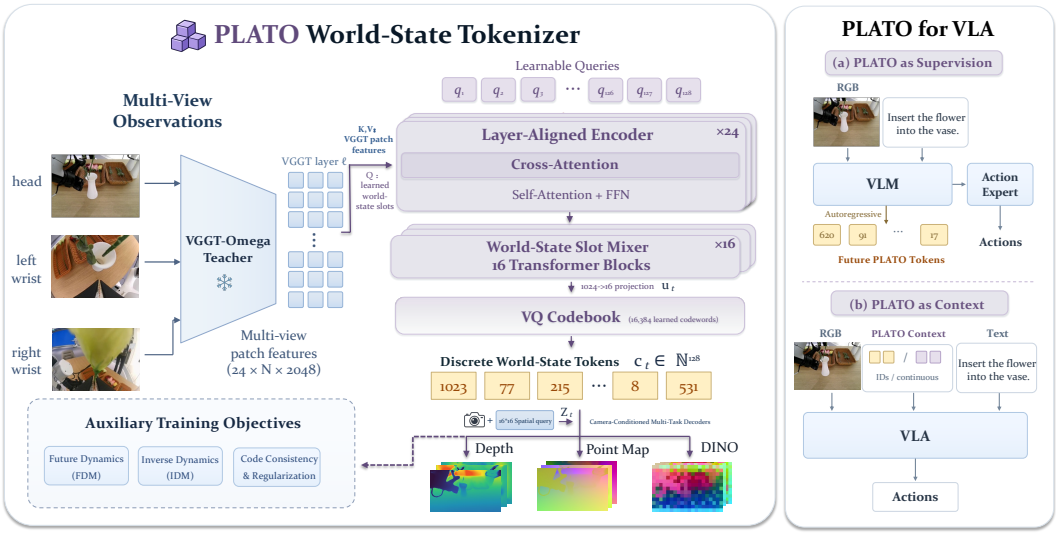


Figure 2: **The PLATO world-state interface.** Learned queries compress multi-view features into continuous and discrete states through reconstruction, dynamics, and consistency training. The frozen tokenizer then supplies future-state targets and current-scene context for VLAs.

instead aligns intermediate VLA features with representations from a pretrained geometric model (Li et al., 2026b). More recently, VGA formulates robotic manipulation as a vision-to-geometry mapping and conditions action generation on pretrained 3D representations (Song et al., 2026). We address a complementary question: how to compress multiple rich geometric features into a unified manipulation-oriented scene representation. PLATO aggregates multi-view information into a fixed token budget, with semantic and dynamics objectives guiding what survives compression.

Predictive learning for robotic manipulation. Predicting visual outcomes provides supervision beyond action imitation. 3D-VLA connects an LLM to diffusion models for goal-image and point-cloud generation (Zhen et al., 2024), while Seer conditions inverse dynamics on forecasted visual states (Tian et al., 2025). FLARE aligns diffusion-policy features with future observation embeddings (Zheng et al., 2025), and LingBot-VLA 2.0 distills current and future queries using depth and DINO-Video teachers (Wu et al., 2026b). Our distinction lies in the prediction interface: PLATO turns future-state learning into discrete next-token prediction rather than dense generation or continuous feature alignment.

Discrete visual and action representations. TiTok compresses images into short one-dimensional token sequences (Yu et al., 2024). For robot learning, LAPA (Ye et al., 2025) and UniVLA (Bu et al., 2025b) derive latent actions from frame transitions to support policy pretraining. G0.5 further uses a cross-embodiment action tokenizer and a unified autoregressive stream for reasoning and action prediction (Liu et al., 2026). PLATO instead tokenizes the physical scene, rather than image appearance alone or motion between observations. Its codes describe the state in which manipulation occurs, requiring neither an action input nor a future observation.

3 METHOD

We introduce PLATO, a compact world-state interface for VLA models (Figure 2) that compresses multi-view observations while preserving geometry, semantics, and manipulation-relevant information. Its continuous features and discrete state tokens support both perceptual context and autoregressive future-state prediction. We describe its construction in Section 3.1, learning objectives in Section 3.2, and VLA integration in Section 3.3.

3.1 TOKENIZING MULTI-VIEW WORLD STATES

Given multi-view RGB observations x_t at time t, a frozen VGGT-Ω backbone extracts hierarchical patch features. We compress these features into 128 learned scene queries through successive cross-attention blocks, each reading the corresponding backbone layer across views. A stack of self-attention blocks integrates information across scene tokens, which are then mapped to pre-quantization features through a learned projection.

To make this representation usable as categorical prediction targets, we quantize each feature using a vector-quantized codebook (Van Den Oord et al., 2017):

$$U_t = E(x_t), \qquad c_{t,i} = \arg\max_k \cos(u_{t,i}, e_k), \tag{1}$$

where E denotes the encoding pipeline, $u_{t,i}$ is its i-th pre-VQ feature, and e_k is a codebook vector. The tokenizer exposes both U_t and the discrete sequence $\mathbf{c}_t = (c_{t,1}, \ldots, c_{t,128})$. A learned recovery decoder maps the selected codebook vectors back to scene features, denoted Z_t. Reconstruction and dynamics objectives operate on these recovered quantized features, requiring useful information to survive the discrete bottleneck.

3.2 LEARNING GEOMETRY, SEMANTICS, AND ACTION RELEVANCE

Preserving spatial and semantic content. The compressed representation should retain both spatial structure and scene content. We therefore use a camera-conditioned decoder to reconstruct depth, pointmaps, and patch-wise DINO features from Z_t, with targets provided by the frozen VGGT-Ω teacher. Camera roles and available calibration condition the per-view readout. The reconstruction objective averages the three losses over current and future observations:

$$\mathcal{L}_{\text{rec}} = \frac{1}{2} \sum_{\tau \in \{t, t+h\}} \left(\lambda_d \mathcal{L}^{\tau}_{\text{depth}} + \lambda_p \mathcal{L}^{\tau}_{\text{pointmap}} + \lambda_g \mathcal{L}^{\tau}_{\text{DINO}} \right). \tag{2}$$

Depth uses confidence-weighted Smooth-L_1 in log space, pointmaps use validity-masked Smooth-L_1, and DINO features use mean patch-wise cosine distance. Together, these complementary targets constrain geometry and semantics.

Connecting scene changes to robot motion. Reconstruction alone does not explicitly identify which information matters for manipulation. We therefore link the compressed scene representation to recorded robot-state transitions through forward and inverse dynamics, following predictive manipulation models (Tian et al., 2025). Let s_t denote the absolute robot state, comprising end-effector poses and gripper states, and let h denote the temporal interval. The forward-dynamics model F predicts future scene features given the current representation and robot-state transition; the inverse-dynamics model I recovers the future robot state from current and future scene features:

$$\mathcal{L}_{\text{FDM}} = \ell_f(F(Z_t, s_t, s_{t+h}, h), \text{sg}(Z_{t+h})), \qquad \mathcal{L}_{\text{IDM}} = \ell_s(I(Z_t, Z_{t+h}, s_t, h), s_{t+h}). \tag{3}$$

Here, ℓ_f combines Smooth-L_1 and cosine distance, ℓ_s is masked Smooth-L_1, and sg denotes stop-gradient. Both objectives act on the recovered quantized features, encouraging the limited token budget to retain information connecting robot motion to physical scene changes.

Training the discrete interface. We first train the continuous encoder with current-scene reconstruction, then introduce quantization and jointly train reconstruction and dynamics prediction. Commitment ($\mathcal{L}_{\text{com}}$) and codebook-diversity losses ($\mathcal{L}_{\text{div}}$) regularize the bottleneck. To stabilize encoding under incomplete or noisy observations, we apply view dropout and Gaussian noise and match augmented outputs to detached clean references: $\mathcal{L}_{\text{code}}$ is cross-entropy on augmented code predictions using clean IDs as labels, while $\mathcal{L}_{\text{feat}}$ aligns augmented and clean pre-VQ features through cosine distance.

The full tokenizer objective is

$$\mathcal{L}_{\text{tokenizer}} = \mathcal{L}_{\text{rec}} + \lambda_F \mathcal{L}_{\text{FDM}} + \lambda_I \mathcal{L}_{\text{IDM}} + \lambda_{\text{com}} \mathcal{L}_{\text{com}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}} + \lambda_{\text{code}} \mathcal{L}_{\text{code}} + \lambda_{\text{feat}} \mathcal{L}_{\text{feat}}. \tag{4}$$

We optimize the student encoder, bottleneck projections, recovery decoder, and prediction heads while keeping VGGT-Ω frozen and updating the codebook by exponential moving average.

3.3 PLATO AS SUPERVISION AND CONTEXT

Once trained, the tokenizer is frozen and reused in two complementary roles: predicting future world states and conditioning on the current scene.

Future-state supervision. We encode observations h control steps ahead into discrete targets $\mathbf{c}_{t+h}$ and train the VLM to predict them alongside actions (Zhen et al., 2024; Wu et al., 2026b). Let $\mathcal{I}_t$ denote the current policy inputs, including RGB observations, language instructions, and robot state. The auxiliary prediction loss and joint VLA objective are

$$\mathcal{L}_S = -\frac{1}{128}\sum_{i=1}^{128} \log p(c_{t+h,i} \mid \mathcal{I}_t, c_{t+h,<i}), \qquad \mathcal{L}_{\text{VLA}} = \mathcal{L}_{\text{action}} + \lambda_S \mathcal{L}_S. \tag{5}$$

The original action objectives and their weights remain unchanged. Attention masking prevents the action branches from accessing the future-state branch, allowing supervision to shape the shared VLM without supplying future information to action prediction. Deployment retains the original policy inputs and requires neither PLATO encoding nor future-token generation.

Current-scene context. The same representation can also provide additional perceptual context for VLA training. We encode current observations and map continuous features U_t through a trainable linear projection, or discrete IDs $\mathbf{c}_t$ through a trainable VLM embedding table distinct from the tokenizer's VQ codebook. The resulting tokens supplement RGB and language inputs for action prediction. Thus, supervision shapes what the VLM learns, while context provides additional scene information for control.

4 EXPERIMENTS

We evaluate PLATO as future-state supervision (Section 4.1) and current-scene context (Section 4.2) in real-world and simulated manipulation. We then analyze how PLATO supervision and context shape VLM representations, focusing on 3D awareness and cross-view consistency (Section 4.3). Finally, we compare PLATO with alternative representations and prediction interfaces, and ablate dynamics supervision and training-source coverage (Section 4.4).

4.1 PREDICTION: PLATO AS SUPERVISION

4.1.1 REAL-WORLD POST-TRAINING

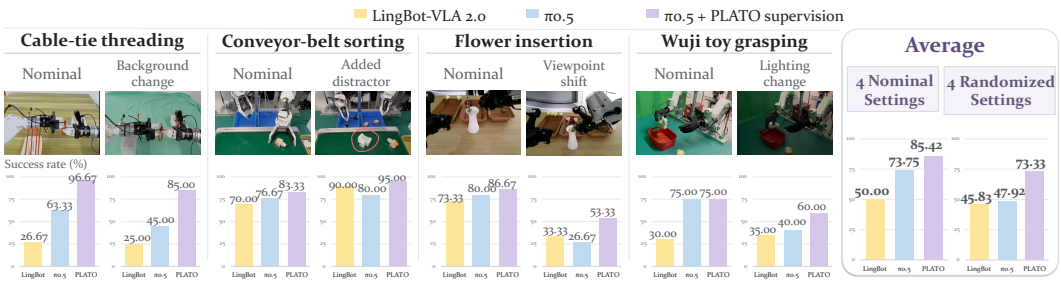


Figure 3: **Real-world post-training with PLATO supervision.** Top: nominal and randomized settings for four manipulation tasks. Bottom: success rates (%) for LingBot-VLA 2.0 (Wu et al., 2026b), $\pi_{0.5}$ (Intelligence et al., 2025), and $\pi_{0.5}$ with PLATO supervision.

Experimental setup. We post-train $\pi_{0.5}$ for 20k steps with or without PLATO supervision on four real-world tasks excluded from tokenizer training: cable-tie threading, conveyor-belt sorting, Wuji dexterous-hand toy grasping, and sequential insertion of three flowers. Each task is evaluated under nominal and randomized conditions, with perturbations covering tablecloth changes, distractors, lighting variations, and camera viewpoint shifts. We also evaluate LingBot-VLA 2.0 (Wu et al., 2026b), a closely related baseline that similarly augments action learning with current- and future-state perceptual supervision, using dual-query distillation from LingBot-Depth and DINO-Video.

Results. PLATO supervision improves nominal manipulation and, more strongly, robustness under randomization (Figure 3). Task-averaged success increases from 73.75% to 85.42% under nominal conditions and from 47.92% to 73.33% under randomization, with gains in all four randomized settings. Cable-tie threading benefits most: success rises from 63.33% to 96.67% under nominal conditions and from 45.00% to 85.00% with background changes. The supervised policy typically inserts the tie on its first attempt, consistent with more reliable spatial alignment. It also outperforms LingBot-VLA 2.0 in all eight settings, with task-averaged advantages of 35.42 and 27.50 percentage points under nominal and randomized conditions, respectively. These gains are consistent with the benefit of consolidating geometric and semantic supervision into a compact, interaction-relevant state vocabulary rather than predicting separate dense perceptual targets.

4.1.2 PRETRAINING EFFECTS

Table 1: **Pretraining effects of PLATO supervision.** Success rates (%) are shown for clean/randomized (C/R) RoboTwin2.0, LIBERO-Plus, and RoboCasa Human-50.

Model / pretraining objective	RoboTwin2.0 Clean / Randomized	LIBERO-Plus	RoboCasa Human-50
(a) Published policies (reference)			
$\pi_{0.5}$ (Intelligence et al., 2025)	82.7 / 76.8	77.4	41.4
StarVLA (Community, 2026)	88.2 / 88.3	77.0	–
ABot-M0 (Yang et al., 2026)	86.1 / 85.1	80.5	–
Being H0.5 (Luo et al., 2026)	–	78.5	53.5
LingBot-VLA (Wu et al., 2026a)	86.5 / 85.3	–	–
InternVLA-A1.5 (Ma et al., 2026)	93.3 / 93.0	84.8	–
(b) Our downstream evaluations			
Actions only	88.08 / 88.60	81.25	48.67
Actions + PLATO supervision	**89.72 / 89.18**	**88.82**	**55.00**
Gain (percentage points)	+1.64 / +0.58	+7.57	+6.33

Experimental setup. Starting from the same pretrained Wall-OSS (Yu et al., 2026) checkpoint, we train action-only and PLATO-supervised variants for 70k steps on the same mixture of approximately 1.01M multi-embodiment episodes. Both checkpoints are then post-trained on RoboTwin2.0 (Chen et al., 2025), LIBERO (Liu et al., 2023), and RoboCasa Human-50 (Nasiriany et al., 2024) under matched protocols using only action losses. Models post-trained on LIBERO are evaluated on LIBERO-Plus (Fei et al., 2025) for perturbation robustness; RoboCasa Human-50 tests post-training with limited data.

Results. Table 1(b) shows that PLATO supervision during large-scale robot pretraining improves downstream success across all three benchmarks. The largest gain occurs on LIBERO-Plus, where the performance improves from 81.25% to 88.82% (+7.57 points) across six of seven categories, with the largest gains under perturbed initial states (+35.74), lighting changes (+4.12), and altered backgrounds (+5.76). Future world state supervision may encourage the VLM to encode robot–object geometry and task-relevant transitions, reducing reliance on incidental visual shortcuts that action-only training may exploit. Beyond LIBERO-Plus, PLATO improves success by 1.64 and 0.58 points on clean and randomized RoboTwin2.0, respectively, and by 6.33 points on RoboCasa Human-50 (48.67% to 55.00%), with the latter suggesting more effective learning from limited data, potentially aided by richer semantic and spatial information in the pretrained VLM features.

4.2 PERCEPTION: PLATO AS CONTEXT

Experimental setup. We evaluate continuous pre-VQ context on LIBERO-Plus and discrete context on real-world flower insertion. For LIBERO-Plus, both variants start from the same PLATO-supervised pretrained checkpoint (Section 4.1.2) and undergo action-only post-training with or without context. For flower insertion, we compare $\pi_{0.5}$ with its supervision-only and context-plus-supervision variants. All three are trained at the same fixed camera viewpoint and evaluated under nominal and shifted viewpoints.

Table 2: **Benefits of PLATO context on LIBERO-Plus.** Success rates (%) with and without continuous context across seven perturbation categories. Overall is weighted by the number of evaluation episodes in each category.

Post-training Method	Camera	Robot init.	Language	Lighting	Background	Noise	Layout	Overall
action only	80.30	**84.13**	86.01	96.50	95.45	97.69	85.64	88.82
+ context	**85.99**	82.39	**86.08**	**96.58**	**96.75**	**97.75**	**87.30**	**89.88**

Results. Context provides its clearest gains under viewpoint shifts. On LIBERO-Plus, overall success rises from 88.82% to 89.88% (+1.06 points), led by camera shifts (+5.69 points; Table 2). For flower insertion, discrete context adds 6.67 points under the nominal view and 40 points under the shifted view relative to supervision alone (Figure 4(D)). After the viewpoint change, the baseline often grasps too high on the thin flower stem, whereas the context-augmented policy reaches lower and grasps securely. Together, these results suggest that PLATO context reduces reliance on view-specific image-to-action correspondences. Beyond supplying geometric information, we find that PLATO contexts also encourage cross-view consistency in learned RGB representations(Sec 4.3.2).

4.3 How PLATO Shapes VLM Representations

We conduct an in-depth study of how PLATO improves action accuracy and generalization. We find that PLATO changes what the VLM encodes: supervision makes geometry more linearly decodable, and context reduces view-dependent clustering.

4.3.1 Supervision Strengthens Geometric Representations

Experimental setup. We compare the frozen 70k-step checkpoints pretrained with and without PLATO supervision (Section 4.1.2). Using identical RGB inputs and episode-disjoint splits, we fit matched linear ridge probes to recover per-patch depth and camera-frame surface normals. We report errors averaged equally across six representation levels.

Table 3: **Layer-averaged geometric readout from frozen VLMs.** Errors are averaged equally over the vision merger, VLM layers 6/12/18/24, and final normalized features; lower is better.

Benchmark	Depth AbsRel ↓		Depth RMSE ↓		Normal error (°) ↓	
	Action-only	+ Scene sup.	Action-only	+ Scene sup.	Action-only	+ Scene sup.
LIBERO	0.0977	**0.0764**	0.1766	**0.1530**	7.53	**5.56**
RoboTwin	0.0573	**0.0531**	0.0745	**0.0700**	9.65	**9.15**
RoboCasa	0.1595	**0.1270**	0.1874	**0.1496**	16.29	**11.81**

Analysis. PLATO supervision consistently improves geometric decodability (Table 3): all three errors decrease on every benchmark and at every probed representation level. On RoboCasa, averaging across the six probe locations, depth AbsRel, RMSE, and normal error decrease by 20.4%, 20.2%, and 27.5%, respectively. The results indicate that PLATO supervision substantially refines the VLM toward stronger 3D-aware representations, providing a plausible representational basis for improved manipulation accuracy and generalization. Full results are provided in Appendix A.2.1.

4.3.2 Context Promotes Cross-View Consistency

Experimental setup. We study the flower insertion task, training all policies at one camera viewpoint and evaluating them under the training and shifted views. We compare $\pi_{0.5}$, + PLATO supervision, and + PLATO supervision + context. Using 82 progress-matched cross-view pairs, we mean-pool pre-transformer RGB tokens and visualize them with PCA followed by t-SNE.

Analysis. Figure 4 shows that context, not supervision alone, narrows the cross-view feature gap. The base $\pi_{0.5}$ RGB features form strongly view-dependent domains, and this separation largely remains after adding PLATO supervision. In contrast, adding PLATO context causes RGB features from the training and shifted views to substantially overlap, despite the policy being trained only at the original viewpoint. This representation shift closely mirrors the behavioral result. Adding context raises shifted-view success from 53.33% to 93.33%. This suggests that the viewpoint-robustness

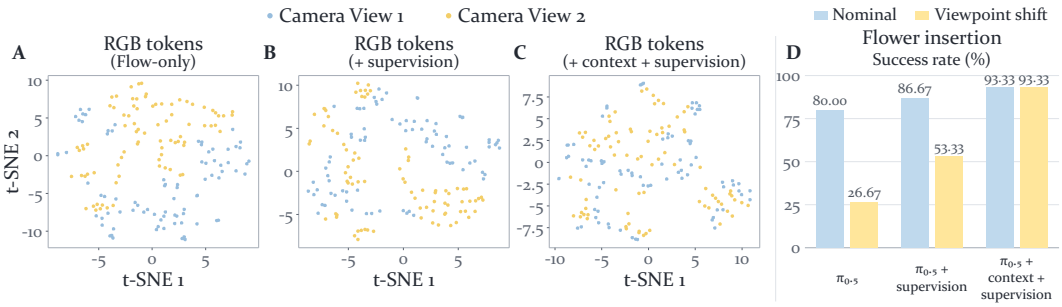


Figure 4: **PLATO context promotes cross-view consistency in RGB features. (A–C)** t-SNE of RGB features from the three variants; context produces cross-view overlap. **(D)** Context raises shifted-view success from 53.33% to 93.33%.

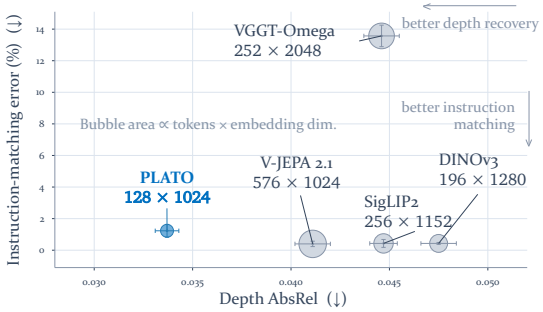
benefit of PLATO context is not solely due to providing additional spatial input at inference; it also substantially refines the VLM's visual representations toward more viewpoint-invariant features. Detailed results are provided in Appendix A.2.2.

4.4 Representation Comparisons and Ablations

We examine the information retained by the compact representation, the choice of future-state targets, and the roles of dynamics supervision and training-source coverage.

4.4.1 Compactness, Geometry, and Semantics

(a) Geometry–semantics-compactness trade-off



(b) Frozen-representation probes

Representation	Tokens × Dim.	Depth AbsRel ↓	Instruction accuracy (%) ↑
VGGT-Omega	252 × 2048	0.0446	86.43
DINOv3	196 × 1280	0.0475	99.57
SigLIP2	256 × 1152	0.0447	99.57
V-JEPA 2.1	576 × 1024	0.0411	99.60
PLATO	**128 × 1024**	**0.0337**	**98.77**

Figure 5: **Geometry and instruction matching in compact representations.** (a) Bubble area: native token count × embedding dimension; axes show depth AbsRel and instruction-matching error (lower left is better). (b) Mean depth AbsRel and instruction accuracy across three seeds.

Experimental setup. We compare frozen PLATO, VGGT-Ω (Wang et al., 2026), DINOv3 (Siméoni et al., 2025), SigLIP2 (Tschannen et al., 2025), and V-JEPA 2.1 (Mur-Labadia et al., 2026) representations using matched depth and instruction probes on RoboTwin data. For PLATO, we read the 128 high-dimensional pre-MLP scene features; for the alternatives, we use final-layer patch features. Depth AbsRel measures agreement with normalized VGGT pseudo-depth, while a frozen SigLIP text encoder supports balanced instruction matching with different-task negatives.

Analysis. Figure 5 shows that PLATO achieves a favorable compactness–informativeness trade-off. It obtains the lowest depth AbsRel (0.0337), a 24.4% reduction relative to VGGT-Ω final layer features, while maintaining a high instruction-matching accuracy of 98.77%, comparable to the strongest semantic representations. VGGT-Ω's native depth decoder fuses multi-layer features, whereas PLATO consolidates them into its final representation, helping explain its stronger depth readout under final-feature probing. At the same time, PLATO uses only 128 tokens, fewer than all alternatives (196–576 tokens). Together, these results show that PLATO compresses strong geometric and semantic information into a compact world-state representation.

Table 4: **Prediction interfaces and tokenizer design choices.** (a) RoboTwin2.0 success rates (%) with alternative future-state objectives. (b) RoboTwin2.0 success rates (%) under a fixed $\pi_{0.5}$ post-training recipe, using PLATO supervision from tokenizers trained with or without FDM/IDM objectives. (c) Reconstruction and dynamics losses on 100 held-out ABC-130k samples, using all, half, or a quarter of the training dataset families.

(a) Prediction interfaces

Future target	Clean	Random.	Combined
None (action only)	69.80	70.62	70.21
Cosmos discrete 2D	73.72	70.20	71.96
Continuous pre-VQ 3D	69.54	70.80	70.17
Depth–DINO queries	66.14	66.90	66.52
PLATO discrete 3D	**73.48**	**73.02**	**73.25**

(b) FDM/IDM

Condition	✗	✓
Clean	86.52	**87.34**
Random.	86.42	**86.80**
Combined	86.47	**87.07**

(c) Training-source coverage

Loss ↓	Full	1/2	1/4
Depth	**0.0080**	0.0087	0.0086
Pointmap	**0.00073**	0.00077	0.00079
DINO	**0.2699**	0.3014	0.2968
FDM	**0.1093**	0.1362	0.1305
IDM	**0.0201**	0.0278	0.0347

4.4.2 Why 3D? Why Discrete?

Experimental setup. We compare against three alternative forms of future state supervision that differ in target representation and prediction interface. On RoboTwin2.0, we train all variants for 40k steps from Qwen3.5-2B with randomly initialized action experts. Cosmos (Agarwal et al., 2025) predicts discrete 2D appearance tokens autoregressively; continuous pre-VQ regression autoregressively predicts the corresponding continuous scene features; and a LingBot-VLA 2.0-inspired Depth–DINO baseline (Wu et al., 2026b) uses learned queries to regress dense future depth and semantic features.

Analysis. The benefits of future-state supervision depend on how the target is represented and predicted, not simply on adding an auxiliary loss. PLATO achieves the highest combined success, 73.25% (+3.04 points over action-only; Table 4(a)). Cosmos performs similarly on clean scenes but trails by 2.82 points under randomization, consistent with an advantage for action-relevant 3D targets over appearance reconstruction. Continuous pre-VQ and Depth–DINO regression trail PLATO by 3.08 and 6.73 points, respectively, and neither improves combined success over action-only training. Together, these results favor PLATO's joint design: action-relevant 3D world states, represented discretely and predicted autoregressively.

4.4.3 Dynamics Supervision and Training-Source Diversity

Experimental setup. We compare tokenizers with and without FDM/IDM objectives under the same downstream $\pi_{0.5}$ action/flow recipe. Separately, we train on all, half, or a quarter of the dataset families and evaluate the same 100 ABC-130k (Allshire et al., 2026) samples, excluded from tokenizer training.

Analysis. FDM/IDM supervision raises combined RoboTwin2.0 success from 86.47% to 87.07% (+0.60 points; Table 4(b)), suggesting that dynamics constraints help PLATO retain information more useful for downstream policy learning. The full training mixture also achieves the lowest loss across all five held-out ABC-130k metrics (Table 4(c)), supporting the value of broad training-source coverage for out-of-distribution reconstruction and dynamics prediction.

5 Conclusion

We introduced PLATO, a compact world-state interface that unifies VLA perception and prediction. Across simulation and real-world tasks, future-state supervision improves downstream control when applied during pretraining or post-training, while context further strengthens viewpoint robustness. Our central insight is that a unified world-state modality does more than enrich VLA inputs: it can reshape the learned representations toward greater 3D awareness and cross-view consistency. However, there are two limitations. First, the limited coverage of tokenizer training data may constrain PLATO's generalization to a broader range of scenes and interactions. Second, its reliance on VGGT-generated targets may propagate teacher errors into the learned representation, potentially limiting the fidelity of the encoded world state.

AI USE STATEMENT.

Generative AI tools were used to support research ideation and the organization of methodological and experimental workflows, including brainstorming possible formulations, refining the presentation of the method, and suggesting directions for analysis. They were also used to retrieve and identify related work, draft portions of the manuscript, improve readability, and assist with LaTeX formatting. The authors made all final decisions regarding the method, experimental design, implementation, experimental execution, analysis, and interpretation of the results. All AI-assisted ideas, retrieved references, and manuscript text were reviewed and independently verified by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide implementation and experimental details throughout the paper and appendix to support reproducibility of our main findings. Section 3 describes the world-state representation, tokenizer learning objectives, and integration into VLA models as future-state supervision and current-scene context. Section 4 presents the real-world and simulation settings, baseline comparisons, representation analyses, and ablation studies. Detailed tokenizer configurations, training-data composition, sampling, preprocessing, and loss formulations are provided in Appendix A.1. Appendix A.4 reports the training budgets, batch sizes, optimization settings, loss weights, and future-state offsets. We further specify the alternative future-target implementations in Appendix A.3 and the geometry-probing and cross-view analysis protocols in Appendix A.2.

REFERENCES

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.

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025a.

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*, 2025b.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a.

Vincent Chen, Starrick Liu, Regis Cheng, Dance Yang, Shalfun Li, Ryan Yu, Lucy Liang, Hang Su, Roy Gan, Hao Wang, et al. Dmuon: Efficient distributed muon training with near-adam overhead. *arXiv preprint arXiv:2606.27153*, 2026b.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

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.

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.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

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.

Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026a.

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. In *International Conference on Learning Representations*, volume 2026, pp. 132324–132345, 2026b.

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.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, et al. G0. 5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026.

Hao Luo, Ye Wang, Wanpeng Zhang, Sipeng Zheng, Ziheng Xi, Chaoyi Xu, Haiweng Xu, Haoqi Yuan, Chi Zhang, Yiqing Wang, et al. Being-h0. 5: Scaling human-centric robot learning for cross-embodiment generalization. *arXiv preprint arXiv:2601.12993*, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, and Nicolas Ballas. V-jepa 2.1: Unlocking dense features in video self-supervised learning. In *European Conference on Computer Vision*, pp. 671–689. Springer, 2026.

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.

Ryan Punamiya, Simar Kareer, Zeyi Liu, Josh Citron, Ri-Zhao Qiu, Xiongyi Cai, Alexey Gavryushin, Jiaqi Chen, Davide Liconti, Lawrence Y Zhu, et al. Egoverse: An egocentric human dataset for robot learning from around the world. *arXiv preprint arXiv:2604.07607*, 2026.

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.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 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)→ g): Vision-geometry backbones over language and video models. 2026.

Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Emu: Generative pretraining in multimodality. In *International conference on learning representations*, volume 2024, pp. 12352–12380, 2024.

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*, volume 2025, pp. 92033–92052, 2025.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736. PMLR, 2023.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Omega. *arXiv preprint arXiv:2605.15195*, 2026.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026a.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026b.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025.

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*, 2026.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024.

Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, et al. Wall-oss-0.5 technical report. *arXiv preprint arXiv:2605.30877*, 2026.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

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.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

A APPENDIX

A.1 PLATO IMPLEMENTATION DETAILS

A.1.1 NETWORK ARCHITECTURE

The frozen VGGT-Ω backbone provides patch features from 24 layers, with feature width 2048. At each layer, 128 learned scene queries attend to the corresponding features from all available views. The scene-query encoder has width 1024 and 16 attention heads. A stack of self-attention blocks then processes the scene tokens before projection to the 16-dimensional pre-quantization features U_t.

Each row of $U_t \in \mathbb{R}^{128\times16}$ is assigned to one of 16,384 codebook entries by cosine similarity, as in Equation 1. The selected vectors are passed through the recovery decoder to obtain Z_t. The camera-conditioned reconstruction decoder and both dynamics heads read Z_t. The high-dimensional pre-MLP features used in the representation comparison have shape 128×1024; they precede the projection that produces U_t and are a different readout from the 128×16 continuous targets in Appendix A.3.2.

Table 5: PLATO tokenizer architecture.

Component	Configuration
VGGT-Ω	24 layers; feature width 2048; frozen
Scene-query encoder	128 queries; width 1024; 24 layers; 16 heads
Pre-VQ transformer	16 layers
Quantization	One codebook; 16,384 entries; code dimension 16
Bottleneck attention	16 heads
Head-mixing / projection / expansion	Widths 512 / 256 / 512
Recovery decoder	TiTok-style (Yu et al., 2024); width 1024; 12 layers
Camera adapter	Three views; 16×16 spatial queries; 16 heads
Dynamics heads	State width 20; 4 layers; 8 heads; horizon 32

The backbone remains frozen throughout tokenizer training. The scene queries, student transformers, bottleneck projections, recovery decoder, camera-conditioned decoder, and prediction heads receive gradient updates. Codebook vectors are updated by exponential moving average (EMA).

A.1.2 TRAINING DATA AND SAMPLING

The tokenizer corpus contains 16 datasets, with 1,645,918 episode entries and 1,157,054,248 runtime frames in the data inventory. Table 6 lists the sources. These counts describe tokenizer training; the VLA pretraining mixture in the main paper contains 1.01M episodes.

Table 6: Sources used to train the tokenizer. Dataset counts refer to named sources, before splitting them into local sampling families.

Group	Datasets	Sources
Core benchmarks	4	RoboTwin (Chen et al., 2025), RoboDojo (Chen et al., 2026a), RoboCasa (Nasiriany et al., 2024), LIBERO (Liu et al., 2023)
Open-source datasets	10	Agibot World (Bu et al., 2025a), Bridge (Walke et al., 2023), DROID (Khazatsky et al., 2024), EgoVerse (Punamiya et al., 2026), InternData-A1 (Tian et al., 2026), RoboCOIN (Wu et al., 2025), EgoScale (Zheng et al., 2026), Galaxea (Jiang et al., 2025), RoboMIND (Wu et al., 2024), RT-1 (Brohan et al., 2022)
Internal datasets	2	Internal real-robot and UMI data

The sampler uses 25 local families, each with weight 1 and a 20,000-window cap. A family therefore contributes $1/25 = 4\%$ of the nominal sampling quota. RoboCOIN is split into six families (24%), RoboMIND into four (16%), and Bridge into two (8%). Each of the remaining seven open-source datasets contributes one family (4%). The four core benchmarks contribute 16% in total, and the

two internal datasets contribute 8%. Figure 6 distinguishes these nominal sampling shares from the fractions of frames in the underlying inventory.

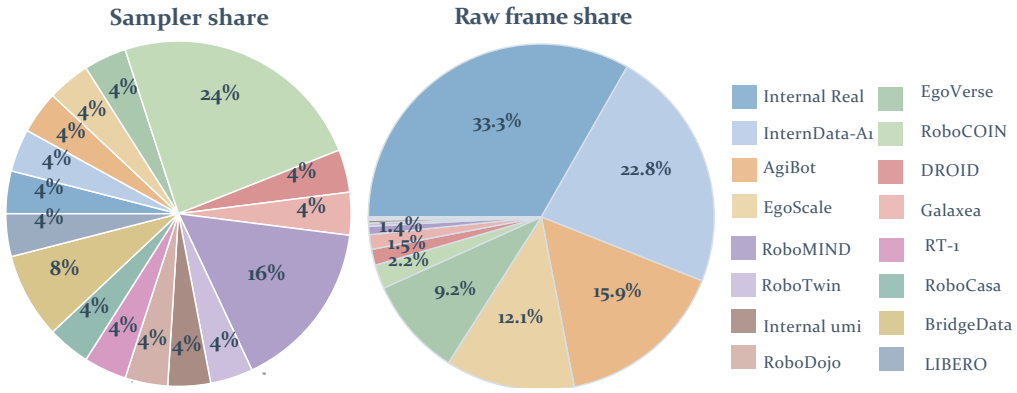


Figure 6: Tokenizer training data. Left: nominal shares of sampled windows under family-balanced sampling. Right: shares of runtime frames in the data inventory.

The inventory contains 384,961,961 frames from the internal real-robot source, 3,310,048 from RoboTwin, 1,427,717 from RoboDojo, 559,340 from RoboCasa, and 204,843 from LIBERO. The other sources account for the remaining frames.

Observations use three camera slots: head, left wrist, and right wrist, with role IDs 0, 1, and 2. Robot states contain absolute end-effector poses and gripper states; Euler rotations are converted to row-wise 6D rotation representations. Each tokenizer training window contains a current observation, a 32-step action horizon, and a future observation at interval h. VGGT inputs use 256-pixel resolution and patch size 16. Camera-validity and robot-state masks are retained when computing the losses. VLA future-state offsets are listed separately in Appendix A.4.

A.1.3 Camera-conditioned query decoding

For each output camera, the decoder constructs a spatial query grid conditioned on camera role and calibration. Its inputs include the role ID, intrinsics, extrinsics, and view- and pose-validity flags. Extrinsics are converted to a camera center and a 6D rotation. Camera centers are expressed relative to the valid head camera and scaled by the median non-reference camera baseline. Intrinsics are normalized by image width and height. An MLP embeds the resulting 13-dimensional geometry vector; invalid pose metadata is replaced by a learned null-pose embedding.

For view v, the queries are

$$Q_v = Q_{\text{grid}} + Q_{\text{coord}} + Q_{\text{shared}} + E_{\text{role}}(r_v) + E_{\text{geom}}(g_v), \tag{6}$$

where the first three terms are the learned spatial grid, 2D coordinate features, and a shared learned query embedding. The queries cross-attend to Z_t, followed by a residual feed-forward block and layer normalization. Outputs are stacked over views and reshaped to $[B, V, H_{\text{grid}}, W_{\text{grid}}, D]$ for dense prediction. All views read the same recovered scene representation. During training, camera-pose dropout replaces selected valid pose conditions with the null-pose embedding.

A.1.4 Commitment and codebook-diversity losses

Let $q(u)$ denote the codebook vector selected for pre-VQ feature u. The commitment penalty is

$$\ell_{\text{com}}(u) = \|u - \text{sg}(q(u))\|_2^2 . \tag{7}$$

The quantizer applies an internal weight of 0.25 to its reduced commitment loss. The losses returned for the current and future observations are summed to form $\mathcal{L}_{\text{com}}$, which receives the outer weight $\lambda_{\text{com}} = 0.1$ in Equation 4. This penalty updates the encoder; the detached codebook vectors are updated by EMA.

Codebook diversity is measured using the batch-averaged assignment distribution. For batch item b, scene position i, and code k, let $d_{bik} = \cos(u_{b,i}, e_k)$. We use

$$p_{bik} = \mathrm{softmax}_k(10 d_{bik}), \qquad \bar{p}_k = \mathrm{mean}_{b,i}\, p_{bik}, \qquad \mathcal{L}_{\mathrm{div}} = -H(\bar{p}), \tag{8}$$

with $\lambda_{\mathrm{div}} = 0.01$. The entropy is computed over the aggregate distribution, rather than separately for each scene.

A.1.5 Reconstruction and dynamics losses

Depth, pointmap, and DINO targets are provided by the frozen VGGT-Ω teacher. Reconstruction is evaluated on the decoder outputs from Z_t and Z_{t+h}, with equal weight for the two times as in Equation 2.

The pointwise depth error is

$$\ell_d = \mathrm{SmoothL1}\left(\log(\max(\hat{d}, 10^{-5})), \log(\max(d, 10^{-5}))\right). \tag{9}$$

Invalid or non-positive teacher depths are masked out; a finite replacement is used before evaluating the logarithm. Confidence weights are normalized by their per-view mean and clipped to $[0, 10]$. The pointmap loss uses Smooth-L_1 and retains a target only when all three coordinates are finite. The DINO loss is the cosine distance between predicted and teacher patch features. Losses use masked means, with depth additionally weighted by confidence. View-validity masks are applied before the per-sample reduction.

The forward model predicts Z_{t+h} from (Z_t, s_t, s_{t+h}, h), with the target Z_{t+h} detached. Its loss combines Smooth-L_1 and cosine distance. The inverse model predicts s_{t+h} from (Z_t, Z_{t+h}, s_t, h) using masked Smooth-L_1. Samples without valid state transitions are excluded from the dynamics loss reduction. Both models operate on recovered quantized features, as in Equation 3.

A.1.6 Input augmentation and consistency

We apply view dropout and Gaussian image noise when forming augmented observations. View dropout includes head-only inputs. Gaussian noise is applied with probability 0.30; its standard deviation is sampled from $(0.002, 0.005, 0.01, 0.02)$ with probabilities $(0.50, 0.30, 0.15, 0.05)$, respectively. The consistency branch starts at step 1,000 and reaches its full weight at step 5,000.

For an augmented observation, the reference is a clean encoding of the same scene. The clean pass does not update the codebook, and its features and code assignments are detached. With E_{code} denoting the matrix of codebook vectors, the code-consistency loss is

$$\ell_{\mathrm{code}} = \frac{1}{\log 16384}\, \mathrm{CE}\left(\frac{\mathrm{normalize}(U^{\mathrm{aug}})\, \mathrm{normalize}(E_{\mathrm{code}})^\top}{0.1}, \mathrm{sg}(c^{\mathrm{clean}})\right). \tag{10}$$

Normalization is along the feature dimension, and cross-entropy is averaged over scene positions. Feature consistency uses

$$\ell_{\mathrm{feat}} = \frac{1}{128} \sum_{i=1}^{128} \left[1 - \cos\left(u_i^{\mathrm{aug}}, \mathrm{sg}(u_i^{\mathrm{clean}})\right)\right]. \tag{11}$$

For each term, losses from the current and future observations are averaged, followed by a batch mean. Both outer weights are 0.1.

A.1.7 Training stages and objective weights

Training begins with current-scene reconstruction through the continuous encoder. We then introduce vector quantization and jointly optimize reconstruction and forward–inverse dynamics on the recovered features. Commitment, diversity, and augmented-to-clean consistency regularize this quantized stage. Table 7 gives the coefficients in Equation 4; the internal commitment scale is specified in Appendix A.1.4.

Table 7: Objective weights for quantized tokenizer training.

Objective	Weight
Depth / pointmap / DINO reconstruction	1.0 / 1.0 / 1.0
Forward / inverse dynamics	0.05 / 0.05
Commitment (outer weight)	0.1
Codebook diversity	0.01
Code / pre-VQ feature consistency	0.1 / 0.1

A.1.8 VLA SUPERVISION AND CURRENT-SCENE CONTEXT

The tokenizer is frozen for VLA training. Future-state supervision uses the 128 discrete IDs c_{t+h} as autoregressive targets. We average the cross-entropy over the 128 prediction positions and combine it with the original action objective using the experiment-specific coefficient reported in the training configuration tables. The action branches cannot attend to the future-state targets. A policy trained with supervision alone does not run the tokenizer or generate future-state tokens during action inference.

For current-scene context, continuous pre-VQ features U_t are mapped into the VLM embedding space by a trainable linear projection. Alternatively, the current IDs c_t index a trainable policy embedding table, distinct from the tokenizer's VQ codebook. Both forms provide 128 context positions alongside RGB and language inputs. These policies encode the current observation at inference time.

The LIBERO-Plus context experiment starts from the same PLATO-supervised checkpoint and uses action-only downstream post-training, with or without continuous current-scene context. The flower-insertion experiment compares action-only training, future-state supervision, and future-state supervision with discrete current-scene context. All three flower-insertion variants are trained with the same fixed camera viewpoint.

A.2 DETAILED REPRESENTATION ANALYSIS

A.2.1 LAYER-WISE GEOMETRY PROBES

We freeze the Wall-OSS checkpoints after 70k pretraining steps and fit matched linear ridge probes to the vision merger, decoder layers 6, 12, 18, and 24, and the final normalized VLM output. Train, validation, and test splits have disjoint episode identities. Depth and normal targets are patch-aligned: LIBERO and RoboTwin use VGGT pseudo-geometry, while RoboCasa uses simulator ground truth. Ridge strength is selected on the validation split before evaluation on the test split.

Table 8 reports all six readouts. The averages in the main paper give each readout equal weight. Layer 24 is measured before the final normalization and is therefore reported separately from the final normalized output.

A.2.2 CROSS-VIEW T-SNE

We analyze the three flower-insertion checkpoints trained for 20k steps: flow-only, flow with PLATO supervision, and flow with supervision and current PLATO context. For each observation, the 768 RGB tokens from three camera streams are mean-pooled before the first VLM block to obtain one 2048-dimensional vector. Language, state-text, and PLATO context tokens are excluded from this readout.

The analysis uses 82 pairs of observations matched by normalized progress in two episodes with different camera configurations. The same 164 observations are used for every checkpoint. The two episodes may also differ in object layout, so the pairs are progress-matched observations rather than simultaneous captures of an identical scene.

Each checkpoint is projected independently. We first apply PCA with 50 components, without per-channel standardization, and then t-SNE with perplexity 30, early exaggeration 12, cosine distance, random initialization, automatic learning rate, random seed 0, and 2,000 iterations. Silhouette scores

Table 8: Layer-wise geometry probes on frozen VLA features. Values are averaged over test frames using valid image patches. LIBERO and RoboTwin use VGGT pseudo-geometry, with depth RMSE in median-normalized relative units. RoboCasa uses simulator ground truth, with depth RMSE in metres. Normal error is in degrees. Lower is better for every metric; bold marks the better value within each pair.

	Depth AbsRel ↓		Depth RMSE ↓		Normal error (°) ↓	
Representation	Action-only	+ PLATO	Action-only	+ PLATO	Action-only	+ PLATO
LIBERO — VGGT pseudo-geometry						
Vision merger	0.0949	**0.0783**	0.1717	**0.1544**	6.76	**5.33**
Layer 6	0.0911	**0.0723**	0.1665	**0.1472**	6.75	**5.09**
Layer 12	0.0983	**0.0743**	0.1752	**0.1496**	7.36	**5.25**
Layer 18	0.1049	**0.0770**	0.1852	**0.1542**	7.88	**5.54**
Layer 24	0.0996	**0.0786**	0.1820	**0.1566**	8.26	**6.00**
Final	0.0974	**0.0777**	0.1787	**0.1560**	8.14	**6.16**
RoboTwin — VGGT pseudo-geometry						
Vision merger	0.0568	**0.0527**	0.0725	**0.0694**	9.15	**8.74**
Layer 6	0.0547	**0.0499**	0.0711	**0.0660**	9.31	**8.70**
Layer 12	0.0569	**0.0520**	0.0741	**0.0686**	9.64	**8.99**
Layer 18	0.0592	**0.0540**	0.0770	**0.0716**	9.90	**9.26**
Layer 24	0.0587	**0.0553**	0.0768	**0.0727**	9.99	**9.56**
Final	0.0577	**0.0548**	0.0752	**0.0715**	9.92	**9.65**
RoboCasa — simulator ground truth						
Vision merger	0.1513	**0.1167**	0.1775	**0.1395**	14.85	**11.16**
Layer 6	0.1504	**0.1184**	0.1786	**0.1402**	15.23	**11.01**
Layer 12	0.1555	**0.1227**	0.1852	**0.1453**	15.95	**11.29**
Layer 18	0.1643	**0.1303**	0.1936	**0.1544**	16.89	**11.82**
Layer 24	0.1686	**0.1365**	0.1955	**0.1586**	17.47	**12.75**
Final	0.1671	**0.1374**	0.1941	**0.1598**	17.33	**12.85**

use Euclidean distances and camera configuration as the label. Trustworthiness uses Euclidean 10-nearest-neighbor structure from PCA-50 to t-SNE.

Table 9: Camera-configuration separation in pooled RGB features. Lower silhouette scores indicate less separation in the corresponding space. Higher trustworthiness indicates better preservation of local neighborhoods by t-SNE. All readouts exclude PLATO context tokens.

Metric	Flow-only	+ PLATO supervision	+ PLATO context + supervision
Original RGB silhouette	0.09441	0.09674	**0.05968**
PCA-50 silhouette	0.09917	0.10012	**0.06198**
t-SNE silhouette	0.12842	0.13112	**0.00187**
Trustworthiness	0.93714	0.94342	**0.95833**

Adding current context reduces the original-feature silhouette from 0.09674 to 0.05968 and the PCA-50 silhouette from 0.10012 to 0.06198. The corresponding t-SNE values are 0.13112 and 0.00187. The supervision-only and context-plus-supervision variants share the same initialization, training data, RGB augmentation, and future-state objective. Since the readout precedes the first VLM block, the measured difference is in the RGB features learned during training.

A.3 Alternative target implementation details

The alternative-target comparison uses Qwen3.5-2B with randomly initialized action experts, trained on RoboTwin2.0 for 40k steps. Each auxiliary target is derived from the observation at $t + 32$, with a 32-step action horizon. The action objective is

$$\mathcal{L}_{\text{action}} = \mathcal{L}_{\text{AR}} + 10\mathcal{L}_{\text{flow}}, \qquad \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{action}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}. \tag{12}$$

The auxiliary encoders are frozen. Auxiliary branches read the observed prefix and are isolated from the action branches. The configurations in Table 10 differ in target type, view coverage, prediction length, and auxiliary weight.

Table 10: Alternative future-state targets on RoboTwin2.0. Prediction slots exclude branch delimiters. Cosmos additionally predicts EOS; the two regression branches do not use categorical EOS supervision.

Target	Views	Prediction slots	Target shape	λ_{aux}
PLATO discrete 3D	3	128	128 categorical IDs	0.1
Cosmos	1	256	256 categorical IDs	0.1
Continuous pre-VQ	3	128	128×16	1.0
Depth–DINO	3	8	$756 \times (1 + 1024)$	0.1

A.3.1 Cosmos discrete 2D target

We resize the future front-view image to 256×256 and rescale pixel values from $[0, 1]$ to $[-1, 1]$. The frozen Cosmos-0.1-Tokenizer-DI16x16 encoder produces a 16×16 index map over a 64,000-entry vocabulary. The map is flattened in row-major order and its IDs are mapped to policy vocabulary entries. Wrist views are not used as targets in this configuration.

The sequence is placed between BOS and EOS in the auxiliary branch. Shifted teacher forcing uses BOS to predict the first code, each code to predict the next, and the final code to predict EOS. Let $y_{1:256}$ be the flattened code map, $y_{257} = \text{EOS}$, and $\mathcal{P}_t$ be the observed prefix. The loss is

$$\mathcal{L}_{\text{Cosmos}} = -\frac{1}{257}\sum_{i=1}^{257} \log p_\theta\left(y_i \mid \mathcal{P}_t, \text{BOS}, y_{<i}\right), \tag{13}$$

with outer weight 0.1. The auxiliary block follows the FAST action block in sequence order, but the attention mask prevents it from reading action tokens and prevents either action branch from reading the future-image targets.

A.3.2 CONTINUOUS PRE-VQ TARGET

Following continuous visual autoregression (Sun et al., 2024), this variant predicts the pre-VQ features of the frozen PLATO encoder. The three future views produce $U_{t+32} \in \mathbb{R}^{128\times 16}$. Targets are detached and normalized:

$$\bar{u}_i = \frac{\mathrm{sg}(u_i)}{\max(\|u_i\|_2, \epsilon)}, \qquad \epsilon = 10^{-6}. \tag{14}$$

The pre-quantization projection is retained; code IDs are not used as regression labels.

The BOS hidden state predicts $\bar{u}_1$. For $i > 1$, the position containing the projected target $\bar{u}_{i-1}$ predicts $\bar{u}_i$. The input adapter is

$$a_i = W_{\mathrm{in}}\,\mathrm{LN}(\bar{u}_i) + b_{\mathrm{in}}. \tag{15}$$

The 128 prediction readouts come from BOS and the first 127 continuous target positions. The branch is causal and isolated from both action branches.

A separate LayerNorm–linear head maps each selected hidden state h_i to a 16-dimensional prediction:

$$\hat{u}_i = W_{\mathrm{out}}\,\mathrm{LN}(h_i) + b_{\mathrm{out}}, \qquad \bar{\hat{u}}_i = \frac{\hat{u}_i}{\max(\|\hat{u}_i\|_2, \epsilon)}. \tag{16}$$

The auxiliary loss is

$$\mathcal{L}_{\mathrm{preVQ}} = \frac{1}{128}\sum_{i=1}^{128}\left[1 - \bar{\hat{u}}_i^{\top}\bar{u}_i + 0.1\,\mathrm{SmoothL1}_{\beta=0.1}\left(\bar{\hat{u}}_i, \bar{u}_i\right)\right]. \tag{17}$$

Smooth-L_1 is averaged over the 16 feature coordinates. The loss is then averaged over positions and the batch, with outer weight 1.0. Continuous targets and their delimiters are excluded from discrete cross-entropy supervision. The input adapter and prediction head are trainable; the future-observation encoder remains frozen.

A.3.3 DEPTH–DINO QUERY TARGET

The frozen VGGT-Ω teacher processes three future views at 224×288 resolution. Each view produces a 14×18 grid, giving 756 target locations. At each location, the target contains one log-depth value and one 1024-dimensional DINO feature.

A depth pixel is valid when its depth is finite and positive and its confidence is finite. Invalid pixels receive zero weight. Within each view, confidence weights are divided by their spatial mean and clipped to $[0, 10]$. For patch region $\mathcal{R}_j$, the target is

$$D_j = \frac{\sum_{p\in\mathcal{R}_j} w_p \log\big(\max(d_p, 10^{-5})\big)}{\sum_{p\in\mathcal{R}_j} w_p}. \tag{18}$$

The numerator and denominator are area-pooled separately. A patch enters the depth loss only if its pooled denominator exceeds 10^{-6} and its camera view is valid. The DINO target $f_j \in \mathbb{R}^{1024}$ is normalized to unit length.

Eight learned query embeddings are inserted into the auxiliary branch. They contain no future-image features and use a causal mask to read the observed prefix without reading the action branches. Their final hidden states form $Q \in \mathbb{R}^{8\times d}$. Two independent resampling heads produce

$$\hat{D} = f_{\mathrm{depth}}(Q) \in \mathbb{R}^{756\times 1}, \qquad \hat{F} = f_{\mathrm{DINO}}(Q) \in \mathbb{R}^{756\times 1024}. \tag{19}$$

Each head applies LayerNorm and a linear projection to obtain 256-dimensional memory, then cross-attends to it with 756 learned output queries in one four-head attention block. A residual feed-forward block with GELU and hidden width 256 is followed by LayerNorm and a linear output layer. The depth and DINO heads have separate query banks and parameters.

Let $\mathcal{V}_d$ contain valid depth patches across the batch and $\mathcal{V}_g$ contain patches from valid camera views. The losses are

$$\mathcal{L}_{\mathrm{depth}} = \frac{1}{|\mathcal{V}_d|}\sum_{j\in\mathcal{V}_d}\mathrm{SmoothL1}_{\beta=0.1}(\hat{D}_j, D_j), \tag{20}$$

$$\mathcal{L}_{\mathrm{DINO}} = \frac{1}{|\mathcal{V}_g|}\sum_{j\in\mathcal{V}_g}\left[1 - \cos(\hat{F}_j, f_j)\right]. \tag{21}$$

Depth and semantic masks are separate: an invalid depth estimate does not remove the corresponding DINO target. The combined objective is

$$\mathcal{L}_{\text{query}} = 0.5\mathcal{L}_{\text{depth}} + 0.5\mathcal{L}_{\text{DINO}}, \tag{22}$$

with outer weight 0.1. The query inputs are learned parameters, so this branch does not use shifted future-feature teacher forcing. The teacher and dense prediction heads are used during training only.

A.4 Training hyperparameters and future-state offsets

Table 11 lists the VLA training settings. All listed recipes use DMuon (Chen et al., 2026b), a learning rate of 5×10^{-5}, a cosine schedule, and 1k warmup steps. The large-scale comparison starts both variants from the same pretrained Wall-OSS checkpoint and trains for 70k steps on the same 1.01M-episode mixture. One variant uses action losses alone; the other also uses future-state supervision. Both then receive matched action-only downstream post-training.

Table 11: VLA training recipes. Loss entries give outer coefficients. The action-only controls omit the auxiliary objective. A dash indicates that downstream training uses no future-state target.

Experiment	Steps	Target	GPUs	Batch	Loss weights
Large-scale VLA pretraining	70k	$t+32$	24	432	AR 1; flow 10; PLATO 0.1
RoboTwin downstream post-training	100k	–	16	288	AR 1; flow 10
LIBERO / RoboCasa-Human50 downstream post-training	20k	–	8	144	AR 1; flow 10
Alternative-target comparison	40k	$t+32$	8	128	AR 1; flow 10; auxiliary weight in Table 10
Real-world post-training	20k	$t+16$	8	128	Flow 1; PLATO 0.01

Downstream post-training uses 100k steps on RoboTwin2.0 and 20k steps on LIBERO and RoboCasa-Human50. Policies post-trained on LIBERO are evaluated on LIBERO-Plus. The real-world experiments start from $\pi_{0.5}$ and use 20k post-training steps on cable-tie threading, conveyor-belt sorting, Wuji dexterous-hand toy grasping, and sequential insertion of three flowers. These four tasks are excluded from tokenizer training.

A.5 Transfer across VLA initializations

Table 12 collects the matched comparisons for $\pi_{0.5}$, Wall-OSS, and Qwen3.5-2B with randomly initialized action experts. For Wall-OSS, future-state supervision is applied in the 70k-step pretraining stage, followed by action-only downstream post-training. For $\pi_{0.5}$ and Qwen3.5-2B, it is applied during the respective 20k-step real-world and 40k-step RoboTwin recipes.

Table 12: Success rates (%) with and without PLATO supervision. Gains are percentage-point differences between the displayed values within each initialization and evaluation setting.

Initialization	Evaluation	Action only	+PLATO	Gain
$\pi_{0.5}$	Real-world nominal / randomized	73.75 / 47.92	85.42 / 73.33	+11.67 / +25.41
Wall-OSS	RoboTwin clean / randomized	88.08 / 88.60	89.72 / 89.18	+1.64 / +0.58
Wall-OSS	LIBERO-Plus	81.25	88.82	+7.57
Wall-OSS	RoboCasa-Human50	48.67	55.00	+6.33
Qwen3.5-2B	RoboTwin clean / randomized	69.80 / 70.62	73.48 / 73.02	+3.68 / +2.40
Qwen3.5-2B	RoboTwin combined	70.21	73.25	+3.04

Within each comparison, the variants share the initialization, training data, action horizon, optimizer, and evaluation protocol. Absolute success rates across rows also reflect differences in the backbone, training stage, and evaluation suite.

Table 13: **Full real-world evaluation.** Successes/rollouts are reported for nominal (N) and randomized (R) settings; percentages are shown in parentheses. The context-plus-supervision variant is evaluated only on flower insertion.

	Cable-tie threading		Conveyor sorting		Flower insertion		Wuji grasping	
Method	N	R	N	R	N	R	N	R
LingBot-VLA 2.0	8/30 (26.67)	5/20 (25.00)	21/30 (70.00)	18/20 (90.00)	11/15 (73.33)	5/15 (33.33)	6/20 (30.00)	7/20 (35.00)
$\pi_{0.5}$	19/30 (63.33)	9/20 (45.00)	23/30 (76.67)	16/20 (80.00)	12/15 (80.00)	4/15 (26.67)	15/20 (75.00)	8/20 (40.00)
$\pi_{0.5}$ + PLATO supervision	29/30 (96.67)	17/20 (85.00)	25/30 (83.33)	19/20 (95.00)	13/15 (86.67)	8/15 (53.33)	15/20 (75.00)	12/20 (60.00)
$\pi_{0.5}$ + PLATO supervision + context	–	–	–	–	14/15 (93.33)	14/15 (93.33)	–	–

A.6 Real-World Experiment Details

Hardware and tasks. We evaluate four real-world tasks across three robotic platforms and end effectors. Cable-tie threading uses a Tianji robotic arm equipped with Gripper-H, an H-shaped press-actuated gripper. Conveyor-belt sorting and sequential flower insertion use the EX001 dual-arm platform equipped with Gripper-G, a G-shaped finger-driven gripper. Toy grasping uses the first-generation Wuji dexterous hand. The tasks respectively evaluate precision insertion, dynamic manipulation, long-horizon manipulation, and dexterous grasping. All task data are excluded from PLATO tokenizer training.

Generalization settings. Each task is evaluated under both its nominal configuration and one task-specific perturbation. For cable-tie threading, we replace the tablecloth and cable-tie box; for conveyor sorting, we introduce distractor objects; for Wuji grasping, we change the illumination. For flower insertion, we shift the deployment viewpoint by changing `lift_waist` from 0.37 to 0.40 and camera pitch from 0.25 to 0.05. These perturbations preserve the task objective while changing visual appearance or observation geometry.

Evaluation protocol. We compare $\pi_{0.5}$, LingBot-VLA 2.0, and $\pi_{0.5}$ with PLATO supervision under the same task settings. The context-plus-supervision variant is additionally evaluated on flower insertion to assess viewpoint robustness. Success is measured over complete task rollouts, and Table 13 reports successful trials over total rollouts together with success rates. The physical setups and representative full rollouts are shown in the supplementary demo videos.