Spatial-Aware VLA Pretraining through Visual-Physical Alignment from Human Videos

Supplementary Material

A. Fusion Layer

The fusion layer design is inspired by VLM-3R[21], which follows a cross-attention mechanism, where semantic visual features attend to 3D spatial features extracted by the 3D encoder. Given vision features $V_{sem} \in \mathbb{R}^{N_v \times d_v}$ and spatial features $V_{spa} \in \mathbb{R}^{N_s \times d_s}$, we first project them into a shared attention space, and then perform cross-attention where visual tokens query 3D spatial tokens. The output is projected back to the vision feature dimension using an output projection, resulting in F_{spa}. To integrate the spatial information while maintaining the pretrained visual semantics, the fusion layer applies a residual connection with a learnable scaling parameter α:

$$V_f = V_{sem} + \alpha F_{spa} \tag{9}$$

Finally, dropout and layer normalization are applied to stabilize optimization.

B. Training Details

Stage 1 We pretrain `VIPA-VLA` with `Hand3D-visual` for a single epoch in the first stage, cost around 6 hours on 8 NVIDIA A800 GPUs. Only the fusion layer is optimized, with AdamW optimizer using the learning rate of $1e-5$. We set warm up ratio to 0.03 and the weight decay to 0.01, with a cosine scheduler. The global batch size is 32. For the fusion layer, the spatial scaling parameter α is initialized as 0.5. The image frames are resized to 448×448.

Stage 2 In the second stage we use `Hand3D-action` to optimize the fusion layer and the LLM backbone of `VIPA-VLA`, requiring around 20 hours on 8 NVIDIA A800 GPUs. The training hyper-parameters are kept the same as in the first stage.

Stage 3 In the post-training stage, we freeze the visual encoder and the 3D encoder, and the global batch size is 128, with a learning rate $5e-5$. The warm up ratio is 0.05 and the weight decay is $1e-5$. For both the LIBERO benchmark and the real robot tasks, we train `VIPA-VLA` for 30K steps, requiring around 5 hours on 8 NVIDIA A800 GPUs.

C. Motion Tokenization

In the second pretraining stage, we tokenize each wrist trajectory point (x_t, y_t, z_t) into three discrete motion tokens. To convert continuous 3D coordinates into discrete indices, we apply a uniform discretization over pre-defined bounded ranges. For each axis $a \in \{x, y, z\}$, we define a clipping range: $a \in [a_{min}, a_{max}]$, and in our implementation, the x and y axes share the range $[-0.5, 0.5]$, while the z axis uses $[0, 1]$. This effectively discretizes a $1m^3$ physical space directly in front of the camera into a structured token space. Each coordinate is discretized into one of K bins, $m^a \in \{0, ..., K-1\}$, where we use $K = 1024$ in our experiments. Thus, each 3D waypoint (x_t, y_t, z_t) is converted into a triplet of motion tokens m_t^x, m_t^y, m_t^z, yielding the final tokenized motion sequence $(m_1^x, m_1^y, m_1^z, ..., m_t^x, m_t^y, m_t^z)$.

D. Details of `Hand3D`

D.1. Curation Details and Statistics

As stated in Section 3.2.1, we collect human demonstration videos from three sources: **(1) Motion capture datasets**: Arctic[20], HOI4D[46], FPHA[24], H2O[39], OAKINK2[83], TACO[47], Dex-YCB[11], **(2) VR-recorded datasets**: EgoDex[29] and **(3) Pseudo-annotated datasets**: Taste-Rob[90]. Following UniHand[49], we align all hand annotations to the unified MANO[63] representation. For datasets with 3D hand joint annotations, we fit the MANO parameters via gradient optimization. For Taste-Rob, which is a video-only dataset, we estimate the hand MANO parameters with HaWoR[85]. For instruction labeling, each video is first segmented into 10s chunks and Gemini-2.5-Flash[17] is applied to generate chunk-level and second-level annotations including task instructions and hand-object interactions. Then we utilize Gemini-2.5-Pro to construct three kinds of VQA-style tasks based on the annotations: instructional motion generation, motion translation and contextual motion prediction. The detailed composition and distribution of the `Hand3D-visual` and `Hand3D-action` are provided in Table 6 and Table 7.

D.2. Examples of `Hand3D-visual`

We provide some examples of the `Hand3D-visual` in Figure 7. The dataset covers four categories of manipulation-related tasks, encompassing diverse forms of spatial understanding essential for interaction. Such diversity ensures that the pretraining stage exposes the model to a wide range of manipulation scenarios, providing rich supervision for learning robust visual–physical alignment.

Figure 7. Examples of `Hand3D-visual`.

Table 6. Data distribution of `Hand3D-visual`.

Category	Count	Proportion
Sources		
Arctic	100,772	33.5%
OakInk2	45,926	15.3%
TACO	39,087	13.0%
H2O	32,390	10.8%
HOI4D	28,977	9.6%
EgoDex	28,200	9.4%
FPHA	12,918	4.3%
Taste-Rob	7,181	2.4%
Dex-YCB	4,917	1.6%
Task Types		
Task Completion	206,409	68.7%
Spatial Relations	74,887	24.9%
Hand Movements	18,867	6.2%
Camera Movements	205	0.1%
Total	300,368	100%

E. Real Robot Tasks

We provide detailed configurations of the real robot tasks introduced in Section 4.3. Each task is further decomposed into several sub-tasks, which are used to measure fine-grained execution performance.

- **Put-Three-Obj**: The task requires the robot to open the drawer, sequentially pick and place three fruits (apple, banana, plum) from the table into the drawer, and finally close the drawer. We define five sub-tasks: *open drawer*, *pick & place apple*, *pick & place banana*, *pick & place plum*, and *close drawer*. For the unseen environment setup, the table surface is modified by covering it with tablecloths of unseen colors.
- **Wipe-Board**: The task requires the robot to picks up a cloth and wipes the pen marks off the whiteboard. There are three sub-tasks defined: *pick up cloth*, *wipe pen marks*, and *clean entire whiteboard*. For the unseen environment setup, we change the color of the marker used to draw the pen marks.
- **Water-Plant**: The task requires the robot to pick up a watering can, locate the plant, and water it by pressing the spray handle. There are three sub-tasks defined: *pick up watering can*, *locate plant*, and *press spray handle*.

During evaluation, we randomize the placement of objects on the table and the distribution of pen marks on the whiteboard for each trial. This setup ensures that the learned policies are tested on their generalization ability rather than memorization of fixed configurations.

Table 7. Data distribution of `Hand3D-action`.

Category	Count	Proportion
Sources		
EgoDex	758,050	73.5%
Arctic	104,032	10.1%
TACO	77,100	7.5%
OakInk2	54,470	5.3%
HOI4D	19,812	1.9%
H2O	17,386	1.7%
Task Types		
Instructional Motion Generation	610,192	59.2%
Contextual Motion Prediction	280,545	27.2%
Motion Translation	140,113	13.6%
Total	1,030,850	100%

F. Additional Simulation Results

In this section, we provide supplementary experiments conducted in the RoboCasa[55] simulation environment, which poses significantly greater challenges than LIBERO—featuring more diverse layouts, cluttered scenes, and more complex visual observations. Such environments demand more accurate and robust 3D spatial understanding from VLA models. To evaluate this setting, we adopt a three-view input configuration and train `VIPA-VLA` using only 50 human demonstrations per task, for a total of 60K training steps with a global batch size of 256. Training

Table 8. **Success rates (%) on the RoboCasa benchmark.** Models are evaluated on 24 tasks (8 for *Pick & Place*, 6 for *Doors / Drawers*, 10 for *Others*), with each task evaluated across 50 trials.

Model	Pick & Place	Doors / Drawers	Others	Avg.
GR00T N1	18.6	50.2	39.1	36.0
$\pi_{0.5}$	**21.5**	57.8	44.9	41.4
VIPA-VLA	20.8	**67.7**	**52.8**	**45.8**

takes approximately 40 hours on 8 NVIDIA A800 GPUs. Following the LIBERO setup, both the visual encoder and the 3D encoder are frozen during training. We use a learning rate of $5e-5$, a warm up ratio of 0.05, and a weight decay of $1e-5$.

Across the diverse RoboCasa tasks, VIPA-VLA achieves the best overall performance, demonstrating strong generalization under multi-view observations and limited demonstration data. Notably, our model attains significant improvements in the *Doors/Drawers* categories ($+9.9\%$), which require precise spatial localization. These results highlight that the Spatial-Aware Pretraining effectively equips the model with more reliable 2D–3D grounding, enabling accurate spatial reasoning even in visually diverse and geometrically challenging environments.

G. Additional Real Robot Task

To better reflect the gains of our approach, we additionally include a more challenging real-robot task and compare with stronger baselines. As shown in Figure 8, the task requires a first-person-view dual-arm robot to insert flowers into a vase, involving precise 3D localization under dynamic viewpoints, which is particularly challenging for spatial grounding. We evaluate VIPA-VLA and $\pi_{0.5}$ over 20 trials. VIPA-VLA performs better in this challenging task with a 30% success rate, while $\pi_{0.5}$ has a 15% success rate, which we attribute to improved visual–physical alignment learned from human video pretraining, particularly under egocentric and dynamically changing viewpoints.

Figure 8. A more challenging real robot task.

H. Ablation Study on Fusion Layer Capacity

Our design uses a single cross-attention fusion layer for the efficient visual–physical alignment. Here we conduct an additional ablation with a 2-layer cross-attention fusion module on LIBERO. As shown in Table 9, the results are not better.

Table 9. Ablation study of the fusion layer capacity.

# Layers	L-S	L-O	L-G	L-L	Avg.
1	92.6	97.2	94.2	85.6	92.4
2	90.4	97.4	93.6	82.6	91.0

I. Qualitative Examples

Figure 9 presents qualitative examples of VIPA-VLA executing real robot tasks. Across diverse scenes and object configurations, the model demonstrates reliable spatial grounding and robust generalization: it accurately localizes objects, adjusts to varying layouts, and completes multi-step manipulation sequences with consistent trajectories. These examples highlight the model's ability to transfer its spatial-aware pretraining to real-world settings, enabling precise and stable task execution even under unseen spatial variations.

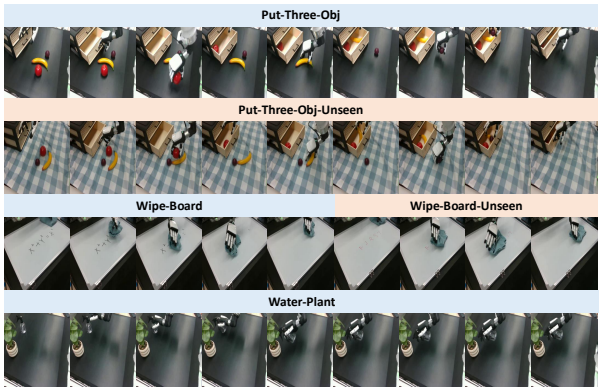

Figure 9. Qualitative examples of VIPA-VLA performing real robot tasks.

J. Failure Case Analysis

Figure 10 illustrates typical failure cases of VIPA-VLA and the baseline model InternVL-3.5 on real robot tasks. For VIPA-VLA, failures generally occur in fine-grained manipulation steps—such as slight misalignment when grasping a small object—while the overall spatial localization and action planning remain correct. These errors often arise from subtle control inaccuracies rather than misunderstanding the physical space. In contrast, the baseline model frequently fail due to inaccurate spatial grounding: they may reach toward incorrect regions or misjudge object positions that reflect poor 2D–3D correspondence. Such failures indicate weaker spatial understanding, causing the policy to be stuck before meaningful manipulation begins.

In the unseen Wipe-Board task (Figure 11), VIPA-VLA consistently succeeds in grasping the cloth, and typically reaches the correct region with appropriate contact with the board, but sometimes fails with fine-grained execution is-

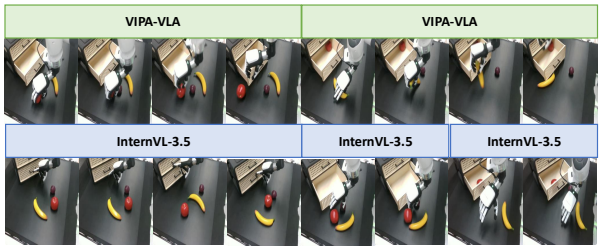

Figure 10. Failure examples of VIPA-VLA and InternVL-3.5 performing real robot tasks.

sues (e.g., insufficient wiping pressure). In contrast, baselines often fail at earlier stages, including (i)inaccurate spatial localization leading to failed grasping and (ii) incorrect depth estimation, applying excessive downward motion. This suggests that VIPA-VLA improves spatial localization and interaction grounding with unseen visual inputs.

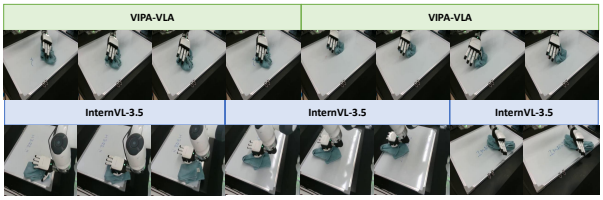

Figure 11. Failure examples of VIPA-VLA and InternVL-3.5 performing the unseen Wipe-Board task.