## APPENDIX

### A. Point Cloud Generation

This section provides additional implementation details for constructing the hand and end-effector point clouds used in GALA. Unless otherwise specified, all point clouds contain $N = 1024$ points and are expressed in a wrist- or root-centered local coordinate frame.

*a) Human Hand Canonicalization.:* For each RGB frame, WiLoR [27] estimates the left- and right-hand MANO [25] meshes together with 21 hand keypoints. For each detected hand, let $w_t$ denote the wrist position, $k_t^{\mathrm{idx}}$ and $k_t^{\mathrm{pky}}$ denote the index- and pinky-MCP joints, respectively, and let

$$\bar{k} * t = \frac{1}{4} \sum * j \in \mathcal{M} k_t^j \tag{15}$$

denote the mean position of the four MCP joints $\mathcal{M}$.

We construct a palm-aligned coordinate frame as

$$\begin{aligned} &\tilde{y}_t = \mathcal{N}(\bar{k}_t - w_t), \qquad \tilde{x}_t = \mathcal{N}(k_t^{\mathrm{idx}} - k_t^{\mathrm{pky}}), \\ &z_t = \mathcal{N}(\tilde{x}_t \times \tilde{y}_t), \quad x_t = \mathcal{N}(\tilde{y}_t \times z_t), \quad y_t = \mathcal{N}(z_t \times x_t). \end{aligned} \tag{16}$$

where $\mathcal{N}(v) = v/\|v\|_2$ denotes $\ell_2$ normalization. The resulting orthonormal basis is followed by a fixed axis-convention transformation shared across datasets.

The MANO mesh vertices are first centered at the wrist and then transformed into this local coordinate frame. We subsequently sample $N = 1024$ surface points from the mesh, with each triangular face selected in proportion to its surface area and points sampled uniformly within the selected face.

*b) Robot End-Effector Point Clouds.:* For each robot embodiment, we obtain the geometry and kinematic structure from its URDF or MJCF model. We identify the links belonging to the relevant end effector, including all articulated finger links for dexterous hands and the finger or jaw links for parallel-jaw grippers.

For each link $l$, we pre-sample a set of surface points

$$\mathcal{S}^{e,h} * l = \left\{ s^{e,h} * l, i \right\} * i = 1^{N_l} \tag{17}$$

from its collision or visual mesh, with the number of samples $N_l$ proportional to the link surface area. Given the recorded embodiment state at time $t$, we first convert it into the corresponding joint configuration $q_t^e$. Forward kinematics then provides the transformation $T * t, l^e \in SE(3)$ from each end-effector link frame to the root-centered coordinate frame. The transformed point set is

$$\tilde{\mathcal{S}}^{e,h} * t, l = \left\{ T * t, l^e, \bar{s}^{e,h} * l, i \right\} * i = 1^{N_l}, \tag{18}$$

where $\bar{s}$ denotes the homogeneous coordinate of a sampled surface point.

We aggregate the transformed samples from all end-effector links,

$$\tilde{P}^{e,h} * t = \bigcup * l \in \mathcal{L}^{e,h} \tilde{\mathcal{S}}_{t,l}^{e,h}, \tag{19}$$

and resample the resulting set to $N = 1024$ points to obtain $P_t^{e,h}$. The same procedure is used for dexterous hands and parallel-jaw grippers; embodiment-specific processing is restricted to converting the recorded state into the joint configuration required for forward kinematics.

*c) Validity Handling.:* If a human hand is not detected or its reconstruction is considered invalid, we replace its point cloud with zeros and set the corresponding validity mask to $m_t^h = 0$. Valid hand observations use $m_t^h = 1$. The validity mask is used throughout training so that missing hands do not contribute to the corresponding hand-geometry objectives.

### B. Fine-Grained Motion Probe Architecture and Training Protocol

We freeze all representation encoders, feature backbones, and VQ codebooks, and train only a lightweight MLP probe on the post-quantization embeddings, rather than on the discrete VQ indices. For each latent stream $\mathbf{X}_m \in \mathbb{R}^{T_m \times D_m}$, we employ an independent branch encoder

$$\mathbf{h}_m = \mathrm{GELU}\left(\mathbf{W}_m \,\mathrm{vec}\left(\mathrm{LN}(\mathbf{X}_m)\right)\right) \in \mathbb{R}^{256}. \tag{20}$$

The 21-dimensional current robot state is processed by $\mathrm{LN}{-}\mathrm{Linear}(21, 128) - \mathrm{GELU}$. The encoded state, visual latent, and available hand-representation branches are concatenated and passed to a prediction head

$$\hat{\mathbf{y}} = \mathrm{Linear}_{256\to 18}\left(\mathrm{Dropout}_{0.1}\left(\mathrm{GELU}\left(\mathrm{Linear}_{d_{\mathrm{in}}\to 256}(\mathbf{h})\right)\right)\right), \tag{21}$$

which predicts the 18-dimensional motion target $\mathbf{y} = [\Delta\mathbf{p}, \Delta\mathbf{r}, \Delta\mathbf{q}_{\mathrm{finger}}]$, consisting of a 3D wrist translation, a 3D rotation vector, and up to 12 articulation dimensions.

The visual latent contains four 128-dimensional tokens. GALA uses its shared bimanual geometric latent through the two slots of the common hand interface, while METIS and Native Kinematics provide separate left- and right-hand latent streams, each containing four 128-dimensional tokens. GALA w/o PC uses only the state and visual branches. Since OPFA does not learn a visual latent, we pair it with the same frozen visual latent produced by GALA w/o PC, matched by sample identifier. Its geometric input comprises four post-quantization streams corresponding to the left/right hands at the start and goal frames. Each OPFA stream is represented by a 1024-dimensional quantized embedding. Missing hand streams are zero-filled. We use the same branch width, prediction head, and optimization protocol for all methods, while retaining the method-specific number and dimensionality of input streams.

The target dimensions are standardized using statistics computed only from valid entries in the training split. We minimize the sum of block-wise Smooth-L1 losses over translation, rotation, and articulation:

$$\mathcal{L}_{\mathrm{probe}} = \sum_{b \in \{\mathrm{pos,rot,finger}\}} \frac{\sum_i \mathbf{m}_{i,b}\, \mathrm{SmoothL1}\left(\hat{\mathbf{y}}_{i,b}, \tilde{\mathbf{y}}_{i,b}\right)}{\sum_i \mathbf{m}_{i,b}}, \tag{22}$$

where $\mathbf{m}_{i,b}$ masks unavailable target dimensions. Full-pose XHand samples supervise all three blocks, whereas samples containing only embodiment-native joint annotations supervise the articulation block. Accordingly, position and rotation metrics are computed only on samples with valid wrist targets, while finger error is computed over all samples with valid articulation targets.

The probe is optimized using AdamW with a learning rate of $10^{-3}$, weight decay of $10^{-4}$, and a batch size of 256. We train for at most 100 epochs and select the checkpoint with the lowest validation loss, using early stopping with a patience of 10 epochs. Training samples are drawn with task- and episode-balanced sampling. The common evaluation intersection contains 57,472 samples, divided at the episode level into 43,580 training, 7,008 validation, and 6,884 test samples. We report the mean and sample standard deviation over three fixed random seeds, $\{42, 43, 44\}$.

### C. RoboCasa Simulation Experiment Settings

We conduct simulation experiments on the RoboCasa GR-1 Tabletop benchmark, which is built upon the RoboCasa simulation framework and uses the Fourier GR-1 humanoid robot equipped with two dexterous hands. The policy receives RGB observations, proprioceptive states, and language instructions as inputs, and predicts coordinated upper-body and hand actions in a closed-loop manner. The benchmark contains 24 tabletop manipulation tasks, with 1,000 human-teleoperated demonstrations provided for each task.

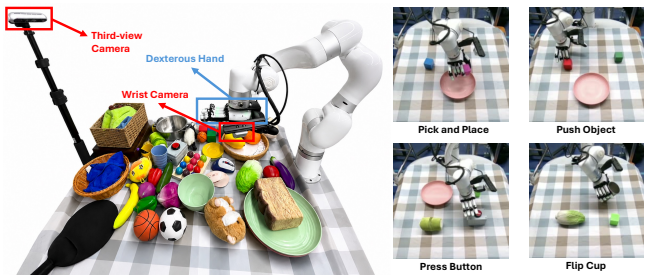


Fig. 4. Real-world platform setup and task overview

**Tasks.** The benchmark consists of 24 pick-and-place tasks covering diverse source regions, target receptacles, manipulated objects, and interaction requirements. These tasks can be divided into two groups:

- **Object Placement with Container Closing (6 tasks):** placing a cup into a drawer, a can into a drawer, a potato into a microwave, milk into a microwave, a bottle into a cabinet, and wine into a cabinet. In addition to grasping and placing the specified object, the robot is required to close the corresponding drawer, microwave, or cabinet to complete the task.
- **Novel-Object Transfer (18 tasks):** moving a specified novel object from a cutting board to a basket, cardboard box, pan, pot, or tiered basket; from a placemat to a basket, bowl, plate, or tiered shelf; from a plate to a bowl, cardboard box, pan, or another plate; and from a tray to a cardboard box, plate, pot, tiered basket, or tiered shelf.

These tasks evaluate a broad range of manipulation capabilities, including language-conditioned object grounding, spatial reasoning, dexterous grasping, cross-container object transfer, bimanual coordination, and long-horizon interaction with articulated furniture. A rollout is considered successful only when the specified object reaches the target receptacle and, for tasks involving articulated containers, the corresponding container is successfully closed.

For a fair comparison, all methods use the same VLA architecture, training data, optimization settings, and evaluation protocol; the only difference is the latent-action modeling method. Each method is evaluated for 50 closed-loop episodes on each of the 24 tasks. Each episode is limited to 720 simulation steps. We report the success rate for each task and the unweighted average success rate across all 24 tasks.

### D. Real-World Experiment Settings

Our realworld experiments are conducted on a robotic platform consisting of an Xarm7 manipulator equipped with a Robotera XHand, resulting in 18 DoF (6 for the Xarm7 and 12 for the XHand) in total. The policy takes RGB observations from a third-view camera and a wrist-mounted camera as visual inputs, as well as language instruction, as shown in Fig 4.

**Tasks.** To evaluate realworld performance, we choose a set of comprehensive manipulation tasks, including: pick and place, push object, press button, and flip cup.

- Pick and Place: the robot was asked to place a specified item into a plate.
- Push Object: the robot was asked to push a designated item forward.
- Press Button: the robot was asked to press a button.
- Flip Cup: the robot was asked to flip the blanket upside down and stand it upright.

These tasks test GALA's language grounding, spatial understanding, and dexterity capabilities. Each method is tested for 50 trials per task under the same task initialization and success criteria, and we report success rates.

**Baselines.** We selected several representative baselines as below. When reproducing baselines, we use the same downstream demonstrations and evaluation settings as GALA.

- **OpenVLA** [1] is an open-source VLA that learns manipulation skills from robotic datasets by predicting discrete action tokens jointly conditioning on visual observations and natural language instructions.
- **UniVLA** [13] learns transferable task-centric latent actions from large-scale videos to enable cross-embodiment pretraining for VLAs.
- **OpenVLA-OFT** [2] proposes an efficient fine-tuning strategy that improves the inference speed and performance of OpenVLA [1].
- $\boldsymbol{\pi_0}$ [4] is a generalist VLA that generates continuous robot actions via flow-matching.
- $\boldsymbol{\pi_{0.5}}$ [5] improves $\pi_0$ with enhanced training and scaling to better support open-world robotic manipulation.
- **HARP-VLA** [14] learns human–robot-aligned representations from temporally aligned human and robot videos, enabling human demonstrations to improve the generalization of downstream VLAs.

Following the settings of the original paper itself, in our realworld experimental settings, OpenVLA and UniVLA uses third-view image as visual input, and OpenVLA-OFT, $\pi_0$, $\pi_{0.5}$ and HARP-VLA uses third-view and wrist view images, which is consistent with GALA.

### E. Implementation Details and Hyperparameters

*a) Stage-1: Latent Action Pretraining:* We train the Stage-1 latent action model using AdamW with a learning rate of $1 \times 10^{-4}$ and a weight decay of $1 \times 10^{-2}$. Training is performed on eight GPUs with a per-GPU batch size of 8, resulting in a total batch size of 64. We use BF16 mixed precision and clip the gradient norm to 0.1. All reported Stage-1 results use the checkpoint at 35,000 optimization steps.

The visual and geometric branches each produce four latent action tokens with a token dimension of 128. The visual and geometric codebook sizes are 16 and 8, respectively. The VQ commitment coefficient is set to 0.25. Each hand point cloud contains 1,024 points, and the coefficient of the Chamfer reconstruction loss is set to 1.0. The forward and backward transition objectives are weighted equally.

Pair-consistent point-cloud augmentation is applied to the start and goal point clouds. Rotation is sampled within $\pm 5^\circ$ along each axis, isotropic scaling is sampled from $[0.97, 1.03]$, and translation is sampled from $[-0.01, 0.01]$ along each normalized coordinate axis.

*b) Stage-2: Geometry-Aware VLA Co-Training:* We initialize the geometry-aware latent-action tokenizer from the Stage-1 checkpoint at 35,000 optimization steps and keep it frozen throughout Stage-2. The VLA is trained for 20,000 optimization steps with a learning rate of $5 \times 10^{-5}$. Training is performed on eight GPUs with a per-GPU batch size of 32 and one gradient accumulation step, resulting in an effective batch size of 256. Model checkpoints are saved every 4,000 steps.

The policy predicts absolute joint actions and is jointly supervised by the continuous action-generation objective and the discrete latent-action objective. We use four supervised latent action tokens. The action loss is assigned a weight of 1.0, while the auxiliary bridge loss is assigned a weight of 0.1. Gradients from the action objective are propagated through the vision-language backbone rather than being detached. During co-training, we optimize the LLM, multimodal projector, diffusion action model, and bridge embeddings, while keeping the visual encoder and the visual component of the bridge frozen.

We use FlashAttention-2 for attention computation and disable FSDP activation checkpointing. The training data consist of all 24 RoboCasa GR-1 teleoperation datasets in the LeRobot v2.0 format,

TABLE VI

HYPERPARAMETERS FOR STAGE-1 LATENT ACTION PRETRAINING.

| Hyperparameter | Value |
|---|---|
| Visual action tokens | 4 |
| Geometric action tokens | 4 |
| Latent dimension | 128 |
| Visual codebook size | 16 |
| Geometric codebook size | 8 |
| VQ commitment coefficient | 0.25 |
| Points per hand | 1,024 |
| Chamfer loss coefficient | 1.0 |
| Forward/backward loss weights | 1.0 / 1.0 |
| Rotation augmentation | $\pm 5^\circ$ |
| Scale augmentation | $[0.97, 1.03]$ |
| Translation augmentation | $[-0.01, 0.01]$ |
| Optimizer | AdamW |
| Learning rate | $1 \times 10^{-4}$ |
| Weight decay | $1 \times 10^{-2}$ |
| Per-GPU batch size | 8 |
| Number of GPUs | 8 |
| Effective batch size | 64 |
| Training precision | BF16 mixed |
| Gradient clipping | 0.1 |
| Training steps | 35,000 |

with uniform sampling weights across tasks. We reserve 1% of each dataset for validation and additionally sample 0.1% for denoised validation, using a fixed random seed of 42. Standard validation is performed every 200 optimization steps, while denoised evaluation is performed every 10,000 steps. Each evaluation uses 10 batches with a batch size of 32.

For the GR-1-only experiments, we adopt the settings described above. For the multi-embodiment experiments, we double both the per-device batch size and the number of training steps to achieve more thorough convergence on multi-embodiment data.

### *F. Training Data*

Stage-1 latent action pretraining and Stage-2 policy co-training use the same training corpus, summarized in Table VIII. For datasets with multiple processed variants, we retain only the camera-based version of EgoDex and HOI4D and the teleoperated version of RoboCasa-GR1.

TABLE VII

HYPERPARAMETERS FOR STAGE-2 GEOMETRY-AWARE VLA CO-TRAINING.

| Hyperparameter | Value |
|---|---|
| Stage-1 tokenizer checkpoint | 35,000 steps |
| Supervised latent action tokens | 8 |
| Action representation | Absolute |
| Action loss weight | 1.0 |
| Bridge loss weight | 0.1 |
| Detach VLM for action loss | No |
| Tune LLM | Yes |
| Tune multimodal projector | Yes |
| Tune diffusion action model | Yes |
| Tune bridge embedding | Yes |
| Tune visual encoder | No |
| Tune bridge visual module | No |
| Attention implementation | FlashAttention-2 |
| Learning rate | $5 \times 10^{-5}$ |
| Per-GPU batch size | 32 |
| Number of GPUs | 8 |
| Gradient accumulation steps | 1 |
| Effective batch size | 256 |
| Training steps | 20,000 |
| FSDP activation checkpointing | Disabled |

TABLE VIII

TRAINING DATA USED IN BOTH STAGE-1 AND STAGE-2. FRAME COUNTS ARE COMPUTED OVER ALL RETAINED CAMERA STREAMS AFTER PREPROCESSING.

| Dataset | Embodiment | Frames |
|---|---|---|
| DROID | Gripper | 27,044,326 |
| EgoDex | Human hand | 15,649,158 |
| HOI4D | Human hand | 891,600 |
| RoboCasa-GR1 | RoboCasa hand | 5,820,277 |
| V2R-Robot | XHand | 987,915 |
| V2R-Human | Human hand | 231,727 |
| **Total** | | **50,625,003** |