Appendix

A. Training Details

A.1. UniDex-VLA Flow-Matching Loss

To train UniDex-VLA, we minimize a conditional flow-matching loss:

$$L^\tau(\theta) = \mathbb{E}_{p(A_t|o_t),\, q(A_t^\tau|A_t)} \left[\left\| v_\theta(A_t^\tau, o_t) - u(A_t^\tau \mid A_t) \right\| \right], \tag{5}$$

where $\tau \in [0,1]$ and $q(A_t^\tau \mid A_t) = \mathcal{N}\big(\tau A_t,\ (1-\tau)I\big)$ is a linear-Gaussian probability path. We sample $A_t^\tau = \tau A_t + (1-\tau)\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ and compute the target conditional vector field $u(A_t^\tau \mid A_t) = A_t - \epsilon$. The network is trained such that the predicted vector field $v_\theta(A_t^\tau, o_t)$ approximates $u(A_t^\tau \mid A_t)$.

At inference time, we integrate the learned vector field using a forward Euler scheme to generate a denoised action chunk:

$$A_t^{\tau+\delta} \;=\; A_t^\tau + \delta\, v_\theta(A_t^\tau,\, o_t)\,,$$

with step size $\delta = 0.1$ and initial condition $A_t^0 \sim \mathcal{N}(0, I)$.

A.2. UniDex-VLA Pretraining

During pre-training, we use 8 NVIDIA H800 GPUs with a total batch size of 128. The model used for subsequent post-training is trained for 3 epochs ($\sim$ 30k steps), which takes around 24 hours. We adopt the AdamW optimizer and a cosine learning-rate scheduler with an initial learning rate of 1e-4. The learning rate is decayed by a factor of 0.95 at the 2nd epoch. The weight decay is set to 1e-10, and we apply gradient clipping with a maximum norm of 1.0.

A.3. UniDex-VLA Post-training

During post-training, we use 2 NVIDIA H800 GPUs for each task, with a total batch size of 8. We use the AdamW optimizer without a learning rate scheduler and set the initial learning rate to 2.5e-5. For common data, we train the model for 50 epochs ($\sim$ 3k steps), which takes around 4 hours. For DemoGen [59] augmented data, we train the model for 2 epochs ($\sim$ 1.8k steps), which takes around 2.5 hours. The weight decay is set to 1e-10, and we again use gradient clipping with a maximum norm of 1.0.

A.4. Baselines

For all baselines (DP [13], DP3 [67], and π_0 [7]), we post-train the models until convergence on the validation set.

For DP [13] and DP3 [67], we use the AdamW optimizer with an initial learning rate of 1e-4. The state horizon is set to 4 and the action horizon to 32. We use a batch size of 32 and train for 400 epochs.

For π_0 [7], we use the AdamW optimizer with an initial learning rate of 2.5e-5. The batch size is set to 8 and the model is trained for 50 epochs. The number of diffusion steps is set to 10.

For our UniDex-VLA baseline without pretraining, we use the same training hyperparameters as UniDex-VLA with pretraining.

B. Human-in-the-loop Retargeting GUI

To minimum human efforts in our human-in-the-loop retargeting process, we develop a human-friendly web-based GUI, as shown in Fig. 14. Through this interface, users can adjust dummy base links, IK parameters, and other retargeting settings to obtain satisfactory robot trajectories.

C. FAAS Details

Here we show the details for the 32 dimensions encoding dexterous hand joints. Dimensions 0–4, 5–9, 10–14, 15–19, and 20–24 correspond to the thumb, index, middle, ring, and little fingers, respectively. Dimensions 25–26 are reserved for extra wrist joints of Shadow hands. Dimensions 27–31 are left unused for new hands. The detailed joint mappings of the robotic hands used in FAAS are shown in Fig. 15.

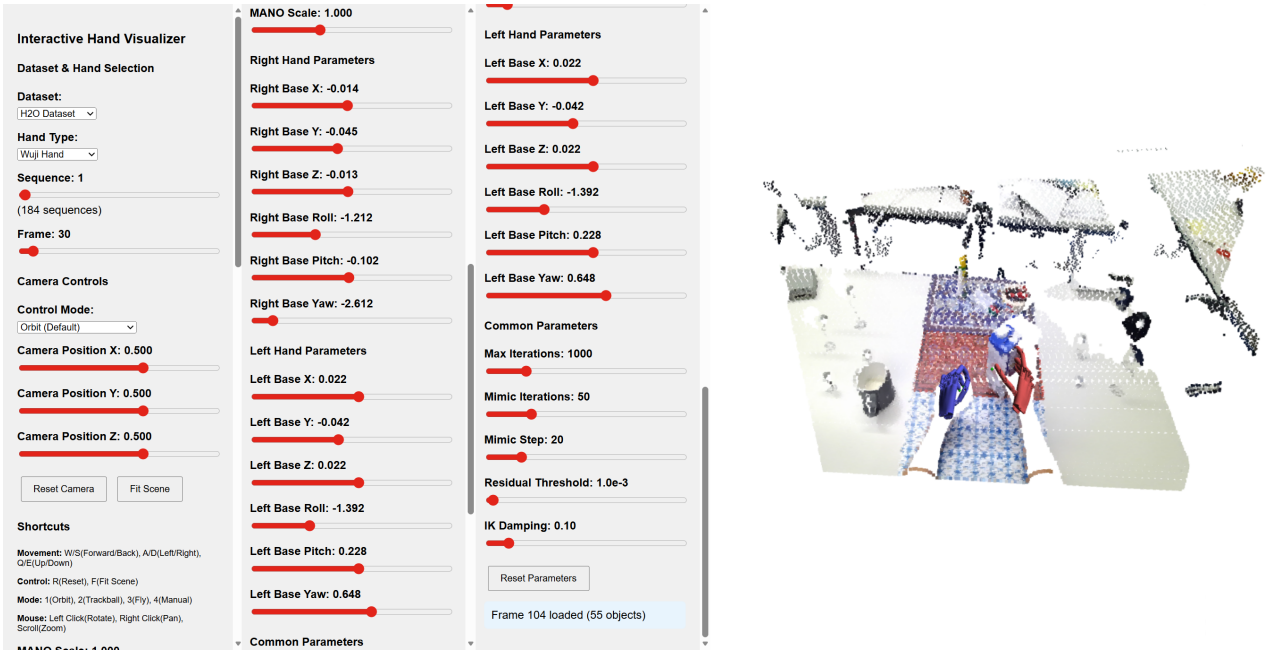


Figure 14. Human-friendly web-basedGUI for retargeting human demonstrations to robot executions. Users can adjust the IK parameters, dummy links, and other settings through the GUI to obtain satisfactory retargeted robot trajectories.

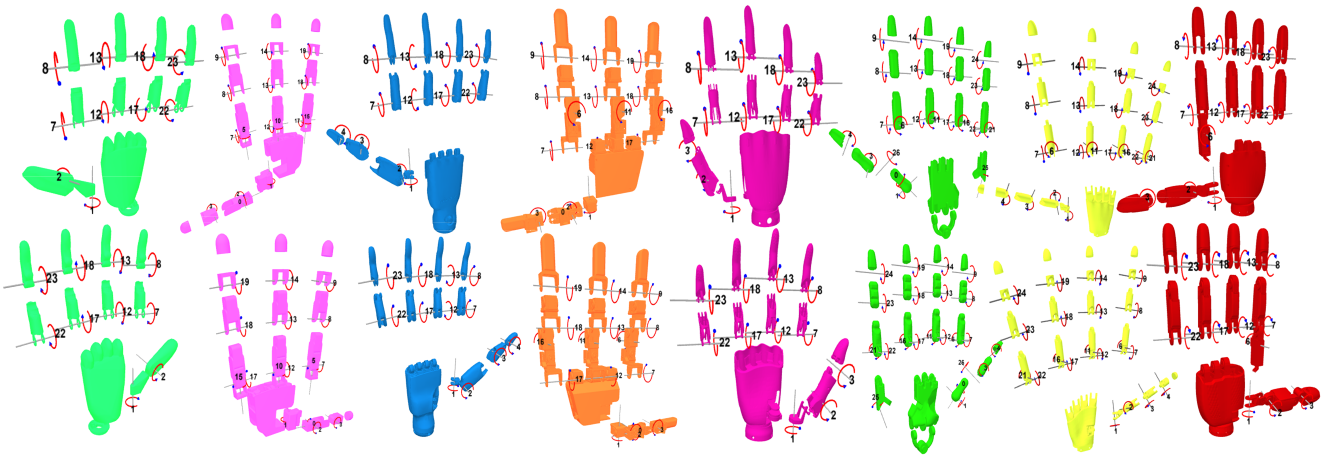


Figure 15. Joint mappings of different robotic hands used in FAAS. From left to right are Ability, Allegro, Inspire, Leap, Oymotion, Shadow, Wuji, and Xhand. The two rows show different views of the joint mappings on the right hand.

D. UniDex-Cap Setup Calibration

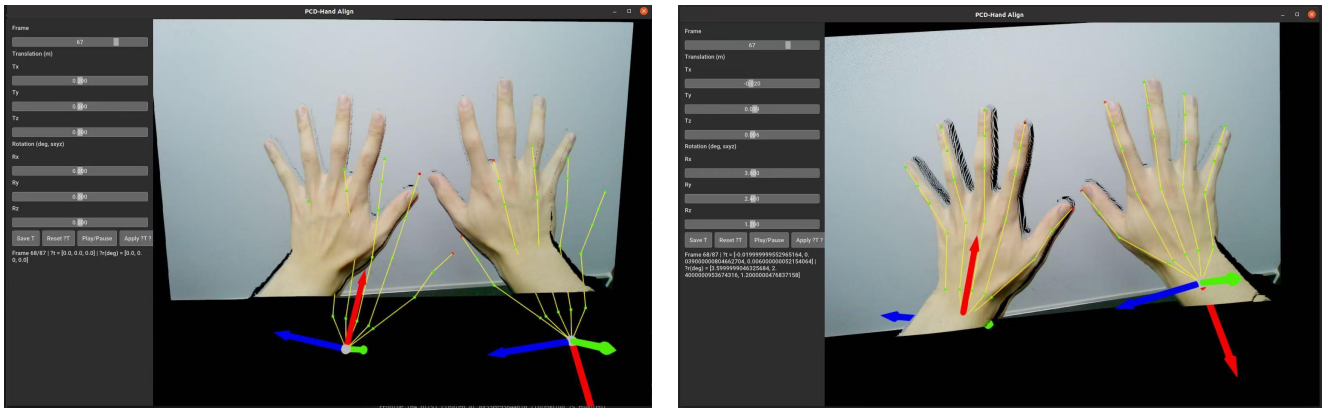
UniDex-Cap combines an Apple Vision Pro (for hand and head poses, denoted $\{P_{\text{VP}}\}$) and an Intel RealSense L515 (for RGB-D). Because Vision Pro does not expose third-party RGB-D video recording, we physically couple the two sensors with a custom 3D-printed mount that rigidly fixes their relative pose. This mechanical constraint ensures that the extrinsic transform between the Vision Pro and the RealSense remains stable for a given user.

As shown in Fig. 16, we provide a lightweight GUI to estimate the remaining constant extrinsics with minimal manual effort. The user records a short calibration clip and then uses a slider-based interface to adjust the hand and wrist poses in the Vision Pro coordinate frame—visualized as a skeleton—until they align with the 3D hand point cloud captured by the RealSense camera. The slider values directly correspond to the transform $T_{\text{RS}}^{\text{VP}}$. Once this transform is determined, all Vision

Pro poses are converted into the RealSense camera frame, yielding temporally aligned hand and head trajectories:

$$P_{\mathrm{RS}} \;=\; T_{\mathrm{RS}}^{\mathrm{VP}}\, P_{\mathrm{VP}}, \tag{6}$$

where P_{VP} and P_{RS} are represented in homogeneous coordinates. This pipeline produces temporally synchronized, geometrically consistent annotations suitable for downstream retargeting and post-training.



(a) Before Calibration **(b) After Calibration**

Figure 16. GUI for UniDex-Cap calibration. (a) shows the initial state before calibration; (b) shows the calibrated result where the hand poses captured by Vision Pro align with the 3D point cloud captured by the RealSense L515 camera.

E. Core Contribution List

The main contributions of the core contributors are as follows:
Gu Zhang: Project lead. Developed the overall dataset construction pipeline, model architecture, and unified action space; built the robot system infrastructure; and wrote the paper.
Qicheng Xu: Led VLA model training; optimized the dataset construction and policy inference pipelines; and contributed to paper writing.
Haozhe Zhang: Led dataset processing; improved the robot system and the human–robot data capture pipeline; and contributed to paper writing.
Jianhan Ma: Implemented retargeting algorithms; and developed dataset visualizations and contributed to early-stage exploration.
Long He: Implemented DemoGen algorithm; and contributed to paper writing.
Yiming Bao: Collected robot data and human data; and contributed to DemoGen algorithm implementation.