

UniHM: Unified Dexterous Hand Manipulation with Vision Language Model

Appendix

A USE OF LLMs

We used an GPT to polish the experiments and introduction sections. We also use the Nano to generate the icon in Figure 1.

B IMPLEMENTATION DETAILS

B.1 RETARGETING

Hand Pose Retargeting. The DexYCB (Chao et al., 2021) and OakInk (Yang et al., 2022) datasets provide MANO pose sequences (Loper et al., 2023), object point clouds, and associated object trajectories. We employ Dex-Retargeting (Qin et al., 2023) to map MANO joint configurations to multiple robotic hands via a differentiable inverse-kinematics objective with joint limits and device-specific kinematic calibration. Given a MANO pose $q_{\text{MANO}}(t)$, the device configuration is:

$$q_{\text{dev}}(t) = \arg \min_{q \in [q_{\min}, q_{\max}]} \sum_k w_k \|f_k(q) - f_k(\phi_{\text{dev}}(q_{\text{MANO}}(t)))\|_2^2 + \lambda \|q - q_{\text{prev}}\|_2^2, \quad (\text{B1})$$

where f_k denote forward-kinematics constraints on fingertip/phalange keypoints, ϕ_{dev} aligns MANO and device frames and scales link lengths, and q_{prev} is the previous solution for temporal smoothness. We generate retargeted sequences for Allegro, Shadow, Schunk, LEAP, and Ability hands, and for the Panda gripper (mapping the thumb-index aperture to a scalar opening width via linear scaling with clamping).

Target Trajectory Generation. Let $M_{\text{ext}} \in SE(3)$ denote the camera extrinsics that map world coordinates to the camera frame, and let $\mathcal{T}_{\text{obj}}^{\text{cam}}(t) \in SE(3)$ be the object pose in the camera frame at time t . The object target trajectory in the world frame used during training is

$$\mathcal{T}_{\text{tar}}(t) = M_{\text{ext}}^{-1} \mathcal{T}_{\text{obj}}^{\text{cam}}(t). \quad (\text{B2})$$

We time-align $\mathcal{T}_{\text{tar}}(t)$ with the retargeted hand poses using the dataset timestamps.

B.2 UNIFIED HAND TOKENIZER

Unified Codebook Training. We adopt a shared VQ-VAE codebook $\mathcal{Z} = \{\mathbf{e}_k\}_{k=1}^K$ initialized with Shadow Hand due to its data availability and widely used. For any hand type h , denote its encoder and decoder by E_h and D_h . Given an input sequence chunk $\mathbf{x}^{(h)}$, the encoder output, quantization, and reconstruction follow the main text:

$$\mathbf{z}_e^{(h)} = E_h(\mathbf{x}^{(h)}), \quad (\text{B3})$$

$$c = Q(\mathbf{z}_e^{(h)}) = \arg \min_{k \in [K]} \|\mathbf{z}_e^{(h)} - \mathbf{e}_k\|_2^2, \quad (\text{B4})$$

$$\mathbf{z}_q^{(h)} = \mathbf{e}_c, \quad (\text{B5})$$

$$\hat{\mathbf{x}}^{(h)} = D_h(\mathbf{z}_q^{(h)}) = D_h(\mathbf{e}_c), \quad (\text{B6})$$

where $Q(\cdot)$ performs a nearest neighbor lookup in the codebook, finding the code vector with the minimum squared Euclidean distance. We train with the standard reconstruction and VQ losses:

$$\mathcal{L}_{\text{rec}} = \|\mathbf{x}^{(h)} - D_h(\mathbf{z}_q^{(h)})\|_2^2, \quad (\text{B7})$$

$$\mathcal{L}_{\text{vq}} = \left\| \text{sg} \left[\mathbf{z}_e^{(h)} \right] - \mathbf{z}_q^{(h)} \right\|_2^2 + \beta \left\| \mathbf{z}_e^{(h)} - \text{sg} \left[\mathbf{z}_q^{(h)} \right] \right\|_2^2, \quad (\text{B8})$$

where $\text{sg}[\cdot]$ is the stop-gradient operator and $\beta > 0$ is the commitment weight.

To avoid codebook collapse with a purely statistical procedure, we maintain a “cold-code” table based on per-epoch usage counts. Let $n_k^{(t)}$ be the usage of code k during epoch t , computed from token indices $\{c_b^{(t)}\}_{b=1}^{B^{(t)}}$:

$$n_k^{(t)} = \sum_{b=1}^{B^{(t)}} \mathbb{1} \left[c_b^{(t)} = k \right], \quad (\text{B9})$$

where $\mathbb{1}(\cdot)$ denotes the indicator function that evaluates to 1 if the condition inside its brackets is true, and 0 otherwise. Define the cold set at epoch t by a count threshold τ_c :

$$\mathcal{S}^{(t)} = \left\{ k \in [K] \mid n_k^{(t)} < \tau_c \right\}. \quad (\text{B10})$$

At scheduled refresh epochs $t \in \mathcal{E}_{\text{reset}}$, we selectively update only the codes belonging to the cold set. Collect a buffer $\mathcal{B}^{(t)}$ of encoder outputs since the previous refresh and run K-Means with $R = |\mathcal{S}^{(t)}|$ to obtain centroids $\{\mu_r\}_{r=1}^R$. We directly replace the cold codes with the newly computed centroids. With an injective assignment $\pi : \mathcal{S}^{(t)} \rightarrow \{1, \dots, R\}$:

$$\mathbf{e}_k \leftarrow \mu_{\pi(k)} \quad \text{for } k \in \mathcal{S}^{(t)}, \quad \mathbf{e}_k \leftarrow \mathbf{e}_k \quad \text{for } k \notin \mathcal{S}^{(t)}. \quad (\text{B11})$$

Thus, only the codes in the cold-code table are refreshed at those epochs, while frequently used codes remain unchanged. The objective for this stage is:

$$\mathcal{L}_{\text{loss}} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{vq}}. \quad (\text{B12})$$

Multi-Encoder and Decoder Training. To integrate a new hand morphology, we first align the new encoder to the reference encoder via retargeted pairs. Let E_{ref} be the reference (Shadow Hand) encoder and $(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{ref}})$ be a retargeted pair:

$$\mathcal{L}_{\text{distill}} = \|E_{\text{new}}(\mathbf{x}_{\text{new}}) - E_{\text{ref}}(\mathbf{x}_{\text{ref}})\|_2^2. \quad (\text{B13})$$

After encoder alignment, we train the new encoder–decoder pair within the shared VQ-VAE using the same reconstruction and VQ losses:

$$\mathcal{L}_{\text{total}}^{(h)} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{vq}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}} \quad (\text{B14})$$

, where $\lambda_{\text{distill}} \geq 0$ controls the contribution of the distillation term during the encoder alignment stage.

Training Details. We train a unified codebook with capacity $K = 8192$ on DexYCB sequences. Since the inputs are 1D pose and short-motion signals, we instantiate encoders/decoders with either MLP or 1D convolutional backbones and ablate both choices, as shown in Table B1. When training the reference encoder, we set the $\beta = 0.25$ and train the network with a learning rate $1e-4$. We also update the codebook at epochs 50, 100, 150, For the multi-encoder and multi-decoder training, the cold-code threshold τ_c is set to 1, and the distillation weight λ_{distill} is set to 0.1.

Table B1: A comparison of the validation set performance for the MLP and 1D-Conv models.

		Allegro	Shadow	Schunk	LEAP	Ability	Panda Gripper	Overall
MAE	MLP	0.0268	0.0450	0.0293	0.0348	0.0432	0.0221	0.0350
	1D-Conv	0.0216	0.0297	0.0221	0.0257	0.0327	0.0182	0.0256
RMSE	MLP	0.0545	0.0886	0.0656	0.0656	0.0913	0.0630	0.0736
	1D-Conv	0.0465	0.0654	0.0551	0.0531	0.0705	0.0519	0.0581

B.3 PHYSICAL OPTIMIZATION

Physical Model. We expand the contact term used in the per-frame objective. Let $s_i(q_t)$ denote the i -th fingertip position (forward kinematics) in the world frame and $\mathcal{T}_{\text{tar}}(t)^{-1}$ be the object-frame transform at time t . Query the nearest neighbor $(\mathbf{p}_i, \mathbf{n}_i)$ on $\mathcal{P}_{\text{obj}}$ in the object frame, and define the signed point-to-plane distance:

$$d_i(q_t) = \mathbf{n}_i^T (\mathcal{T}_{\text{tar}}(t)^{-1} s_i(q_t) - \mathbf{p}_i). \quad (\text{B15})$$

We adopt an asymmetric, smooth penalty $f(d)$ that is continuous and slope-matched at $d = 0$:

$$f(d) = \begin{cases} \frac{\alpha}{2}d^2, & d \geq 0 \text{ (outside)} \\ \frac{\alpha}{k^2} \left(e^{-kd} + kd - 1 \right), & d < 0 \text{ (inside)} \end{cases} \quad (\text{B16})$$

At $d = 0$, f is continuous and has matched first derivatives:

$$\lim_{d \rightarrow 0^\pm} f(d) = 0, \quad \lim_{d \rightarrow 0^+} f'(d) = 0, \quad \lim_{d \rightarrow 0^-} f'(d) = \frac{\alpha}{k}(1 - e^{-k \cdot 0}) = 0. \quad (\text{B17})$$

Moreover, $f''(0^-) = f''(0^+) = \alpha e^{-k \cdot 0} \geq 0$, so f is twice-differentiable and convex. Stacking per-fingertip residuals gives:

$$r_{\text{contact},i}(q_t) = \sqrt{2\lambda_c f(d_i(q_t))}, \quad \mathcal{E}_{\text{contact}}(q_t) = \frac{1}{2} \|r_{\text{contact}}(q_t)\|_2^2 = \lambda_c \sum_i f(d_i(q_t)), \quad (\text{B18})$$

with $\lambda_c > 0$. For Gauss–Newton, we linearize the residuals at the current iterate. Let $x_i(q_t) = \mathcal{T}_{\text{tar}}(t)^{-1} s_i(q_t)$ and $J_{x,i}(q_t) = \partial x_i / \partial q_t$ (the kinematic Jacobian in the object frame). Treating $(\mathbf{p}_i, \mathbf{n}_i)$ as fixed within an inner iteration:

$$\frac{\partial d_i(q_t)}{\partial q_t} = \mathbf{n}_i^T J_{x,i}(q_t), \quad \frac{\partial r_{\text{contact},i}}{\partial q_t} = \sqrt{2\lambda_c} \frac{f'(d_i(q_t))}{2\sqrt{f(d_i(q_t))} + \epsilon^2} \mathbf{n}_i^T J_{x,i}(q_t), \quad (\text{B19})$$

which yields the contact Jacobian rows that form $J_t = \partial r_{\text{contact}}(q_t) / \partial q_t$ with a small $\epsilon > 0$.

Generative Prior and Temporal Prior. We keep the generative and temporal priors from the main text and unify their roles as dynamic smoothers from two information sources—semantic intent (generator) and history (velocity/acceleration). The generative prior anchors q_t to q_t^{gen} :

$$\mathcal{E}_{\text{gen}}(q_t, q_t^{\text{gen}}) = \frac{1}{2} (q_t - q_t^{\text{gen}})^T \mathbf{W}_{\text{gen}} (q_t - q_t^{\text{gen}}), \quad \mathbf{W}_{\text{gen}} \succ \mathbf{0}. \quad (\text{B20})$$

The temporal prior penalizes first- and second-order differences:

$$\mathcal{E}_{\text{time}}(q_t, q_{t-1}^{\text{opt}}, q_{t-2}^{\text{opt}}) = \frac{1}{2} \|q_t - q_{t-1}^{\text{opt}}\|_{\mathbf{W}_{\text{vel}}}^2 + \frac{1}{2} \|(q_t - q_{t-1}^{\text{opt}}) - (q_{t-1}^{\text{opt}} - q_{t-2}^{\text{opt}})\|_{\mathbf{W}_{\text{acc}}}^2, \quad (\text{B21})$$

where $\mathbf{W}_{\text{vel}} \succ \mathbf{0}$ and $\mathbf{W}_{\text{acc}} \succ \mathbf{0}$ (often $\mathbf{W}_{\text{vel}} = \lambda_{\text{vel}} I$, $\mathbf{W}_{\text{acc}} = \lambda_{\text{acc}} I$). Define the per-frame objective:

$$\mathcal{E}_t(q_t, q_t^{\text{gen}}, q_{t-1}^{\text{opt}}, q_{t-2}^{\text{opt}}) = \mathcal{E}_{\text{contact}}(q_t) + \mathcal{E}_{\text{gen}}(q_t, q_t^{\text{gen}}) + \mathcal{E}_{\text{time}}(q_t, q_{t-1}^{\text{opt}}, q_{t-2}^{\text{opt}}). \quad (\text{B22})$$

Let $r_{\text{contact}}(q_t)$ denote the stacked contact residuals. Linearizing $r_{\text{contact}}(q_t + \Delta q_t) \approx r_{\text{contact}}(q_t) + J_t \Delta q_t$ and expanding the two quadratic priors around q_t give:

$$\mathcal{E}_t(q_t + \Delta q_t) \approx \frac{1}{2} \|r_{\text{contact}}(q_t) + J_t \Delta q_t\|_2^2 + \frac{1}{2} \Delta q_t^T (\mathbf{W}_{\text{gen}} + \mathbf{W}_{\text{vel}} + \mathbf{W}_{\text{acc}}) \Delta q_t + \Delta q_t^T \tilde{\mathbf{W}}, \quad (\text{B23})$$

where:

$$\tilde{\mathbf{W}} \triangleq \mathbf{W}_{\text{gen}}(q_t - q_t^{\text{gen}}) + \mathbf{W}_{\text{vel}}(q_t - q_{t-1}^{\text{opt}}) + \mathbf{W}_{\text{acc}}((q_t - q_{t-1}^{\text{opt}}) - (q_{t-1}^{\text{opt}} - q_{t-2}^{\text{opt}})). \quad (\text{B24})$$

Setting the gradient w.r.t. Δq_t to zero and adding Levenberg–Marquardt damping $\lambda \geq 0$ yields the normal equations:

$$(J_t^T J_t + \mathbf{W}_{\text{gen}} + \mathbf{W}_{\text{vel}} + \mathbf{W}_{\text{acc}} + \lambda I) \Delta q_t = -J_t^T r_{\text{contact}}(q_t) - \tilde{\mathbf{W}}. \quad (\text{B25})$$

For $t < 2$, we use boundary priors consistent with the main text, e.g., $q_{-2}^{\text{opt}} = q_{-1}^{\text{opt}} = q_0^{\text{gen}}$.

Optimization Procedure. Given the decoded grasping trajectory $\mathcal{Q}_{\text{gen}}$, the object point cloud $\mathcal{P}_{\text{obj}}$, and the target pose trajectory $\mathcal{T}_{\text{tar}}(t)$, we refine each frame by Gauss–Newton with LM damping while freezing nearest-neighbor correspondences inside each inner iteration. In our implementation, we set $\alpha = k = 1$ and use a small positive constant ϵ to handle cases where $f(d)$ approaches zero.

In practical environments, obtaining precise point clouds is often challenging. A significant advantage of employing the kernel function $f(d)$ is the inherent robustness it provides to the segmentation process.

As shown in Fig. B1, the kernel function is relatively flat in a neighborhood of $d = 0$, which implies that, even when the segmented point cloud is contaminated with noise, the resulting cost term does not deviate substantially from that constructed from the ground-truth point cloud, and thus the optimization still converges to the correct solution in the presence of noisy inputs. We further demonstrate the optimization results under noisy conditions (for a clearer illustration of the grasp point, we have omitted the remaining structure of the dexterous hand, plotting only the object point cloud and the positions of the hand’s fingers), as shown in Fig. B2.

Algorithm 1 Physical-guided dynamic refinement

Require: Generated sequence $\mathcal{Q}_{\text{gen}} = \{q_t^{\text{gen}}\}_{t=0}^T$, point cloud $\mathcal{P}_{\text{obj}}$, target poses $\{\mathcal{T}_{\text{tar}}(t)\}_{t=0}^T$, weights λ_c , $\mathbf{W}_{\text{gen}}$, $\mathbf{W}_{\text{vel}}$, $\mathbf{W}_{\text{acc}}$, damping $\lambda \geq 0$

Ensure: Refined sequence $\mathcal{Q}_{\text{opt}} = \{q_t^{\text{opt}}\}_{t=0}^T$

- 1: Set boundary priors $q_{-2}^{\text{opt}} = q_{-1}^{\text{opt}} = q_0^{\text{gen}}$
- 2: **for** $t = 0$ to T **do**
- 3: Initialize $q_t \leftarrow q_t^{\text{gen}}$
- 4: **repeat**
- 5: Transform fingertips $x_i(q_t) = \mathcal{T}_{\text{tar}}(t)^{-1} s_i(q_t)$
- 6: For each fingertip i , find $(\mathbf{p}_i, \mathbf{n}_i) = \text{NN}(x_i(q_t), \mathcal{P}_{\text{obj}})$ and determine inside/outside sign
- 7: Compute $d_i(q_t)$, $r_{\text{contact}}(q_t)$, and $J_t = \partial r_{\text{contact}} / \partial q_t$ with correspondences fixed
- 8: Form $\tilde{\mathbf{W}}$ using $q_t, q_{t-1}^{\text{opt}}, q_{t-2}^{\text{opt}}$
- 9: Solve $(J_t^T J_t + \mathbf{W}_{\text{gen}} + \mathbf{W}_{\text{vel}} + \mathbf{W}_{\text{acc}} + \lambda I) \Delta q_t = -J_t^T r_{\text{contact}}(q_t) - \tilde{\mathbf{W}}$
- 10: Update: $q_t \leftarrow q_t + \Delta q_t$; adjust λ by decrease/increase of $\mathcal{E}_t$
- 11: **until** converged or max iters
- 12: $q_t^{\text{opt}} \leftarrow q_t$
- 13: **end for**

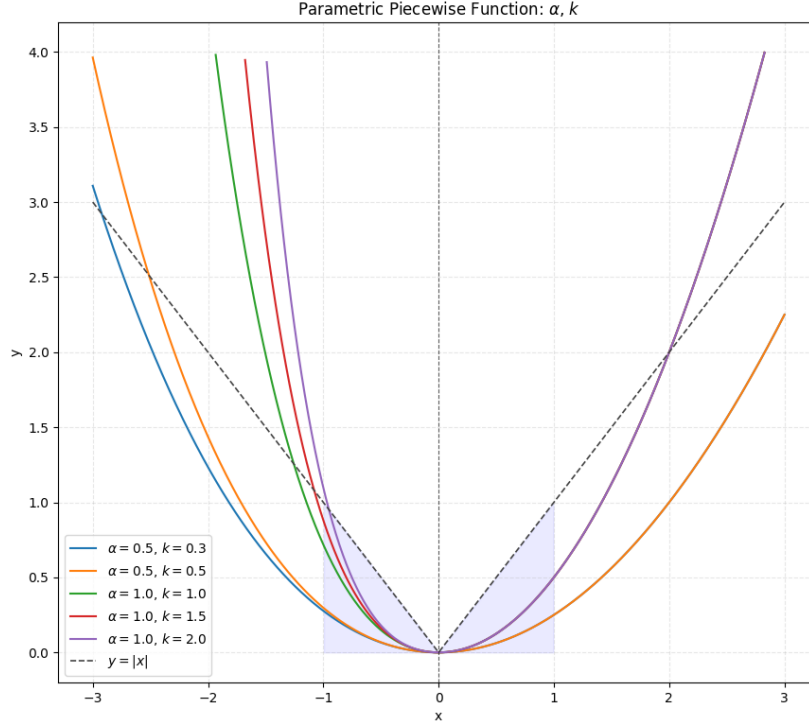


Figure B1: The function plots under varying values of α and k are displayed. Note that α and k control the curve behavior for $x > 0$ and $x < 0$, respectively. For comparison, the curve of $y = |x|$ is plotted using a dashed line.

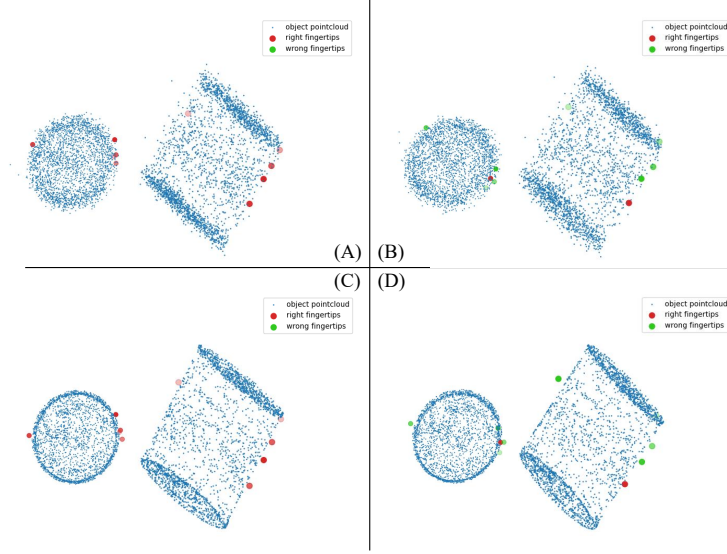


Figure B2: Optimization results when the point cloud includes noise. Here, the black points represent the object point cloud, and the red/green points denote the positions of the dexterous hand’s fingertips. (A) Optimization using the $f(d)$ kernel, visualized on the noisy input. (B) Optimization using the Euclidean distance, visualized on the noisy input. (C) The $f(d)$ kernel optimization result projected onto the clean noise-free point cloud. (D) The Euclidean distance optimization result projected onto the noise-free point cloud.

C MODEL INSTRUCTION

C.1 TEXT TOKENIZER

We employ a pretrained language model from the Qwen series, specifically Qwen3, as the core component for our language input pipeline. This choice is motivated by its strong performance across a wide range of natural language tasks and its robust, efficient tokenizer.

The Qwen3 tokenizer operates on a byte-level Byte-Pair Encoding algorithm (Sennrich et al., 2015). This method ensures that all possible input characters are representable, and it is particularly effective at handling diverse linguistic inputs, including code and technical terms, without resorting to unknown tokens. The tokenizer’s vocabulary size is V_{tok} , and each input sequence S is tokenized into a sequence of tokens $T = (t_1, t_2, \dots, t_N)$, where N is the sequence length. The tokenization process can be formally expressed as:

$$\text{tokenize}(S) \rightarrow T. \quad (\text{C1})$$

This token sequence T is then converted into a sequence of token embeddings, which serve as the input to the subsequent layers of our model. We utilize the tokenizer’s built-in padding and truncation functionalities to handle variable-length sequences, ensuring a consistent input shape for the model.

C.2 POINT CLOUD ENCODER

We use a PointNet-based architecture (Qi et al., 2017a;b) as our point cloud feature extractor, which is a key component of our model. The network directly consumes raw point cloud data, which is an unordered set of n points, with each point represented by its (x, y, z) coordinates. The PointNet architecture addresses the permutation invariance of point clouds by using a symmetric function to aggregate information from each point.

Specifically, the network first applies a shared multi-layer perceptron to each point individually to transform the input features. This is followed by a max-pooling layer, which acts as a symmetric function to aggregate the point-wise features into a global feature vector. This process can be concisely described as:

$$f(X) = \max_{i=1,\dots,n} \{h(x_i)\}, \quad (\text{C2})$$

where $X = \{x_1, \dots, x_n\}$ is the input point cloud, h represents the shared MLP, and $f(X)$ is the resulting global feature vector. The network is configured to output several feature vectors, which are then concatenated and fed into the subsequent main model for further processing.

C.3 TRAJECTORY ENCODER

Our model takes the target trajectory $\mathcal{T}_{\text{tar}}$ as input. We encode this trajectory using a two-layer MLP to obtain a feature representation $\mathbf{z}(t)$. The MLP processes the pose $\mathbf{p}_{\text{tar}}(t)$ at each timestep t and is defined as:

$$\mathbf{z}(t) = \sigma_2 (\mathbf{W}_2 \sigma_1 (\mathbf{W}_1 \mathbf{p}_{\text{tar}}(t) + \mathbf{b}_1) + \mathbf{b}_2), \quad (\text{C3})$$

where σ_1 and σ_2 are non-linear activation functions. The hidden layer dimension is set to 512 during training.

C.4 MASKED TRAINING

During training, we progressively mask the ground truth hand poses to encourage the model to learn the temporal relationships within a sequence. This approach, which has been shown to be highly effective in fields like language modeling (e.g., BERT (Liu et al., 2019)) and computer vision (e.g., DINOv2 (Oquab et al., 2023)), allows the model to generate hand poses from a given context.

Specifically, we feed the ground truth trajectory into the VQ-VAE encoder and then use a mask to conceal the corresponding hand poses. The masked hand pose tokens are replaced with a single, learnable token. This masking process can be formalized as:

$$\mathcal{Q}_{\text{masked}} = M \odot E(\mathcal{Q}) + (1 - M) \odot \mathcal{T}_{\text{mask}}, \quad (\text{C4})$$

where $\mathcal{Q}$ is the ground truth hand pose sequence, E is the VQ-VAE encoder, M is a binary mask, and $\mathcal{T}_{\text{mask}}$ is the learnable token.

We implement this masking strategy with a progressive curriculum across training epochs. For the first 20% of epochs, we do not mask any hand poses. From 20% to 80% of the epochs, we linearly increase the masking ratio until it reaches 100%. Finally, during the last 20% of epochs, all hand poses are consistently masked, forcing the network to perform full generation. We train the model for 100 epochs using the AdamW optimizer with a learning rate of $1\text{e-}4$.

C.5 TARGET TRAJECTORY PLANNER

We use CLiPort to plan the target trajectory $\mathcal{T}_{\text{tar}}$. Specifically, we take RGB-D and instructions as input data. Then, based on GenH2R (Wang et al., 2024), we generate a smooth trajectory in the output space, which is then encoded using the aforementioned trajectory encoder and fed into the model.

D ADDITIONAL VISUALIZATION

We employ Sapien (Xiang et al., 2020) as a visualization engine for observing and validating the generated results. We also present some additional visualizations here.

E REAL-WORLD SETUP

The experimental setup utilizes a Franka manipulator arm equipped with Panda Hand, an XHand dexterous hand, and Inspire Hand in Fig.E1. The 7-degree-of-freedom (DoF) Franka provides a

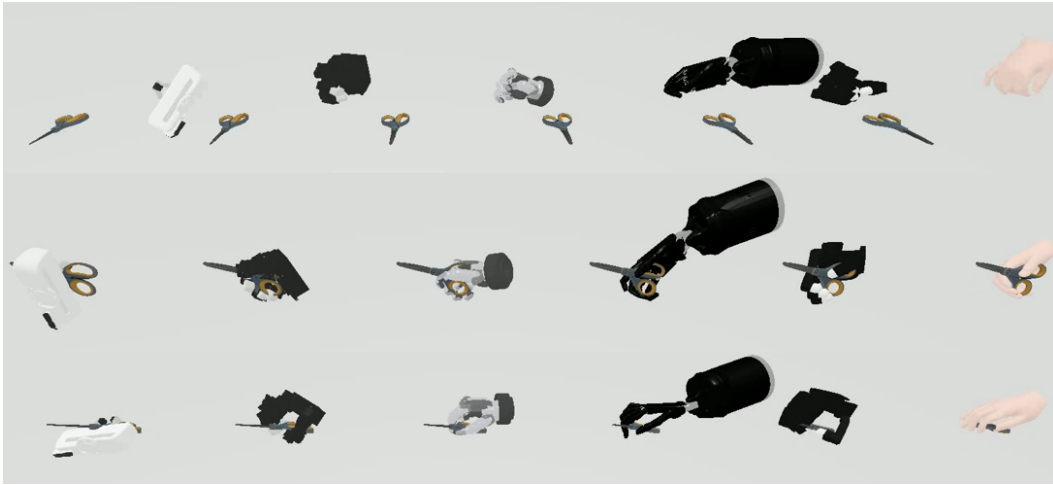


Figure D1: The visualization results of our grasping sequence in Sapien. More real world examples could be found in the video.

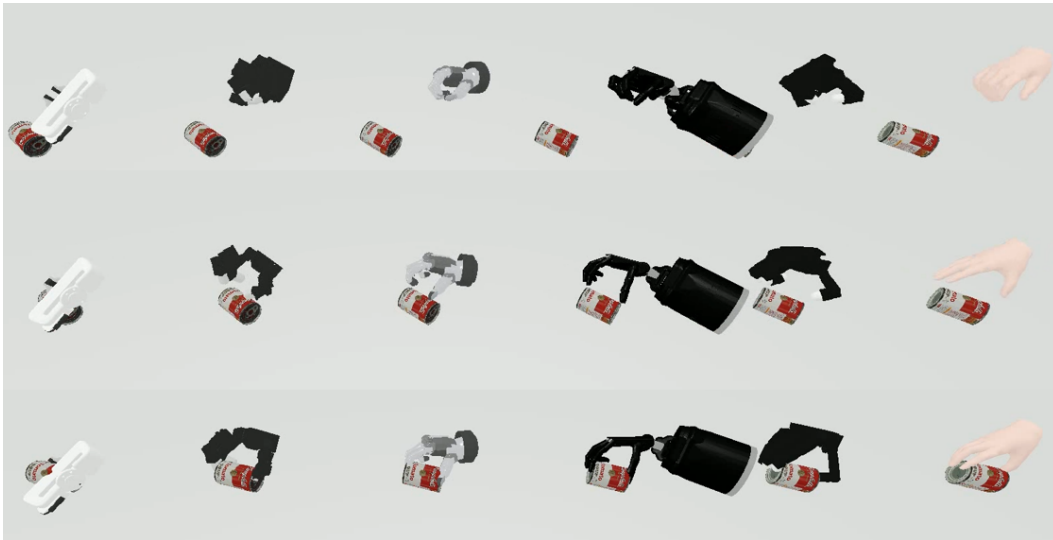


Figure D2: The visualization results of our grasping sequence in Sapien. More real world examples could be found in the video.

workspace comparable to that of a human arm. The 2-DOF Panda Hand, 12-DOF XHand, and 6-DOF Inspire Hand, with overall dimensions similar to a human hand, feature a proportionally elongated pinky finger. This specific design requires an adaptive scaling heuristic to ensure natural and fluid motion.

We employ a Zed camera to acquire RGB-D observations, apply PointSAM(You & Wu, 2025; Wang et al., 2025a; Wu et al., 2025) to segment the target point cloud, and subsequently leverage SnowflakeNet (Xiang et al., 2023; 2021) to reconstruct a complete point cloud that serves as input to our model.

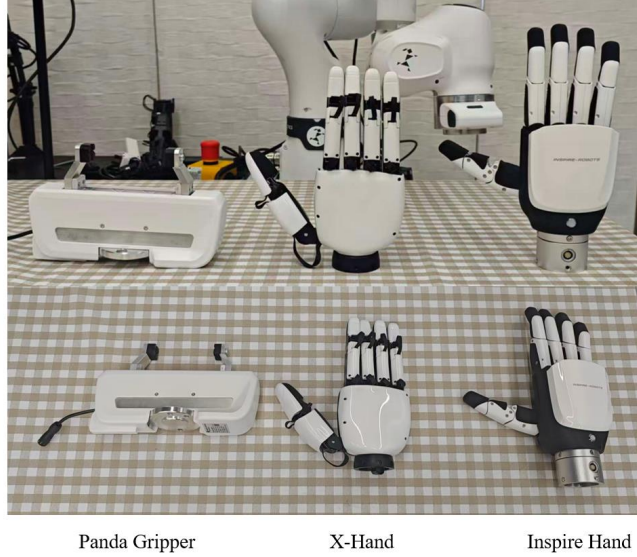


Figure E1: Real-World Cross-embodiment Set up

F EVALUATION METRIC

MPJPE (Mean Per-Joint Position Error). For each sequence, we compute the Euclidean error of every hand joint in every frame, Then average over joints and time:

$$\text{MPJPE}_i = \frac{1}{T_i J} \sum_{t=1}^{T_i} \sum_{j=1}^J \|p_{t,j}^{(i)} - g_{t,j}^{(i)}\|_2. \quad (\text{F1})$$

The final MPJPE reported

$$\text{MPJPE} = \frac{1}{N} \sum_{i=1}^N \text{MPJPE}_i. \quad (\text{F2})$$

where N is the number of test sequences. **FPL (Final Position Location Error)** FPL measures how close the final hand placement is to the ground truth. Let $c_T^{(i)}$ and $\hat{c}_T^{(i)}$ be the 3D positions of the hand root / palm center at the last frame T_i of sequence i :

$$\text{FPL}_i = \|\hat{c}_{T_i}^{(i)} - c_{T_i}^{(i)}\|_2, \quad \text{FPL} = \frac{1}{N} \sum_i \text{FPL}_i. \quad (\text{F3})$$

FOL (Final Orientation Location Error). FOL measures the orientation error of the hand at the final frame. Let $R_{T_i}^{(i)}$ and $\hat{R}_{T_i}^{(i)}$ be the ground-truth and predicted rotation matrices of the hand root; the orientation error is

$$\theta_i = \arccos\left(\frac{\text{trace}((R_{T_i}^{(i)})^\top \hat{R}_{T_i}^{(i)}) - 1}{2}\right), \quad (\text{F4})$$

converted to degrees. We report

$$\text{FOL} = \frac{1}{N} \sum_i \theta_i. \quad (\text{F5})$$

Feature extractor. We first embed every hand-object interaction sequence x (either real or generated) using a pretrained motion encoder $f(\cdot)$ that consumes the full temporal joint trajectory (and object pose) and outputs a fixed-dimensional feature vector $z = f(x) \in \mathbb{R}^d$. We use the same encoder for all methods and compute all distributional metrics in this learned feature space.

FID (Fréchet Inception Distance). Let $\{z_k^{\text{real}}\}_{k=1}^{M_r}$ be the features of real test sequences and $\{z_k^{\text{gen}}\}_{k=1}^{M_g}$ the features of generated sequences. We estimate the empirical means and covariances:

$$\mu_r = \frac{1}{M_r} \sum_k z_k^{\text{real}}, \quad \Sigma_r = \text{Cov}(\{z_k^{\text{real}}\}); \quad (\text{F6})$$

$$\mu_g = \frac{1}{M_g} \sum_k z_k^{\text{gen}}, \quad \Sigma_g = \text{Cov}(\{z_k^{\text{gen}}\}), \quad (\text{F7})$$

and compute

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}). \quad (\text{F8})$$

Diversity. Diversity measures how varied the generated sequences are in the same feature space. Given the set of features $\{z_k\}_{k=1}^M$ for a method, we compute

$$\text{Div} = \frac{2}{M(M-1)} \sum_{1 \leq a < b \leq M} \|z_a - z_b\|_2. \quad (\text{F9})$$