DexRefine: Refine Human Motion to Physically Feasible Robotic Actions

Hyesung Lee, Si-Hwan Heo, Sungwook Yang

***Abstract*— Human motion capture is a scalable data source for dexterous manipulation, but direct retargeting often fails due to embodiment mismatches and missing physical interactions. We present DexRefine, a framework that learns task-space residual actions to refine human motion into physically feasible robot actions. When deployed on a physical robot, DexRefine successfully refines synthetic Human-Object Interaction (HOI) trajectories in real-time, achieving robust sim-to-real transfer. Experiments on a challenging "bottle reorientation" task show our method substantially outperforms a baseline, reducing Object Trajectory and Orientation Error (OTE/OOE) by approximately 52% and 50%, respectively, while also attaining a significantly higher success rate. Our Video Link: https://youtu.be/6SmPZBl0zdM.**

I. INTRODUCTION

Recent advances in artificial intelligence have elevated imitation learning (IL) [1]–[6] to a leading paradigm for robotic dexterous manipulation, enabling versatile control of high-degree-of-freedom (DoF) robot hands. However, collecting the large-scale, high-quality demonstrations required for this approach remains prohibitively expensive: teleoperation systems demand specialized hardware and extensive operator time. To alleviate this burden, researchers are increasingly turning to publicly available human motion-capture (mocap) and video datasets [7]–[9], which can be mined at scale with minimal overhead.

However, directly leveraging human mocap data on robots remains challenging. Previous retargeting algorithms [10]–[12] only partially bridge the embodiment gap between the human and the robot. Furthermore, the inherent noise and coarse resolution of mocap fail to capture the fine-grained details of human dexterity, often yielding suboptimal trajectories and task failures. Moreover, camera-based mocap and pose-estimation pipelines [13]–[15] capture only the kinematic relationship between the hand and the object, providing no information about underlying physical interactions such as contact forces.

To compensate for these missing physical dynamics, recent studies augment coarse mocap priors with simulation-based reinforcement learning (RL), optimizing for residual or direct joint-space actions under object-centric reward signals [16]–[21]. Closest to our work, Chen et al. [16] adopt a hierarchical IL + RL framework in which imitation generates wrist trajectories, and a low-level RL policy refines them in task space while producing the corresponding hand-joint actions. However, full joint-space exploration for hand motion generation remains a significant burden for reinforcement learning.

Building on this perspective, we present **DexRefine**, a framework that refines human motion into physically feasible robot actions using task-space residual reinforcement learning. Our approach refines the robot's wrist and fingertip poses in the task-space by the RL policy trained to maintain the object trajectory in alignment with a reference trajectory. We validated our method using a Human-Object Interaction (HOI) mocap dataset for a "bottle reorientation" task. Our experiments reveal two key findings: (1) Learning task-space residuals is significantly more stable and sample-efficient than training the robot directly in joint space; and (2) Compared to a baseline that relies on simple inverse-kinematics(IK)-based retargeting, DexRefine effectively refines synthetic HOI trajectories, achieving a high success rate while reducing object tracking error by about half.

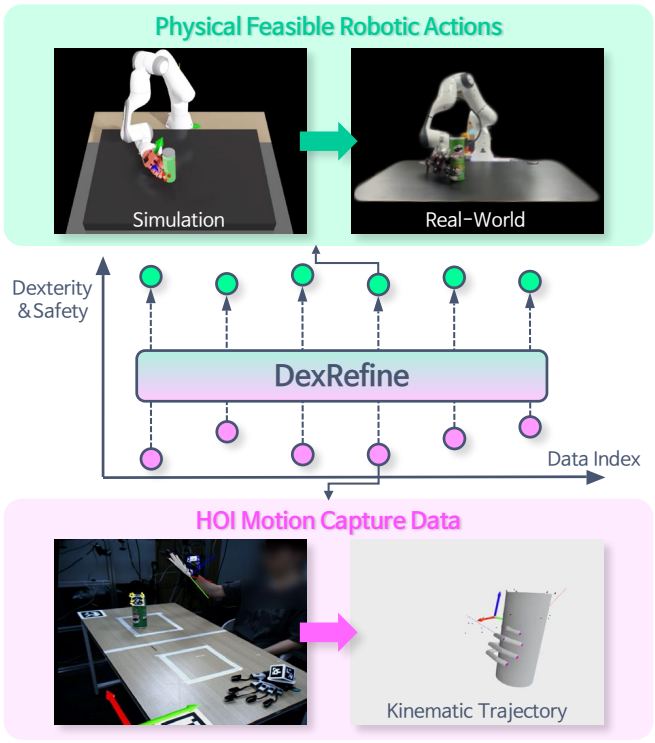

Fig. 1: DexRefine can refine coarse and unsafe kinematic HOI mocap data into physically feasible robotic actions, enhancing both dexterity and safety.

II. METHOD

Fig. 2 provides an overview of our proposed framework. In the following sections, we describe each component in detail.

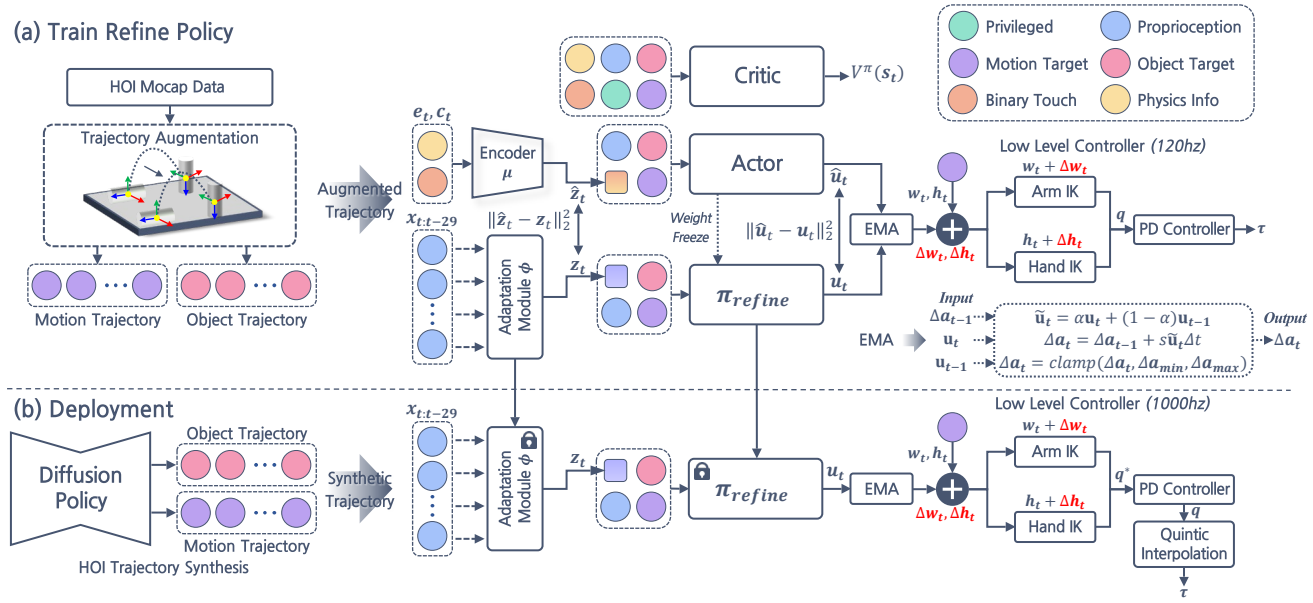

Fig. 2: **Overview of DexRefine Framework:** (a) A refinement policy, π_{refine}, is trained via an asymmetric actor-critic RL method to refine augmented human motions to feasible robot actions. To enable sim-to-real transfer, a student policy reconstructs a teacher's latent vector encoding object physics and binary contact states using only proprioceptive history, (b) At deployment, a Diffusion Policy generates a coarse HOI trajectory conditioned on the initial object pose, which π_{refine} then refines into an executable robot action.

A. HOI Mocap Data Collection

We collected HOI motion data using a Manus glove and a multi-view motion capture system in the same environment as [22]. For the "bottle reorientation" task, 89 demonstration trajectories were collected under randomized object positions. Each trajectory contains the fingertip positions $\mathbf{p}_f \in \mathbb{R}^3$ measured in the wrist frame, the object 6D pose $(\mathbf{R}_o, \mathbf{t}_o) \in \mathbb{SE}(3)$ defined in the world frame, and the wrist 6D pose $(\mathbf{R}_w, \mathbf{t}_w) \in \mathbb{SE}(3)$ also defined in the world frame. The dataset is:

$$\mathcal{D} = \left\{ \tau_i = \left[\left(\mathbf{p}_f^t, \mathbf{R}_o^t, \mathbf{t}_o^t, \mathbf{R}_w^t, \mathbf{t}_w^t \right)_{t=1}^{T} \right] \right\}_{i=1}^{N}, \tag{1}$$

where $N = 89$ and T is the trajectory length.

B. Task Space Residual Action

The primary objective of our DexRefine Policy is to refine the coarse human motion into physically feasible robot actions. At each time step t, the policy π_{refine} outputs a raw control signal $\mathbf{u}_t$. To stabilize training and prevent abrupt changes, this signal is processed through several steps. First, $\mathbf{u}_t$ is clamped element-wise to $[-1, 1]$ and then smoothed using an exponential moving average (EMA) filter:

$$\tilde{\mathbf{u}}_t = \alpha \mathbf{u}_t + (1 - \alpha)\mathbf{u}_{t-1}, \tag{2}$$

where we set the smoothing factor $\alpha = 0.8$. This smoothed signal $\tilde{\mathbf{u}}_t$ is then used to incrementally update the residual action $\Delta\mathbf{a}_t$:

$$\Delta\mathbf{a}_t = \Delta\mathbf{a}_{t-1} + s\tilde{\mathbf{u}}_t\,\Delta t, \tag{3}$$

where Δt is the policy inference interval and s is a component-wise scaling factor (0.05 for wrist translation, 0.2 for wrist rotation, and 0.05 for finger motion).

The residual action $\Delta\mathbf{a}_t = \left(\Delta\mathbf{w}_t,\ \Delta\mathbf{h}_t\right)$ consists of the residual wrist motion $\Delta\mathbf{w}_t = \left(\Delta\mathbf{R}_{w,t},\ \Delta\mathbf{t}_{w,t}\right)$ and fingertip translations $\Delta\mathbf{h}_t$, comprising an 18-DoF action space ($\Delta\mathbf{a}_t \in \mathbb{R}^{18}$). To ensure physical feasibility, the cumulative residuals are clamped within predefined ranges:

$$\Delta\mathbf{p}_{\text{wrist}} \in [-0.1,\ 0.1]\ \text{m}, \quad \Delta\mathbf{r}_{\text{wrist}} \in [-40^\circ,\ 40^\circ],$$
$$\Delta\mathbf{p}_{\text{finger}} \in [-0.04,\ 0.04]\ \text{m}.$$

The final action is then computed by adding this residual to the initial coarse action, $\mathbf{a}_t^{\text{final}} = \mathbf{a}_t + \Delta\mathbf{a}_t$. This task-space command is mapped to joint-space targets q^* using inverse kinematics (IK); specifically, we employ an analytical solver for the hand and a damped least-squares solver [23] for the arm. Before execution by the low-level PD controllers, the resulting joint trajectories are refined using quintic interpolation to suppress jitter and ensure smooth tracking. We optimized our policy using Proximal Policy Optimization (PPO) [24]. Detailed reward formulation and training details are provided in Appendix.

C. Sim-to-Real Transfer

To enable sim-to-real transfer of DexRefine policy, we employ domain randomization, an asymmetric actor–critic architecture, and a distillation strategy similar to [25]. The privileged teacher policy has access to simulation-only information, encoding object physics properties $\mathbf{e_t} \in \mathbb{R}^{18}$ (e.g., mass, pose, friction) and binary tactile signals from object contact $\mathbf{c_t} \in \{0, 1\}^{11}$ into a 16-dimensional latent

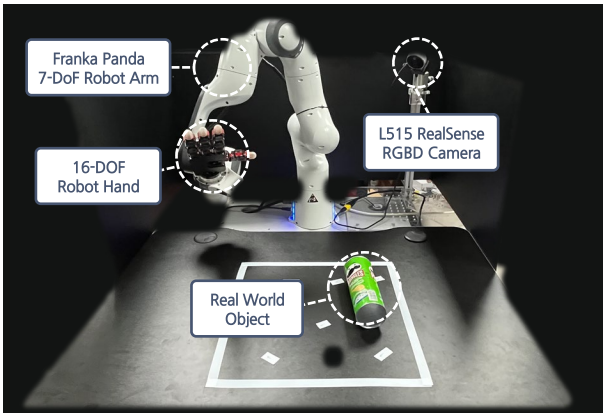

Fig. 3: The experimental setup in the real world.

vector $\hat{\mathbf{z}}_t = \mu(\mathbf{e_t}, \mathbf{c_t})$. Since this privileged information is unavailable at deployment, the student policy learns to reconstruct this latent representation from proprioceptive history. Specifically, a GRU-based [26] adaptation module ϕ processes a 30-step history of joint positions and previous actions, $\mathbf{x}_{t-29:t}$ where $\mathbf{x}_t = [\mathbf{q}_t; \mathbf{a}_t] \in \mathbb{R}^{46}$, to produce its own latent vector $\mathbf{z}_t = \phi(\mathbf{x}_{t-29:t})$. The distillation is performed through an online DAgger [27], where the teacher's actor network is frozen and copied to the student. The student policy is then trained to mimic both the teacher's actions and its latent vector by minimizing a weighted objective:

$$\mathcal{L} = \lambda_1 \|\hat{\mathbf{u}}_t - \mathbf{u}_t\|_2^2 + \lambda_2 \|\mathbf{z}_t - \hat{\mathbf{z}}_t\|_2^2, \tag{4}$$

where $\mathbf{u}_t$ is the teacher's action and $\hat{\mathbf{u}}_t$ is the student's action. We set the loss weights to $\lambda_1 = 0.1$ and $\lambda_2 = 1.0$ to prioritize accurate latent representation matching.

D. HOI Trajectory Synthesis for Deployment

In real-world applications, it is infeasible to provide pre-captured HOI trajectories for every possible initial pose of an object. To address this, we first expand the spatial distribution of a collected set of 1,000 HOI demonstrations via data augmentation: we randomly translate each trajectory in the x–y plane, interpolate its start and end segments to match the robot's initial pose, and apply random z-axis rotations exploiting the object's axial symmetry. To overcome the remaining limitation that a finite set of augmented trajectories cannot cover all possible initial poses, we leverage this augmented dataset to train a diffusion policy [1]. This policy is trained to broaden the spatial coverage beyond the mocap distribution and takes the current state $\mathbf{s}_t = [\mathbf{o}_t, \mathbf{q}_t^o, \mathbf{w}_t, \mathbf{q}_t^w, \mathbf{h}_t]$ to output a one-step delta $\Delta\mathbf{a}_t = [\,\Delta\mathbf{o}_t,\ \mathbf{r}_t^o,\ \Delta\mathbf{w}_t,\ \mathbf{r}_t^w,\ \Delta\mathbf{h}_t\,]$. These deltas, consisting of positional changes $(\Delta\mathbf{o}_t, \Delta\mathbf{w}_t, \Delta\mathbf{h}_t)$ and continuous 6D rotation parameters $(\mathbf{r}_t^o, \mathbf{r}_t^w)$, are recursively accumulated to produce a full trajectory. As a result, the policy can generate diverse and feasible HOI trajectories across a wide range of initial object poses without requiring explicit ground-truth motion for every case.

III. Experimental Results

Our evaluation focuses on the following two key aspects: **Effectiveness of Task-Space Residual Actions:** We investigate whether learning residual actions in the task space (i.e., for the wrist and fingertips) achieves more efficient and stable control compared to directly predicting actions in the joint space, given the same task-space motion guidance. **Real-World Performance Comparison:** We quantify the extent to which DexRefine executes more physically feasible motions than a standard Inverse Kinematics based retargeting baseline, using object-conditioned human-object interaction (HOI) trajectories generated by a diffusion model.

A. Experimental Setup

Our real-world setup (Fig. 3) consists of a Franka Emika Panda arm equipped with a fully actuated 16-DoF robotic hand [28] for dexterous manipulation. An external Intel RealSense L515 RGBD camera observes the workspace. To obtain the initial 6-DoF object pose in the robot base frame, we use a two-stage perception pipeline. First, we perform camera robot calibration to estimate the rigid transform between the camera and robot's base frames. Then, we leverage FoundationPose [29] to estimate the object pose in the camera frame. Composing the two transforms yields the object pose in the base frame for each trial.

B. Residual Action in Task Space vs. Direct Action in Joint Space

Under identical task-space guidance, we compare four control configurations by combining wrist control (task-space residual or direct joint-space) with fingertip control (task-space residual or direct joint-space). We train each policy for 5,000 iteration steps using 4,096 parallel environments with the same seed and compare the final success rate. As summarized in Fig. 4(a), task-space residual learning consistently outperforms direct joint-space prediction. The optimal configuration, using Cartesian residuals for both the wrist and hand, yields the highest success rate at approximately 70%. This is a dramatic improvement over using joint-space for the wrist, which keeps the success rate below 15%. This finding highlights that learning corrective actions in Cartesian space, particularly for the wrist, is the most critical component for success in this dexterous manipulation task.

C. Real-World Deployment Performance: DexRefine vs. IK Retargeting

We evaluate the distilled student policy in ten real-world trials, each with a random initial object pose. For each trial, a diffusion model generates a synthetic human–object interaction (HOI) trajectory conditioned on the initial object pose, which serves as the reference motion. We compare two controllers executing this same reference trajectory: (i) *Synthetic HOI + DexRefine*, which refines the reference into physically feasible robot actions, and (ii) *Synthetic HOI + IK*, which directly retargets the reference using inverse kinematics.

Performance is measured by success rate and object trajectory errors: Object Translation Error (OTE, m) and Object Orientation Error (OOE, deg). In this evaluation, OTE/OOE quantify reference-tracking fidelity—the translational and rotational deviation between the executed object pose and

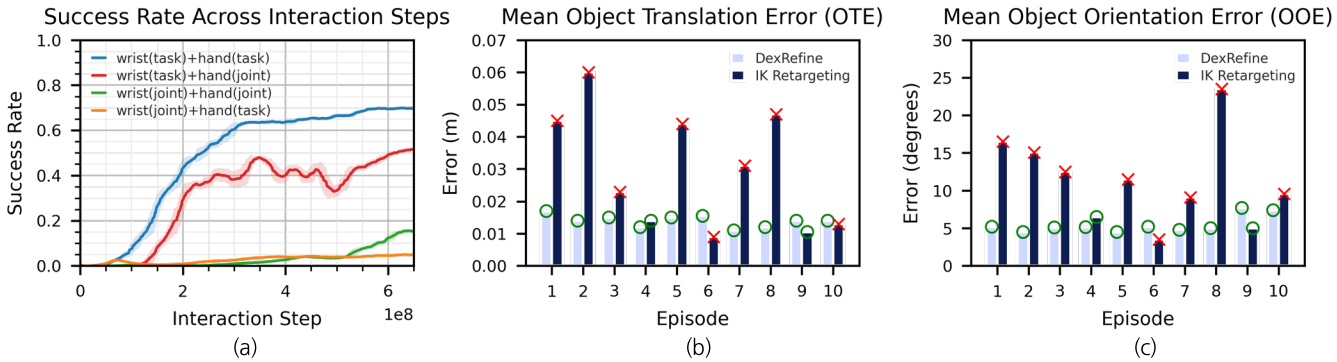

Fig. 4: Action-space comparison and real-world object-pose errors: (a) task-space residual vs. direct joint-space (wrist & fingertips), (b) per-episode Object Trajectory Error (OTE, m) over 10 real-world trials, (c) per-episode Object Orientation Error (OOE, deg) over the same trials. (× denotes a failed episode and ○ denotes a successful episode.)

TABLE I: Real-world deployment: success rate and aggregated object errors across 10 trials. OTE: object translation error (m); OOE: object orientation error (degree).

Method	Success Rate	OTE (m)	OOE (°)
Synthetic HOI + DexRefine (Ours)	**10/10**	**0.015 ± 0.011**	**5.82 ± 6.10**
Synthetic HOI + IK	2/10	0.031 ± 0.046	11.75 ± 18.98

the diffusion-generated reference at each time step. Errors are averaged per episode and then aggregated across all ten trials. We define task success as placing the object in a stable, upright position near the target location. DexRefine's task-space residual actions are designed to reduce these tracking errors online via small wrist/fingertip corrections.

As shown in Table I and Figs. 4(b) and (c), DexRefine markedly outperformed the IK baseline, reducing OTE by approximately 52% and OOE by about 50%. Critically, DexRefine achieved a perfect 100% success rate across all trials. This contrasts sharply with the IK baseline, which failed in 8 out of 10 trials. The baseline's failures were due to severe errors, including floor collisions (four trials), grasp failure (one trial), or a combination of both (three trials). Fig. 5 provides a qualitative comparison, visually highlighting typical failure modes of the IK baseline, such as grasp failure and wrist collisions, in contrast to the successful execution by DexRefine.

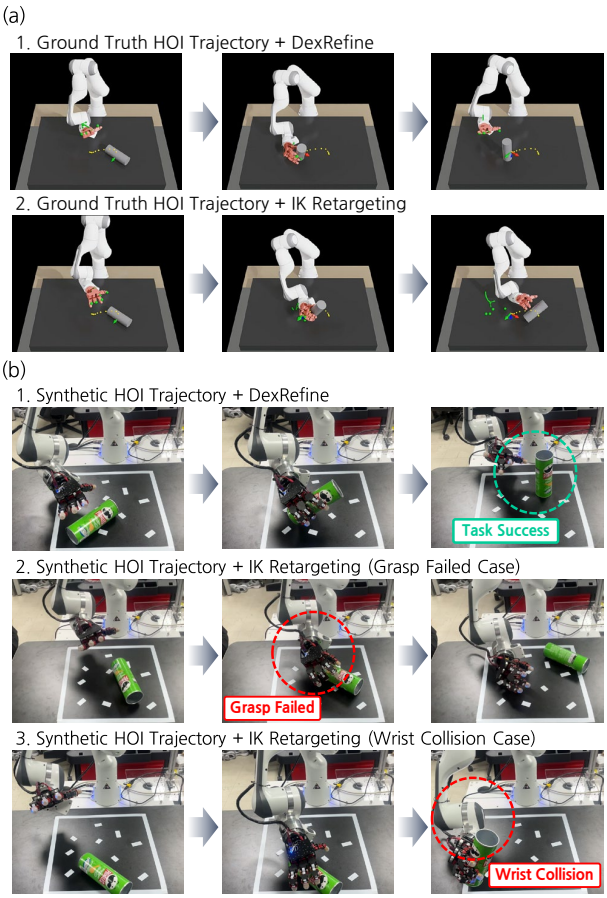

Fig. 5: Comparison of DexRefine with IK Retargeting in (a) a simulated environment and (b) a real-world deployment.

IV. Conclusion

In this paper, we presented DexRefine, a residual learning framework for refining human motions into physically feasible robot actions. We demonstrated that learning residual actions in task space accelerates RL training and effectively bridges the embodiment gap. The efficacy of our approach was particularly evident in our bottle reorientation task, where the robot's different arm-wrist configuration made it difficult to directly mimic the human's natural wrist-centric strategy. DexRefine overcame this physical limitation by generating an in-hand manipulation strategy to rotate the object—a dexterous behavior absent from the original mocap data. This results demonstrate that our method can produce emergent behaviors that augment physical interaction. Despite these promising results, a key limitation is that our method has been validated on only a single task. Therefore, our future work will focus on extending DexRefine to a more diverse range of tasks and objects.

REFERENCES

[1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[2] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[3] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[4] L. Heng, H. Geng, K. Zhang, P. Abbeel, and J. Malik, "Vitacformer: Learning cross-modal representation for visuo-tactile dexterous manipulation," *arXiv preprint arXiv:2506.15953*, 2025.

[5] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[6] J. Cao, Q. Zhang, J. Sun, J. Wang, H. Cheng, Y. Li, J. Ma, K. Wu, Z. Xu, Y. Shao, *et al.*, "Mamba policy: Towards efficient 3d diffusion policy with hybrid selective state models," *arXiv preprint arXiv:2409.07163*, 2024.

[7] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield, *et al.*, "Dexycb: A benchmark for capturing hand grasping of objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9044–9053.

[8] Z. Fan, O. Taheri, D. Tzionas, M. Kocabas, M. Kaufmann, M. J. Black, and O. Hilliges, "Arctic: A dataset for dexterous bimanual hand-object manipulation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 12 943–12 954.

[9] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "Amass: Archive of motion capture as surface shapes," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5442–5451.

[10] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox, "Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system," *arXiv preprint arXiv:2307.04577*, 2023.

[11] Z.-H. Yin, C. Wang, L. Pineda, K. Bodduluri, T. Wu, P. Abbeel, and M. Mukadam, "Geometric retargeting: A principled, ultrafast neural hand retargeting algorithm," *arXiv preprint arXiv:2503.07541*, 2025.

[12] C. M. Kim, B. Yi, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, "Pyroki: A modular toolkit for robot kinematic optimization," *arXiv preprint arXiv:2505.03728*, 2025.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[14] H. Xu, H. Li, Y. Wang, S. Liu, and C.-W. Fu, "Handbooster: Boosting 3d hand-mesh reconstruction by conditional synthesis and sampling of hand-object interactions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 159–10 169.

[15] J. Kim, J. Kim, J. Na, and H. Joo, "Parahome: Parameterizing everyday home activities towards 3d generative modeling of human-object interactions," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 1816–1828.

[16] Y. Chen, C. Wang, Y. Yang, and C. K. Liu, "Object-centric dexterous manipulation from human motion data," *arXiv preprint arXiv:2411.04005*, 2024.

[17] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik, *et al.*, "Dexteritygen: Foundation controller for unprecedented dexterity," *arXiv preprint arXiv:2502.04307*, 2025.

[18] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid, "Vividex: Learning vision-based dexterous manipulation from human videos," *arXiv preprint arXiv:2404.15709*, 2024.

[19] S. Zhao, X. Zhu, Y. Chen, C. Li, X. Zhang, M. Ding, and M. Tomizuka, "Dexh2r: Task-oriented dexterous manipulation from human to robots," *arXiv preprint arXiv:2411.04428*, 2024.

[20] K. Li, P. Li, T. Liu, Y. Li, and S. Huang, "Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6991–7003.

[21] X. Liu, J. Adalibieke, Q. Han, Y. Qin, and L. Yi, "Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references," *arXiv preprint arXiv:2502.09614*, 2025.

[22] S. Park, S. Lee, M. Choi, J. Lee, J. Kim, J. Kim, and H. Joo, "Learning to transfer human hand skills for robot manipulations," in *1st Workshop on X-Embodiment Robot Learning*.

[23] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," in *Conference on Robot Learning*. PMLR, 2023, pp. 1722–1732.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[27] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[28] C. Ahn, S. Park, and D. Hwang, "Effectiveness of kinesthetic sensing in in-hand rotation of objects with an eccentric center of mass," in *ICRA 2025 Workshop"Handy Moves: Dexterity in Multi-Fingered Hands"Paper Submission*.

[29] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.

[30] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

APPENDIX

A. Reward Function

The total reward is composed of several task-specific terms and regularization penalties (see Table II for parameter settings). We represent the pose of a rigid body by its position $\mathbf{p} \in \mathbb{R}^3$ and orientation $\mathbf{q} \in \mathbb{R}^4$, where $\mathbf{q}$ is a unit quaternion. The overall reward is defined as:

$$r = r_{\text{obj}} + r_{\text{hand}} + r_{\text{wrist}} + r_{\text{contact}} - p \tag{5}$$

1) Reward Components: The task rewards encourage successful manipulation behaviors. The primary object reward, r_{obj}, promotes transferring the object from its current pose $(\mathbf{p}_{\text{obj}}^{\text{cur}}, \mathbf{q}_{\text{obj}}^{\text{cur}})$ to the goal pose $(\mathbf{p}_{\text{obj}}^{\text{goal}}, \mathbf{q}_{\text{obj}}^{\text{goal}})$. This reward provides a smooth and dense signal that increases as the object approaches its goal. Importantly, it is gated by an indicator function $\mathbb{1}_{\text{contact or goal}}$, which activates only when the hand is in contact with the object or the task is completed. We define the object's position and rotation errors as follows: the position error is $e_{p,\text{obj}} = \left\|\mathbf{p}_{\text{obj}}^{\text{cur}} - \mathbf{p}_{\text{obj}}^{\text{goal}}\right\|_2$, and the rotation error is $e_{R,\text{obj}} = \text{dist}_{\text{rot}}\big(\mathbf{q}_{\text{obj}}^{\text{cur}}, \mathbf{q}_{\text{obj}}^{\text{goal}}\big)$, where $\text{dist}_{\text{rot}}(\cdot,\cdot)$ denotes the rotation distance.

$$r_{\text{obj}} = w_{\text{obj}} \, \exp\big(- a_{\text{obj}}^{\text{pos}} \, e_{p,\text{obj}} - a_{\text{obj}}^{\text{rot}} \, e_{R,\text{obj}} \big) \mathbb{1}_{\text{contact or goal}}. \tag{6}$$

Auxiliary pose rewards guide both the hand and wrist configurations. For notational consistency, we adopt the error notation introduced for the object—namely, the position error $e_{p,\cdot}$ and the rotation error $e_{R,\cdot}$—with the subscript indicating

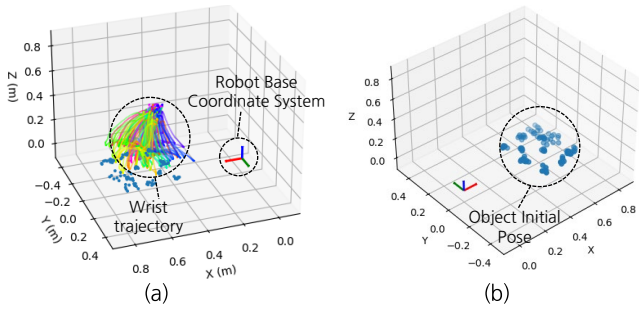

Fig. 6: Trajectory distribution of the collected HOI data: (a) Wrist trajectory distribution in the robot base coordinate system, (b) Initial object position distribution in the robot base coordinate system.

the entity. The hand reward, r_{hand}, aligns the fingertips with their goals, shaping a pre-grasp posture:

$$r_{\text{hand}} = w_{\text{hand}} \exp\big(- a^{\text{pos}}_{\text{hand}} e_{p,\text{hand}} \big). \tag{7}$$

The wrist reward, r_{wrist}, guides the end-effector toward its goal pose:

$$r_{\text{wrist}} = w_{\text{wrist}} \exp\big(- a^{\text{pos}}_{\text{wrist}} e_{p,\text{wrist}} - a^{\text{rot}}_{\text{wrist}} e_{R,\text{wrist}} \big). \tag{8}$$

2) Penalty Components: Penalties are introduced to regularize behavior, ensuring safety. The collision penalty, $p_{\text{collision}}$, discourages unintended contact between the arm and the environment. It is proportional to the forces measured at links 6, 7, and 8 of the Franka Emika Panda, which are located near the end-effector and thus most susceptible to collision:

$$p_{\text{collision}} = w_{\text{collision}} \sum_{i=1}^{3} F_i. \tag{9}$$

Control regularization terms limit excessive joint movements and energy expenditure. The velocity penalty regulates joint speed:

$$p_{\text{vel}} = w_{\text{vel}} \left\| \mathbf{v} \right\|_2 . \tag{10}$$

The torque penalty discourages large actuation effort:

$$p_{\text{torque}} = w_{\text{torque}} \left\| \boldsymbol{\tau} \right\|_2 . \tag{11}$$

Finally, the work penalty approximates mechanical energy consumption through the joint-wise product of torque and velocity magnitudes:

$$p_{\text{work}} = w_{\text{work}} \left(|\mathbf{v}| \cdot |\boldsymbol{\tau}| \right). \tag{12}$$

The total penalty is the sum of these components:

$$p = p_{\text{collision}} + p_{\text{vel}} + p_{\text{torque}} + p_{\text{work}}. \tag{13}$$

B. Observation Space

The teacher policy is trained using observations composed of proprioceptive, privileged, object, and target information, as summarized in Table III. For the proprioceptive observation, the current step is stacked together with the previous two steps, which indirectly provides information about velocity and acceleration to the policy.

TABLE II: Reward function parameters.

Parameter	Symbol	Value
Reward Weights		
Object reward weight	w_{obj}	30.0
Hand reward weight	w_{hand}	10.0
Wrist reward weight	w_{wrist}	5.0
Contact reward weight	w_{contact}	0.05
Penalty Weights		
Collision penalty weight	$w_{\text{collision}}$	0.01
Velocity penalty coefficient	w_{vel}	0.0002
Torque penalty coefficient	w_{torque}	0.0003
Energy penalty coefficient	w_{energy}	0.0005
Pose-Alignment Coefficients		
Object position coefficient	$a^{\text{pos}}_{\text{obj}}$	10.0
Object rotation coefficient	$a^{\text{rot}}_{\text{obj}}$	20.0
Hand position coefficient	$a^{\text{pos}}_{\text{hand}}$	10.0
Wrist position coefficient	$a^{\text{pos}}_{\text{wrist}}$	5.0
Wrist rotation coefficient	$a^{\text{rot}}_{\text{wrist}}$	5.0

C. HOI Trajectory Augmentation during RL Training

To prevent our policy from overfitting to the limited spatial distribution of our collected motion capture (mocap) data, we introduce an online data augmentation scheme during the RL training phase. Figure 4 illustrates that the original Human-Object Interaction (HOI) data is primarily concentrated within a specific workspace region ($x \in [0.4, 0.8]$m and $y \in [-0.2, 0.2]$m relative to the robot's base frame). At the start of each training episode, the augmentation process begins by uniformly sampling a reference HOI trajectory from the dataset. Subsequently, a random 2D translation offset, $\Delta p = (\Delta x, \Delta y)$, is sampled from a uniform distribution and applied to the entire trajectory. Since this direct translation can create a spatial discontinuity between the robot's fixed initial pose and the start of the shifted trajectory, we perform a final smoothing step. To ensure a continuous and feasible motion, the final augmented wrist trajectory is generated by linearly interpolating its initial segment between the robot's actual starting wrist pose and the first pose of the translated reference trajectory. This strategy allows the policy to train on a wider variety of target locations, significantly improving its generalization capabilities.

D. Training Details

We train our policies using IsaacLab [30] as the simulation environment. The physics timestep is set to $\Delta t = 1/120$ s, while the control frequency of the policy is $1/30$ s by applying a decimation factor of 4. The Teacher Policy is trained for 5,000 policy steps using 4,096 parallel environments, equivalent to approximately 250 days of simulated experience. The Student Policy is trained for 650 policy steps with 2,048 parallel environments, corresponding to approximately 15 days of simulated experience. The detailed hyperparameters used for training with PPO are summarized in Table IV.

E. Domain Randomization

To facilitate sim-to-real transfer, we applied extensive domain randomization on robot dynamics, object properties,

TABLE III: Observations used for training the Teacher Policy.

Category	Component	Dim
Privileged	Hand joint velocities	16
	Hand joint torques	16
	Fingertip orientations (quat)	16
	Fingertip velocities	24
	Fingertip forces	24
	Object velocity (lin + ang)	6
	Arm joint velocities	7
	Arm joint torques	7
	Total privileged dim	**116**
Proprioceptive	Arm joint positions	7
	Hand joint positions	16
	Previous joint actions	23
	Wrist position (kinematics-derived)	3
	Wrist orientation (quat, kinematics-derived)	4
	Fingertip positions (world)	12
	History buffer (3-step stacked)	**195**
Object Physics & Contact Encoding	Object position	3
	Object orientation (quat)	4
	Object mass	1
	Center of mass	7
	Material encoding	3
	Binary contact info	11
	Total object dim	**29**
Motion Target	Goal object position	3
	Goal object orientation (quat)	4
	Goal fingertip positions	12
	Goal wrist orientation (quat)	4
	Goal wrist position	3
	Total target dim	**26**
Total observation dim		**366**

TABLE IV: PPO hyperparameters

Parameter	Value
Value loss coefficient	1.0
Clipped value loss	True
Clipping parameter	0.2
Entropy coefficient	0.005
Number of learning epochs	5
Number of mini-batches	4
Number of steps for env	32
Learning rate	5.0×10^{-4}
Learning rate schedule	Adaptive
Discount factor	0.99
GAE parameter	0.95
Desired KL divergence	0.01
Maximum gradient norm	1.0

and environment conditions. The details of the randomized parameters and their ranges are summarized in Table V.

F. Network Architecture

Both the actor and critic networks are implemented as multilayer perceptrons (MLPs) with hidden layer sizes of [1024, 1024, 512, 256], using ELU activations. To facilitate sim-to-real transfer, we employ an asymmetric actor–critic architecture, where privileged information is provided exclusively to the critic network.

For the actor network, object-related physics information and binary touch information are encoded into a 16-dimensional latent vector through an encoder network with hidden layers [128, 64] and ReLU activations. The latent representation is then concatenated with the remaining observations and processed by the main actor network to generate the final policy output.

TABLE V: Domain randomization settings used in our training.

Parameter	Type	Distribution	Range
Robot			
Mass	Scaling	uniform	[0.5, 1.5]
Friction	Scaling	uniform	[0.8, 1.2]
Joint Limits (lower)	Scaling	uniform	[0.95, 1.05]
Joint Limits (upper)	Scaling	uniform	[0.95, 1.05]
Joint Stiffness	Scaling	uniform	[0.5, 1.5]
Joint Damping	Scaling	uniform	[0.5, 1.5]
Object			
Mass	Scaling	uniform	[0.1, 1.0]
Friction	Scaling	uniform	[0.5, 1.1]
Restitution	Scaling	uniform	[0.0, 0.4]
Desk			
Friction	Scaling	uniform	[0.5, 1.1]
Restitution	Scaling	uniform	[0.0, 0.4]
Environment			
Gravity	Additive	uniform	[0.0, 0.5]