A Minimalist Retargeting-Guided Reinforcement Learning Recipe for Dexterous Manipulation

Yunhai Feng[1] **Natalie Leung**[1] **Jiaxuan Wang**[1] **Lujie Yang**[2] **Haozhi Qi**[2] **Preston Culbertson**[1]
[1]Cornell University [2]Amazon FAR

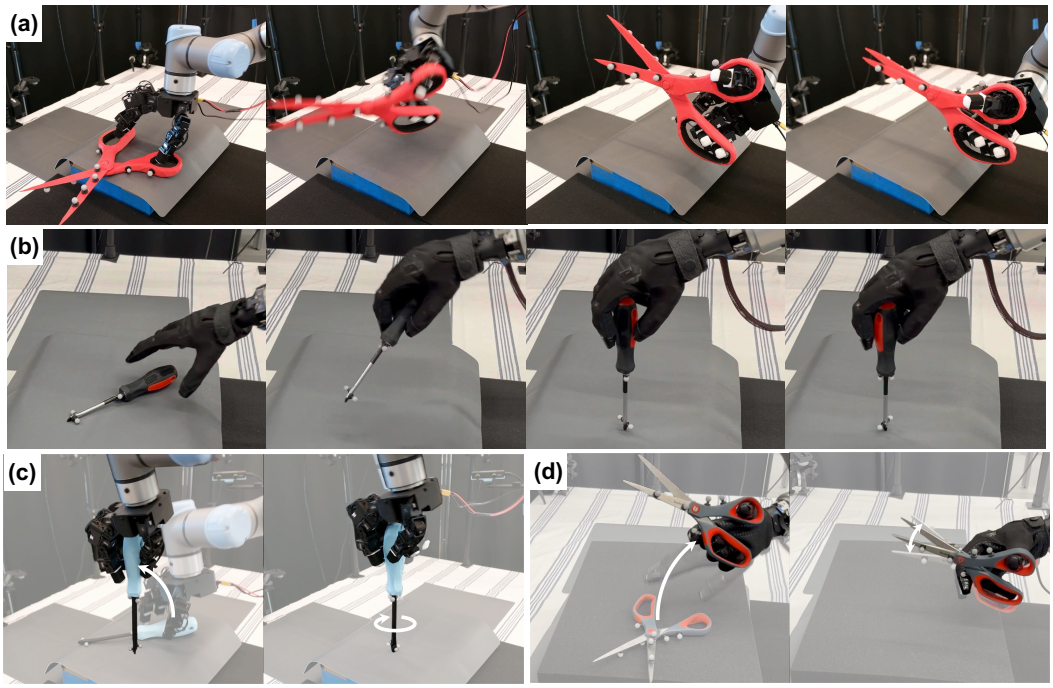


Figure 1: **Dexterous Tool Use Tasks.** We present a minimalist retargeting-guided RL pipeline for learning dexterous manipulation from a single human demonstration. We evaluate our method on four contact-rich task-hand settings across two robot hands: (a) `LEAP-Scissors`, (b) `WUJI-Screwdriver`, (c) `LEAP-Screwdriver`, and (d) `WUJI-Scissors`.

Abstract: Recent work in humanoid whole-body control has found success with a simple recipe: retarget human motion to robot kinematic references, then train policies via reinforcement learning (RL) to track them. But how does this recipe transfer to dexterous manipulation? The answer is not obvious, as manipulation involves complex, contact-rich dynamics and requires delicate regulation of contact modes and forces. We present REGRIND, a minimalist retargeting-guided RL pipeline that learns dexterous manipulation policies from a single human demonstration. REGRIND retargets human hand-object motion to a robot reference that preserves hand-object spatial and contact relationships, trains a residual RL policy in simulation to track object-centric keypoints along that reference, and transfers the resulting policy zero-shot to hardware with careful system identification. The resulting policies produce fluid, human-like behavior on two different multi-fingered hands across contact-rich tool-use tasks, including operating a pair of scissors and turning a screwdriver. Through systematic hardware experiments, we identify and analyze the key factors that govern sim-to-real transfer in dexterous manipulation, offering practical guidance for retargeting-based learning in contact-rich settings. Videos and code are available at `https://yunhaifeng.com/REGRIND`.

Keywords: Dexterous Manipulation, Motion Retargeting, Sim-to-Real

1 Introduction

Humanoid robots and anthropomorphic hands share the human body's kinematic structure, opening the possibility of learning skills directly from the vast amount of human motion data. In humanoid whole-body control, this has led to a simple and increasingly effective recipe: retarget a human motion reference to the robot, train a policy in simulation to track the retargeted reference, and deploy the policy on hardware [1, 2]. Dexterous manipulation, however, has largely followed a different path. Current manipulation skills are often learned from teleoperated demonstrations collected directly on real robots [3, 4, 5], with limited use of either human motion data or massive simulation.

While recent work has explored using humanoid-style retargeting pipelines for dexterous hands, results have largely been confined to simulation or open-loop replays of simulation motions [6, 7]. One hypothesis for this limited real-world success is that simple kinematic retargeting, while closely matching the human hand pose, ignores the hand-object interaction and produces physically implausible trajectories that are a poor reference for downstream RL. This raises a natural question: *does preserving interactions during retargeting improve RL for contact-rich dexterous manipulation?*

In this work, we answer this question by presenting REGRIND: **RE**targeting-**G**uided **R**e**IN**forcement learning for **D**exterous manipulation, a minimalist pipeline for learning contact-rich dexterous manipulation skills from a single human demonstration. We first retarget human hand-object motion to an interaction-preserving robot reference that maintains the spatial relationships between the hand and the object. To mitigate the scarcity of multi-fingered dexterous manipulation data, we dynamically augment the retargeted reference with perturbed initial configurations to provide broader coverage of the task space. We then train a residual RL policy in simulation to track object-centric keypoints along this reference, using the retargeted trajectories as both a motion prior and reset distribution for exploration. The resulting policies transfer to real multi-fingered hands on challenging tool-use tasks, including operating a pair of scissors and turning a screwdriver, which require coordinated finger motion, complex contact transitions, and interactions that leverage friction.

At the same time, our study reveals an important distinction between locomotion and manipulation. Although retargeting-based learning can produce highly capable manipulation policies [6, 7], dexterous manipulation exhibits substantially greater sensitivity to sim-to-real discrepancies. Manipulation hinges on rich, sustained contact between the robot and the object, and small modeling errors in friction, compliance, or geometry compound quickly when the policy's success depends on those interactions. Our results suggest that successful sim-to-real transfer requires not only interaction-preserving retargeting but also thoughtful design of observation and action spaces, appropriate domain randomization and curricula, and careful system identification. Together, these ingredients form a practical recipe for sim-to-real dexterous manipulation.

Our contributions are as follows:

- We propose a general and minimalist framework for learning dexterous manipulation from a single human demonstration. We release the code and data for the community to reproduce the results and build upon our work.
- We apply a simple yet effective data augmentation strategy that exposes the RL policy to diverse trajectories for improved robustness and generalization.
- We evaluate our method on challenging contact-rich dexterous manipulation tasks and demonstrate its superior performance over existing methods, especially the value of interaction-preserving motion retargeting.

2 Related Work

Dexterous Manipulation with Reinforcement Learning. Reinforcement learning (RL) has driven much of the progress in dexterous manipulation. While some works have trained manipulation policies directly in the real world [8, 9, 10, 11], the high cost of real-world interaction and the sample inefficiency of RL have made training in simulation followed by sim-to-real transfer a more practical approach. Such sim-to-real efforts have progressed from reorienting a cube and solving a Rubik's cube [12, 13, 14] to reorienting diverse objects [15, 16, 17, 18], and, increasingly, to contact-rich,

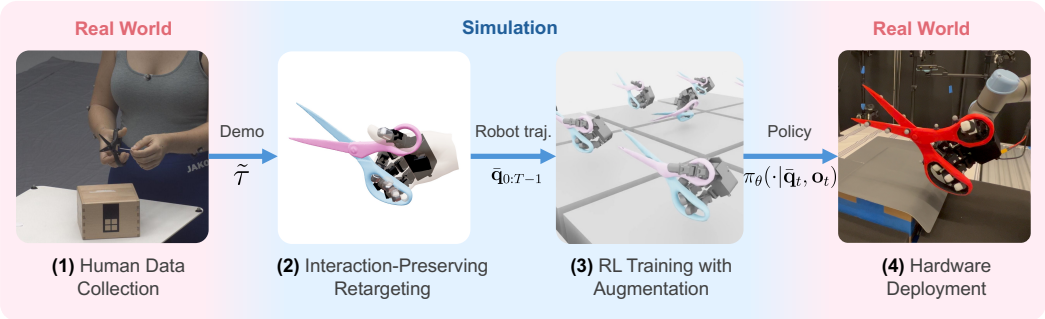


Figure 2: **Method Overview.** We follow a real-to-sim-to-real pipeline to learn an RL policy from 3D human data. (1) Given a human demonstration with hand and object poses, we (2) retarget it to the robot with interaction-aware motion retargeting, (3) train a policy to track the retargeted trajectory with large-scale RL in simulation, and then (4) transfer the policy zero-shot to the real world. We dynamically augment the reference trajectory during RL training to make the policy generalizable to different initial configurations.

bimanual, and tool-use tasks [19, 20, 21]. However, scaling RL across tasks remains bottlenecked by reward engineering and exploration. A growing line of work injects external priors such as retargeted reference trajectories [22, 7] or coarse plans from vision-language-model [23] to improve learning efficiency. We follow this direction, using human hand motion—faithfully transferred via interaction-preserving retargeting—as a prior that simplifies reward design and biases exploration toward reliable human-like strategies.

Dexterous Manipulation with Imitation Learning. Propelled by scalable data collection systems [3, 5, 4] and expressive policy classes [24, 25], imitation learning (IL) has become the central paradigm for real-world robot manipulation by learning visuomotor policies directly from real-robot demonstrations and thereby sidestepping the sim-to-real gap. However, IL relies on accurate on-robot action data, which is difficult to capture for high-degree-of-freedom dexterous hands. A range of data collection pipelines have emerged for multi-fingered hands, including systems that capture human hand motion with motion capture gloves, VR headsets, or RGB cameras and then retarget it onto robot hands [26, 4, 27, 28], wearable hand exoskeletons that mechanically couple human and robot fingers for portable, in-the-wild collection [29, 30, 31], and methods that augment human demonstrations with physics-based simulation [32]. Our pipeline does not require collecting robot-specific data; instead, it transfers raw human motion to robot trajectories via motion retargeting, placing it close to work that leverages motion-capture and video data as broad sources of human motion [7, 6, 33, 34, 35].

Motion Retargeting from Human Data. Retargeting human motion onto the robot and training policies in simulation to track the reference has become an increasingly common recipe for humanoid whole-body control [36, 37, 1, 2]. For dexterous hands, retargeting has mainly supported teleoperation and demonstration generation through per-frame kinematic optimization [38, 28] or learned mappings from human video [39, 40]. More recent pipelines incorporate object context through functional retargeting to demonstrated object states [7], physics-informed retargeting [41], learned human-to-robot transfer [42, 6], trajectory-guided refinement in simulation [43], or sparse object-centric rewards and hand poses to guide RL [22]. However, hand-centric retargeting can still yield physically implausible contact structures, and prior object-aware methods either target dataset generation and imitation, emphasize simulation-only tracking, or rely on sparse object or hand cues rather than dense interaction structure. We adapt the interaction mesh formulation [44, 2] from humanoid loco-manipulation to dexterous hand–object manipulation and use the interaction-preserving reference as an exploratory restart distribution for efficient closed-loop sim-to-real RL.

3 Method: Retargeting-Guided Reinforcement Learning

3.1 Problem Statement

Robot policy learning with reinforcement learning (RL) can be formulated as a finite-horizon Markov Decision Process (MDP) defined by $\langle \mathcal{S}, \mathcal{A}, P, \rho_0, r, T \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are continuous state

and action spaces, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics, ρ_0 is the initial state distribution, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, and T is the horizon. The goal is to find the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi, s_0\sim\rho_0}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1})$ is a trajectory sampled from the policy π, and $\gamma \in [0, 1)$ is the discount factor.

The manipulation task is specified by a human demonstration $\tilde{\tau} = (\tilde{s}_0, \tilde{s}_1, \ldots, \tilde{s}_{T-1})$, and the reward encourages the robot to track the human demonstration, e.g., $r(s_t, a_t) = \exp(-\text{dist}(s_t, \tilde{s}_t))$ with some distance metric $\text{dist}(\cdot, \cdot)$. Here, $\tilde{s}_t \in \tilde{\mathcal{S}}$ is the demonstration state that contains information about the object and the human hand at timestep t. Note that $\mathcal{S}$ and $\tilde{\mathcal{S}}$ are two different spaces because of the embodiment gap between the robot and the human. For instance, $s_t \in \mathcal{S}$ contains robot joint positions while $\tilde{s}_t \in \tilde{\mathcal{S}}$ may only provide the hand keypoints. We assume the object state representation is the same in both spaces, thus, a reasonable distance metric can be the distance between two object poses in $\text{SE}(3)$.

One challenge of RL is exploration. Kakade and Langford [45] introduced the concept of exploratory restart distribution to guide the exploration process, which is later popularized by DeepMimic [46] as Reference State Initialization (RSI). Instead of optimizing the objective in Eq. (1), the policy is optimized with respect to an exploratory restart distribution μ:

$$\max_\pi \mathbb{E}_{\tau\sim\pi, s_0\sim\mu}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]. \tag{2}$$

It was also shown that it is important for μ to be close to the state visitation distribution $d_{\rho_0}^{\pi^*}$ of the optimal policy π^*[45], motivating a natural choice of μ as the distribution that supports the states from the human demonstration in our case. Due to the embodiment gap discussed earlier, we convert the demonstration states $\tilde{s}_t$ to the robot states s_t using a motion retargeting process and use the retargeted trajectory as the exploratory restart distribution.

The rest of the section is organized as follows. We first introduce the interaction-aware motion retargeting process to convert the human demonstration to a robot trajectory which preserves the hand-object interaction semantics in Section 3.2. Then, we introduce the RL training pipeline with the retargeted robot trajectory as the reference motion in Section 3.3. The full pipeline of our method is illustrated in Fig. 2.

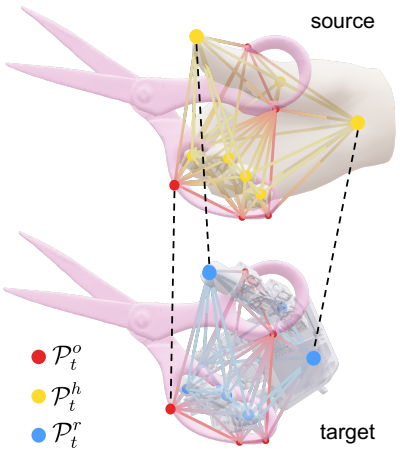


Figure 3: **Interaction meshes.** We construct the source interaction mesh with the object and hand keypoints in the demonstration (top), and the target interaction mesh with the same object keypoints and the robot keypoints that correspond to the hand keypoints (bottom). The dashed lines connect the corresponding keypoints in the two meshes.

3.2 Interaction-Aware Motion Retargeting

The goal of motion retargeting is to convert the human demonstration into a robot trajectory that mimics the human motion. The demonstration consists of the MANO hand keypoints [47] and the object configuration at each timestep, where the object configuration is specified by a 6D pose (and additional joint position(s) if the object is articulated).

We formulate the problem of motion retargeting as an optimization problem on the robot configuration $\mathbf{q}_{0:T-1} = \{(\mathbf{T}_t^r, \mathbf{q}_t^r)\}_{t=0}^{T-1}$, where $\mathbf{T}_t^r$ is the robot wrist

pose and $\mathbf{q}_t^r$ is the vector of robot joint positions at timestep t. The key design decision is selecting a metric to measure how well the robot trajectory mimics the human demonstration. Importantly, we need to consider both the human-robot embodiment gap and the complex interaction between the robot hand and the object, a key difference from prior works that perform pure kinematic retargeting without considering the object.

Inspired by OmniRetarget [2], we define an interaction mesh-based retargeting objective to encourage robot motion to preserve the semantics of manipulating the object, such as the spatial and contact relationships between the robot hand and the object. We define semantic keypoints on the human hand and the object as $\mathcal{P}_t^h$ and $\mathcal{P}_t^o$, respectively. They constitute the set of source points in the demonstration $\tilde{\mathcal{P}}_t = \mathcal{P}_t^o \cup \mathcal{P}_t^h$. Given the robot configuration $\mathbf{q}_t$, we define a set of robot keypoints $\mathcal{P}_t^r(\mathbf{q}_t)$, which semantically correspond to the human hand keypoints $\mathcal{P}_t^h$. The robot keypoints $\mathcal{P}_t^r(\mathbf{q}_t)$ and the original object keypoints $\mathcal{P}_t^o$ constitute the set of target points $\mathcal{P}_t(\mathbf{q}_t) = \mathcal{P}_t^o \cup \mathcal{P}_t^r(\mathbf{q}_t)$. We measure the interaction-preserving property by the deformation of interaction meshes (see Fig. 3 for an illustration). The optimization program is formulated as

$$\begin{aligned} \bar{\mathbf{q}}_{0:T-1} = \underset{\mathbf{q}_{0:T-1}}{\arg\min} \sum_{t=0}^{T-1} D\left(\mathcal{M}(\tilde{\mathcal{P}}_t), \mathcal{M}(\mathcal{P}_t(\mathbf{q}_t))\right) + \lambda \sum_{t=1}^{T-1} \|\mathbf{q}_t - \mathbf{q}_{t-1}\|_2^2 \\ \text{s.t.} \quad \mathbf{q}_{0:T-1} \in \mathcal{Q}, \end{aligned} \tag{3}$$

where λ is a weighting factor and $\mathcal{Q}$ is the feasible set of the robot configuration, including the kinematic constraints of the robot and collision avoidance constraints. The connectivity between the keypoints $\mathcal{P}$ are decided by Delaunay tetrahedralization [48] to form the interaction mesh $\mathcal{M}(\mathcal{P})$ with vertices $\mathcal{P}$ and edges $\mathcal{E}$. We minimize the deformation energy between the two meshes, which is defined as the change of the Laplacian coordinates of the meshes:

$$D\left(\mathcal{M}(\tilde{\mathcal{P}}_t), \mathcal{M}(\mathcal{P}_t(\mathbf{q}_t))\right) = \sum_i \left\| L_i(\tilde{\mathcal{P}}_t) - L_i(\mathcal{P}_t(\mathbf{q}_t)) \right\|_2, \tag{4}$$

where $L_i(\mathcal{P})$ is the Laplacian coordinate of the i-th vertex in the mesh $\mathcal{M}(\mathcal{P}) = (\mathcal{P}, \mathcal{E})$, defined as

$$L_i(\mathcal{P}) = \mathbf{p}_i - \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{p}_j, \tag{5}$$

where $\mathbf{p}_i$ is the coordinate of the i-th vertex and $\mathcal{N}_i = \{j \in \mathcal{P} : (i,j) \in \mathcal{E}\}$ is the neighborhood of the i-th vertex. The second term in the objective (Eq. (3)) encourages temporal smoothness of the trajectory. The program is solved sequentially for each timestep t using a Sequential Quadratic Programming (SQP)-style solver. See App. A for implementation details.

3.3 Reference Motion-Guided Policy Learning

Using the retargeted robot trajectory $\bar{\mathbf{q}}_{0:T-1}$, i.e., the solution of Eq. 3, as the reference motion, we learn an RL policy to track it using a generalizable keypoint tracking reward and reference state initialization. We use $\bar{\cdot}$ to denote the "nominal" quantity from the retargeted trajectory to distinguish it from the actual states in RL.

Action Space. The policy π_θ learns a residual action on top of the reference motion. The control target is then computed as $\mathbf{q}_t^{\text{target}} = \bar{\mathbf{q}}_t + \boldsymbol{\alpha} \odot \pi_\theta(\bar{\mathbf{q}}_t, \mathbf{o}_t)$, where $\bar{\mathbf{q}}_t$ is the nominal robot configuration from the retargeted trajectory, $\mathbf{o}_t$ is the observation, and $\boldsymbol{\alpha}$ is a scaling vector controlling the action magnitude for each degree of freedom, and $\odot$ denotes element-wise product. This control target is then sent to a low-level PD controller.

Observation Space. We use asymmetric actor-critic observations. The actor observation contains the configurations of the object and the robot, the last action $\mathbf{a}_{t-1}$, as well as a phase variable $\phi \in [0, 1]$, with $\phi = 0$ indicating the start of the motion and $\phi = 1$ indicating the end of the motion [46]. The object configuration consists of the object pose, including position $\mathbf{p}_t^o \in \mathbb{R}^3$ and orientation $\mathbf{R}_t^o \in \mathbb{R}^6$, represented by Rot6D [49], and the joint position $\mathbf{q}_t^o \in \mathbb{R}^{n_o}$ for articulated objects, where n_o is the number of object joints. These values are obtained from a motion capture

system. The robot proprioception consists of all joint positions $\mathbf{q}_{t-1:t}$ from the current and the previous timesteps. We use the observations that can be relatively reliably obtained in the real-world system and avoid using noisy sensor measurements such as velocity for the policy to reduce the sim-to-real transfer gap. In addition to the above-mentioned observations, the critic also has access to the fingertip positions and joint velocities of the robot.

Reward Function. Our reward function is designed to be generalizable to different tasks and objects, and to be easy to implement, without requiring an explicit contact prior as in [7, 23]. Instead, the prior for hand-object interaction is implicitly encoded in the reference trajectory generated by the interaction-preserving retargeting process. The main reward is a proximity reward incentivizing close object tracking. We leverage a keypoint-based distance metric to measure the discrepancy between the current object configuration and the target object configuration:

$$\epsilon_{\text{object}} = \text{dist}_{\text{KP}}(\mathcal{P}_t^o, \bar{\mathcal{P}}_t^o) = \frac{1}{N_k} \sum_{i=1}^{N_k} \left\| \mathbf{p}_{t,i}^o - \bar{\mathbf{p}}_{t,i}^o \right\|_2, \tag{6}$$

where $\mathbf{p}_{t,i}^o$ and $\bar{\mathbf{p}}_{t,i}^o$ are the current and target positions of the i-th keypoint on the object, and $\bar{\mathcal{P}}_t^o = \{\bar{\mathbf{p}}_{t,i}^o\}_i$. Compared to translation and rotation errors defined for rigid bodies, this distance metric is generally applicable to both rigid and articulated objects. The reward is then shaped with an exponential kernel: $r_{\text{object}} = \exp\left(-\epsilon_{\text{object}}/\sigma\right)$, where σ controls the sharpness of the reward. We also include object velocity and wrist pose tracking rewards, as well as penalties for large action magnitude and action rate to incentivize smooth motion. See App. B for all RL training details.

Reference Motion-Guided Training. As motivated in Sec. 3.1, we adopt reference state initialization (RSI) [46] and use the retargeted trajectory as the reference motion to guide the policy learning with a restart state distribution μ. We sample the restart state from the retargeted trajectory, and initialize both the robot and the object to the corresponding states at the start of an episode, with a small perturbation to provide a broader coverage of potential high-value states. We also terminate the episode if the object deviates too much from the reference trajectory. Using a tight object deviation threshold is helpful to prevent the policy from deviating from the reference trajectory too much, since residual RL policies may not be able to recover from large deviations.

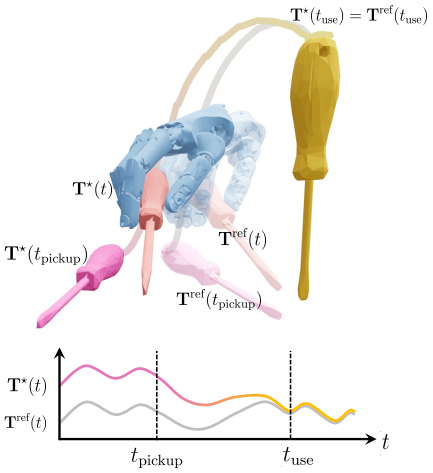


Figure 4: **Data augmentation.** $\mathbf{T}^{\text{ref}}$ and $\mathbf{T}^\star$ denote the original and augmented retargeted trajectory, respectively. $\mathbf{T}^\star$ is interpolated towards $\mathbf{T}^{\text{ref}}$ between the timestep t_{pickup} (when the object is picked up) and the timestep t_{use} (when the subject starts to use the tool at a target position). See Appendix B.3 for more details.

Data Augmentation. Since our pipeline learns from a single human demonstration, the retargeted trajectory only covers one instantiation of the task, which limits the spatial generalization of the learned policy. We therefore perform data augmentation to synthesize a diverse set of reference trajectories for training a more robust and generalizable RL policy. We randomize the initial object configuration by perturbing its position and orientation around the demonstrated configuration ($\pm$5 cm for position and $\pm$30 degrees for orientation), which defines a rigid transformation $\Delta\mathbf{T}_0 \in \text{SE}(3)$ between the perturbed and the original initial object pose. To generate a new reference trajectory consistent with this perturbed start, we apply a time-varying rigid transformation $\Delta\mathbf{T}(t)$ to every state of the retargeted trajectory, obtained by interpolating between $\Delta\mathbf{T}_0$ at the start and the identity transformation at the goal. This warps the beginning of the trajectory toward the augmented initial state while smoothly returning it to the original goal state by the end. Fig. 4 shows an example of the augmented trajectory for picking up and turning a screwdriver.

Different from OmniRetarget [2], which reruns the retargeting procedure to generate a new robot trajectory for the augmented object trajectory, we directly apply the same rigid transformation to both the object pose and the robot hand root pose at each timestep without re-solving the optimization problem of retargeting. This enables us to dynamically generate an infinite number of augmented trajectories at RL training time while still preserving the relative spatial and contact relationships captured by the original retargeting process.

Domain Randomization and Curriculum. To further robustify the learned policy, we apply domain randomization to the physics simulation parameters (e.g., friction, motor gains, time delays, etc.) and add observation noise during RL training. In addition, we apply random pushes to both the object and the robot with a curriculum on the push strength. We also adopt gravity curriculum to gradually increase the gravity magnitude from zero to its full value, which is useful for tasks involving picking up objects.

4 Experiments

We design experiments to answer the following key questions:

- **(Q1)** Does interaction-preserving retargeting improve retargeting fidelity?
- **(Q2)** Does preserving interactions during retargeting improve downstream RL performance?
- **(Q3)** Are the policies trained with REGRIND able to transfer to real-world dynamics?
- **(Q4)** Can the policies replicate the demonstration from different initial states?

4.1 Experiment Setup

Robots and Tasks. Our method is evaluated on two dexterous manipulation tasks: `Scissors` from the ARCTIC dataset [50], and `Screwdriver` from our own data collection setup (see App. C for demonstration collection details), covering both rigid and articulated objects. Each task is evaluated on two anthropomorphic robot hands: a 16-DoF LEAP hand [51] and a 20-DoF WUJI hand. The robot hand is mounted on a 7-DoF UR5e robot arm. The tasks are illustrated in Fig. 1. In the scissors task, the robot picks up a pair of scissors and operates them at a desired orientation; in the screwdriver task, it grasps the screwdriver and turns it continuously while holding it upright. Due to its size, the LEAP hand cannot perform semantically similar grasps on normal-sized tools, so we use 3D-printed oversized objects for the LEAP hand, and real objects for the WUJI hand. To isolate the dynamics gap in sim-to-real transfer, we feed object poses from a motion capture system directly to the policy at deployment time, which minimizes perception errors and prevents them from acting as a confounder.

Evaluation Metrics. We use the following two metrics to evaluate the policy performance:

- Object Tracking Error (Err.): The average distance between the current object keypoint positions and the target object keypoint positions (Eq. 6).
- Success rate (SR): The ratio of successful trials over the total number of trials. See App. D for detailed success criteria for each task.

Baselines. We compare our method with two representative baselines for learning dexterous manipulation from human demonstrations.

- SPIDER [41], a physics-based retargeting method that generates dynamically feasible robot trajectories. The retargeted trajectories can be executed as open-loop policies or used as reference motions for downstream RL.
- DexMachina [7], which also trains RL policies to track kinematic retargeted robot trajectories, but without considering the robot-object interaction semantics in the retargeting process.

In addition, we compare with a simple baseline that retargets the human motion to the robot with differential inverse kinematics (using Mink [52]) and then trains an RL policy with the same settings as our method, named Mink IK + RL.

4.2 Results

(Q1) Retargeting Quality.

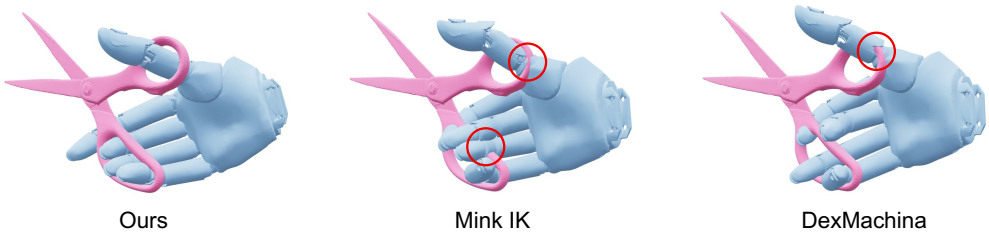


Figure 5: **Qualitative Retargeting Results.** Sample retargeted configurations produced by our method, Mink IK, and DexMachina for the `WUJI-Scissors` task. Our method produces a human-like grasp that preserves the hand-object interaction semantics, while the other two baselines produce physically implausible or unstable grasps that are not suitable for RL training. The red circles highlight significant hand-object penetrations.

We visualize the resulting robot configurations produced by different retargeting strategies for a qualitative comparison in Fig. 5. Both the Mink IK + RL and DexMachina baselines use simple IK-based retargeting. They produce large robot-object penetrations in the IK solutions (highlighted in red circles), which can lead to instability when initializing the simulator to these states during RL training. DexMachina uses a simulation-based post-processing step to eliminate these penetrations, but this projection can push the fingers to undesired positions, as no useful information is provided for the simulator to resolve the collision along the "correct direction" to preserver hand-object interaction. These suboptimal states are likely to harm RL training instead of providing consistent guidance.

(Q2) Simulation performance comparison.

Table 1: **Simulation Performance.** Object keypoint tracking error (Err.) and success rates (SR) on the 4 tasks in simulation. SPIDER results are averaged over 5 seeds; others are averaged over 3 seeds $\times$ 1024 rollouts.

Method	`LEAP-Scissors`		`LEAP-Screwdriver`		`WUJI-Scissors`		`WUJI-Screwdriver`	
	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑
REGRIND (Ours)	**5.6**$\pm$**6.4**	**99.8**$\pm$**0.3%**	**5.4**$\pm$**5.8**	**99.7**$\pm$**0.0%**	**5.3**$\pm$**13.1**	**98.7**$\pm$**1.3%**	6.5$\pm$15.5	98.8$\pm$1.3%
SPIDER	176.2$\pm$1.6	0.0$\pm$0.0%	134.5$\pm$78.9	0.0$\pm$0.0%	119.0$\pm$32.7	0.0$\pm$0.0%	116.9$\pm$16.1	0.0$\pm$0.0%
DexMachina	10.1$\pm$9.5	22.3$\pm$17.7%	8.3$\pm$6.7	**99.7**$\pm$**0.1%**	67.2$\pm$31.8	0.0$\pm$0.0%	**5.9**$\pm$**9.7**	**99.3**$\pm$**0.1%**
Mink IK + RL	12.6$\pm$21.4	2.0$\pm$2.9%	17.5$\pm$4.5	0.0$\pm$0.0%	38.9$\pm$29.1	0.0$\pm$0.0%	10.0$\pm$12.0	3.1$\pm$1.3%

Table 1 shows the performance of our method and the baselines on the four tasks in simulation. For all methods except SPIDER, we report the tracking error and success rate of the final RL policies. Across many hyperparameter settings, SPIDER produces object trajectories that deviate significantly from the original demonstration and are not useful initializations for residual RL. However, since SPIDER produces physically plausible, MPC-style trajectories that attempt to track the demonstration, we compare its retargeted trajectories against the trajectories produced by our closed-loop RL policies. Our method consistently achieves low object tracking error and near perfect success rates on all tasks. DexMachina also achieves near perfect success rates on the screwdriver tasks. However, its performance degrades on the more challenging scissors tasks, which involve complex object geometry and hand-object interactions. Both the Mink IK + RL and SPIDER baselines struggle with the contact-rich tasks considered here.

(Q3) Sim-to-real transfer. We deploy the RL policies that achieve reasonable performance in simulation to the real world and show the performance in Table 2. Our method transfers reliably on three of the four tasks: `LEAP-Scissors`, `LEAP-Screwdriver`, and `WUJI-Screwdriver`. Surprisingly,

Table 2: **Real-World Performance.** Object keypoint tracking error (Err.) and success rates (SR) in the real world.

Method	LEAP-Scissors		LEAP-Screwdriver		WUJI-Scissors		WUJI-Screwdriver	
	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑	Err. (mm) ↓	SR ↑
REGRIND (Ours)	**27.9**±**2.0**	**9 / 10**	**16.0**±**2.9**	**10 / 10**	65.2±21.6	0 / 10	**9.6**±**2.5**	**9 / 10**
DexMachina	228.7±99.9	0 / 10	122.7±85.4	2 / 10	—	—	47.3±45.0	5 / 10
Mink IK + RL	—	—	—	—	—	—	41.8±47.2	0 / 10

despite strong simulation performance, the DexMachina policies transferred poorly on both screwdriver tasks. One hypothesis is that, without the stronger regularization provided by our interaction-preserving retargeting process, DexMachina policies are more prone to exploiting simulation artifacts, resulting in unstable grasps and overly aggressive motions that overshoot or hit the table during real-world deployment. Instead, our method produces human-like grasps that are robust to the real-world environment. None of the methods transferred successfully for the `WUJI-Scissors` task, likely due to the larger sim-to-real gap with the non-backdrivable motors on the WUJI hand and the inaccurate mesh of the real scissors.

(Q4) Generalization to Different Initial Configurations.

To test the generalization capability of our policies beyond the demonstrated configurations, we evaluate the policies with randomly sampled initial object configurations. The position perturbations are sampled uniformly within ±5 cm along the x and y axes, and the orientation perturbations are sampled uniformly within ±30 degrees around the z axis. Table 3 shows the performance of our method on three tasks in the real world. We observe that the policies trained with our method generalize well to different initial states, achieving similar performance to the performance on the demonstrated configurations. This demonstrates the effectiveness of our simple dynamic data augmentation strategy at RL training to improve the robustness and generalizability of the policy.

Table 3: **Generalization Performance.** Object keypoint tracking error (Err.) and success rates (SR) of our method on randomized initial configurations in the real world.

Task	Err. (mm) ↓	SR ↑
`LEAP-Scissors`	27.4±2.7	8 / 10
`LEAP-Screwdriver`	16.3±2.1	10 / 10
`WUJI-Screwdriver`	10.3±2.2	9 / 10

5 Limitations

We are optimistic about retargeting-based learning for dexterous manipulation, as we have demonstrated promising results on challenging tasks and gained a better understanding of the challenges of sim-to-real transfer in contact-rich manipulation. At the same time, several limitations remain. First, our method relies on motion capture to obtain the object state at deployment time. Distilling the state-based RL policy into a vision-based policy is a natural next step to make the system more applicable to in-the-wild scenarios. Second, our pipeline still requires careful system identification to enable sim-to-real transfer; a promising direction is to adapt the policy at test time using context inferred from interactions with the real world.

Acknowledgments

This project was supported in part by the Department of the Navy, Office of Naval Research under ONR award number N00014-25-1-2086. Model training was performed using the Unicorn shared computing cluster at Cornell University. Yunhai Feng would like to thank the PoRTaL Lab for support with motion capture prototyping, Paige Yun for help with the motion capture infrastructure, and Samuel Jin and other members at the Praxis Lab for helpful technical discussion.

References

[1] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv:2508.08241*, 2025.

[2] L. Yang, X. Huang, Z. Wu, A. Kanazawa, P. Abbeel, C. Sferrazza, C. K. Liu, R. Duan, and G. Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. In *International Conference on Robotics and Automation (ICRA)*, 2026.

[3] T. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

[4] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

[5] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. In *Conference on Robot Learning (CoRL)*, 2025.

[6] K. Li, P. Li, T. Liu, Y. Li, and S. Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2025.

[7] Z. Mandi, Y. Hou, D. Fox, Y. Narang, A. Mandlekar, and S. Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation. In *International Conference on Machine Learning (ICML)*, 2026.

[8] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[9] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning (CoRL)*, 2020.

[10] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 2025.

[11] L. Dodeja, O. Biza, S. Vats, S. Hart, S. Tellex, R. Walters, K. Schmeckpeper, and T. Weng. When life gives you bc, make q-functions: Extracting q-values from behavior cloning for on-robot reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2026.

[12] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.

[13] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *International Conference on Robotics and Automation (ICRA)*, 2023.

[14] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv:1910.07113*, 2019.

[15] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 2023.

[16] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning (CoRL)*, 2023.

[17] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning (CoRL)*, 2023.

[18] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang. Lessons from learning to spin "pens". In *Conference on Robot Learning (CoRL)*, 2025.

[19] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik. Twisting lids off with two hands. In *Conference on Robot Learning (CoRL)*, 2024.

[20] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. In *Conference on Robot Learning (CoRL)*, 2025.

[21] K. Kedia, T. G. W. Lum, J. Bohg, and C. K. Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. In *Robotics: Science and Systems (RSS)*, 2026.

[22] T. G. W. Lum, O. Y. Lee, C. K. Liu, and J. Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. In *Conference on Robot Learning (CoRL)*, 2025.

[23] V. de Bakker, J. Hejna, T. G. W. Lum, O. Celik, A. Taranovic, D. Blessing, G. Neumann, J. Bohg, and D. Sadigh. Scaffolding dexterous manipulation with vision-language models. In *Neural Information Processing Systems (NeurIPS)*, 2025.

[24] 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*, 2025.

[25] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024.

[26] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik, M. Lambeta, T. Wu, P. Abbeel, and M. Mukadam. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv:2502.04307*, 2025.

[27] K. Shaw, Y. Li, J. Yang, M. K. Srirama, R. Liu, H. Xiong, R. Mendonca, and D. Pathak. Bimanual dexterity for complex tasks. *arXiv:2411.13677*, 2024.

[28] 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. In *Robotics: Science and Systems (RSS)*, 2023.

[29] H.-S. Fang, B. Romero, Y. Xie, A. Hu, B.-R. Huang, J. Alvarez, M. Kim, G. Margolis, K. Anbarasu, M. Tomizuka, E. Adelson, and P. Agrawal. Dexop: A device for robotic transfer of dexterous human manipulation. *arXiv:2509.04441*, 2025.

[30] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. In *Conference on Robot Learning (CoRL)*, 2025.

[31] T. Tao, M. K. Srirama, J. J. Liu, K. Shaw, and D. Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv:2505.07813*, 2025.

[32] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. J. Fan, and Y. Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[33] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision (ECCV)*, 2022.

[34] R. Zheng, D. Niu, Y. Xie, J. Wang, M. Xu, Y. Jiang, F. Castañeda, F. Hu, Y. L. Tan, L. Fu, T. Darrell, F. Huang, Y. Zhu, D. Xu, and L. Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv:2602.16710*, 2026.

[35] G. Zhang, Q. Xu, H. Zhang, J. Ma, L. He, Y. Bao, Z. Ping, Z. Yuan, C. Lu, C. Yuan, T. Liang, X. Tian, M. Shao, F. Zhang, M. Ding, Y. Gao, H. Zhao, H. Zhao, and H. Xu. Unidex: A robot foundation suite for universal dexterous hand control from egocentric human videos. *arXiv:2603.22264*, 2026.

[36] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[37] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025.

[38] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[39] A. Sivakumar, K. Shaw, and D. Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. In *Robotics: Science and Systems (RSS)*, 2022.

[40] K. Shaw, S. Bahl, and D. Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning (CoRL)*, 2023.

[41] C. Pan, C. Wang, H. Qi, Z. Liu, H. Bharadhwaj, A. Sharma, T. Wu, G. Shi, J. Malik, and F. Hogan. Spider: Scalable physics-informed dexterous retargeting. In *International Conference on Intelligent Robots and Systems (IROS)*, 2026.

[42] S. Park, S. Lee, M. Choi, J. Lee, J. Kim, J. Kim, and H. Joo. Learning to transfer human hand skills for robot manipulations. *arXiv:2501.04169*, 2025.

[43] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid. Vividex: Learning vision-based dexterous manipulation from human videos. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[44] E. S. Ho, T. Komura, and C.-L. Tai. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics (TOG)*, 2010.

[45] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

[46] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 2018.

[47] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. In *ACM SIGGRAPH Asia (SIGGRAPH Asia)*, 2017.

[48] H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *International Meshing Roundtable (IMR)*, 2005.

[49] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2023.

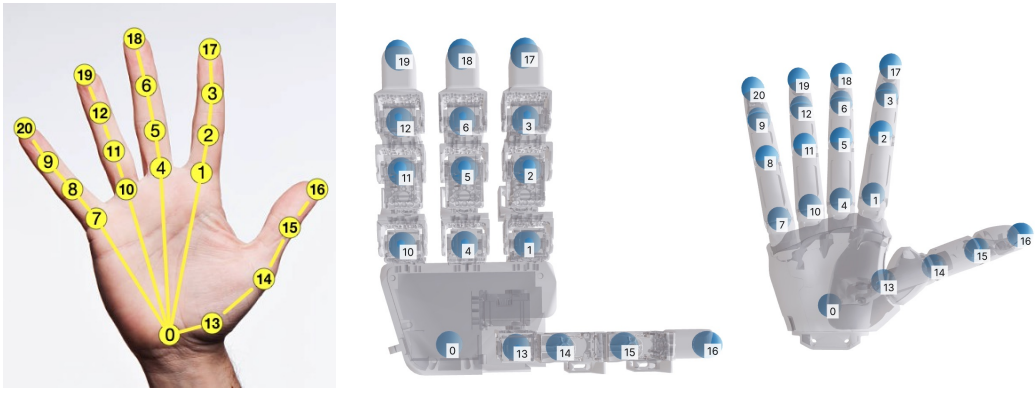
[51] K. Shaw, A. Agarwal, and D. Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. In *Robotics: Science and Systems (RSS)*, 2023.

[52] K. Zakka. Mink: Python inverse kinematics based on MuJoCo, 2026.

A Motion Retargeting Details

A.1 Keypoints and Correspondences

To construct the interaction meshes, we sample 50 keypoints on each object. For scissors, we sample in the region of the handles where the fingers are making contact with. For screwdriver, we uniformly sample on the entire object surface. We define 21 hand keypoints on the joints, following the widely used MANO model[1]. Figure 6 shows the MANO keypoints and the corresponding robot keypoints we define on the robot URDFs.



MANO keypoints | LEAP Hand keypoints | WUJI Hand keypoints

Figure 6: **Hand keypoint correspondence.** MANO human hand keypoints and the corresponding robot keypoints on the LEAP hand and WUJI hand.

A.2 Solver Details

We follow OmniRetarget [2] for the interaction mesh-based retargeting process. As in Sec. 3.2, $\mathbf{q}_t$ denotes the robot actuated configuration—the floating base together with all finger joint coordinates—while the demonstration object pose is held fixed and not optimized. The trajectory-level program in Eq. (3) is solved *sequentially*: at each timestep t we solve for $\mathbf{q}_t^*$, warm-started from $\mathbf{q}_{t-1}^*$. Within each frame we run a small number of Sequential Quadratic Programming (SQP) iterations: the objective is quadratically approximated, the hard constraints in $\mathcal{Q}$ are linearized at the current iterate, and the convex subproblem is solved with MOSEK via Drake's `MathematicalProgram`[2].

A.3 Feasible Set $\mathcal{Q}$

The kinematic constraints in Eq. (3) are enforced as hard constraints on $\mathbf{q}_t$ at each timestep. We write the feasible set as

$$\mathcal{Q} = \left\{ \mathbf{q}_t : \mathbf{q}_{\min} \le \mathbf{q}_t \le \mathbf{q}_{\max},\ \mathbf{v}_{\min}\,\Delta t \le \mathbf{q}_t - \mathbf{q}_{t-1} \le \mathbf{v}_{\max}\,\Delta t,\ \phi_j(\mathbf{q}_t) \ge 0,\ \forall j \right\}, \quad (7)$$

where $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are joint limits, $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ bound the per-frame configuration change, Δt is the demonstration timestep, and $\phi_j(\mathbf{q}_t)$ is the signed distance for the j-th collision pair.

Joint limits. We require $\mathbf{q}_{\min} \le \mathbf{q}_t \le \mathbf{q}_{\max}$, with limits taken from the robot model. For the floating base, quaternion components are constrained to $[-1, 1]$ before renormalization.

Velocity bounds. To encourage smooth retargeted motion, we bound the frame-to-frame change as $\mathbf{v}_{\min}\,\Delta t \le \mathbf{q}_t - \mathbf{q}_{t-1} \le \mathbf{v}_{\max}\,\Delta t$, matching the velocity-limit constraint used in OmniRetarget.

[1] https://mano.is.tue.mpg.de

[2] https://github.com/RobotLocomotion/drake

Non-penetration. For each active collision pair j, we enforce non-penetration via $\phi_j(\mathbf{q}_t) \geq 0$. In our dexterous manipulation setting we activate robot–object and robot–environment (table/ground) pairs. Adjacent geometries that should not generate constraints (e.g. palm–finger links) are filtered at plant setup.

B RL Training Details

B.1 Simulation

We use IsaacSim 5.1.0[3] for large-scale parallel simulation. Simulation uses $\Delta t_{\text{sim}} = 1/120\,\text{s}$ with control decimation 4, so the policy is evaluated every $\Delta t = 4\Delta t_{\text{sim}} = 1/30\,\text{s}$. Episodes last $9.0\,\text{s}$ (≈ 270 control steps).

B.2 Reward Functions

We train with a dense reward that compares the simulated state to time-indexed targets from a re-targeted reference motion. At each control step t, the scalar reward is a weighted sum of shaped terms:

$$r_t \;=\; \sum_k w_k\, \phi_k\big(\mathbf{e}_{k,t}\big), \tag{8}$$

where w_k is the term weight, $\mathbf{e}_{k,t}$ is the corresponding tracking or regularization error, and ϕ_k maps the error to $[0, 1]$ (or a binary penalty). Most tracking terms use an exponential kernel on a scalar error e:

$$\phi_{\text{lin}}(e;\, \sigma) = \exp\Big(-\frac{e}{\sigma}\Big)\,. \tag{9}$$

Object linear/angular velocity matching, action magnitude, and action-rate terms use a squared-error cost c with

$$\phi_{\text{sq}}(c;\, \sigma) = \exp\Big(-\frac{c}{\sigma^2}\Big)\,. \tag{10}$$

Table 4: Reward terms and hyperparameters.

Term	**Error / cost**	**Shaping**	$\boldsymbol{\sigma}$	**Weight** w_k
Object keypoint position	mean $\|\mathbf{p}_i - \bar{\mathbf{p}}_i\|_2$ over keypoints	(9)	0.02	1.5
Object linear velocity	$\|\mathbf{v} - \bar{\mathbf{v}}\|_2^2$	(10)	1.0	1.0
Object angular velocity	$\|\boldsymbol{\omega} - \bar{\boldsymbol{\omega}}\|_2^2$	(10)	3.14	1.0
Wrist position	$\|\mathbf{p}_{\text{wrist}} - \bar{\mathbf{p}}_{\text{wrist}}\|_2$	(9)	0.02	0.05
Wrist orientation	quaternion error magnitude	(9)	0.2	0.05
Action ℓ_2 magnitude	$\text{mean}_j(a_j^2)$	(10)	1.0	0.5 (WUJI only)
Action rate ℓ_2	$\text{mean}_j(a_j - a_{j,t-1})^2$	(10)	0.5	1.0
Action out of bounds	sum of violations outside $[-1, 1]$	(9)	1.0	1.0
Early termination	$\mathbb{I}$[terminated $\wedge$ $\neg$demo end]	–	–	-10.0

B.3 Data Augmentation

Referernce state initialization (RSI) samples a random demo phase at reset. Trajectory augmentation applies a per-episode perturbation, blended out between timesteps $t_{\text{pickup}}, t_{\text{use}}$ on object and hand targets.

Let $(\Delta\mathbf{p}, \Delta\psi)$ be a sampled perturbation transformation, with $\Delta\mathbf{p}_{xy} \in [-5, 5]^2\,\text{cm}$, $\Delta\mathbf{p}_z = 0$, and $\Delta\psi \in [-30°, 30°]$. The blend weight (linear interpolation of perturbation strength) is computed as:

$$w(t) = \begin{cases} 1, & t < t_{\text{pickup}}, \\ 1 - \dfrac{t - t_{\text{pickup}}}{t_{\text{use}} - t_{\text{pickup}}}, & t_{\text{pickup}} \leq t \leq t_{\text{use}}, \\ 0, & t > t_{\text{use}}. \end{cases} \tag{11}$$

[3] https://github.com/isaac-sim/IsaacSim/releases/tag/v5.1.0

Table 5: Detailed domain randomization and curriculum terms.

DR term	LEAP	WUJI
Physics		
Object CoM offset	scissors: $x, y, z = \pm 2.5, \pm 1, \pm 0.5$ cm; screwdriver: $x, y, z = \pm 0.5, \pm 3, \pm 0.5$ cm	scissors: $x, y, z = \pm 1.2, \pm 0.5, \pm 0.2$ cm; screwdriver: $x, y, z = \pm 0.5, \pm 2, \pm 0.5$ cm
Object geometry scale	—	scissors: $x, y \in [1.0, 1.05]$, $z \in [0.95, 1.0]$; screwdriver: $x, y \in [1.0, 1.05]$, z=1.0
Robot/table/object friction	robot $[0.7, 1.3]$; table, object $[0.5, 1.5]$	robot, table, object $[0.5, 1.5]$
Robot/object mass scale	$[0.95, 1.05]$ / $[0.8, 1.2]$	
Joint k_p, k_d scale	$[0.9, 1.1]$	
Finger default-joint offset	$[-0.03, 0.03]$ rad	—
Curriculum		
Gravity	$0 \rightarrow -9.81\,\mathrm{m/s^2}$ by env step 130k (resampled each reset)	
Random push	inactive $<$ 130k; max at 170k: $v \in [-0.5, 0.5]$ m/s, $\omega \in [-1, 1]$ rad/s; interval $[1, 5]$ s	
Observations		
Additive noise	pos ± 2 mm; rot ± 0.02 rad; joints ± 0.02 rad	
Time lag	$[0, 2]$ control steps	

The reference poses $(\mathbf{p}_t^{\mathrm{ref}}, \mathbf{R}_t^{\mathrm{ref}})$ are then transformed to the augmented trajectory as:

$$\mathbf{p}^{\star}(t) = \mathbf{p}_t^{\mathrm{ref}} + w(t)\,\Delta\mathbf{p}, \tag{12}$$

$$\mathbf{R}^{\star}(t) = \mathbf{R}_z\big(w(t)\,\Delta\psi\big)\,\mathbf{R}_t^{\mathrm{ref}}. \tag{13}$$

Hand wrist and fingertip positions use the same rigid map about the unperturbed object root $\mathbf{p}_{\mathrm{obj},t}^{\mathrm{ref}}$:

$$\mathbf{p}_{\mathrm{hand}}^{\star}(t) = \mathbf{R}_z\big(w(t)\,\Delta\psi\big)\big(\mathbf{p}_{\mathrm{hand},t}^{\mathrm{ref}} - \mathbf{p}_{\mathrm{obj},t}^{\mathrm{ref}}\big) + \mathbf{p}_{\mathrm{obj},t}^{\mathrm{ref}} + w(t)\,\Delta\mathbf{p}; \tag{14}$$

finger joint references are unchanged.

B.4 Domain Randomization & Curriculum

We apply domain randomization (DR) through Isaac Lab *event terms*. Each parallel environment samples its own parameters. The detailed domain randomization setup is shown in Table 5.

Curriculum. Gravity and random pushes depend on the number of environment steps. Gravity ramps from zero to $9.81\,\mathrm{m/s^2}$ by step 130k (Table 6). Pushes on robot and object activate from step 130k, reaching $\pm 0.5\,\mathrm{m/s}$ linear and $\pm 1.0\,\mathrm{rad/s}$ angular velocity at 170k, every 1–5 s.

Table 6: Gravity curriculum: sampled g_z bounds (m/s^2; g_x=g_y=0). Columns are environment-step thresholds; each cell gives the resampled range at reset.

Step $\geq$ 0	20k	30k	40k	50k	60k	70k	80k	90k	100k	110k	120k	130k
$g_{\min}$	0	0	0.5	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	9.81
$g_{\max}$	0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	9.81	9.81	9.81

Observation noise. We apply uniform noise on policy observations at every timestep (position $\pm 2, \mathrm{mm}$, orientation $\pm 0.02, \mathrm{rad}$, joints $\pm 0.02, \mathrm{rad}$). We also apply continuous observation lag to model the delay in the real-world system (uniformly sampled in $[0, 2]$ control steps per reset, with shared lag within object state, wrist pose, and hand joints).

B.5 Termination Conditions

The episode is early terminated if any of the following conditions is met: i) the object is out of the workspace; ii) the hand is moving too far away from the object; iii) the object is deviated from the target object configuration by more than a threshold (keypoint position error $> 0.15\,\mathrm{m}$); iv) for

the WUJI hand, we also terminate the episode if the contact force between the fingertip and the table is greater than a threshold (10N) for safety concerns as the real WUJI hand motors are not back-drivable.

B.6 RL Algorithm

We use PPO implemented in RSL-RL[4]. The hyperparameters are summarized in Table 7.

Table 7: RL algorithm and policy hyperparameters.

Hyperparameter	**Value**
Steps per env per iteration	24
Learning rate	10^{-3}
LR schedule	adaptive (target KL)
Desired KL	0.01
Learning epochs per iteration	5
Mini-batches per iteration	4
Discount γ	0.998
GAE λ	0.95
PPO clip ϵ	0.2
Entropy coefficient	0.002
Value loss coefficient	1.0
Max gradient norm	1.0
Actor MLP hidden dims	1024, 512, 256, 128
Critic MLP hidden dims	1024, 512, 256, 128
Activation	ELU
Actor obs. normalization	yes
Critic obs. normalization	yes
Initial action noise std	0.5 (scalar, diagonal Gaussian)

With $N_{\text{env}} = 4096$ parallel environments, each iteration collects $N_{\text{env}} \times 24 = 98{,}304$ transitions; each PPO update uses $5 \times 4 = 20$ optimizer passes with mini-batch size $98{,}304/4 = 24{,}576$. With the retargeted motion serving as the base action in residual RL, we initialize the last layer of the actor to produce zero-mean residual actions, and use a smaller initial action noise standard deviation (0.5).

C Human Demonstration Collection

We use the data from the ARCTIC dataset[5] for the scissors task. For the screwdriver task, we collect the data from our own setup. We use a motion capture system to capture the object pose. To get the hand keypoint labels, we use the HaMeR hand pose estimation model[6], which produces the 21 MANO keypoints given an RGB image. We set up four calibrated cameras at different viewpoints, and triangulate the 3D hand keypoints from the 2D image keypoints for each timestep. See Fig. 7 for an example of the estimated hand keypoints.

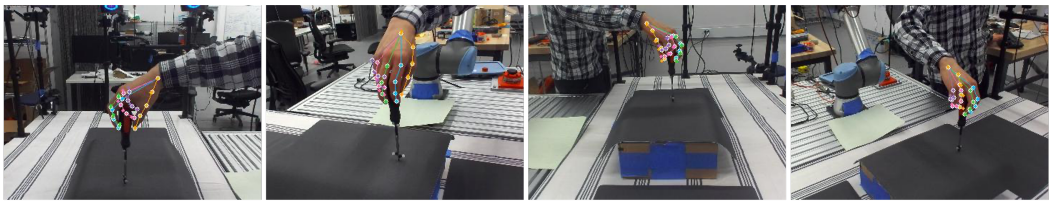


Figure 7: **Hand pose estimation.**

[4] https://github.com/leggedrobotics/rsl_rl
[5] https://github.com/zc-alexfan/arctic
[6] https://github.com/geopavlakos/hamer

D Evaluation Details

D.1 Task Success Criteria

We define the task success criteria as follows.

Scissors. Let z_t be the object root height and q_t the revolute joint angle. Define the average blade direction $\mathbf{d}_t$ as the unit vector along the mean of the two blades' body $+x$ axes in world frame (the top blade rotated by q_t about the joint axis). An timestep is *upright* if

$$z_t - z_0 > 0.05 \cdot s \text{ m} \quad \text{(lifted off the table)} \tag{15}$$

$$\mathbf{d}_t \cdot [0, -1, 0]^\top > 0.7 \quad \text{(properly oriented: body } x \text{ axis aligned with world } -y) \tag{16}$$

where z_0 is the initial height and s is the task object scale (s=2 for the 3D-printed LEAP Hand scissors, s=1 for the real-size WUJI scissors). Success holds if some contiguous upright segment contains both an opening and a closing sweep of at least $20°$ each, i.e. maximum joint increase and decrease within that segment are each $\geq 20°$.

Screwdriver. Let $\mathbf{y}_t$ be the screwdriver body $+y$ axis in world frame. A step is *upright* when $-\mathbf{y}_t \cdot [0, 0, 1]^\top > 0.9$ (body y axis aligned with world $-z$ axis). While upright, we accumulate rotation about world $+z$ from successive orientations; the accumulator resets whenever the object is not upright. Success requires the final timestep to be upright and the accumulated spin magnitude to be at least one full turn ($\geq 2\pi$).

D.2 Baseline Details

SPIDER We use the official implementation of SPIDER[7] with the MuJoCo Warp workflow.

DexMachina We use a custom implementation of DexMachina-style retargeting and RL under the same framework as our method. We apply the same collision-aware post-processing step on top of the IK retargeted trajectories as in DexMachina. The downstream RL training is the same as our method.

Mink IK + RL This baseline simply replaces the retargeted trajectories in our method with the IK retargeted trajectories from Mink[8].

E Real-World Robot System Details

Hardware Setup. The system consists of a 7-DoF UR5e robot arm and a 16-DoF LEAP hand (or 20-DoF WUJI hand) and a motion capture system with 9 cameras for object pose estimation. See Fig. 8 for the system setup.

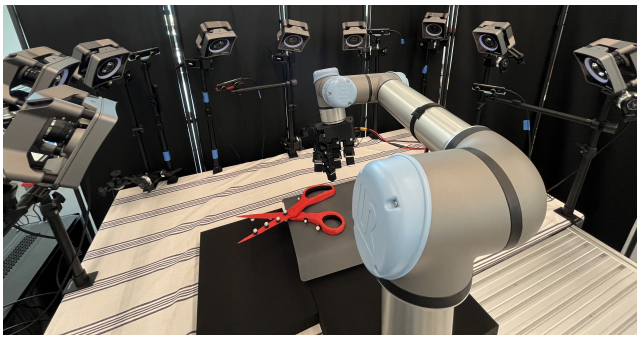


Figure 8: **Real-World Robot System.**

[7] https://github.com/facebookresearch/spider
[8] https://github.com/kevinzakka/mink

System Identification. We perform system identification to align the robot dynamics between simulation and the real-world system. We record the policy rollouts in simulation, replay the joint targets on the real robot, and compare the response curves with the simulated ones. Fig. 9 shows the response curves for one trajectory on the LEAP hand. We observe that the real-robot response is slightly delayed by 1-2 timesteps ($\approx 30 - 60\,\mathrm{ms}$), which motivates us to apply a time lag in the observation pipeline in simulation to model the delay in the real-world system.

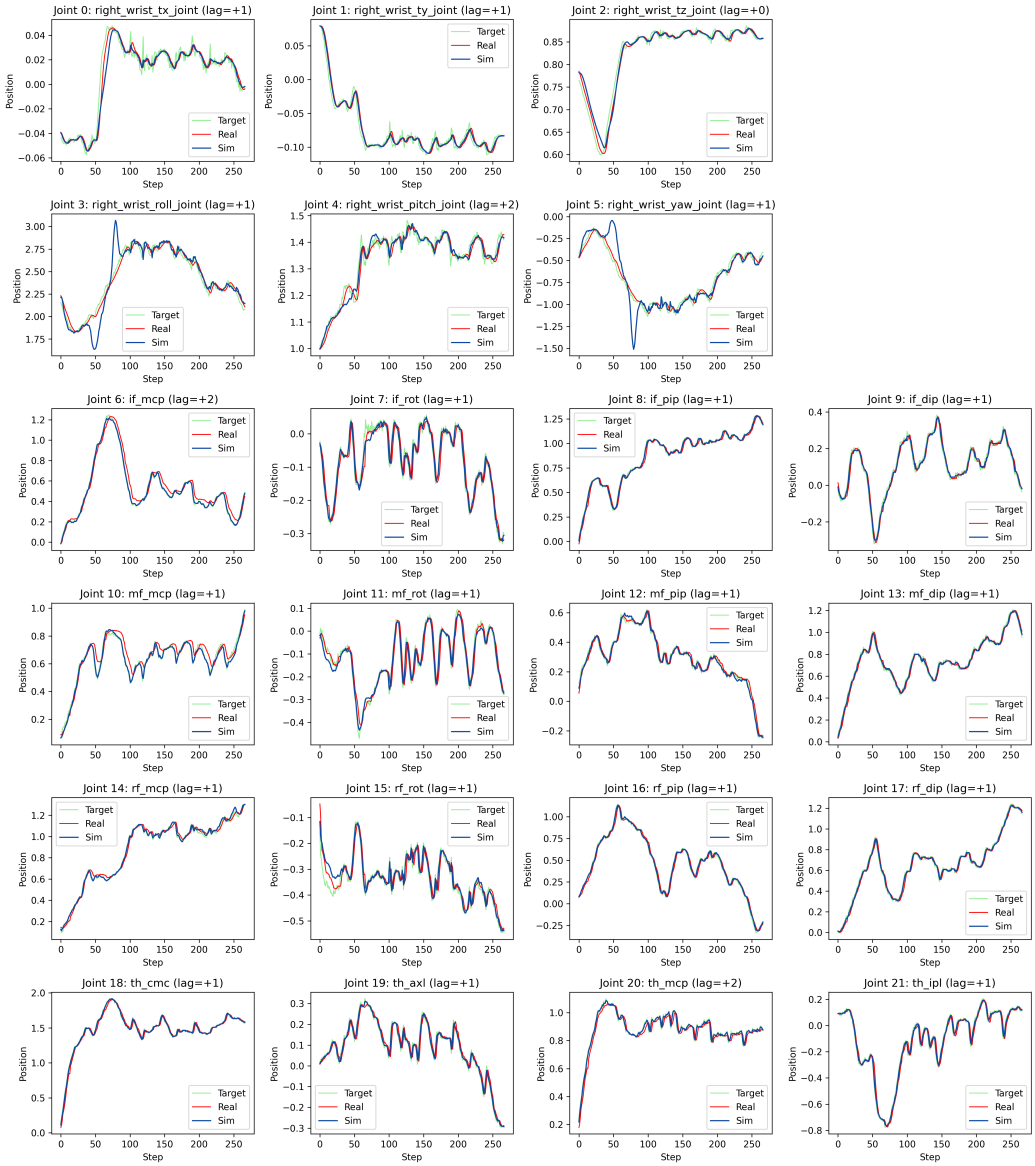


Figure 9: **Response curves for system identification.** Green: desired joint position; Red: real-robot response; Blue: simulated response.