REDEXT: LEARNING A GENERALIZABLE RESIDUAL POLICY FOR DEXTEROUS RETARGETING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dexterous retargeting often requires fitting robot controls to each human hand-object motion, making large collections expensive to process. Offline inverse kinematics provides inexpensive nominal controls, but cannot correct errors caused by contact and object dynamics. We present REDEXT, a two-stage reinforcement-learning framework that learns one residual policy to execute new trajectories from inverse-kinematics commands. The key is to learn reusable feedback by varying both the commands the policy corrects and the motions it practices. We first perturb successful source commands while keeping their motion targets fixed, so the policy learns to correct varied execution errors. Rollout-guided training then broadens trajectory coverage through fresh on-policy learning, without treating rollout actions as imitation labels. Once trained, the shared policy executes new trajectories with frozen weights and supports optional local or shared adaptation for better tracking. On new trajectories screened for initial stability in single-hand simulation, frozen REDEXT achieves higher success and shorter processing times than the evaluated per-trajectory methods. Shared adaptation further raises success from 55.08% to 80.08% across 256 test trajectories. Ablations show that control perturbations ease transfer to kinematic commands, and expanded IK training broadens success coverage. The training recipe extends to four additional robot hands, each with a separately trained policy.

1 INTRODUCTION

Dexterous retargeting turns demonstrated hand-object motion into robot behavior (Antotsiou et al., 2018; Qin et al., 2022). Motion datasets offer diverse reference trajectories (Chao et al., 2021; Zhan et al., 2024), but executing them requires more than geometric correspondence. Inverse kinematics provides inexpensive robot-specific nominal controls without optimizing contact forces or object dynamics. Physics-based methods such as SPIDER (Pan et al., 2025) optimize controls for individual trajectories in simulation. A shared feedback controller could reuse experience across motions and avoid fitting each new trajectory separately (Figure 1).

Shared residual policies offer a way to reuse such feedback by correcting nominal controls (Silver et al., 2018; Johannink et al., 2019). DexH2R (Zhao et al., 2025) and DexTrack (Liu et al., 2025) learn shared corrections across kinematic trajectories; DexTrack combines reinforcement learning with action imitation from mined trajectory demonstrations. SPIDER also trains residual feedback over its optimized controls for humanoid motion. Our focus is learning shared dexterous feedback from successful source controls and transferring it to nominal inverse-kinematics controls, which we call IK priors, on new trajectories. The source controls guide simulator interaction during initial training. Subsequent learning uses IK without optimized action labels for additional tasks.

Reusing feedback requires handling two distinct changes. Replacing source controls with IK changes the nominal commands and execution errors even when the reference stays the same. Moving to a new trajectory also changes the required hand-object motion. A policy that handles the first change may struggle with the second. Our training combines variation in nominal controls with exposure to a broader set of trajectories.

We introduce REDEXT, a two-stage framework for **Re**sidual learning for **Dex**terous re**T**argeting. The first stage trains a shared residual policy on success-screened seed tasks. A performance-adaptive curriculum applies temporally correlated perturbations to successful source commands,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

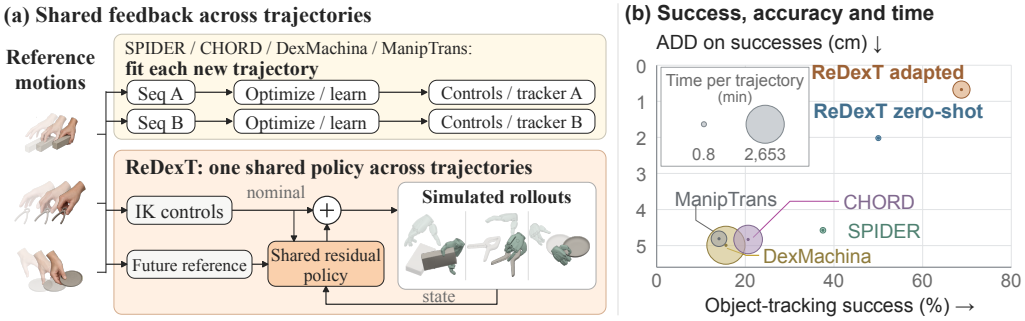

Figure 1: REDEXT reuses one policy to retarget new motions. Left: different reference trajectories share residual feedback over IK controls, producing physical rollouts without further physics-based fitting. Right: on 32 OOD trajectories, frozen execution combines higher success with lower reported processing cost than the external baselines; local adaptation further improves tracking. ADD averages each method's successful trajectories. Circle area is proportional to the square root of post-IK time per trajectory; smaller circles indicate faster processing. Upstream training is excluded.

exposing the policy to varied execution errors for the same reference. Conditioned on the execution state, reference, and nominal command, the policy learns feedback through reinforcement learning.

The second stage extends training beyond the source tasks. Clean-IK rollouts identify current successes and selected failures, which join the seeds for continued on-policy learning around their original IK controls. At deployment, one frozen policy corrects offline IK commands for each new trajectory (Figure 2). This provides an inexpensive first pass over a collection of motions; optional shared adaptation refines that collection with a single policy.

On the comparison subset of 32 trajectories, frozen REDEXT achieves 50.00% success, versus 37.50% for SPIDER and 14.06% to 20.63% for the evaluated learning-based methods. It processes each trajectory in a reported 0.8 minutes, less than these methods require; local adaptation raises success to 68.75%. On the full test set of 256 trajectories, frozen success reaches 55.08%, exceeding matched-budget full-pool training by 5.86 percentage points in the evaluated runs. Shared adaptation raises success to 80.08% and horizon completion to 75.39%. Separately trained frozen policies also achieve higher success than physical replay of SPIDER controls on four additional robot hands.

Our contributions are:

- A shared residual-learning framework that executes new trajectories from offline IK with frozen weights. Training reuses experience across motions without using candidate rollout actions as imitation labels.
- A two-stage training design combining control perturbations with broader IK training. Ablations show reduced source-to-IK sensitivity and examine the roles of control source, candidate selection, and control range.
- A frozen shared policy with higher success and shorter processing times than the evaluated per-trajectory methods, plus further gains through local or shared adaptation. Experiments on five robot hands demonstrate the wider applicability of the training recipe.

2 RELATED WORK

Dexterous retargeting and trajectory generation. Retargeting maps human motion to robot behavior under embodiment and contact constraints. Keypoint and object-relative objectives support task-aware optimization, video imitation, and teleoperation (Antotsiou et al., 2018; Qin et al., 2022; Handa et al., 2020; Qin et al., 2023). Because geometric correspondence alone does not ensure successful contact, physics-based methods optimize or learn robot behavior in simulation (Chen et al., 2024; Mandi et al., 2025). SPIDER (Pan et al., 2025) searches for a control sequence per trajectory and demonstrates PPO residual policies over its optimized controls for humanoid motion. That residual experiment does not report shared dexterous tracking over IK on new trajectories. CHORD (Zhu et al., 2026) guides RL through object-centric contact-wrench matching. ManipTrans (Li et al.,

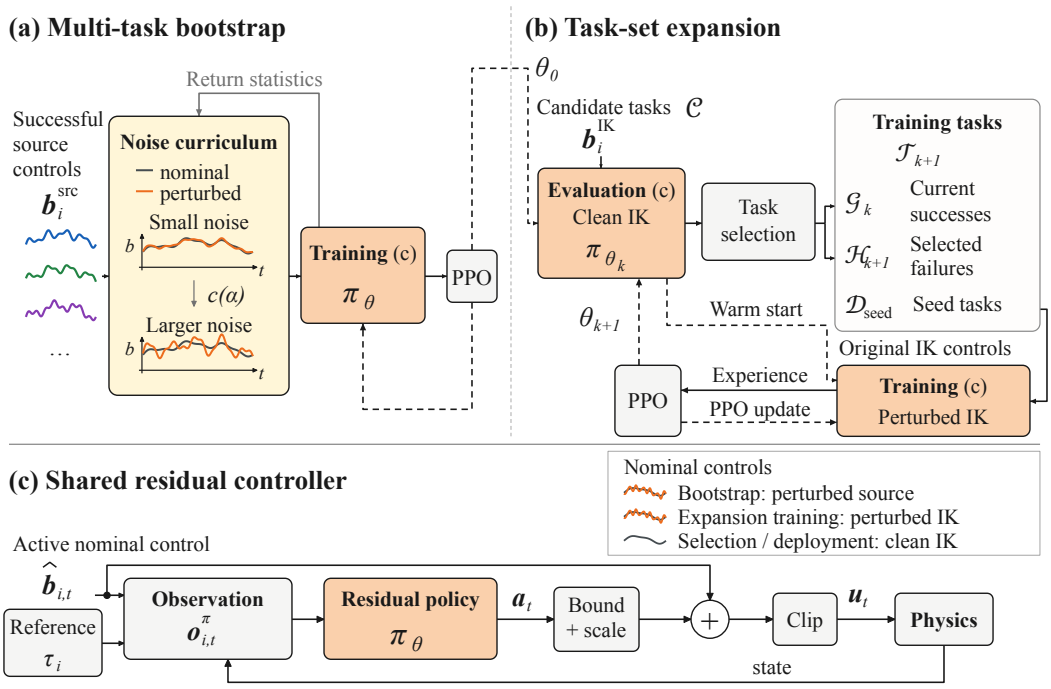

Figure 2: Overview of REDEXT. (a) Bootstrap training learns shared feedback by perturbing successful source controls while keeping motion targets fixed. (b) IK rollouts identify trajectories for continued on-policy training, broadening the policy's experience beyond the seeds. (c) The shared policy combines the execution state, reference motion, and nominal control to predict residual corrections. The same controller is used throughout training and frozen for deployment. Dashed arrows indicate updates or warm starts.

2025) reuses a generalist trajectory imitator and learns task-specific residual refinements. Our shared feedback executes new trajectories from offline IK without policy updates.

Residual policies over control priors. Residual policy learning parameterizes control as a nominal output plus a learned correction (Silver et al., 2018; Johannink et al., 2019). Learned demonstration and skill priors extend this principle beyond hand-designed controllers (Alakuijala et al., 2021; Rana et al., 2023), while online correction and policy refinement adapt existing behaviors (Jiang et al., 2024; Yuan et al., 2025). Residual Flow Steering (Su et al., 2026) combines action corrections with latent-space steering of a flow-matching policy. Residuals can also be shared: ResDex (Huang et al., 2025) learns residual grasping across objects, and DexH2R (Zhao et al., 2025) corrects retargeted primitive actions with a shared residual conditioned on desired hand-object motion. REDEXT builds on this parameterization to learn shared, reference-conditioned feedback. Training perturbs nominal controls at fixed motion targets, then broadens the set of trajectories used for IK training.

Training experience for shared controllers. Shared controllers learn from both online exploration and generated demonstrations. Dexplore (Xu et al., 2025) guides exploration around motion references during training. AdaDexTrack (Adalibieke et al., 2026) combines a distilled generalist tracker with online modulation of references, object representations, and positional targets. UniDexGrasp (Xu et al., 2023) and UniDexGrasp++ (Wan et al., 2023) develop generalist grasping through multi-object learning and curricula. Demonstration mining supplies an additional learning signal for shared tracking. DexTrack (Liu et al., 2025) trains a shared residual tracker over a kinematic baseline with RL and action imitation, alternating training with per-trajectory homotopy demonstration mining. It also evaluates generalization to unseen trajectories and objects. CyberDemo (Wang et al., 2024), DexMimicGen (Jiang et al., 2025), and DexFlyWheel (Zhu et al., 2025) expand demonstrations through augmentation or iterative generation. Probe, Learn, Distill (Xiao et al., 2026) distills task-specific residual rollouts into a vision-language-action policy. In REDEXT, successful seed controls supply nominal behavior for RL. We use candidate rollouts to choose trajectories for fresh on-policy learning over their original IK controls, rather than to collect action-imitation labels.

3 METHOD

Offline IK supplies nominal motion for each trajectory but cannot correct execution errors. REDEXT adds feedback from a shared policy. Training prepares it for the switch from source controls to IK and for new motions. We first vary nominal controls while holding seed trajectories fixed, then broaden trajectory coverage through rollout-guided IK training. Figure 2 shows bootstrap, task-set expansion, and the shared controller in panels a, b, and c.

3.1 PROBLEM FORMULATION AND SOURCE CONTROLS

A task i specifies a robot-object instance, an initial simulator state, a reference trajectory τ_i of desired object, wrist, and fingertip states, and a nominal joint-position control sequence $\boldsymbol{b}_i$. Its active scoring length is T_i frames. Every task has an IK sequence $\boldsymbol{b}_i^{\mathrm{IK}}$, computed offline using robot geometry and joint limits without optimizing contact forces or dynamic feasibility. Seed tasks additionally provide a reference-aligned successful source sequence $\boldsymbol{b}_i^{\mathrm{src}}$ for bootstrap training. These sequences supply nominal behavior for simulator interaction, not action-imitation labels. We denote the disjoint seed and candidate task sets by $\mathcal{D}_{\mathrm{seed}}$ and $\mathcal{C}$. Seed source controls must pass the success criterion below; candidates provide additional trajectories for IK-based training. We obtain source controls with SPIDER (Pan et al., 2025).

At step t, a shared stochastic policy samples $\boldsymbol{a}_t \sim \pi_\theta(\cdot \mid \boldsymbol{o}_{i,t}^{\pi})$. Let $\widehat{\boldsymbol{b}}_{i,t}$ be the active nominal control: curriculum-perturbed during training and clean during standard evaluation. The applied control is

$$\boldsymbol{u}_t = \mathrm{clip}\Big(\widehat{\boldsymbol{b}}_{i,t} + \boldsymbol{D}\,\mathrm{clip}(\boldsymbol{a}_t, -1, 1), \boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\Big), \tag{1}$$

where the diagonal matrix $\boldsymbol{D}$ scales residuals by actuator ranges and control coefficients. The controller applies this command at each step $t \geq 1$.

A single object-tracking criterion evaluates valid recorded rollouts against reference frames at the corresponding simulation times. Let $\bar{e}_p$ and $\bar{e}_R$ be the worst, across active objects, time-averaged translation and geodesic rotation errors. A rollout with control sequence $\boldsymbol{u}$ succeeds iff

$$\mathrm{Succ}(\tau_i, \boldsymbol{u}) = \mathbf{1}[\bar{e}_p \leq 0.10\ \mathrm{m}\ \wedge\ \bar{e}_R \leq 0.50\ \mathrm{rad}]. \tag{2}$$

This criterion defines source-control labels, successful candidates, and evaluation success. Full-horizon completion is a separate diagnostic.

3.2 SHARED RESIDUAL CONTROL

Different trajectories can require different responses to the same tracking error. We therefore condition shared feedback on execution state, nominal control, and upcoming reference motion. The actor combines proprioception, object and wrist tracking state, object-surface proximity and scale, and the active nominal control in $\boldsymbol{x}_{i,t}$; a future-reference token $\boldsymbol{z}_{i,t}^{\mathrm{future}}$ supplies the upcoming motion targets. We encode future relative object, wrist, and fingertip motion with a discrete cosine transform to obtain a compact token. The actor and critic inputs are

$$\boldsymbol{o}_{i,t}^{\pi} = [\boldsymbol{x}_{i,t-\delta}, \boldsymbol{z}_{i,t-\delta}^{\mathrm{future}}], \qquad \boldsymbol{o}_{i,t}^{V} = [\boldsymbol{x}_{i,t}, \boldsymbol{z}_{i,t}^{\mathrm{future}}, \phi_{i,t}, \boldsymbol{\eta}_i], \tag{3}$$

where δ is the sampled observation delay in steps, $\phi_{i,t}$ is normalized trajectory phase, and $\boldsymbol{\eta}_i$ contains privileged physical parameters available only to the critic. A Gaussian MLP actor predicts the residual in Equation 1. One policy is shared across tasks and warm-started across training stages; its weights are frozen for deployment. The reward combines object tracking, trajectory progress, fingertip-object surface-distance agreement, missed-contact penalties, and action regularization. PPO (Schulman et al., 2017) maximizes discounted return averaged over active task and temporal-augmentation pairs.

3.3 SEED-TASK BOOTSTRAP WITH CONTROL PERTURBATIONS

A policy trained over source controls must handle different nominal commands when deployed over IK. We prepare it for this switch by perturbing successful source controls while keeping the

reference fixed. The policy learns feedback for the same desired motion under varied execution errors. Temporally correlated noise sustains these perturbations across steps:

$$\boldsymbol{n}_t = \rho \boldsymbol{n}_{t-1} + \sqrt{1-\rho^2}\boldsymbol{\epsilon}_t, \qquad \widehat{\boldsymbol{b}}_{i,t} = \boldsymbol{b}_{i,t} + c(\alpha)\,\frac{\boldsymbol{D}}{2}\boldsymbol{n}_t, \tag{4}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\rho \in [0,1)$ controls temporal correlation, and $c(\alpha) \in [0,1]$ sets the noise scale at curriculum state α. The perturbed control enters the observation before history sampling: the actor sees its delayed value as in Equation 3, while control composition uses the current value. Clean-IK evaluation disables both delay and noise.

We increase control variation gradually according to training returns. Bootstrap begins at zero noise. Difficulty advances only after the mean of per-task return estimates stabilizes. If performance declines, training returns to the preceding anchor checkpoint and tries an intermediate noise level. Each task-set expansion round inherits the previous checkpoint and curriculum difficulty, now perturbing IK rather than source controls.

3.4 Rollout-Guided IK Training and Deployment

Control perturbations expose the policy to execution errors, but do not add new motion targets. We therefore extend training to candidate trajectories using their IK priors. Each round retains the seeds and revisits currently successful candidates, while selected failures provide experience on unresolved tasks. Clean-IK rollouts determine which candidates enter the next round. Let θ_k be the checkpoint after round k, with θ_0 obtained from seed training. For each candidate, we evaluate one rollout using mean actions and its original clean IK controls. $\mathrm{Rollout}(\pi, \boldsymbol{b})$ denotes the resulting applied-control sequence. The current success set is

$$\mathcal{G}_k = \left\{ i \in \mathcal{C} : \mathrm{Succ}\big(\tau_i, \mathrm{Rollout}(\pi_{\theta_k}, \boldsymbol{b}_i^{\mathrm{IK}})\big) = 1 \right\}. \tag{5}$$

To obtain training experience on failed tasks, we include up to B current failures. With $\mathcal{F}_k = \mathcal{C} \setminus \mathcal{G}_k$ and $M_k = \min(B, |\mathcal{F}_k|)$, selection and the next-round task set are

$$\begin{aligned} \mathcal{H}_{k+1} &= \mathrm{Select}(\mathcal{F}_k, M_k), \qquad |\mathcal{H}_{k+1}| = M_k, \\ \mathcal{T}_{k+1} &= \mathcal{D}_{\mathrm{seed}} \cup \mathcal{G}_k \cup \mathcal{H}_{k+1}. \end{aligned} \tag{6}$$

The default selector chooses failures with the lowest tracking error normalized by the success thresholds. We compare this rule with uniform random selection and other task-set choices in Section 4.4. Each round warm-starts from θ_k and collects fresh stochastic on-policy experience around every selected task's original IK prior, including seeds, under the inherited curriculum. These original IK priors are also available at deployment. Selection rollouts only determine task membership; their actions neither replace the priors nor serve as imitation labels. Test trajectories are excluded from bootstrap, expansion, and configuration selection. Stopping and checkpoint rules are fixed, and updates on test trajectories are evaluated separately as local or shared adaptation. Deployment uses the frozen policy's mean residual action in Equation 1, with control noise and observation delay disabled. Appendix A details implementation, training, and multi-task acceleration.

4 Experiments

We compare shared feedback with per-trajectory fitting on new motions. We also test training choices, adaptation, and applicability to other hands.

4.1 Experimental Setup

Benchmark. We use a simulated right-handed Sharpa hand and 921 hand–object reference trajectories from OAKINK2 (Zhan et al., 2024) and DexYCB (Chao et al., 2021), screened for initial stability. Left-hand OAKINK2 motions are mirrored. Bootstrap uses 192 seeds with SPIDER-optimized controls (Pan et al., 2025), followed by expansion over 473 candidates. The OOD test set contains 256 trajectories: 154 seen-object and 102 unseen-object motions, with object identity defined by the converted assets in the seed and candidate sets. Left/right-hand variants and their mirrored derivatives stay in the same split. Test trajectories are excluded from pretraining and

Table 1: Performance on (A) an OOD subset of 32 trajectories and (B) the full set of 256 trajectories. Best values within each block are bold. SR/HCR are percentages; ADD (m) averages each method's successes. Time* is post-IK minutes per trajectory, excluding upstream training. ‡Shared time includes joint training amortized over 256 targets. †ManipTrans uses its native setup.

(A) OOD subset: 32 trajectories

Method	SR ↑	ADD ↓	Time* ↓
No test-time adaptation			
IK-only	12.50	0.0337	1.1
Full-pool	43.75	**0.0200**	**0.8**
REDEXT	**50.00**	0.0202	**0.8**
Per-trajectory optimization or learning			
SPIDER	37.50	0.0457	**1.2**
CHORD	20.63	0.0483	840.7
DexMachina	15.63	0.0499	2653.2
ManipTrans†	14.06	0.0481	73.7
REDEXT (local)	**68.75**	**0.0067**	109.9

(B) Full OOD set: 256 trajectories

Method	SR ↑	HCR ↑	Time* ↓
No test-time adaptation			
IK-only	15.23	0.78	1.1
IK-RL	11.72	2.34	0.7
IK-RL+Noise	31.64	12.89	**0.6**
Full-pool	49.22	34.38	0.8
REDEXT-B	35.94	20.70	0.8
REDEXT	**55.08**	**39.06**	0.8
Joint adaptation on the target set			
Full-pool (shared)	74.22	71.48	**2.7**‡
REDEXT (shared)	**80.08**	**75.39**	**2.7**‡

configuration selection; only the adaptation experiments update policies on them. Shared-policy comparisons use all 256 test trajectories; costly per-trajectory baselines use a fixed random subset of 32, with 20 seen-object and 12 unseen-object trajectories.

Comparisons. *IK-only* replays IK without learned feedback. Two internal shared residual baselines, motivated by DexH2R (Zhao et al., 2025), train from scratch on the 192 seeds: *IK-RL* without control noise and *IK-RL+Noise* with the noise curriculum. *Full-pool* jointly trains from scratch with IK on all 665 training trajectories, matching REDEXT's architecture, control and noise scales, other settings, and total interactions. REDEXT-B and REDEXT denote the default bootstrap and expanded policies. Both are frozen over clean IK unless marked *local* for independent per-trajectory adaptation or *shared* for joint adaptation on the target set. *Local-RL* learns from scratch for each target. External methods include SPIDER, ManipTrans (Li et al., 2025), and our single-hand adaptations of CHORD (Zhu et al., 2026) and DexMachina (Mandi et al., 2025). Internal Sharpa comparisons share assets, simulator, trajectories, and scoring. CHORD uses Sharpa in Isaac Lab/PhysX; ManipTrans retains its native Shadow hands, reference preprocessing, horizon, and Isaac Gym, as indicated by the dagger in Table 1(A).

Metrics. Mean-error success rate, or SR, follows Equation 2. Horizon completion rate, or HCR, requires a full active-horizon record with no termination signal before the last active step. Instantaneous object errors above 0.10 m or 0.50 rad, or non-finite states, trigger this signal. Finite rollouts continue without threshold-triggered resets and can still pass the mean-error criterion. Rates weight trajectories equally, average repeated attempts per trajectory, and retain failed or unavailable cases in the denominator. Object errors measure average distance of model points, denoted ADD, translation, and rotation; hand errors measure wrist pose, fingertips, and finger joints. We average errors over active frames and then successful trajectories, and also compare common successes after adaptation. Reported processing time covers initialization, target-specific fitting, and rollout after IK, excluding upstream preparation and training. Frozen-policy and SPIDER times reuse the original timing profiles from the 32-trajectory subset. Training uses H200 GPUs; evaluation uses RTX 4090 GPUs. Appendix B details the data and evaluation protocols.

4.2 ZERO-SHOT RETARGETING ON NEW TRAJECTORIES

We first compare frozen shared feedback with fitting each new trajectory. Zero-shot execution uses offline IK without policy updates. On the subset of 32 trajectories, REDEXT reaches 50.00% SR, compared with SPIDER's 37.50% and 14.06% to 20.63% for the evaluated learning-based methods (Table 1(A)). Reported processing takes 0.8 minutes per trajectory, compared with 1.2 for SPIDER and 73.7 to 2653.2 for the learning-based methods. Frozen shared feedback succeeds on more trajectories at lower reported cost; adaptation spends additional computation on better tracking.

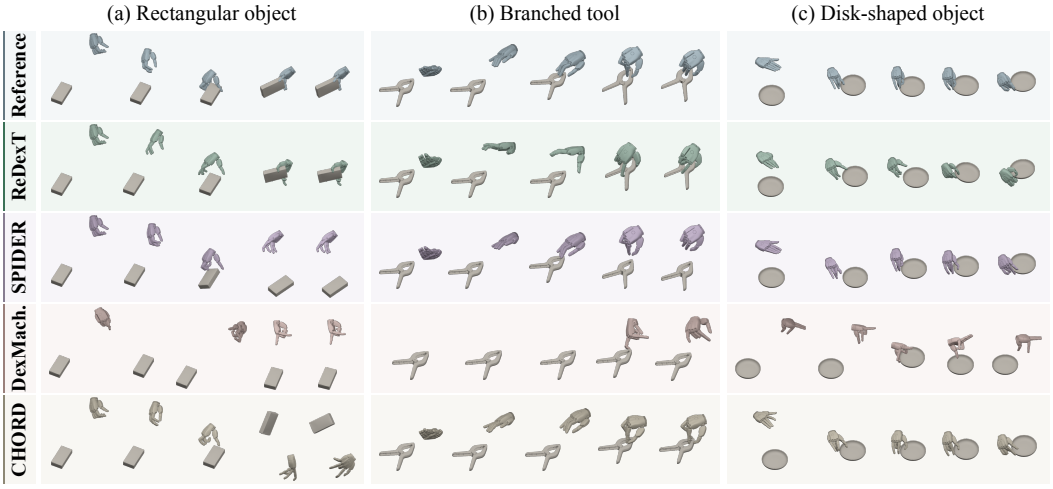

Figure 3: Qualitative comparison on new trajectories with a rectangular object, a branched tool, and a disk. Each row shows five snapshots in temporal order. REDEXT uses its frozen policy; CHORD and DexMachina use their single-hand adaptations.

Table 2: Control noise and trajectory coverage. (A) Frozen bootstrap policies with SPIDER controls and IK; drops are in percentage points. (B) Training-set variants on 256 new trajectories, frozen (F) and after 20k shared updates (S). All use coefficients $(.004, .004, 1.2)$; rates are percentages. Best values are bold within each metric and setting.

(A) Control switch: 192 seeds

Metric	Noise	SP ↑	IK ↑	Drop ↓
SR	Off	89.06	58.33	30.73
SR	On	**90.63**	**86.98**	**3.65**
HCR	Off	**82.81**	40.63	42.19
HCR	On	81.25	**72.92**	**8.33**

(B) OOD test set: 256 trajectories

Route	SR-F ↑	HCR-F ↑	SR-S ↑	HCR-S ↑
Seed cont.	30.08	17.19	62.50	53.13
All candidates	39.45	20.70	66.02	58.59
Success-only	39.84	**23.83**	**70.31**	**65.23**
Lowest-error	40.23	21.88	66.41	60.16
Random-failure	**41.02**	**23.83**	67.19	61.72

On all 256 test trajectories, frozen REDEXT reaches 55.08% SR and 39.06% HCR, adding 15 successes and 12 completed horizons over matched-budget Full-pool training (Table 1(B)). ADD is nearly identical on their 110 common successes, so the main gain is broader success coverage. These single-run gains support the full recipe without isolating its stages. With matched nominal controls and resets, feedback raises IK-only SR from 15.23% to 55.08% and HCR from 0.78% to 39.06%. It corrects 105 of 217 IK failures and retains 36 of 39 IK successes. Figure 3 illustrates the correction: frozen REDEXT follows the rectangular object's reorientation, while DexMachina and CHORD show hand-object separation.

Mean-error success does not guarantee full-horizon execution. Of 141 frozen successes, 41 signal termination before the last active step, and only 72 of the 105 corrected IK failures complete the horizon. Frozen SR is 75.32% on seen-object and 24.51% on unseen-object trajectories; this gap also reflects differences in dataset and duration, so it does not isolate object geometry. Appendices C.1, C.5, and C.7 give results, diagnostics, and further examples.

4.3 BOOTSTRAP ABLATIONS

We test how perturbations affect transfer from source commands to IK. Policies trained over SPIDER controls with and without noise are frozen at 50k updates and tested on the same 192 seeds under both control sources. These diagnostics use coefficients $(.004, .004, 1.2)$, rather than the default $(.005, .10, 1)$ used by both REDEXT and Full-pool above; other bootstrap settings match. Despite similar performance under SPIDER controls, switching to IK reduces SR by 3.65 percentage points with noise, versus 30.73 without it (Table 2(A)). The HCR drop also shrinks, indicating less dependence on the source commands.

This benefit extends to new trajectories. Under clean IK on the full OOD set, SPIDER-base and IK-base bootstrap policies both reach 28.13% SR, or $72/256$ successes, with noise. Their no-noise counterparts reach 12.11% and 13.28%, respectively. The two noisy routes train on the same 192 SPIDER-selected seeds and differ only in control source, so source commands alone provide no aggregate SR advantage at this budget. These IK tests evaluate noise and its schedule together.

Input ablations identify which information helps the policy track a trajectory. Under SPIDER controls, removing object-surface features, future-reference context, or translation centering reduces overall bootstrap success. Removing nominal-control observations instead improves it. Thus geometry and upcoming motion help, but not every input is necessary for this test. Appendix C.2 reports the full training and input ablations, including paired outcomes and settings.

4.4 Expansion and Control-Scale Ablations

We next test continued training on more trajectories. Seed-only continuation and training with all 473 candidates share the ablation configuration, bootstrap initialization, and 50k further IK updates under matched configured environment budgets (Table 2(B)). Broader training raises frozen SR from 30.08% to 39.45%. Expansion resets stage-local curriculum statistics, while seed continuation restores the saved state, so this gain cannot be attributed to exposure alone.

Continued perturbations matter more in these runs than candidate selection. Success-only, Lowest-error, Random-failure, and All candidates span 39.45% to 41.02% frozen SR, a difference of just four successes. All four exceed seed continuation in frozen SR and HCR, but these single runs do not establish a reliable selection advantage. Lowest-error reaches 40.23% frozen SR with noise in both stages, versus 25.78% with bootstrap noise alone and 23.44% without noise. Broader IK training and continued perturbations thus have stronger support than the selection heuristic.

Independent frozen evaluations improve from $92/256$ successes at bootstrap to $141/256$ at R5, but 15 of 425 candidates solved during training fail at R5. Control range also matters: changing the coupled residual and noise scales raises frozen SR from 40.23% to 55.08% at fixed depth, whereas adding a hidden layer with the ablation control scales reaches 41.80%. Appendix C.3 gives the retention and training-design comparisons.

4.5 Adaptation to New Trajectories

Per-trajectory adaptation. We test whether pretraining accelerates learning on each new trajectory. Local-RL starts randomly, ReDexT-B from the default bootstrap, and ReDexT from the expanded policy. All receive 5,000 PPO updates under identical clean-IK training, or 30.72 million interactions per trajectory, excluding upstream training. ReDexT reaches 70.31% SR and 66.02% HCR, compared with 22.66% and 4.30% from random initialization and 66.41% and 59.77% from bootstrap (Figure 4). At 1,501 updates, ReDexT already reaches 67.19% SR and 61.72% HCR, exceeding the bootstrap initialization's 5,000-update endpoint on both metrics.

Local adaptation also strengthens the external comparison: on the subset of 32 trajectories, SR rises from 50.00% to 68.75% (Table 1(A)). Local adaptation takes a reported 109.9 minutes per trajectory, compared with 840.7 for CHORD and 2653.2 for DexMachina, while achieving higher success. On 55 common successes from the full OOD set, the expanded initialization reduces all seven object and hand errors. ADD falls to 0.00614 m, compared with 0.01664 m from bootstrap and 0.05903 m from random initialization. Bootstrap nevertheless retains slightly higher SR on unseen-object and OAKINK2 trajectories.

Shared-policy adaptation. Shared adaptation updates one policy across new motions. After 20,000 joint updates on all 256 targets with matched interaction budgets, ReDexT reaches 80.08% SR and 75.39% HCR, compared with 74.22% and 71.48% from Full-pool. The default model takes about 8.1 hours on one H200 for the entire target set. Horizon completion also improves: 38 of 41 frozen mean-only successes become horizon-complete, and 97 of 100 completed horizons remain so. On 139 common successes, ADD falls from 0.0194 to 0.0101 m, showing that adaptation improves object tracking accuracy as well as success rate.

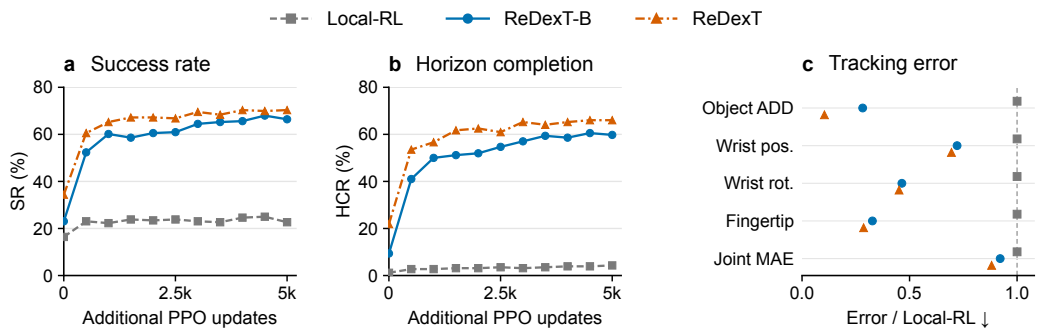

Figure 4: Adaptation to 256 new trajectories from random, bootstrap, and expanded initializations. (a,b) Success and horizon completion over 5,000 PPO updates. (c) Final object and hand errors on 55 common successes, normalized by Local-RL for each metric; lower is better.

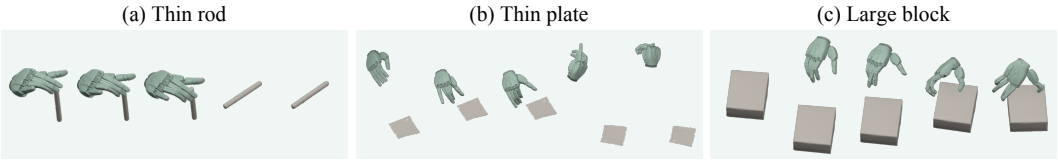

Figure 5: Grasping failures of frozen REDEXT on a thin rod, a thin plate, and a large block. Each example shows five snapshots in temporal order.

Failure analysis. Shared adaptation resolves 66 of 115 frozen SR failures but loses two previous successes, leaving 51 failures. Figure 5 shows frozen-policy failures on a thin rod, a thin plate, and a large block. These examples motivate examining object dimensions and contact geometry, without establishing either as the cause. Other observed failures include displacement during approach and drops after lifting. Failure at the current training budget does not establish physical infeasibility. Appendices C.4 and C.5 give adaptation results and outcome transitions.

4.6 APPLICABILITY TO OTHER HANDS

We test the recipe on Allegro, Inspire, XHand, and Shadow, training each policy from scratch. On the same 256 OOD trajectories, frozen policies achieve 44.92–51.95% SR; 20k shared updates raise this range to 67.19–78.52%. Before adaptation, they exceed physical replay of SPIDER controls by 8.20 to 13.67 percentage points under the same evaluation protocol. Adaptation also lowers ADD on each hand's common successes. Across the four additional hands, shared adaptation adds 57–70 net successes per hand while losing only 1–3 frozen successes, expanding coverage while preserving most previously successful motions. Across all five hands, adaptation yields larger gains in HCR than in SR, improving full-horizon execution as well as success under the average-error criterion. Each hand uses its own training pool and control scales; weights are not transferred between hands. Appendix C.6 gives detailed results and examples.

5 CONCLUSION

REDEXT learns shared feedback for new trajectories through two-stage training. Control perturbations ease source-to-IK transfer, while training on additional trajectories broadens coverage. The frozen policy processes new motion collections with higher success and shorter processing times than the evaluated per-trajectory methods. When greater coverage or accuracy is needed, joint adaptation refines the collection with one policy, raising success from 55.08% to 80.08% on 256 trajectories. Separately trained policies extend the approach to four additional hands.

Limitations and future work. Our experiments cover single-hand manipulation in simulation, with a separate policy for each hand. Extending the method to coordinated bimanual manipulation and sharing one residual policy across embodiments are natural next steps. The scale and diversity of the motion data also remain limited, motivating larger datasets and evaluation on longer, more complex trajectories. Validation on physical hands remains open.

AI USE STATEMENT

Generative AI tools assisted manuscript drafting, language revision, and LaTeX formatting. They also supported figure design and generated illustrative human hand-object images for Figure 1; these images are not experimental observations. The authors reviewed and revised the AI-assisted material and take responsibility for the final text, claims, and artifacts in this submission.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the method and evaluation protocol. Appendix A documents policy inputs, control, optimization, and task-set expansion; Appendix B specifies the data, fixed splits, evaluation metrics, and baseline adaptations.

REFERENCES

Jianibieke Adalibieke, Qianwei Han, Xueyi Liu, Yuzhe Qin, and Li Yi. Adadextrack: Dynamic modulation for adaptive and generalizable dexterous manipulation tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28021–28031, 2026.

Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021.

Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Task-oriented hand motion retargeting for dexterous manipulation imitation. In *European Conference on Computer Vision Workshops*, pp. 287–301. Springer, 2018.

Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. DexYCB: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9044–9053, 2021. doi: 10.1109/CVPR46437.2021.00893.

Yuanpei Chen, Chen Wang, Yaodong Yang, and C Karen Liu. Object-centric dexterous manipulation from human motion data. *arXiv preprint arXiv:2411.04005*, 2024.

Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 9164–9170. IEEE, 2020.

Ziye Huang, Haoqi Yuan, Yuhui Fu, and Zongqing Lu. Efficient residual learning with mixture-of-experts for universal dexterous grasping. In *International Conference on Learning Representations*, volume 2025, pp. 87492–87510, 2025.

Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv:2405.10315*, 2024.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930. IEEE, 2025.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6991–7003. IEEE, 2025.

Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. In *International Conference on Learning Representations*, volume 2025, pp. 85970–85994, 2025.

Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation. *arXiv preprint arXiv:2505.24853*, 2025.

Chaoyi Pan, Changhao Wang, Haozhi Qi, Zixi Liu, Homanga Bharadhwaj, Akash Sharma, Tingfan Wu, Guanya Shi, Jitendra Malik, and Francois Hogan. SPIDER: Scalable physics-informed dexterous retargeting. *arXiv preprint arXiv:2511.09484*, 2025.

Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pp. 570–587. Springer, 2022.

Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In *Conference on Robot Learning*, pp. 2095–2104. PMLR, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL `https://arxiv.org/abs/1707.06347`.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Entong Su, Tyler Westenbroek, Anusha Nagabandi, and Abhishek Gupta. RFS: Reinforcement learning with residual flow steering for dexterous manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3868–3879. IEEE, 2023.

Jun Wang, Yuzhe Qin, Kaiming Kuang, Yigit Korkmaz, Akhilan Gurumoorthy, Hao Su, and Xiaolong Wang. Cyberdemo: Augmenting simulated human demonstration for real-world dexterous manipulation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17952–17963. IEEE, 2024.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026.

Sirui Xu, Yu-Wei Chao, Liuyu Bian, Arsalan Mousavian, Yu-Xiong Wang, Liangyan Gui, and Wei Yang. Dexplore: Scalable neural control for dexterous manipulation from reference scoped exploration. In *Conference on Robot Learning*, pp. 2184–2199. PMLR, 2025.

Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4737–4746. IEEE, 2023.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025.

Xinyu Zhan, Lixin Yang, Yifei Zhao, Kangrui Mao, Hanlin Xu, Zenan Lin, Kailin Li, and Cewu Lu. OAKINK2: A dataset of bimanual hands-object manipulation in complex task completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 445–456, 2024.

Shuqi Zhao, Xinghao Zhu, Yuxin Chen, Chenran Li, Yichen Xie, Xiang Zhang, Mingyu Ding, and Masayoshi Tomizuka. Dexh2r: Task-oriented dexterous manipulation from human to robots. *IEEE/ASME Transactions on Mechatronics*, 2025.

Kefei Zhu, Fengshuo Bai, YuanHao Xiang, Yishuai Cai, Xinglin Chen, Ruochong Li, Xingtao Wang, Hao Dong, Yaodong Yang, Xiaopeng Fan, et al. Dexflywheel: A scalable and self-improving data generation framework for dexterous manipulation. *Advances in Neural Information Processing Systems*, 38:4236–4265, 2025.

Xinghao Zhu, Zixi Liu, Shalin Jain, Chenran Li, Milad Noori, Michael Andres Lin, Huihua Zhao, John Welsh, Mrinal Verghese, Wei Liu, et al. Learning dexterous manipulation using contact wrench guidance from human demonstration. *arXiv preprint arXiv:2607.00033*, 2026.

APPENDIX

A Method and Training Details **14**

A.1 IK-Derived Nominal Control 14

A.2 Control and Simulation Parameters 14

A.3 Actor and Critic Observations 14

A.4 Reward and Termination 15

A.5 Optimization and Prior-Transfer Curriculum 15

A.6 Bootstrap, Task Selection, and Deployment 16

A.7 Scalable Multi-Task Training 17

B Benchmark and Evaluation Protocols **17**

B.1 Trajectory Corpus and Screening 17

B.2 Evaluation Metrics 18

B.3 Baseline Implementations 19

B.4 Per-Trajectory Learning and Adaptation 20

B.5 Compute Accounting 20

C Additional Results and Analysis **22**

C.1 Frozen Execution and Generalization 22

C.2 Training Priors and IK Transfer 23

C.3 Task-Set Expansion 24

C.4 Adaptation to New Trajectories 25

C.5 Recovery and Execution Boundaries 26

C.6 Additional Embodiments 28

C.7 Qualitative Results 31

A METHOD AND TRAINING DETAILS

A.1 IK-DERIVED NOMINAL CONTROL

SPIDER's offline kinematic preprocessing maps demonstrated palm poses and fingertips to nominal robot-joint targets, retaining the object pose as the task reference. MuJoCo relaxes soft palm/object pose welds and fingertip connect constraints while preserving robot kinematics and joint limits. Hand-object collisions and actuation are disabled. Optional pre-contact opening interpolates each finger from an open pose using binary contact onset; IK does not optimize contact forces or dynamic feasibility.

The first frame uses multiple random initializations; subsequent frames start from the preceding solution. Scalar joints are clamped to their limits and smoothed with a three-frame, quaternion-aware moving average. The resulting targets are interpolated to the simulator step and terminally held; they remain fixed throughout training and deployment.

A.2 CONTROL AND SIMULATION PARAMETERS

This subsection specifies the default Sharpa configuration. The diagonal scale for actuator j in Equation 1 is

$$D_{jj} = (u_{\max,j} - u_{\min,j}) m c_j, \qquad c_j = \begin{cases} 0.005 & \text{floating-wrist translation,} \\ 0.10 & \text{floating-wrist rotation,} \\ 1.0 & \text{hand joint,} \end{cases} \tag{7}$$

with $m = 1.0$.

Simulation. The pinned stack is MuJoCo 3.7.0, MuJoCo-Warp 3.7.0.1, and Warp 1.12.1, also used in the evaluated environment. Control and physics both advance by $0.01\,\mathrm{s}$ at 100 Hz, with one physics substep per action. The integrator is `implicitfast`; the Newton solver uses 20 iterations and 50 line-search iterations, with gravity $(0, 0, -9.81)\,\mathrm{m/s^2}$. The configuration field `ctrl_dt=1.5` controls sequence chunking, not the residual policy's action interval.

Actuation. The six floating-wrist coordinates and 22 finger joints use position servos, with $k_p = 1000$ for the wrist and $k_p = 300$ for the fingers. `dampratio=1` determines velocity feedback gains at model compilation. Wrist control ranges are ± 5 m for translation and ± 6.2 rad for rotation; finger controls inherit joint ranges. Joint damping is 100 for the wrist and zero for the fingers, with armature 1. The default experiments impose no additional actuator force/torque limits; control-position clipping does not constitute a torque bound.

Contact and assets. Explicit contact pairs specify friction $(1, 1, 0.1, 0, 0)$ rather than combining geom friction automatically. Thumb/index-object pairs use `condim=4`; other hand-object and object-table pairs use `condim=3`. All use `solref` $= (0.02, 1)$, `solimp` $= (0.9, 0.95, 0.001, 0.5, 2)$, and zero collision margin. Ordinary hand-table collisions are disabled. We augment the Sharpa Wave `wave_01` asset with a floating wrist and tracking sites, retaining its explicit hand inertial parameters. Objects use saved convex collision parts. In the 256 held-out scenes, collision density is $1000\,\mathrm{kg/m^3}$; MuJoCo computes object mass and inertia from this geometry rather than measured object masses. Mesh scale is $(1, 1, 1)$, without additional runtime mass or scale changes.

A.3 ACTOR AND CRITIC OBSERVATIONS

The shared state contains hand joint positions and velocities; current object pose and velocity, target pose, and pose error; current and target wrist poses and their error; robot-body poses in the wrist frame; target human fingertip/ keypoint positions in the target-wrist frame; current and target object-in-wrist poses and their error; object-surface proximity offsets; three-dimensional object scale; and the current nominal control. Nominal-position subtraction centers world-frame positions and nominal wrist translation by the base wrist translation, leaving rotations and velocities unchanged.

Proximity features are wrist-frame offsets from each robot-body origin, simulated fingertip surface, and target fingertip to five nearest points on a 500-point sampled object surface. For simulated fingertips, we first use 16 cached surface samples per finger, then select the five nearest pairs.

The actor concatenates the shared state and future-reference token. Training uniformly samples one valid frame from a three-frame actor history, delaying the entire input, including nominal control and future token. The critic uses current shared state, current token, and normalized phase; privileged inputs add object mass and the diagonal of its inertia tensor. Object velocity is visible to both networks. Control composition uses current values, and clean evaluation fixes the actor delay to zero.

The future token samples 16 target frames from step t through $t + 15$, covering $0.15\,\mathrm{s}$ at $0.01\,\mathrm{s}$ intervals and repeating the terminal frame beyond the horizon. Poses contain positions and unit quaternions in $wxyz$ order, with signs propagated within each window to make adjacent quaternion dot products nonnegative. For object or wrist target pose $(\boldsymbol{p}_t, \boldsymbol{q}_t)$,

$$\boldsymbol{p}_{t+h}^{\mathrm{rel}} = R(\boldsymbol{q}_t)^\top (\boldsymbol{p}_{t+h} - \boldsymbol{p}_t), \qquad \boldsymbol{q}_{t+h}^{\mathrm{rel}} = \boldsymbol{q}_t^{-1} \otimes \boldsymbol{q}_{t+h}, \tag{8}$$

where $h \in \{0, \ldots, 15\}$, R is the rotation matrix, and $\otimes$ is quaternion multiplication. Fingertips are expressed in each frame's target-wrist frame and differenced from frame t. We concatenate relative object and wrist poses and fingertip differences, apply an orthonormal DCT-II along time, retain coefficients 0–7, and flatten the coefficient-feature array.

A.4 Reward and Termination

Let e_p, e_R be the tracked object's instantaneous translation and rotation errors evaluated against the post-step reference target. Let L_i count the temporally resampled reward-reference frames, including frame zero but excluding extra terminal hold and batch padding. Define $s_t = t/\max(L_i - 1, 1)$, with $t = 1$ after the first physical step. This progress term is not clipped: for $L_i > 1$, it reaches 1 at $t = L_i - 1$ and continues growing during terminal hold, when target poses remain at the final frame. Set $w_t = 2$ if $e_p < 0.05\,\mathrm{m}$ and $e_R < 0.3\,\mathrm{rad}$, otherwise $w_t = 1$. The task reward is

$$r_{i,t}^{\mathrm{task}} = w_t \left[\exp(-15 e_p) + \exp(-3 e_R) + 2 s_t^2\right]. \tag{9}$$

Let $d_{f,t}^R$ be the nearest distance from a simulated fingertip site to the sampled object surface at its current pose, and $d_{f,t}^H$ the distance from the reference fingertip to the target object surface. Set $d_{f,t} = |d_{f,t}^R - d_{f,t}^H|$ and

$$g(d) = \frac{\exp[-10\,\mathrm{clip}(d, 0, 0.05)] - \exp(-0.5)}{1 - \exp(-0.5)}. \tag{10}$$

For F fingertips, the complete reward is

$$\begin{aligned} r_{i,t} = r_{i,t}^{\mathrm{task}} &+ \frac{0.5}{F} \sum_{f=1}^{F} g(d_{f,t}) \\ &- \frac{0.5}{F} \sum_{f=1}^{F} \mathbf{1}[d_{f,t}^H < 0.01 \ \wedge \ d_{f,t}^R > 0.015] - 0.2 \max(\|\boldsymbol{a}_t\|_2 - 1, 0), \end{aligned} \tag{11}$$

where $\boldsymbol{a}_t$ is the Gaussian action before clipping in Equation 1.

Training episodes end at the control-sequence horizon, including any terminal-reference hold, on non-finite state, or when tracked-object error exceeds $0.10\,\mathrm{m}$ or $0.50\,\mathrm{rad}$. These checks follow each physics step and use the post-step reference target, as does the reward. PPO treats threshold failures as terminal; only horizon completion is a time-limit truncation with value bootstrapping. Evaluation records the first termination signal but continues finite rollouts for the prescribed duration without threshold-triggered resets. The trajectory-mean oracle and completion diagnostic are defined in Appendix B.2.

A.5 Optimization and Prior-Transfer Curriculum

Networks and noise schedule. The default actor and critic are separate 512–512–256 ELU MLPs with online input normalization. The default curriculum starts at $\alpha = 1$ and decreases by 0.2 to zero.

Equation 4 uses $\boldsymbol{n}_0 = \mathbf{0}$, $\rho = 0.90$, and $c(\alpha) = (\alpha_{\text{start}} - \alpha)/(\alpha_{\text{start}} - \alpha_{\min})$ for $\alpha_{\text{start}} > \alpha_{\min}$. Fixed-noise variants use $c = 0$ for no noise or $c = 1$ for maximum noise. Gaussian perturbations have scale $c(\alpha)\boldsymbol{D}/2$, with no hard amplitude bound; composed controls are clipped to actuator limits.

Advancement and rollback. The standard schedule uses a 10,000-iteration initial warm-up, a 1,500-iteration stage warm-up, and checks every 500 iterations. Each PPO iteration averages the latest up to 50 completed episode returns per trajectory/time-scale pair, weighting pairs equally and empty buffers as zero. Returns are undiscounted and not normalized by episode length. For the latest $W = 1{,}000$ iteration-level means, let μ, σ be the mean and population standard deviation, b the least-squares slope against $0, \ldots, W-1$, and μ_1, μ_2 the half-window means. With $s = \max(|\mu|, 1)$,

$$\text{slope} = \frac{Wb}{s}, \qquad \text{volatility} = \frac{\sigma}{s}, \qquad \text{decline} = \frac{\max(\mu_1 - \mu_2, 0)}{\max(|\mu_1|, 1)}. \tag{12}$$

After warm-up, advancement requires a complete finite window, $\mu \geq 1.0$, relative slope in $[-0.1, 0.01]$, volatility ≤ 0.01, and decline ≤ 0.10. A trial is eligible for rollback after 2,000 iterations if relative slope is below -0.1. Before advancing, the runner saves a policy/optimizer anchor; rollback restores it and retries midway between the last successful and failed α values. If a later stage is mastered, the prior failed boundary is used once to bisect the next step. Training stages have fixed PPO iteration budgets. Expansion stages inherit curriculum difficulty, restarting the PPO iteration count and curriculum statistics.

Table 3: PPO settings for shared-policy training.

Rollout and objective		**Optimization**	
Parameter	**Value**	**Parameter**	**Value**
Steps/environment	24	Learning rate	10^{-3}
Discount γ	0.99	PPO epochs / minibatches	5 / 4
GAE λ	0.95	Clip ratio	0.2
Value coefficient	1.0	Gradient norm	1.0
Entropy coefficient	0.002	Target KL	0.01
Initial policy std.	0.3	LR schedule	adaptive

A.6 Bootstrap, Task Selection, and Deployment

The default procedure uses 50,000 bootstrap updates on the 192 seeds, followed by five 10,000-update expansion rounds. Each round evaluates one clean-IK mean-action rollout per candidate, retains the seeds, and trains on current successes plus up to $B = 64$ selected failures. All selected tasks, including seeds, use their original IK controls for fresh on-policy learning. Rounds inherit the policy, optimizer, and curriculum difficulty while resetting stage-local progress; held-out trajectories never enter these updates. Success sets are recomputed rather than accumulated. The historical union $\mathcal{A}_0 = \emptyset$, $\mathcal{A}_{k+1} = \mathcal{A}_k \cup \mathcal{G}_k$ is used only to report coverage and retention, not to choose the next training set.

Clean-IK deployment. Freeze the policy, disable control noise and observation delay, and reset to the fixed initial state. At each step, the time-indexed IK control enters the observation and composes with the mean residual action through Equation 1, closing the loop through the simulator. Candidate selection uses this procedure. The matched IK-only diagnostic sets the residual to zero; scoring covers only the active reference interval (Appendix B.2).

Failure selection. The default selector ranks failures by normalized tracking error,

$$d_i^{(k)} = \max\left(\frac{\bar{e}_{p,i}^{(k)}}{\epsilon_p}, \frac{\bar{e}_{R,i}^{(k)}}{\epsilon_R}\right), \qquad (\epsilon_p, \epsilon_R) = (0.10\,\text{m}, 0.50\,\text{rad}), \tag{13}$$

and takes the lowest-error $M_k = \min(64, |\mathcal{F}_k|)$ tasks. Success-only uses $B = 0$; random selection samples the same capped count uniformly without replacement. The controlled selector study uses

the ablation configuration, while joint all-candidate training includes every candidate after bootstrap. Selected training-set sizes vary as policies diverge.

A.7 Scalable Multi-Task Training

Temporal augmentation uses scales $\{0.5, 1.0, 2.0\}$, resampling targets and nominal controls, scaling durations, and dividing reference velocities by the time scale. Each GPU merges task worlds into one MuJoCo-Warp model with ragged reference buffers and vectorized rewards; NCCL all-reduce synchronizes gradients for multi-GPU runs. Observation construction uses CUDA graphs, and TF32 is disabled.

Stage settings. Default pretraining, frozen evaluation, and local/shared adaptation use the same time step, actuation, and contact parameters. No mass, friction, inertia, or external-force randomization is enabled. Bootstrap and expansion use the temporal scales above and the base-control noise curriculum with $\rho = 0.9$ and scale increasing from zero to $\boldsymbol{D}/2$. Default frozen evaluation and local/shared adaptation use original speed and clean IK without this noise. Actor training samples the recent three-frame history; evaluation uses the current frame.

B Benchmark and Evaluation Protocols

The controlled benchmark uses a simulated right-handed Sharpa hand, known hand-object trajectories and geometry, and fixed initial states. The offline IK generator, assets, actuation limits, and success oracle remain fixed.

B.1 Trajectory Corpus and Screening

The seed, candidate, and held-out sets in Table 4 are disjoint over processed trajectory IDs. Native right-hand, corresponding left-hand, and mirrored derivatives are kept within the same split, so these related records do not cross training and held-out sets. OAKINK2 includes native right-hand and converted left-hand trajectories; DexYCB contains right-hand trajectories. The candidate pool includes successful and failed source controls. Seen/unseen labels indicate whether the post-conversion object asset occurs in the seed or candidate set. Mirroring reflects both the sequence and its object mesh; a mirrored asset is not marked seen merely because its native counterpart shares the same source object ID. These labels remain fixed across stages and refer to ReDexT's training exposure, not each baseline's pretraining. The held-out trajectories are not used for configuration selection. Policies are updated on these trajectories only in the explicitly labeled local and shared adaptation studies; frozen results precede these updates on target trajectories. Internal comparisons use all 256 held-out trajectories. External methods use a fixed random subset of 32. Its 20 seen-object trajectories comprise 7 OAKINK2 and 13 DexYCB records; all 12 unseen-object trajectories are from OAKINK2. Unavailable cases remain in the denominator.

Table 4: Trajectory counts by dataset and training or OOD split.

Split	OAKINK2	DexYCB	Total
Seed	104	88	192
Expansion candidates	193	280	473
Held-out, object seen	66	88	154
Held-out, object unseen	102	0	102
Total	**465**	**456**	**921**

Left-to-right conversion. We reflect positions across $y = 0$ with $\boldsymbol{M} = \mathrm{diag}(1, -1, 1)$ and map quaternions in (w, x, y, z) order as $(q_w, q_x, q_y, q_z) \mapsto (q_w, -q_x, q_y, -q_z)$. Mirrored wrist, finger, and object trajectories replace the right-hand fields; left-hand fields are cleared. Object meshes receive the same reflection, with reversed triangle winding to preserve outward-facing geometry.

Eligibility and source controls. A trajectory must have finite object positions and move at least $0.05\,\mathrm{m}$ from its initial object position. Each available source control is screened over its first 100 simulated frames and rejected if the object falls by more than $0.01\,\mathrm{m}$; eligibility requires at least one control to pass. This screen applies to all splits, including held-out trajectories, and does not require full-horizon source success. Every seed is paired with a *source-success control* that additionally passes Equation 2. SPIDER supplies these controls in our experiments. A *source-failure task* has no SPIDER control that passes the oracle within the fixed optimization budget; this label does not establish intrinsic infeasibility or IK quality.

Dataset composition and duration. The held-out seen-object subset contains 66 OAKINK2 and 88 DexYCB trajectories; all 102 unseen-object trajectories are OAKINK2. Their mean active durations at original speed are 9.26, 2.00, and $11.80\,\mathrm{s}$, respectively. Seen/unseen comparisons therefore combine object novelty with dataset and duration differences. Restricting evaluation to OAKINK2 fixes the source while leaving durations and motions unmatched.

B.2 Evaluation Metrics

Recorded states and temporal alignment. Our MuJoCo-Warp rollouts save states after each physics step and omit the reset state. Object and hand metrics compare each saved state with the reference targets for the same simulation time, consistent with reward and termination. Targets remain at the final reference frame during terminal hold. Saved times are relative to reset; the simulator takes an initialization step before the reset snapshot. External rollout and scoring protocols are specified in Appendix B.3.

Active length and terminal hold. T_i counts active reference frames for scoring, excluding extra terminal padding. It is distinct from the reward length L_i and the executed control-sequence length: the latter can include a variable terminal hold. In the 256 original-speed bootstrap evaluations checked here, hold ranges from 0 to 148 steps, or 0 to 1.48 s. The 310 extra reference frames added at loading are an indexing buffer, not 3.1 s of mandatory execution. Batch padding equalizes tensor lengths without extending a task's active scoring interval.

Success and completion. Controlled object scoring resamples reference trajectories to the rollout time step and compares temporally aligned states over the active interval. Recorded-frame coverage requires

$$\frac{\min(N_i^{\mathrm{saved}}, T_i)}{T_i} \geq 0.99, \tag{14}$$

where N_i^{saved} counts saved states, including any after a termination signal; non-finite values are checked separately. Extra hold frames do not enter the denominator. Object errors average the available scored frames. Equation 2 tests these means independently of the first termination signal. For multi-object records, scoring takes the worst error over active objects. An object is static if its mean reference translation and rotation displacements are below $10^{-3}\,\mathrm{m}$ and $5 \times 10^{-2}\,\mathrm{rad}$; if all objects are static, all are evaluated. ManipTrans uses the native protocol below.

Let ℓ_i count steps through and including the first termination signal, using the recorded length when no signal occurs. Completion requires

$$H_i = \mathbf{1}\left[\ell_i \geq T_i \ \wedge \ N_i^{\mathrm{saved}} \geq T_i\right]. \tag{15}$$

A first signal at the last active step, $\ell_i = T_i$, therefore still counts as completion. HCR does not require every active step to stay within the instantaneous thresholds. With $S_i = \mathrm{Succ}(\tau_i, \boldsymbol{u})$, HCR is $N^{-1}\sum_i H_i$ and joint success rate is $N^{-1}\sum_i S_i H_i$. Here N is the fixed trajectory count, including failed and unavailable cases. Recorded coverage and completion are distinct: saved arrays can include steps after the first termination signal.

Tracking-error aggregation. Object ADD, translation, and geodesic rotation errors average active frames within each successful trajectory, then weight trajectories equally. ADD uses corresponding object-mesh vertices without symmetry minimization. Adaptation additionally reports quality on common successful trajectories.

Hand tracking. Wrist and fingertip errors compare world-frame robot forward kinematics with demonstrated hand targets using Euclidean position error and geodesic wrist rotation error. Joint error is the mean absolute deviation of the 22 finger coordinates from the IK-derived robot reference. Hand and object metrics use the same temporal alignment. Errors average over each active trajectory, then equally across trajectories; Table 14 uses the same 55 three-way common successes as the object-quality comparison.

Paired IK outcomes. For each trajectory i, let S_i^{IK} and S_i^{res} denote the IK-only and residual-execution outcomes under Equation 2. The paired evaluations use the same nominal IK controls and reset states. Trajectory-level recovery converts an IK failure into a successful residual execution; preservation retains an IK success, and harm denotes losing one. Their rates are

$$r_{\mathrm{imp}} = \frac{\sum_i (1 - S_i^{\mathrm{IK}}) S_i^{\mathrm{res}}}{\sum_i (1 - S_i^{\mathrm{IK}})}, \qquad r_{\mathrm{pres}} = \frac{\sum_i S_i^{\mathrm{IK}} S_i^{\mathrm{res}}}{\sum_i S_i^{\mathrm{IK}}}, \qquad r_{\mathrm{harm}} = \frac{\sum_i S_i^{\mathrm{IK}} (1 - S_i^{\mathrm{res}})}{\sum_i S_i^{\mathrm{IK}}}. \tag{16}$$

Rates with zero denominators are undefined; otherwise $r_{\mathrm{pres}} + r_{\mathrm{harm}} = 1$.

B.3 Baseline Implementations

Internal comparisons use common motion, robot and object assets, simulator, evaluator, and offline IK controls. IK-only executes this prior with zero residual. IK-RL trains a shared residual policy from random initialization on the 192 seed IK priors, without nominal-control noise. This follows the kinematic-prior/shared-residual design of DexH2R (Zhao et al., 2025), while using our observations, reward, and robot; it is an internal baseline rather than a reproduction. IK-RL+Noise adds our control-noise curriculum. The IK-RL and IK-RL+Noise rows in Table 1 use their 100k checkpoints; both routes train only on the 192 seeds. Local-RL learns a separate randomly initialized residual policy per trajectory using the observations, action, reward, network, and PPO settings. ReDexT-B denotes the bootstrap-only policy, whereas ReDexT denotes the policy after task-set expansion. The suffixes (local) and (shared) denote independent per-trajectory and joint shared-policy adaptation, respectively.

Full-pool training from scratch. The full-pool baseline jointly trains on all 665 training trajectories from random initialization, replacing source-control bootstrap and staged expansion with IK-based training from the first update. Temporal scales $\{0.5, 1, 2\}$ yield 1,995 augmented tasks. The model uses a 512–512–256 ELU actor and critic, object-point observations, and three-frame actor history. Its pretraining budget matches the default model: 100,000 PPO updates, 51,840 configured environments, and 24 steps per environment per update, giving 124.416 billion nominal interactions. Both completed pretraining runs use 123.28632 billion simulator interactions. Bootstrap, R5, and Full-pool use training and evaluation seed 0 in this comparison. Residual/noise scales and remaining training settings also match, including coefficients $(0.005, 0.10, 1)$. The 256 target trajectories are excluded from pretraining. Subsequent shared adaptation uses clean IK, 8,192 environments, and 20,000 updates on all 256 targets, with an interaction budget matched to the default model's adaptation. The comparison tests the complete training recipe rather than either training stage alone.

Single-hand adaptations. For DexMachina, we add Sharpa, restrict hand-dependent terms to the active right hand, and omit rigid-object articulation error. Pose tracking, hybrid actions, motion/contact guidance, and decaying virtual-object control are retained. Each trajectory receives 5,000 training updates and ten evaluation rollouts from the final policy. SR averages attempts within each trajectory, then weights the 32 trajectories equally; the three unavailable cases remain in the denominator. Tracking errors average successful attempts within each trajectory, then equally across trajectories with at least one success. Time includes per-trajectory training and evaluation.

CHORD. CHORD uses Isaac Lab/PhysX and the right Sharpa hand. We remove the inactive hand from actions, observations, termination, contact, and wrench support, retaining contact-wrench guidance and the virtual-object curriculum. Each trajectory uses 256 environments, 24 steps per update, and 10,000 PPO updates with training seed zero, totaling 61.44 million interactions. The final checkpoint is evaluated ten times with seeds 0–9, with virtual-object assistance disabled and fixed reference timing and initial states. All 320 rollouts record post-physics, pre-reset states over the active horizon. Quality averages successful attempts within each trajectory, then equally across

the seven successful trajectories; ADD uses the common mesh scorer. Time includes training and the ten-rollout evaluation process.

SPIDER. SPIDER uses 1,024 optimization samples, planning horizons of $3.0/1.6$ s, and iteration caps of $16/32$ for OAKINK2/DexYCB. We physically replay the exact 32 optimized control sequences in MuJoCo-Warp with zero residual and actuator clipping, using the original scenes and native scorer. This yields $12/32$ SR successes and $2/32$ horizon completions; saved optimization states are not used as evaluation trajectories. These controls differ from the preprocessed 256-control batch used for the additional-hand comparison. Time retains the original RTX 4090 profile, which uses one task per GPU with rendering disabled and includes initialization, compilation, optimization, execution, and saving. End-to-end time was not remeasured for the physical replay.

Native ManipTrans. ManipTrans uses 31 task-specific checkpoints in Isaac Gym, with native Shadow hands in 22 right-hand and nine left-hand configurations. It retains native preprocessing and the $N - 3$-step horizon, where N is the capped reference length. The first observation uses the reset state without an extra warmup step. Each trajectory has ten attempts with native base-policy sampling and mean residual actions; the mean-error oracle covers the complete native horizon. Embodiment, trajectories, horizon, simulator, and meshes differ from the controlled comparison.

SR averages attempts within each trajectory, then over all 32 IDs. The missing checkpoint's ten slots remain in the denominator, as do 13 interrupted attempts. Quality averages successful attempts per trajectory and then across trajectories, using unique archived mesh vertices for ADD. Time averages training plus ten evaluation/export attempts over 30 complete records, including initialization and I/O; it excludes retargeting, shared pretraining, and training retries.

B.4 Per-Trajectory Learning and Adaptation

For each of the 256 held-out trajectories, Local-RL starts from random initialization, ReDexT-B (local) from the default model's 50k bootstrap checkpoint, and ReDexT (local) from its fifth-round expanded checkpoint. All routes use clean IK priors, the 512–512–256 MLP, unit global residual scale, and coefficients $(.005, .10, 1)$. Observations, reward, and PPO follow Appendices A.3–A.5. Each uses 256 environments, 24 steps per update, and 5,000 updates: $256 \times 24 \times 5000 = 30{,}720{,}000$ interactions per trajectory, excluding upstream bootstrap and expansion. Adaptation restores actor, critic, observation-normalization, and action-distribution state, but resets the optimizer, curriculum, and update counter.

Training disables control noise and temporal augmentation; updates remain local to the target trajectory. Evaluation uses identical trajectory files, fixed object-surface samples and initial states, original speed, mean actions, and actor history length one. We evaluate after 1, 501, 1,001, ..., 4,501, and 5,000 updates and use the final checkpoint for endpoints. Frozen ReDexT-B and ReDexT checkpoints provide true zero-update evaluations; the first saved training checkpoint follows one update, and random initialization has no measured zero-update point.

All 768 per-trajectory training runs complete, with 8,448 evaluation records across the 11 post-update checkpoints. Rates retain the fixed 256-trajectory denominator, and each endpoint uses update 5,000 rather than a per-trajectory best checkpoint. Where runs were retried, the first complete valid attempt is used; attempts are not selected by reward or success.

Mean post-IK training-process plus final-rollout time is 105.35, 97.72, and 98.41 min per trajectory for Local-RL, ReDexT-B (local), and ReDexT (local), respectively, over all 256 trajectories. These times exclude queueing, intermediate evaluations, external metric scoring, retries, and upstream shared training. All methods train on H200 GPUs and use RTX 4090 GPUs for evaluation; these totals combine the two stages. These local-training times do not replace the frozen execution profiles in Appendix B.5.

B.5 Compute Accounting

Per-trajectory wall time. Tables report mean post-IK time per trajectory: initialization, target-specific learning or optimization, and physical rollout. Frozen policies and IK-only require no target-specific learning. This measures the cost of processing a new trajectory after preparing its inputs and,

for shared policies, training the controller. Bootstrap source-control generation, upstream shared training, offline IK, and separate metric scoring are excluded. Baseline preparation follows each method's processing pipeline. Policy training uses H200 GPUs and evaluation uses RTX 4090 GPUs throughout. Timings retain each method's stated concurrency and processing stages.

The default REDEXT and REDEXT-B take 46.11 and 45.92 s per trajectory, respectively, on the fixed 32-trajectory subset. Each uses one worker on the same RTX 4090 and a fresh process per trajectory. This includes environment construction, checkpoint loading, compilation, inference, simulation, and saving. Timing is profiled separately from the independent SR/quality evaluations; only wall time is taken from the profiling batches. IK-RL and IK-RL+Noise retain fixed-32 times of 39.2 and 33.4 s. These fixed-32 profiles also supply the learned frozen-policy times in the 256-trajectory panel; that panel's success rates use all 256 trajectories, but its reported times are not a 256-trajectory average.

For the shared-adapted rows, Table 1 adds joint training time divided by 256 to the reported 0.8 min execution profile, marked ‡. The full-pool report states that its inference time matches the default model under the same timing protocol, also supplying the full-pool frozen entry. Separate profiling records or timing sample counts for the full-pool policy and adapted checkpoints are not reported. Joint-training times in Table 5 cover one policy trained on all 256 targets using a single H200, 8,192 environments, and 20,000 updates. Thus, $T^{\text{joint-adapt}}/256 + T^{\text{execution}}$ is approximately 2.71 min/trajectory for Full-pool and 2.69 for the default model, both reported as 2.7. These derived averages combine H200 training with the reused RTX 4090 execution profile; they are not single-trajectory request latencies or end-to-end measurements from one run.

For an IK-based method, end-to-end per-trajectory cost is

$$T_i^{\text{retarget}} = T_i^{\text{IK}} + T_i^{\text{target-learning}} + T_i^{\text{rollout}}, \tag{17}$$

with $T_i^{\text{target-learning}} = 0$ for frozen execution. An independent CPU profile of current IK preprocessing averages 4.26 s over the fixed 32 trajectories on an AMD EPYC 7542, including model construction, solving, auxiliary replay, and saving. These newly solved priors are separate from the reported policy evaluations.

Cumulative cost. Total cost for new trajectories includes source-control production and screening, IK for training trajectories, bootstrap, candidate-selection rollouts, expansion, and per-trajectory IK, optional adaptation, and rollout. Appendix B.3 specifies the matched nominal and actual interaction budgets; these counts are distinct from elapsed time. For K new trajectories, the average end-to-end time would be

$$\overline{T}(K) = \frac{T^{\text{shared}}}{K} + \frac{1}{K}\sum_{i=1}^{K} T_i^{\text{retarget}}, \tag{18}$$

where T^{shared} includes source-control generation and screening, training-input preparation, bootstrap, selection, and expansion. Table 5 gives H200 training times and inter-stage overhead. The observed 174.13 h window already includes 1.45 h of inter-stage evaluation and switching. Its displayed components sum to 174.12 h, within separate two-decimal rounding of the observed total.

Table 5: Measured wall time for shared adaptation and the stages of default upstream training.

Training phase	**Wall time**
Joint adaptation on 256 targets, 20,000 updates	
Full-pool (Task15)	8 h 7 min 55 s
REDEXT (default, route 11)	8 h 4 min 11 s
Range variant (route 10)	8 h 7 min 17 s
Default upstream training	
Bootstrap: 50,000 updates	85.16 h
Five expansion rounds: 50,000 updates in total	87.51 h
Five inter-stage evaluations and switches	1.45 h
Observed upstream window	174.13 h

Table 6: Performance by object familiarity on the 32-trajectory OOD subset. SR is in percent over 20 seen-object and 12 unseen-object trajectories; bold marks the best rates within each group. †ManipTrans uses its native protocol.

Method	Seen SR ↑	Unseen SR ↑
No test-time adaptation		
IK-only	20.00	0.00
REDEXT-B	60.00	0.00
REDEXT	**70.00**	**16.67**
Per-trajectory optimization or learning		
SPIDER	**50.00**	**16.67**
CHORD	30.00	5.00
DexMachina	25.00	0.00
ManipTrans†	20.00	4.17

Table 7: Frozen and per-trajectory learning results on the 256-trajectory OOD set, with 154 seen-object and 102 unseen-object trajectories. SR is a percentage; bold marks the best frozen results. ADD is in meters over each method's successes.

Method	Seen SR ↑	Unseen SR ↑	ADD ↓
IK-only	22.73	3.92	0.0605
IK-RL	14.29	7.84	0.0436
IK-RL+Noise	47.40	7.84	0.0289
Full-pool	69.48	18.63	0.0198
REDEXT-B	50.00	14.71	0.0218
REDEXT	**75.32**	**24.51**	**0.0197**
Local-RL	29.22	12.75	0.0594

Processing estimates. Serial extrapolation of the measured 4.26 s IK mean gives approximately 1.1 h for all 921 training and target trajectories. Other input preparation and final metric-processing costs remain unmeasured, and original SPIDER optimization is excluded from the upstream window. These records therefore do not establish measured end-to-end cost or a break-even count.

C Additional Results and Analysis

C.1 Frozen Execution and Generalization

Tables 6 and 7 supplement Table 1 with object strata and tracking errors. Quality is averaged over each method's successful trajectories, and seen/unseen labels refer to REDEXT's training exposure. Both REDEXT-B and REDEXT use the default configuration and are frozen; Local-RL uses 5,000 updates per trajectory. The supplementary numerical results retain the complete tables and configuration explorations omitted here. In comparison tables, bold values mark the best reported values within each group; ties at the displayed precision are all bold. Evaluation and timing protocols are given in Appendix B.

Full-pool training from scratch. Paired evaluation of REDEXT and matched-budget Full-pool training gives 110 common SR successes, 31 unique to REDEXT, and 16 unique to Full-pool. ADD on the common successes is 0.01894 m for REDEXT and 0.01892 m for Full-pool. The main gain is therefore broader success coverage, with 15 more successes and 12 more completed horizons in this single-run comparison. Appendix B.3 specifies the matched protocol, and Appendix C.4 reports shared adaptation.

Table 8: Clean-IK transfer after 50k bootstrap updates on 256 OOD trajectories. Noise denotes the training control-noise curriculum. SR/HCR are percentages. ADD and position errors are in centimeters; rotation is in radians. Errors average successful trajectories.

Train prior	Noise	SR ↑	HCR ↑	ADD ↓	Pos. ↓	Rot. ↓
SPIDER	Yes	**28.13**	**10.94**	**3.16**	**3.00**	**0.204**
IK	Yes	**28.13**	9.77	3.44	3.25	0.223
SPIDER	No	12.11	1.17	5.47	5.33	0.239
IK	No	13.28	1.56	4.74	4.61	0.260

Table 9: Paired changes in SR after 50k bootstrap updates. Arrows give the comparison direction; gained/lost are counts of trajectories that become successful/unsuccessful. The 95% intervals use paired trajectory resampling.

Contrast	ΔSR (pp)	95% interval	Gained/lost
Training trajectories ($n = 192$)			
SPIDER → IK, noise	-3.65	$[-7.29, 0.00]$	3/10
SPIDER → IK, no noise	-30.73	$[-37.50, -23.96]$	2/61
Held-out trajectories ($n = 256$)			
Noise off → on, SPIDER training	$+16.02$	$[10.55, 21.88]$	51/10
Noise off → on, IK training	$+14.84$	$[9.38, 20.31]$	47/9
IK → SPIDER training, noise	$+0.00$	$[-5.47, 5.47]$	25/25

Dataset and object strata. The default frozen policy succeeds on $71/88$ DexYCB and $70/168$ OAKINK2 trajectories, including $116/154$ seen-object and $25/102$ unseen-object trajectories. Object novelty co-varies with dataset and duration (Appendix B.1).

C.2 Training Priors and IK Transfer

Bootstrap endpoint and evaluation. All four policies train on the same 192 SPIDER-selected seeds and are evaluated at 50k updates, before expansion. The ablation configuration uses coefficients $(0.004, 0.004, 1.2)$ rather than the default $(0.005, 0.10, 1)$; remaining bootstrap settings match, including 51,840 configured environments and 24 rollout steps. The two noisy routes start randomly and differ only in nominal-control source, sharing architecture, observations, reward, PPO, temporal scales $\{0.5, 1, 2\}$, and noise curriculum.

Transfer on seed trajectories. For each SPIDER-trained policy, we switch the nominal control in both the observation and action composition without updating weights. Table 2(A) shows the resulting seed SR and HCR, with drops computed from unrounded rates; Table 8 evaluates the same checkpoints on OOD trajectories under clean IK. HCR equals joint success in all four evaluations.

Paired outcomes. Each noisy training route succeeds on 72 held-out trajectories: 47 are shared and 25 are unique to each, while both fail on 159. Relative to their no-noise counterparts, held-out SR is higher by 16.02 percentage points for SPIDER training and 14.84 for IK training. Table 9 reports paired bootstrap intervals from 100,000 trajectory resamples. These quantify trajectory-level variation conditional on the evaluated checkpoints, not variation across training runs. Training-trajectory contrasts use SPIDER-trained policies, while OOD contrasts use clean IK evaluation. The comparison evaluates control noise together with its adaptive schedule.

Bootstrap conditioning. We ablate bootstrap inputs and training choices using the ablation configuration, with coefficients $(0.004, 0.004, 1.2)$ and 192 training seeds. Table 10 evaluates new trajectories with SPIDER controls. These diagnostics complement the clean-IK transfer study in Table 8. Removing object-surface features or future-reference context reduces overall SR, although retaining either input does not improve every object stratum. Removing nominal-control observations instead raises SR, so these diagnostics do not establish that every input is necessary.

Table 10: Bootstrap ablations on 256 OOD trajectories. SR is reported in percent overall and for 154 seen-object and 102 unseen-object trajectories. Evaluation uses SPIDER controls.

Variant	All SR ↑	Seen SR ↑	Unseen SR ↑
Bootstrap	27.34	42.21	4.90
w/o object-surface features	23.44	35.06	5.88
w/o future-reference context	25.00	37.66	5.88
w/o translation centering	25.78	37.01	**8.82**
w/o nominal-control observation	**32.81**	**50.00**	6.86
w/o control-noise curriculum	14.45	20.78	4.90

Table 11: Task-set expansion ablations on 256 OOD trajectories. Frozen fifth-round SR/HCR are percentages. Rankings compare IK rows 01–07; row 08 uses SPIDER controls.

ID	Variant	SR ↑	HCR ↑
01	Success-only	39.84	**23.83**
02	Lowest-error	40.23	21.88
03	Random-failure	**41.02**	**23.83**
04	All candidates	39.45	20.70
05	No noise	23.44	6.25
06	Bootstrap noise only	25.78	11.72
07	No bootstrap	39.06	23.05
08	SPIDER-base expansion	41.41	20.31

The object-surface ablation removes proximity and scale features; the future-reference ablation uses one target frame without DCT. Translation centering changes nominal wrist translations, leaving rotations, velocities, tracking errors, and wrist-relative features unchanged. Removing the control-noise curriculum disables noise throughout training. Additional architecture, scale, observation-delay, and temporal-augmentation variants are included in the supplementary numerical results.

Control magnitude. For the SPIDER+Noise policy in Table 8, the mean absolute finger correction across its 174 successful seed rollouts over SPIDER before final clipping is 50.28% of actuator range, and 37.05% of composed wrist and finger controls are clipped. These statistics average executed frames and actuators within each rollout, then successful tasks; residual parameterization does not imply small corrections.

C.3 Task-Set Expansion

Training settings. Table 11 compares eight routes built around the ablation configuration at their fifth-round checkpoints. Routes 01–04, 06, and 08 share the noisy 50,000-update bootstrap; 05 uses a 50,000-update no-noise bootstrap and 07 starts randomly at maximum noise. Route 04 retains bootstrap and uses all 192 seeds and 473 candidates in each subsequent round. Route 07 skips bootstrap but still selects candidates and always retains the 192 seeds. All routes use the 512–512–256 MLP, unit global residual scale, coefficients $(0.004, 0.004, 1.2)$ matching route 09, Seed continuation, temporal scales $\{0.5, 1, 2\}$, and five rounds of 10,000 updates with a configured 51,840-environment budget and 24 steps per environment. Routes 05/06 disable expansion-stage noise; the others use maximum difficulty. Route 08, SPIDER-base expansion, uses SPIDER controls for both training and evaluation and serves as a separate prior diagnostic. The configured expansion budget is 62.208 billion interactions before integer task allocation. Selected task-set sizes and per-task exposure vary by route.

Expansion-time coverage and retention. Current coverage is $|\mathcal{G}_k|/|\mathcal{C}|$; ever-solved coverage uses $\mathcal{A}_{k+1} = \mathcal{A}_k \cup \mathcal{G}_k$. The default model's current candidate successes increase from $246/473$ at bootstrap to $410/473$ at R5, whose training set includes all 665 training trajectories. Of 425 candidates

Table 12: Control range and network depth on 256 OOD trajectories with clean IK. Scale lists wrist translation, wrist rotation, and finger coefficients; depth counts hidden layers. F/S denotes frozen/shared-adapted rates in percent, with 20,000 adaptation updates.

Variant	Scale	Depth	SR-F ↑	SR-S ↑	HCR-F ↑	HCR-S ↑
09 Seed continuation	$(.004, .004, 1.2)$	3	30.08	62.50	17.19	53.13
10 Range	$(.005, .05, 1)$	3	51.95	75.39	37.89	69.92
11 REDEXT (default)	$(.005, .10, 1)$	3	**55.08**	**80.08**	**39.06**	**75.39**
12 Deeper	$(.004, .004, 1.2)$	4	41.80	73.44	25.39	69.14
13 Deeper + range	$(.005, .05, 1)$	4	50.78	79.69	37.11	73.83

solved at least once, 15 fail at R5; another 48 never succeed at a recorded stage, together accounting for 63 failures. Training uses current successes and selected failures, not the historical union. Independent clean-IK evaluations on the same 256 held-out trajectories give 92 and 141 successes at bootstrap and R5. With matched nominal controls and reset states, R5 retains 84 successes, gains 57, and loses eight. Held-out trajectories were not used for configuration selection.

Noise, initialization, and prior. Routes 05–08 use lowest-error selection and are compared with 02. With expansion-stage noise, route 02 reaches 40.23% frozen SR versus 25.78% for bootstrap-only noise; removing noise in both stages yields 23.44%. The no-bootstrap route reaches 39.06%, changing both initialization and subsequent selected task sets. It does not isolate bootstrap on fixed seeds. Route 08 reaches 41.41% under SPIDER evaluation. The four selector endpoints span 39.45–41.02% frozen SR, providing no clear advantage for error-based ranking in these runs. Shared-adaptation endpoints and paired changes for all eight routes are retained in the supplementary numerical results.

Control range and capacity. Table 12 compares Seed continuation and coupled residual/noise scales with network depth. Route 09, Seed continuation, adds 50,000 IK-training updates on the 192 seeds after bootstrap, rather than stopping at REDEXT-B. It restores the checkpoint's optimizer and curriculum state, whereas expansion inherits optimizer state and difficulty but resets stage-local curriculum statistics. Routes 10–13 use 50,000 bootstrap updates and five 10,000-update rounds, with 51,840 configured environments, 24 rollout steps, and lowest-error selection with $B = 64$. The deeper architecture adds one 512-unit layer to both networks.

C.4 ADAPTATION TO NEW TRAJECTORIES

Per-trajectory learning. Table 13 compares local adaptation from random, default bootstrap, and expanded initializations. Each target receives an independently adapted policy with 5,000 updates. The expanded model reaches 180 successes and 169 completed horizons, compared with 170 and 153 from bootstrap and 58 and 11 from random initialization. Every horizon-complete trajectory also passes the mean-error criterion in these runs. The expanded model's advantage is concentrated in seen-object and DexYCB trajectories; bootstrap retains slightly higher SR and HCR on unseen-object and OAKINK2 trajectories. Seen/unseen labels describe upstream exposure, and all trajectories receive target updates. Object novelty co-varies with dataset and duration, as described in Appendix B.1.

Paired outcomes. Final success sets are paired over the same 256 trajectories. Relative to bootstrap, the expanded initialization gains 25 successes and loses 15, a net gain of ten or 3.91 percentage points. This is a higher endpoint in the reported runs, rather than a uniform improvement across trajectories. On the 155 trajectories successful under both pretrained initializations, ADD is 0.00846 m from the expanded model and 0.01268 m from bootstrap.

Tracking quality on common trajectories. Table 14 evaluates all three final policies on the same 55 successful trajectories. The expanded initialization has lower mean errors than both alternatives for all seven object and hand metrics. This controls for which trajectories enter the average, but remains conditional on all three methods succeeding. The own-success errors in Table 13 describe

Table 13: Local adaptation after 5,000 updates per trajectory. SR/HCR are percentages, and errors average each policy's successes. Brackets give stratum sizes.

Metric or stratum	Local-RL	REDEXT-B	REDEXT
All SR/HCR [256]	22.66 / 4.30	66.41 / 59.77	**70.31 / 66.02**
Seen SR/HCR [154]	29.22 / 3.25	72.73 / 64.94	**80.52 / 77.27**
Unseen SR/HCR [102]	12.75 / 5.88	**56.86 / 51.96**	54.90 / 49.02
DexYCB SR/HCR [88]	18.18 / 1.14	69.32 / 63.64	**86.36 / 82.95**
OAKINK2 SR/HCR [168]	25.00 / 5.95	**64.88 / 57.74**	61.90 / 57.14
ADD (m)	0.05937	0.01309	**0.00891**
Position error (m)	0.05877	0.01266	**0.00854**
Rotation error (rad)	0.14950	0.06275	**0.05792**

Table 14: Object and hand tracking errors on the same 55 trajectories successful under all three initializations after 5,000 local updates. Each trajectory has equal weight; lower is better.

Metric	Local-RL	REDEXT-B	REDEXT
Object ADD (m)	0.05903	0.01664	**0.00614**
Object pos. (m)	0.05848	0.01632	**0.00596**
Object rot. (rad)	0.14346	0.05558	**0.03218**
Wrist pos. (m)	0.11669	0.08413	**0.08108**
Wrist rot. (rad)	1.21694	0.56496	**0.54905**
Tip (m)	0.20047	0.06559	**0.05740**
Joint MAE (rad)	0.20790	0.19157	**0.18327**

each policy's outputs, rather than quality on an identical trajectory population. Figure 4(c) shows the normalized common-success errors.

Shared-policy adaptation. The default model adapts jointly to all 256 targets for 20,000 updates, using 8,192 environments, clean IK, original reference speed, and default coefficients $(0.005, 0.10, 1)$. It retains 139 frozen successes, gains 66, and loses two. On the 139 common successes, ADD decreases from 0.0194 to 0.0101 m, position error from 0.0183 to 0.0096 m, and rotation error from 0.1405 to 0.0629 rad. Adaptation therefore improves tracking quality on the same trajectories as well as increasing success coverage.

Shared adaptation from full-pool training. With matched pretraining and adaptation budgets, REDEXT achieves 15 more successes and ten more completed horizons than Full-pool (Table 1). Seen/unseen SR is $90.26/64.71\%$ for REDEXT and $87.01/54.90\%$ for Full-pool. Both succeed on 83 of 88 DexYCB trajectories, while OAKINK2 success counts are 122 and 107, respectively. Equal marginal counts need not imply identical success sets. ADD over each policy's own successes is 0.01017 m and 0.01091 m, respectively. Object labels retain their pre-adaptation meaning; all 256 trajectories receive target updates. These results compare the complete training strategies under the protocol in Appendix B.3.

C.5 RECOVERY AND EXECUTION BOUNDARIES

Across the reported IK-only, IK-RL variants, default frozen policies, local-learning endpoints, and default shared-adapted batches, every horizon-complete rollout also passes the mean-error oracle. HCR and joint SR therefore coincide empirically, not by definition (Appendix B.2). The default frozen policy reaches 55.08% SR and 39.06% joint SR. All 512 frozen and shared-adapted records cover the active trajectory; 41 frozen and 12 adapted successes have an earlier termination signal.

Under matched nominal IK controls and reset states, the default frozen model and IK-only have 36 shared successes, 105 trajectories successful only under REDEXT, three successful only under IK-only, and 112 shared failures. Residual feedback therefore recovers $105/217$ IK-only failures,

Table 15: Trajectory outcomes before and after shared adaptation of the default policy. Entries count trajectories. Mean-only passes the trajectory-mean criterion but not horizon completion; joint success satisfies both.

	After shared adaptation		
Frozen outcome	**Failure**	**Mean-only**	**Joint success**
Failure	49	8	58
Mean-only	1	2	38
Joint success	1	2	97

a recovery rate of 48.39%, and preserves $36/39$ IK-only successes, a preservation rate of 92.31%. The remaining $3/39$ successes are lost, a harm rate of 7.69%. Recovery here refers to success under the mean-error oracle. Of the 105 recovered trajectories, 72 also satisfy horizon completion and 33 do not.

Table 15 separates gains in mean-error success from gains in horizon completion. Shared adaptation benefits both previously failing and mean-only trajectories, although some successful trajectories regress.

Temporal tracking deviations. First crossings among frozen and adapted mean-only successes are predominantly rotational. The 41 frozen cases exceed at least one threshold for 14.29% of active frames on average. Nineteen of these cases and seven of the 12 adapted mean-only successes finish within both thresholds, demonstrating that tracking can return within bounds after a deviation. Thirteen frozen failures cross a threshold at the first saved step and remain failures after adaptation. These post-step diagnostics do not identify contact-level causes.

Hand-reference fidelity. Paired comparisons separate hand-reference fidelity from object tracking. On the 84 trajectories successful under both default REDEXT-B and REDEXT, ADD decreases from 0.0216 to 0.0163 m and fingertip error from 0.0652 to 0.0638 m, whereas wrist-rotation error increases from 0.5345 to 0.5807 rad.

C.6 Additional Embodiments

Experimental question and protocol. These experiments test whether the training recipe applies beyond the default hand model. Allegro, Inspire, XHand, and Shadow each train an independent shared residual policy from scratch, with 50k source-control bootstrap updates followed by five 10k IK-based rounds. The actor and critic retain the 512–512–256 ELU architecture, observation definitions, reward, PPO, and noise-curriculum mechanism. Observation and action dimensions follow each hand's geometry and actuators. Every policy is evaluated on the same 256 OOD capture IDs, comprising 88 DexYCB and 168 OAKINK2 trajectories and disjoint from its own seed and candidate sets. Shared adaptation then updates one policy per hand jointly on these 256 targets for 20k updates, using IK, 8,192 worlds, and no curriculum. Thus frozen evaluation measures execution of held-out trajectories; shared adaptation measures learning on the target set. No policy weights are transferred between hands.

Table 16: Frozen and shared-adapted performance of separate policies for five hands on 256 OOD trajectories. Shared adaptation uses 20k updates. SR and HCR are percentages; ADD is in meters and averages each policy's successes.

	Frozen			Shared-adapted		
Hand	SR ↑	HCR ↑	ADD ↓	SR ↑	HCR ↑	ADD ↓
Sharpa	55.08	39.06	0.01974	80.08	75.39	0.01017
Allegro	51.95	41.80	0.01413	78.52	75.00	0.00958
Inspire	44.92	27.73	0.02006	67.19	65.63	0.00871
XHand	50.00	35.55	0.01777	73.83	68.36	0.01060
Shadow	48.83	33.20	0.02192	76.17	69.53	0.01036

Coverage and adaptation. All four additional policies execute held-out trajectories before target updates (Table 16). Shared adaptation adds 68, 57, 61, and 70 net SR successes for Allegro, Inspire, XHand, and Shadow, respectively, while losing 1, 3, 2, and 2 frozen successes. Their common-success subsets contain 132, 112, 126, and 123 trajectories. On these subsets, ADD decreases from 0.01412 to 0.00877 m, 0.01993 to 0.00905 m, 0.01741 to 0.01000 m, and 0.02162 to 0.00973 m. Thus adaptation improves both coverage and tracking quality, rather than merely changing which trajectories succeed. Frozen HCR remains below SR for every hand: mean object accuracy does not guarantee full-horizon completion. Under the same native scoring, frozen SR exceeds physical replay of SPIDER controls with zero residual by 8.20–13.67 percentage points across these four hands.

Differences from the default setup. Targets are preprocessed and training pools screened separately for each hand. Table 17 records the exposure and control settings. Wrist coefficients scale both residual actions and training base-control noise; the finger coefficient and global action multiplier remain 1.0. Joint ranges and actuator geometry differ, so equal coefficients do not imply equal physical control amplitudes. Training uses 48,000 worlds, compared with Sharpa's 51,840. These hand-specific configurations test the applicability of the recipe, rather than isolating morphology or ranking hand types. We report one trained policy per hand; repeated-training variability is not measured.

Table 17: Training data and control coefficients for each hand. Act. counts actuators including the floating wrist; Union counts trajectories used across the six training stages. Trans./Rot. are wrist translation/rotation coefficients.

Hand	Act.	Seed	Candidate	Union	Trans.	Rot.
Sharpa	28	192	473	665	0.005	0.1000
Allegro	22	192	761	735	0.00833	0.0414
Inspire	18	174	743	673	0.025	0.0414
XHand	18	185	776	743	0.005	0.0419
Shadow	28	162	785	734	0.005	0.0419

Figures 6 to 8 show six frozen-policy examples spanning different object geometries and grasp motions, with five snapshots per hand in temporal order. Performance across the full evaluation set is reported in Table 16. The accompanying visual supplement contains the full gallery.

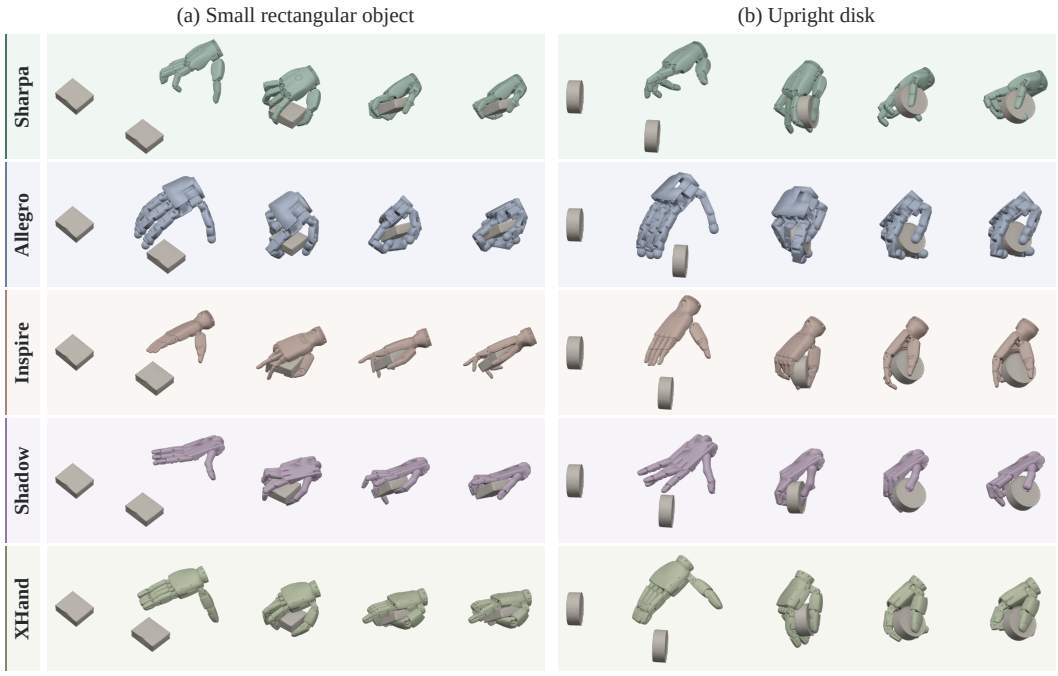

Figure 6: Frozen-policy rollouts across hands with (a) a small rectangular object and (b) an upright disk.

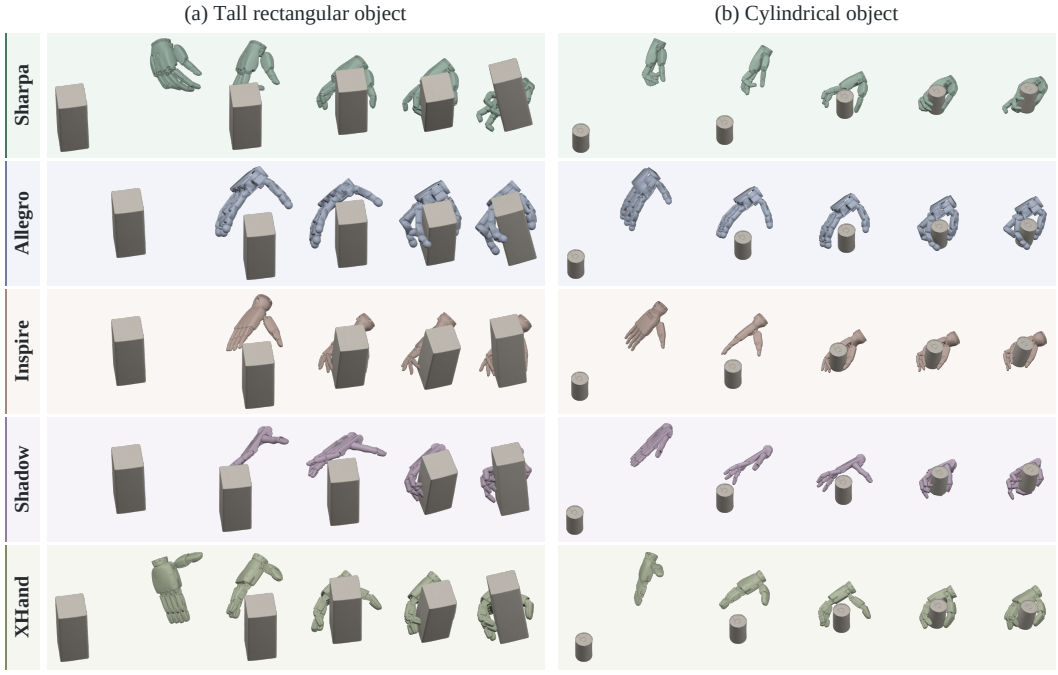

Figure 7: Frozen-policy rollouts across hands with (a) tall rectangular and (b) cylindrical objects.

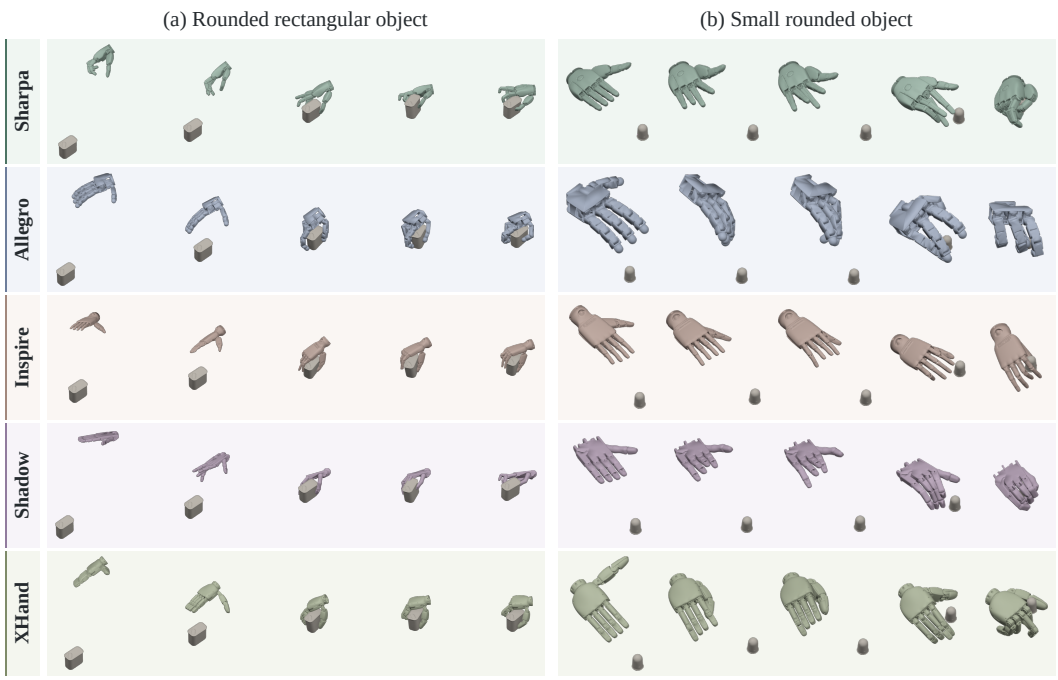

Figure 8: Frozen-policy rollouts across hands with (a) a rounded rectangular object and (b) a small rounded object.

C.7 QUALITATIVE RESULTS

Figures 9 and 10 compare four sequences with different object geometries and scales. REDEXT uses the frozen default R5 policy; DexMachina and CHORD are single-hand adaptations. Each sequence has five snapshots in temporal order. The cropped views align initial object bottoms across methods while preserving later motion within each row; image positions do not measure physical contact.



Figure 9: Qualitative comparison on (a) a bottle-shaped object and (b) a rectangular object.

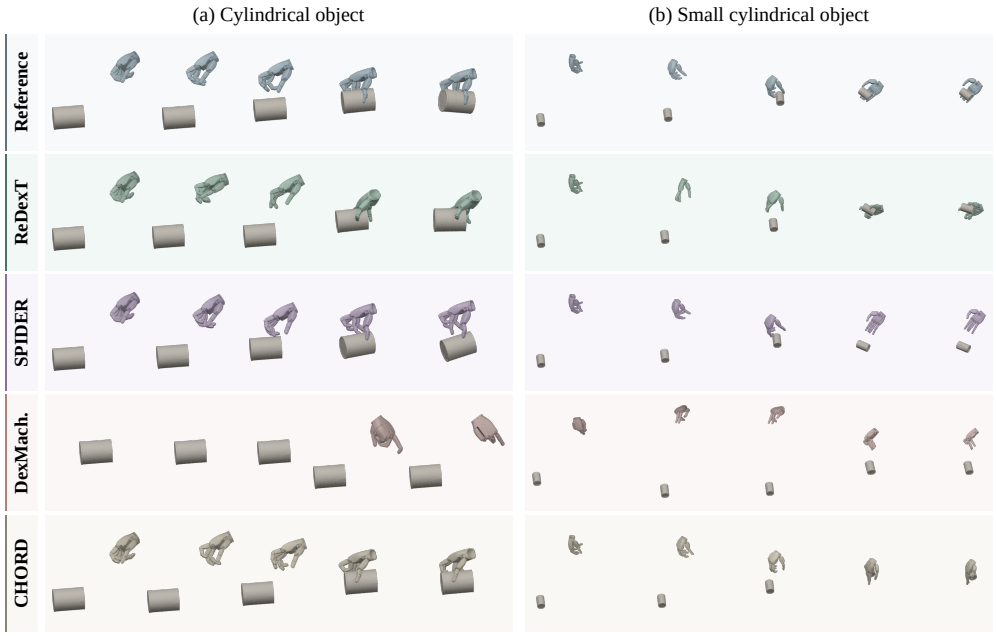

Figure 10: Qualitative comparison on (a) a cylindrical object and (b) a small cylindrical object.