DEXLINK: SYNTHESIZING DEXTEROUS BEHAVIORS FROM CHAINED HAND-OBJECT POSES

Meenal Parakh[†], **Deep Patel**[§], **Beining Han**[†], **Jia Deng**[†]
[†]Princeton University
[§]NEC Laboratories America

ABSTRACT

Synthesizing diverse dexterous manipulation policies is challenging: existing simulation-based approaches rely on human demonstrations (Jiang et al., 2025; Mu et al., 2026) which are difficult to scale, or use generic reward functions (Kedia et al., 2026), which provide limited control over the hand-object interactions. We propose DexLink, which addresses these limitations by synthesizing diverse dexterous behaviors from sparse, reusable hand-object interaction templates. Given key task waypoints, DexLink retargets static interaction templates to target objects using an optimization that preserves spatial hand-object relationships. It then links the retargeted poses into an approximate reference trajectory and uses residual reinforcement learning to refine this reference into a closed-loop policy in simulation. Experiments spanning single-hand, bimanual, articulated, and dynamic tasks demonstrate that DexLink provides fine-grained control over task timing, grasp types, and high-level execution. The modular interaction templates transfer reliably across object scales, instances, and categories; without object-specific human demonstrations, the framework also matches the performance of dense reference-tracking methods. Finally, zero-shot deployment of policies trained purely in simulation shows promise for the simulation-based framework. https://dex-link.github.io/

1 INTRODUCTION

Learning from data is a key paradigm in robotic manipulation and relies on the ability to collect large, diverse behaviors (Khazatsky et al., 2025). For dexterous manipulation, this requires coverage over diverse tasks, objects, and interaction behaviors. Obtaining such coverage is challenging: dexterous hands involve complex, contact-rich dynamics that make teleoperation cumbersome and expensive (Handa et al., 2019; Tao et al., 2025).

Simulation provides a scalable platform for synthesizing dexterous behaviors, with potential for real-world transfer (Jiang et al., 2025; Chen et al., 2023). Yet, specifying which tasks and behaviors to generate, and how, remains an open challenge. Existing work often relies on human demonstrations (Jiang et al., 2025; Li et al., 2025) to generate robot data, creating a scaling bottleneck, or limits generation to only specific skills such as grasping (Ye et al., 2025; Chen et al., 2025). We aim to address these challenges, enabling simulation-based synthesis of general dexterous behaviors.

Towards this goal, we present DexLink, a simulation-based framework that enables generation of dexterous behaviors across a diverse set of tasks and objects, spanning single-hand, bimanual, long-horizon, articulated, and dynamic tasks. Further, DexLink allows control over behavior synthesis: object goal poses, hand-object interactions, task timing, and high-level execution strategies (Figure 1).

Importantly, DexLink provides these capabilities without requiring costly human demonstrations or cumbersome per-task reward engineering, which existing works often depend on. Existing generation methods vary in how they input task-related information, from indirect optimization objectives to full task demonstrations. Indirect objectives such as task-specific reward design (Chen et al., 2021; Kedia et al., 2026) or hand priors (Mandikal & Grauman, 2022; de Bakker et al., 2026) in RL indirectly guide policies toward specific behaviors: while such guidance is easy to obtain, it provides only weak control over learned behaviors and can produce unnatural robot motions or interactions

that fail to satisfy downstream task requirements. In contrast, methods such as Jiang et al. (2025); Li et al. (2025); Mu et al. (2026) provide task supervision using dense demonstrations obtained from teleoperation, motion-capture data, or vision-based annotations, respectively: the dense supervision provides more control over learning but is costly to collect and, when extracted from video, depends on the accuracy of vision modules. Further, the generation is typically tied to the specific task-object pair represented in the demonstration, limiting scalability.

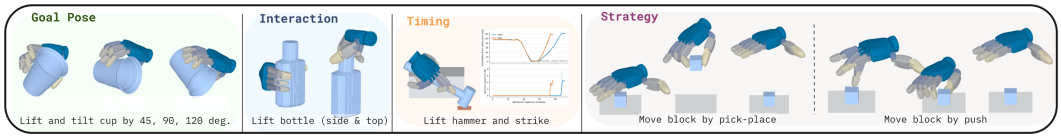

Figure 1: Controllable dexterous behavior axes: goal pose, hand–object interaction, task timing, and high-level strategy.

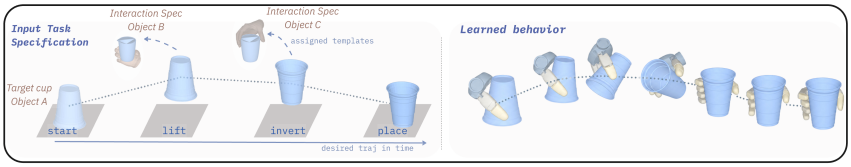

Figure 2: (a) An example task representation provided to the framework. (b) DexLink synthesizes a grounded policy that follows the specification.

DexLink instead achieves the *diversity* and *control* over behavior generation by assuming a sparse, compositional task representation which combines a sequence of object-state waypoints with hand-object interaction templates assigned to selected waypoints (shown in Figure 2, left). The object waypoints describe the desired task progression, while the interaction templates specify the desired spatial relationship between the hand and object. Importantly, these templates only specify local interactions and can be reused across tasks and objects and chained with other templates along the waypoint sequence. Selecting different waypoints and interaction templates allows the target behavior to be varied, or controlled, as desired. These templates can be obtained from motion-capture frames or static hand-object reconstruction methods (Yang et al., 2022; Fan et al., 2023a).

Obtaining a physically grounded trajectory from this compositional task representation presents a challenge: we need to preserve the desired interactions, goals, and phase timings, while remaining dynamically feasible. DexLink addresses this grounding problem by decomposing it into two steps: constructing a reference trajectory that preserves the task intent, and refining the reference for dynamic feasibility. Specifically, an object-aware retargeting optimization adapts each interaction template to the target-object geometry, producing approximate robot anchors. These anchors are linked along the waypoints to form an intermediate reference, which is refined with residual RL in simulation. The intermediate reference narrows RL exploration around the intended behavior, while RL corrects its dynamics, contact, and timing errors to provide a physically grounded trajectory (Figure 2 right).

Our experiments, covering a diverse set of tasks, evaluate the framework's flexibility: interaction templates can be reused across scale, within-category and across-category variations, and generation can be carefully controlled by varying the task specification. Ablations establish the importance of each component, and we show that the framework supports sparse interaction anchors, while performing on par with other approaches for dense object tracking. Finally, zero-shot deployment of policies trained purely in simulation demonstrates the potential for zero-shot real-world transfer.

In summary, our contributions are: (i) a simulation-based framework for synthesizing diverse dexterous behaviors across tasks and objects; (ii) an approach that converts composable sequences of object-state waypoints and hand-object interaction templates into physically grounded policies; and (iii) systematic evaluations in simulation demonstrating coverage over variations in object geometries and tasks, and zero-shot real-world deployment of simulation policies.

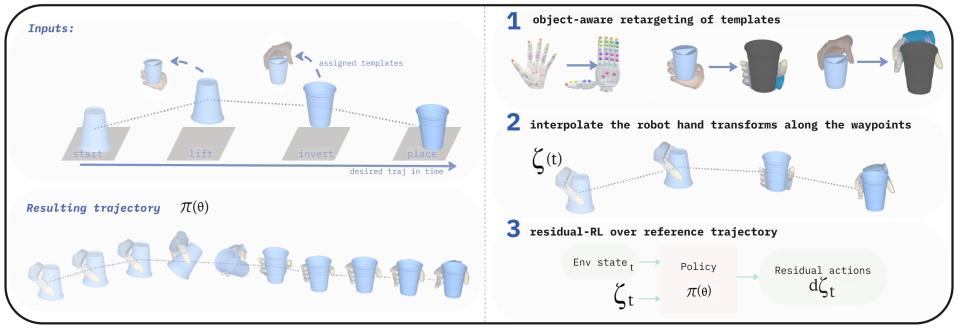

Figure 3: DexLink overview: Given hand-object pose waypoints, object-aware retargeting computes interaction anchors, which are chained to construct an approximate reference trajectory. Residual RL refines the reference into a closed-loop policy.

2 Related Works

Generating Dexterous Demonstrations. Simulation-based approaches specify dexterous behaviors in several ways. One category of methods specifies desired behaviors via reward functions in RL (Chen et al., 2021; Mandikal & Grauman, 2022). While the same reward function can enable learning across diverse objects, generalization to diverse tasks typically requires task-specific reward engineering. Works such as Kedia et al. (2026); de Bakker et al. (2026) can scale to different objects and tasks but cannot control lower-level finger-object interaction patterns, which may matter for downstream tasks. Alternately, dense interaction guidance can be obtained through teleoperation (Jiang et al., 2025), annotated human videos (Hsieh et al., 2025; Mu et al., 2026), or MoCap data (Mandi et al., 2025; Li et al., 2025), avoiding task-specific reward engineering. However, such data is costly to collect and leaves limited room for data augmentation. Generalization to new tasks and objects can be achieved by pretraining on large datasets (Zhao et al., 2025), augmenting demonstrations (Chen et al., 2024), or iteratively expanding object diversity (Zhu et al., 2025). However, using learned priors for generation itself remains challenging: the generation may not adhere to the conditioning and may generate only dominant modes Dhariwal & Nichol (2021), limited by the scale of pretraining. DexLink instead uses sparse interaction templates reused across tasks and objects, while still providing control over interactions at key stages.

Generating Dexterous Grasps. Dexterous grasping is a fundamental component of dexterous behaviors, as many tasks can be decomposed into individual grasp stages (Mu et al., 2026). Methods such as Chen et al. (2025); Ma et al. (2025); Zhong et al. (2025) enable controlled generation of different grasp types from interaction templates. However, these works cannot be easily extended to dynamic tasks, which involve complex interactions that force-closure constraints (Chen et al., 2025), or learned generative models (Ma et al., 2025) cannot output precisely. Although such methods could be incorporated into DexLink for reference generation, we show that an analytical objective is sufficient to transfer interaction templates into a coarse reference that downstream RL can refine.

Human-Robot retargeting. Existing retargeting methods (Qin et al., 2024; Yin et al., 2025; Qin et al., 2022) map human hand poses to robot hand configurations by preserving fingertip positions or vectors between hand keypoints, such as Task Space Vectors. However, these objectives do not account for the target object's geometry. DexLink instead uses object-aware retargeting to optimize robot hand configurations that preserve the source interaction while conforming to the target object.

3 DexLink

Figure 3 provides an overview of the framework. The input is a sequence of object-state waypoints, with hand-object interaction templates assigned to selected waypoints where the desired interaction is important for task success. The output is a closed-loop dexterous policy learned in simulation. The framework consists of two steps: (a) retargeting and composing the selected templates along the task waypoints to construct an approximate reference trajectory, and (b) refining this reference with residual RL.

3.1 TASK SPECIFICATION

The object-state task specification $\mathcal{G} = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_K\}$ is a sequence of desired scene states. Each $\mathbf{s}_k$ is associated with a time t_k and specifies the desired poses of relevant objects and, when applicable, their articulation values. These waypoints encode the final goal and any intermediate states required to complete the task.

At selected waypoints k, the task specification includes a human hand-object interaction template ψ_k assigned to a target scene object with mesh $\mathcal{M}^{\text{tgt}}$:

$$\psi_k = \left(\mathcal{M}_k^{\text{src}}, P_k^{\text{h}}, R_k^{\text{src}\to\text{tgt}}\right).$$

Here, $\mathcal{M}_k^{\text{src}}$ is the source object mesh, P_k^{h} contains the 3D human-hand keypoints expressed in the source-object frame, and $R_k^{\text{src}\to\text{tgt}} \in \text{SO}(3)$ maps the source-object frame to the target-object frame. The source mesh and hand keypoints can be obtained from motion capture or hand-object reconstruction methods. We assume that the chosen templates represent a geometrically meaningful interaction that can be transferred to the target object or a relevant object part, similar to grasp taxonomies (Chen et al., 2025).

Figure 4 shows the rotational alignment. Applying $R_k^{\text{src}\to\text{tgt}}$ expresses the source mesh and hand keypoints in the target-object axes:

$$\bar{\mathcal{M}}_k^{\text{src}} = R_k^{\text{src}\to\text{tgt}} \mathcal{M}_k^{\text{src}}, \qquad \bar{P}_k^{\text{h}} = R_k^{\text{src}\to\text{tgt}} P_k^{\text{h}}.$$

3.2 REFERENCE TRAJECTORY CONSTRUCTION

The reference trajectory need not be dynamically feasible and may contain timing and interaction errors, provided it biases residual RL toward the intended behavior (de Bakker et al., 2026). We construct this reference by retargeting each interaction template ψ_k to the target mesh $\mathcal{M}^{\text{tgt}}$ and interpolating between the resulting robot configurations along the task waypoints (Figure 3, Steps 1-2).

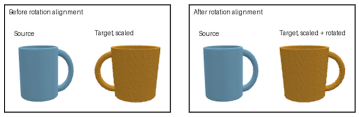

Figure 4: Example of alignment required for task-relevant source and target object parts before retargeting.

Object-aware retargeting. For each template ψ_k, retargeting solves for a robot wrist pose $\mathbf{w}_k^\star \in \text{SE}(3)$, expressed in the target-object frame, and a joint configuration $q_k^\star$ that preserves the source hand-object interaction on the target object. Directly matching hand keypoints is unsuitable when the hand embodiment and object geometry differ. Instead, we preserve relations among hand keypoints using $\mathcal{L}_{\text{cfg}}$ and the spatial relationship between the hand and object using $\mathcal{L}_{\text{rel}}$:

$$(\mathbf{w}_k^\star, q_k^\star) = \arg\min_{\mathbf{w}_k \in \text{SE}(3),\, q_k} \left[\mathcal{L}_{\text{cfg}}\left(\mathbf{w}_k \cdot \text{FK}(q_k), \bar{P}_k^{\text{h}}\right) + \lambda\, \mathcal{L}_{\text{rel}}\left(\mathbf{w}_k \cdot \text{FK}(q_k), \mathcal{M}_k^{\text{tgt}}; \bar{P}_k^{\text{h}}, \bar{\mathcal{M}}_k^{\text{src}}\right)\right].$$

Here, $\text{FK}(q_k)$ denotes the robot hand keypoints returned by forward kinematics in the wrist frame, while $\mathbf{w}_k \cdot \text{FK}(q_k)$ denotes the same keypoints expressed in the target-object frame.

The configuration term preserves geometric relations between hand keypoints, including finger directions (Qin et al., 2022) and palm orientation. Let $\mathcal{C}$ denote the set of relations, and let $\phi_c(P)$ extract relation c from hand keypoints P. We define

$$\mathcal{L}_{\text{cfg}}(P^{\text{r}}, P^{\text{h}}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} d_c\big(\phi_c(P^{\text{r}}), \phi_c(P^{\text{h}})\big),$$

where d_c is cosine distance for normalized directions and squared ℓ_2 distance for unnormalized vectors.

For the hand-object relational term, we compute smoothed BPS-based descriptors (Prokudin et al., 2019) for the human–source and robot–target pairs (Figure 3, Step 1). For each hand keypoint, the descriptor captures its displacement to the object surface D, the corresponding surface normal N, and a normalized object-coordinate feature F. r and h flags correspond to descriptors for the robot and human hand, respectively.

$$\mathcal{L}_{\text{rel}} = \mathcal{L}_{\text{dist}}(D^{\text{r}}, D^{\text{h}}) + \mathcal{L}_{\text{dir}}(D^{\text{r}}, D^{\text{h}}) + \mathcal{L}_{\text{normal}}(N^{\text{r}}, N^{\text{h}}; D^{\text{h}}) + \mathcal{L}_{\text{feat}}(F^{\text{r}}, F^{\text{h}}).$$

The four terms compare hand-object distance, displacement direction, local surface orientation, and normalized object coordinates, respectively. Further details are in the appendix.

Because the descriptor captures spatial hand-object relationships, we can retarget interactions even when the hand and object are not in contact, as in handover tasks. The resulting solution $q_k^\star$ is made penetration-free for simulation.

Chaining waypoints into a reference. Object-aware retargeting produces $(\mathbf{w}_k^\star, q_k^\star)$ at each waypoint with an interaction template. Consider two successive templated waypoints k and ℓ, associated with the same object or articulated part, and let $\alpha_t = (t - t_k)/(t_\ell - t_k)$ for $t \in [t_k, t_\ell]$. We construct the hand reference as

$$\begin{aligned} \mathbf{w}_t^{\text{ref}} &= T_t^{\text{obj}} \cdot \text{Interp}_{\text{SE(3)}}\left(\mathbf{w}_k^\star, \mathbf{w}_\ell^\star; \alpha_t\right), \\ q_t^{\text{ref}} &= \text{Interp}\left(q_k^\star, q_\ell^\star; \alpha_t\right). \end{aligned}$$

Here T_t^{obj} is the interpolated world pose of the corresponding object or part. We interpolate object translations and articulation values linearly, while using spherical interpolation for rotations. Interpolating the wrist transform in the object frame approximately preserves the specified hand-object relationship as the object moves. We apply this process independently for each hand (in case of bimanual tasks). Finally, the constructed trajectory is processed to remove penetrations. The result is a time-indexed reference containing the desired scene state, wrist pose, and finger configuration (Figure 3, Step 2):

$$\xi = \left\{\left(\mathbf{s}_t^{\text{ref}}, \mathbf{w}_t^{\text{ref}}, q_t^{\text{ref}}\right)\right\}_{t=0}^{T}.$$

3.3 Residual RL refinement

The constructed reference provides a nominal motion but may contain errors in timing, contact, or interaction. We therefore train a closed-loop residual policy $\pi_\theta(\cdot; \xi)$ with PPO (Schulman et al., 2017) in simulation, conditioned on ξ, to refine the reference into physically grounded behavior.

At control step t, the policy observes robot proprioception, contact signals, wrist-relative object states, the current reference state ξ_t, and the normalized task progress. It predicts bounded corrections to the reference wrist pose and finger configuration:

$$(\Delta\mathbf{w}_t, \Delta q_t) \sim \pi_\theta(x_t), \qquad \mathbf{w}_t^{\text{cmd}} = \mathbf{w}_t^{\text{ref}} \oplus \Delta\mathbf{w}_t, \qquad q_t^{\text{cmd}} = q_t^{\text{ref}} + \Delta q_t,$$

where x_t denotes the policy observation, and $\oplus$ represents an addition in SE(3). The commanded wrist and finger targets are executed with a low-level PD controller over the joints. The reward function for RL consists of: binary task signals, reference tracking, preservation of specified interactions, and motion smoothness:

$$R_t = \sum_{k \in \mathcal{K}} \bar{w}_k r_k(t) + w_{\text{succ}} \mathbf{1}_{\text{success}}(t), \qquad \mathcal{K} = \{\text{obj}, \text{int}, \text{ref}, \text{smooth}\}.$$

where $\bar{w}_k$ are normalized dense-term weights. The key dense terms are

$$r_{\text{obj}}(t) = \frac{1}{|\mathcal{O}_t|} \sum_{o \in \mathcal{O}_t} \exp\left[-\left(s_p e_{o,t}^p + s_R e_{o,t}^R + s_a e_{o,t}^a\right)\right], \qquad r_{\text{ref}}(t) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \exp\left[-s_{\text{ref}}\left(e_{h,t}^{w,p} + e_{h,t}^{w,R} + e_{h,t}^f\right)\right],$$

$$r_{\text{int}}(t) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \exp\left[-s_{\text{int}}\left(\lambda_{\text{cfg}} \mathcal{L}_{\text{cfg}}^h(t) + \lambda_{\text{rel}} \mathcal{L}_{\text{rel}}^h(t)\right)\right].$$

$\mathcal{O}_t$ is the set of tracked objects and $\mathcal{H}$ is the set of robot hands. The errors e^p, e^R, e^a measure object position, orientation, and articulation relative to the reference scene trajectory, while $e^{w,p}, e^{w,R}, e^f$ measure wrist and finger deviation from the robot reference. The interaction terms $\mathcal{L}_{\text{cfg}}$ and $\mathcal{L}_{\text{rel}}$ reuse the configuration and hand-object objectives introduced in Section 3.2. r_{smooth} is an exponential penalty on the change in joint values in consecutive steps. Consequently, r_{obj} encourages task

progress, r_{int} preserve the specified interaction, r_{ref} provides a nominal motion bias, and r_{smooth} discourages abrupt joint motion.

To parallelize learning across long task horizons, each environment samples an initial reference index and initializes the robot and objects from the corresponding collision-refined reference state. This reset distribution allows different portions of the task to be learned concurrently.

4 EXPERIMENTS

We evaluate whether DexLink provides a useful interface for generating diverse dexterous behaviors. Our experiments address the following questions: **(A)** Does object-aware retargeting preserve interaction diversity, staying close to the specified human-object interactions? **(B)** Can the learned dexterous behaviors be reliably controlled with changes in the task specification, and can interaction templates be reused across variations in object geometry? **(C)** How important is the intermediate reference trajectory, and how do other design choices affect policy learning? **(D)** How does waypoint density affect policy learning? Does the framework's performance remain on par with dense reference-tracking methods? **(E)** Do the learned policies show potential for zero-shot sim-to-real transfer?

Setup. We use the XHand (ROBOTERA) embodiment in all experiments. Interaction templates are drawn from OakInk (Yang et al., 2022), which provides diverse static and functional hand-object interactions, and ARCTIC (Fan et al., 2023b), which provides temporally dense, bimanual interactions. We use the SNOPT solver in Drake (Tedrake & the Drake Development Team, 2019) for object-aware retargeting, and RL policies are trained in the Genesis simulator (Genesis Authors, 2024) with 1,024 parallel environments. Unless stated otherwise, each policy is trained for 50k policy rollout steps, corresponding to 51.2M env transitions, using the same policy architecture and optimizer settings. The experiments accumulate results across 3 seeds, selecting checkpoints based on the best logged train-time task metrics. The checkpoints are evaluated with greedy action selection for 50 episodes. RL runs for dense reference tracking experiments use a different training budget and checkpoint selection.

Metrics. Because the tasks differ in their objectives, we use task-specific *success criteria* as the primary metric for each task. We report additional metrics for measuring the quality of rollouts: (i) *Dynamic Time Warping* (DTW, $\downarrow$) between the policy and reference trajectories; and (ii) *ADD-AUC* for dense object tracking, which measures deviation from the reference motion using mesh-vertex tracking accuracy, computed as the area under the success curve across distance thresholds Mandi et al. (2025).

4.1 RESULTS

Object-aware retargeting. The retargeted configurations provide interaction anchors for downstream policy learning. We therefore evaluate whether they are physically plausible, remain faithful to the source interactions, and preserve interaction diversity.

We evaluate 287 OakInk hand-object interactions spanning 33 object categories and retarget each interaction to XHand while keeping the object geometry fixed. We compare object-aware retargeting with two baselines: (i) fingertip-only retargeting, which uses IK with explicit fingertip-position targets; and (ii) fingertip retargeting augmented with a hand-configuration matching cost (Qin et al., 2022). Both baselines assume that the object geometry remains unchanged and therefore serve as oracle references in this setting.

Object-aware retargeting achieves force-closure feasibility in $\sim 60\%$ of cases at a 1cm contact threshold (and 80% at a 2cm threshold), showing that retargeted interactions are physically plausible but lack accuracy. On the feasible set, we compute the recall of retargeted interactions for each object mesh, achieving ~ 0.8 recall over random chance (0.4), indicating that the retargeted behaviors remain close to the original specified interactions. Finally, we visualize the spread of feasible retargetings over the human-fitted PCA space (using joint angle vector). We see that retargetings remain well-distributed, and that the optimization does not collapse the diversity of the source interactions.

Robustness to object-geometry changes. The previous experiment fixed the object geometry across retargeting. We now show that interactions transfer reliably across object-geometry varia-

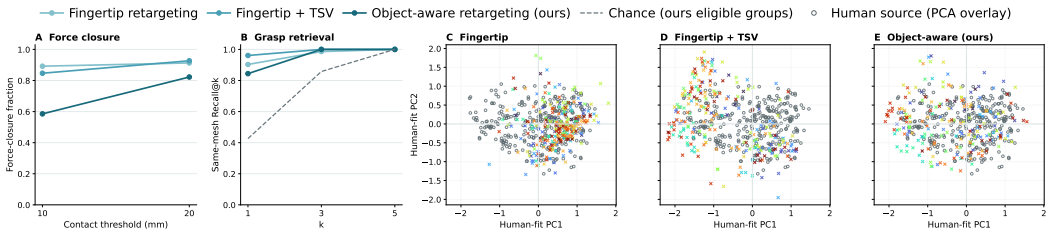

Figure 5: Quality and diversity of retargeted interactions. (A) Fraction of retargetings satisfying force closure at contact thresholds 1 and 2 cm. (B) Retrieval of retargeted interactions using hand contact map similarity compared to the source template, reported with Recallk; dashed line is random retrieval success. (C-E) Feasible robot joint configurations projected onto the first two principal components fitted to the 287 human-hand configurations. Gray circles denote human source templates and colored crosses denote the robot retargetings.

tions, specifically targeting the `lift` task: grasping and raising the object $15cm$ off the table. The task specification contains two waypoints-object on the table and object in the air-with the same interaction template assigned to both. For scale robustness, we transfer five source templates spanning bottle and mug interactions to different target instances, each evaluated at four scales. For instance- and category-level transfer, we compare templates transferred from another object or category with templates obtained on the target object. Tables 1a and 1b report mean lift success.

Table 1: Template transfer to object geometry variations. Reported are mean success for lift tasks.

	Target-object scale			
Hand-object interaction	0.70	0.85	1.00	1.25
Bottle side	1.0	1.0	1.0	0.0
Bottle top	0.33	1.0	1.0	1.0
Mug handle	0.67	1.0	0.67	0.0
Mug side	0.99	1.0	1.0	1.0
Mug top	1.0	1.0	1.0	0.67

(a) Transfer to a target instance across scales.

Transfer type	Source category	Target category	Transferred (source → target)	Exact (target interaction)
Within-category transfer	Mug	Mug	1.0	1.0
	Bottle	Bottle	1.0	1.0
	Trigger sprayer	Trigger sprayer	1.0	1.0
Out-of-category transfer	Apple	Camera	1.0	1.0
	Hammer	Wineglass	1.0	0.67
	Apple	Mouse	1.0	1.0

(b) Instance- & category-level transfer

Overall, we observe perfect lift success on 13/20 scale conditions, concentrated around the nominal scale. Performance is less reliable at extreme scales and varies across RL seeds; for example, the bottle-side grasp becomes unstable at a target-object scale of 1.25. Transferred templates also achieve consistent success across the instance and category-level evaluations (Table 1b), performing comparably to target-specific templates. These results show that a source interaction can guide lifting across changes in object instance, scale, and category without requiring a separately generated grasp for each target.

Task-specification controlling policy behavior. We evaluate control over learned behavior along four axes (Figure 1). *Goal pose* for `lift-and-tilt` changes the mug's final tilt angle; *Interaction* changes the grasp template for bottle lift (covered previously); *Timing* for `hammer-lift-and-strike` alters the timestep of hammer swing waypoints, mainly changing the swing speed; and *Strategy* for `move-block` alters the intermediate waypoint pose for block, resulting in push and pick-place strategies. Each task variant uses a single interaction template. Figure 6 reports task success based on the final object state.

We observe that an interpretable task specification interface can generate diverse variations, with control over goal pose, task strategy, and timing: all task variations achieve $> 66\%$. For the `move-block`, the desired strategy is achieved in $2/3$ seeds, and timing control in the hammer task visibly alters the strength of impact, from $\sim 4N$ for slow motion to $\sim 7.5N$ for fast motion. Some task variants exhibit high variance across seeds, with two seeds succeeding and one failing completely.

Ablations across representative tasks We evaluate the framework's design choices on four representative tasks: a grasp transition (`mug-grasp-change`), articulation (`open-notebook`), sequential manipulation (`stack-3-blocks`), and dynamic bimanual manipulation

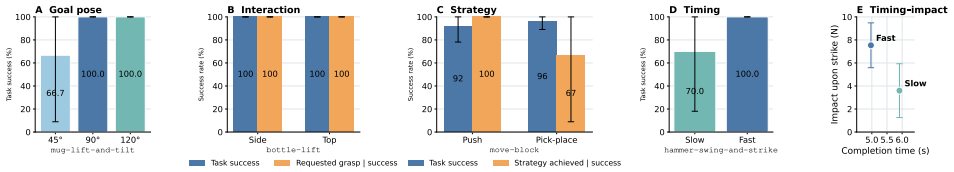

Figure 6: Controllability through task specification. (A) Success for different target mug orientations. (B-C) Task success and strategy-achieved rate (on success traj.). (D-E) Hammer strike success under slow and fast timing, and the effect of timing on impact upon strike.

Table 2: Ablations across four task categories. DTW is evaluated only on successful rollouts.

Method	Mug grasp change			Open notebook		Stack blocks			Banana handover		
	Lift success (%) ↑	Wrist rot. DTW ↓	Ordered grasp success (%) ↑	Success (%) ↑	Artic. DTW ↓	Success (%) ↑	Pos. DTW ↓	Rot. DTW ↓	Success (%) ↑	Pos. DTW ↓	Rot. DTW ↓
Obj. tracking + wrist residual	–	–	–	100.0 ± 0.0	0.044 ± 0.003	0.0 ± 0.0	–	–	–	–	–
No reference traj	–	–	0.0 ± 0.0	–	–	0.0 ± 0.0	–	–	–	–	–
Delta actions	100.0 ± 0.0	0.210 ± 0.042	33.3 ± 57.7	0.0 ± 0.0	–	0.0 ± 0.0	–	–	66.7 ± 57.7	0.065 ± 0.013	0.421 ± 0.041
Initial-only resets	100.0 ± 0.0	0.251 ± 0.058	33.3 ± 57.7	0.0 ± 0.0	–	30.0 ± 52.0	0.007	0.024	0.0 ± 0.0	–	–
No interaction rewards	100.0 ± 0.0	0.314 ± 0.023	0.0 ± 0.0	66.7 ± 57.7	0.042 ± 0.001	0.0 ± 0.0	–	–	33.3 ± 57.7	0.056	0.483
Full method	100.0 ± 0.0	0.163 ± 0.091	92.0 ± 12.2	100.0 ± 0.0	0.055 ± 0.022	86.7 ± 16.2	0.008 ± 0.002	0.035 ± 0.018	97.3 ± 4.6	0.037 ± 0.002	0.371 ± 0.008

(`banana-hand-transfer`). Success is determined from the final scene state: whether the mug is grasped, the notebook is opened, the blocks are stacked, or the banana is transferred to the left hand. We additionally report DTW between the rollout and reference trajectories, separately measuring object position, object orientation, articulation, or wrist orientation as appropriate for each task.

Table 2 shows that removing hand trajectory optimization fails to learn across all tasks within the training budget. Replacing reference-centered residual actions with delta actions (where the policy predicts motion with respect to the current state, instead of the reference trajectory) succeeds in mug-lift and partially on the banana handover, but fails on notebook-opening and block-stacking tasks, and is less reliable in achieveing the specified grasp change in mug task. This highlights the importance of the reference bias for tasks with long horizons or precise motion requirements. Restricting resets to the initial state also reduces performance, particularly on the longer-horizon tasks, showing the benefit of sampling resets along the reference trajectory. Removing the interaction rewards preserves the mug lift success, but fail to achieve the grasp change, and performance also degrades on the notebook, block-stacking, and banana tasks. Overall, the results show that a reference containing both wrist and finger motion, together with reference-centered actions and trajectory-wide resets, is important for reliable learning.

Works such as de Bakker et al. (2026) and Chen et al. (2024) often provide a reference only for the wrist and learn finger actions from scratch. Building on these works, we test another variant of our method: the action space consists of wrist residuals and absolute finger-joint actions. We evaluate this setting on notebook opening and block stacking using wrist residuals and absolute finger actions. It succeeds on notebook opening but fails on the longer-horizon block-stacking task, further demonstrating the value of including finger motion in the reference.

Sparse vs Dense Tracking. We study how densely interaction templates must be specified using `open-notebook` and `banana-handover`, two tasks with dense hand-object motion references from ARCTIC. From each dense reference, we construct three task specifications: templates at every frame, templates at a stride of six frames, and templates only at the start and end of the task. All variants retain the same trajectory duration, with configurations between specified templates obtained through interpolation.

Figure 7 shows that the right template density is task-dependent: open notebook remains stable with sparse waypoints, whereas banana handover benefits from denser temporal guidance. For tasks with discrete hand-object interaction phases (all tasks, except banana-handover), where the interaction type remains constant, a single source template per phase works well in practice.

Dense Reference Tracking Finally, we test the framework's ability to work with dense task specifications using trajectories from Arctic MoCap data. We follow the evaluation setup used by Mandi

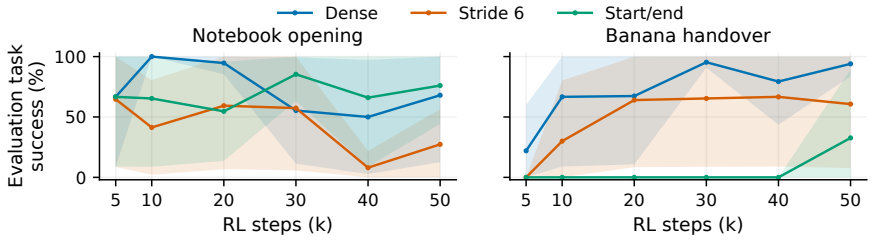

Figure 7: Effect of template density on learning for open-notebook and banana-handover.

et al. (2025). Of the 7 task clips, five use matched object geometry, one is transferred to an in-category size variation, and one is transferred across categories from waffle-iron to a notebook.

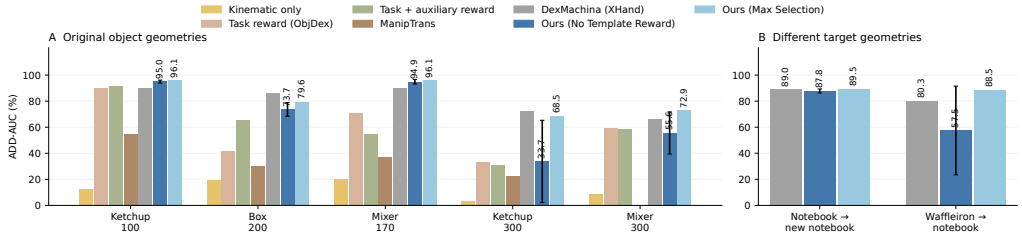

Figure 8: Dense object tracking comparisons. The first five tasks use the original object geometry; the final two transfer the reference to an in-category or out-of-category target. For these transfer cases, the DexMachina values correspond to the original source and are included only as a reference. `Ours (Max Selection)` represents the best seed run (out of 3 seeds). Baseline values are reproduced from the published results of Mandi et al. (2025).

Our RL experiments use a different setting from previous experiments: we train with a maximum budget of 250k rollout steps and use the no-template reward variant of our method, because we find that template rewards can contradict highly dynamic object-motion targets, such as ARCTIC clips. Performance is evaluated using ADD-AUC, which measures object-trajectory tracking accuracy.

Figure 8 compares our method with results from Mandi et al. (2025). Except for two clips with high variance across seeds, DexLink achieves comparable dense-tracking performance while also supporting in-category and cross-category transfer, without requiring a hand-designed curriculum over object motion or gravity, as prior methods commonly do (Mandi et al., 2025; Li et al., 2025).

Real World Deployment. We deploy point-cloud policies trained entirely in simulation on four real objects: a mug and a mustard bottle from the training set, and two zero-shot transfers to different objects: a cup and a lego-block. We achieve object-level transfer through domain randomization during training. Figure 9 reports 10 grasp-and-lift trials per object. In this limited setting, the zero-shot policies successfully grasp and lift all four objects, including the two zero-shot transferred instances. This provides preliminary evidence of sim-to-real transfer of policies trained entirely in simulation.

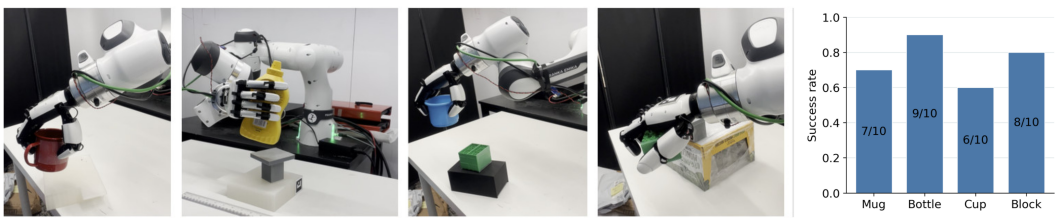

Figure 9: Sim-to-real grasp-and-lift evaluation. Left: successful rollouts with the mug and mustard bottle seen during simulation training and the unseen cup and block. Right: grasp success over 10 rollouts per object.

4.2 Generation cost and Multi-task RL

Tasks can be authored from MoCap references, frames extracted from videos, or an agentic harness with human verification. In our authoring study, estimated manual effort for agentic authoring is < 1 minute per task. Given a task specification, retargeting and RL run automatically without further manual intervention. Retargeting takes ~ 10 minutes on a CPU, while RL requires 1-20 hours of GPU time, depending on the task.

We show that multi-task RL policies can learn related task variations jointly and reduce the additional training needed for new variations. We adjust policy inputs and outputs to enable task-conditioned learning across 30 cup-manipulation variants (Figure 10). The shared-policy results show $\sim$70% success on the training task set and 50% zero-shot success on two new cup scales. Fine-tuning on the new scales reaches comparable performance to training from scratch with 4x fewer training steps.

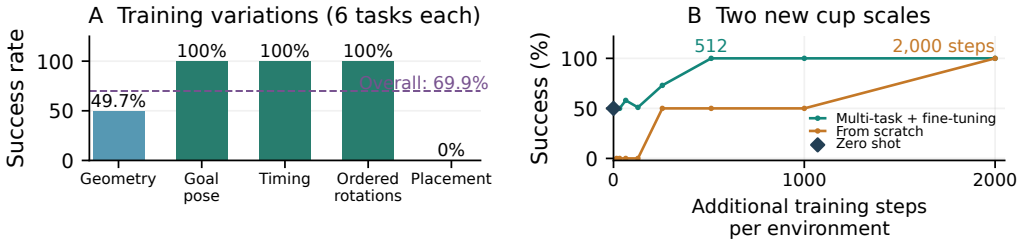

Figure 10: (A) Shared-policy success on 30 training variants (six per category). (B) Zero-shot transfer and fine-tuning on two new cup scales, compared with training from scratch using the same architecture.

5 Conclusion

We presented DexLink, a framework for synthesizing diverse dexterous manipulation data across objects and tasks, covering single-hand, bimanual, long-horizon, articulated, and dynamic tasks. Through systematic experimental analysis, we demonstrated the different capabilities of the framework: control over the generated behaviors along four axes, the reusability of interaction templates (used in specifying the desired behaviors) across object geometry variations, the ability to track both sparse and dense object reference motions, and a systematic ablation to emphasize the contribution of each component.

For task rollouts, please visit the project website.

Limitations

The framework has the following limitations. First, it requires an explicit task specification, which is less readily available than inputs such as videos. We believe current agentic systems are well-suited for this approach of specifying tasks. Second, the scope of the presented real-world experiments is limited to grasp-and-lift tasks. Finally, our experiments target only rigid objects. Sim-to-real transfer for a broader range of tasks and extending the approach to deformable objects are potential future work.

References

Jiayi Chen, Yubin Ke, Lin Peng, and He Wang. Dexonomy: Synthesizing all dexterous grasp types in a grasp taxonomy, 2025. URL https://arxiv.org/abs/2504.18829.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation, 2021. URL https://arxiv.org/abs/2111.03043.

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes, 2023. URL https://arxiv.org/abs/2211.11744.

Yuanpei Chen, Chen Wang, Yaodong Yang, and C. Karen Liu. Object-centric dexterous manipulation from human motion data, 2024. URL https://arxiv.org/abs/2411.04005.

Vincent de Bakker, Joey Hejna, Tyler Ga Wei Lum, Onur Celik, Aleksandar Taranovic, Denis Blessing, Gerhard Neumann, Jeannette Bohg, and Dorsa Sadigh. Scaffolding dexterous manipulation with vision-language models, 2026. URL https://arxiv.org/abs/2506.19212.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Zicong Fan, Maria Parelli, Maria Eleni Kadoglou, Muhammed Kocabas, Xu Chen, Michael J. Black, and Otmar Hilliges. Hold: Category-agnostic 3d reconstruction of interacting hands and objects from video, 2023a. URL https://arxiv.org/abs/2311.18448.

Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J. Black, and Otmar Hilliges. Arctic: A dataset for dexterous bimanual hand-object manipulation, 2023b. URL https://arxiv.org/abs/2204.13662.

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/genesis-world.

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, 2019. URL https://arxiv.org/abs/1910.03135.

Jhen Hsieh, Kuan-Hsun Tu, Kuo-Han Hung, and Tsung-Wei Ke. Dexman: Learning bimanual dexterous manipulation from human and generated videos, 2025. URL https://arxiv.org/abs/2510.08475.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning, 2025. URL https://arxiv.org/abs/2410.24185.

Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C. Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation, 2026. URL https://arxiv.org/abs/2602.16863.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning, 2025. URL https://arxiv.org/abs/2503.21860.

Yiyao Ma, Kai Chen, Kexin ZHENG, and DOU QI. Contact map transfer with conditional diffusion model for generalizable dexterous grasp generation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 79807–79836. Curran Associates, Inc., 2025. doi: 10.52202/085713-2666. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/728a3f18902026dc60b7f7a84c7d2ce4-Paper-Conference.pdf.

Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation, 2025. URL https://arxiv.org/abs/2505.24853.

Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video, 2022. URL https://arxiv.org/abs/2202.00164.

Juncheng Mu, Sizhe Yang, Yiming Bao, Hojin Bae, Tianming Wei, Linning Xu, Boyi Li, Huazhe Xu, and Jiangmiao Pang. Deximit: Learning bimanual dexterous manipulation from monocular human videos, 2026. URL https://arxiv.org/abs/2602.10105.

Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets, 2019. URL https://arxiv.org/abs/1908.09186.

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, 2022. URL https://arxiv.org/abs/2108.05877.

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, 2024. URL https://arxiv.org/abs/2307.04577.

ROBOTERA. XHAND 1. https://www.robotera.com/en/goods1/4.html#/product/XHAND.

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

Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *Robotics: Science and Systems (RSS)*, 2025.

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

Lixin Yang, Kailin Li, Xinyu Zhan, Fei Wu, Anran Xu, Liu Liu, and Cewu Lu. Oakink: A large-scale knowledge repository for understanding hand-object interaction, 2022. URL https://arxiv.org/abs/2203.15709.

Jianglong Ye, Keyi Wang, Chengjing Yuan, Ruihan Yang, Yiquan Li, Jiyue Zhu, Yuzhe Qin, Xueyan Zou, and Xiaolong Wang. Dex1b: Learning with 1b demonstrations for dexterous manipulation, 2025. URL https://arxiv.org/abs/2506.17198.

Zhao-Heng Yin, Changhao Wang, Luis Pineda, Krishna Bodduluri, Tingfan Wu, Pieter Abbeel, and Mustafa Mukadam. Geometric retargeting: A principled, ultrafast neural hand retargeting algorithm, 2025. URL https://arxiv.org/abs/2503.07541.

Siheng Zhao, Yanjie Ze, Yue Wang, C. Karen Liu, Pieter Abbeel, Guanya Shi, and Rocky Duan. Resmimic: From general motion tracking to humanoid whole-body loco-manipulation via residual learning, 2025. URL https://arxiv.org/abs/2510.05070.

Tao Zhong, Jonah Buchanan, and Christine Allen-Blanchette. Grasp2grasp: Vision-based dexterous grasp translation via schrödinger bridges, 2025. URL https://arxiv.org/abs/2506.02489.

Kefei Zhu, Fengshuo Bai, YuanHao Xiang, Yishuai Cai, Xinglin Chen, Ruochong Li, Xingtao Wang, Hao Dong, Yaodong Yang, Xiaopeng Fan, and Yuanpei Chen. Dexflywheel: A scalable and self-improving data generation framework for dexterous manipulation, 2025. URL https://arxiv.org/abs/2509.23829.

A Reference Trajectory Construction Details

A.1 Retargeting Objective

For interaction template ψ_k, the robot keypoints in the target-object frame are $P_k^{\mathrm{r}} = \mathbf{w}_k \cdot \mathrm{FK}(q_k)$, and the aligned human keypoints are $\bar{P}_k^{\mathrm{h}}$. Retargeting compares these keypoints using the hand-configuration loss $\mathcal{L}_{\mathrm{cfg}}$ and compares their relationships to the corresponding objects using $\mathcal{L}_{\mathrm{rel}}$. Figure 11 shows the shared human and robot keypoints.



Figure 11: Hand keypoints used by the retargeting and reward terms, with corresponding points on the human and robot hands.

Hand-configuration similarity. Table 3 defines the pair and triplet relations used by $\mathcal{L}_{\mathrm{cfg}}$.

Table 3: Hand keypoint relations used in $\mathcal{L}_{\mathrm{cfg}}$. The last column indicates whether the vector is normalized before comparison.

Group	Relation set	Norm.
Wrist→tip	$\{\,\texttt{wrist} \to f_{\texttt{tip}} \mid f \in \{\texttt{thumb},\texttt{index},\texttt{middle},\texttt{ring},\texttt{little}\}\,\}$	Yes
Base→tip	$\{\,f_{\texttt{base}} \to f_{\texttt{tip}} \mid f \in \{\texttt{thumb},\texttt{index},\texttt{middle},\texttt{ring},\texttt{little}\}\,\}$	Yes
Thumb-tip opposition	$\{\,\texttt{thumb_tip} \to f_{\texttt{tip}} \mid f \in \{\texttt{index},\texttt{middle},\texttt{ring},\texttt{little}\}\,\}$	No
Palm orientation	$(\texttt{thumb_base} - \texttt{wrist}) \times (\texttt{little_base} - \texttt{wrist})$	Yes

For $s \in \{\mathrm{r}, \mathrm{h}\}$, let $p^s(u)$ denote the position of keypoint u in P^{r} or P^{h}. For a pair (u, v) in Table 3, define

$$v_{uv}^s = p^s(v) - p^s(u), \qquad \hat{v}_{uv}^s = v_{uv}^s / (\|v_{uv}^s\|_2 + \epsilon).$$

For the palm triplet (u, v, w), let

$$c_{uvw}^s = (p^s(v) - p^s(u)) \times (p^s(w) - p^s(u)), \qquad \hat{c}_{uvw}^s = c_{uvw}^s / (\|c_{uvw}^s\|_2 + \epsilon).$$

The loss compares the corresponding relations of the two hands:

$$\mathcal{L}_{\mathrm{cfg}}(P^{\mathrm{r}}, P^{\mathrm{h}}) = \frac{1}{|\mathcal{P}|} \sum_{(u,v)\in\mathcal{P}} d_{\mathrm{pair}}(u, v) + \frac{1}{|\mathcal{T}|} \sum_{(u,v,w)\in\mathcal{T}} d_{\mathrm{triplet}}(u, v, w),$$

where normalized relations use $2(1 - \hat{v}^{\mathrm{r}} \cdot \hat{v}^{\mathrm{h}})$ or $2(1 - \hat{c}^{\mathrm{r}} \cdot \hat{c}^{\mathrm{h}})$, and unnormalized relations use squared Euclidean distance.

Hand-object relational similarity. The hand-object term uses a differentiable BPS-based descriptor (Prokudin et al., 2019). For each object, the optimizer samples a point cloud with surface normals and normalized object-coordinate features. For a hand keypoint b_i (marked in Figure 11) and object surface samples x_j, we avoid a hard nearest-neighbor assignment. Instead, we use exponential

soft-nearest weights

$$a_{ij} = \frac{\exp(-\beta\|b_i - x_j\|_2^2)}{\sum_k \exp(-\beta\|b_i - x_k\|_2^2)}.$$

The descriptor then uses the expected closest point, normal, and feature channels under these weights:

$$\bar{x}_i = \sum_j a_{ij} x_j, \qquad \bar{n}_i = \frac{\sum_j a_{ij} n_j}{\|\sum_j a_{ij} n_j\|_2}, \qquad \bar{f}_i = \sum_j a_{ij} f_j.$$

This exponential weighting makes the hand-object descriptor differentiable with respect to the hand keypoint positions, so the resulting objective can be optimized by SNOPT. The descriptor stores the displacement to the soft-nearest surface point, its normal, and its normalized object-coordinate feature:

$$\Psi_{\mathrm{rel}}(P, \mathcal{M}) = [D, N, F].$$

Using the notation from the main paper, we compute

$$(D^{\mathrm{r}}, N^{\mathrm{r}}, F^{\mathrm{r}}) = \Psi_{\mathrm{rel}}(P_k^{\mathrm{r}}, \mathcal{M}_k^{\mathrm{tgt}}),$$
$$(D^{\mathrm{h}}, N^{\mathrm{h}}, F^{\mathrm{h}}) = \Psi_{\mathrm{rel}}(\bar{P}_k^{\mathrm{h}}, \bar{\mathcal{M}}_k^{\mathrm{src}}).$$

We compare the robot and source BPS descriptors using four terms. Let $d_{r,i}^2 = \|D_i^{\mathrm{r}}\|_2^2$ and $d_{h,i}^2 = \|D_i^{\mathrm{h}}\|_2^2$. The distance term matches the relative hand-object distance by

$$\mathcal{L}_{\mathrm{dist}} = \sum_i \left| 1 - \frac{d_{r,i}^2}{d_{h,i}^2} \right|.$$

The direction term compares normalized displacement directions,

$$\mathcal{L}_{\mathrm{dir}} = \sum_i \left(1 - \hat{D}_i^{\mathrm{r}} \cdot \hat{D}_i^{\mathrm{h}} \right).$$

The normal term compares object normals, emphasizing near-contact relations,

$$\mathcal{L}_{\mathrm{normal}} = \sum_i \left(1 - N_i^{\mathrm{r}} \cdot N_i^{\mathrm{h}}\right) \exp\left(-\frac{d_{h,i}^2}{2(0.05)^2} \right).$$

Finally, when feature channels are present, the feature term is

$$\mathcal{L}_{\mathrm{feat}} = \sum_i |F_i^{\mathrm{r}} - F_i^{\mathrm{h}}|.$$

The hand-object relational loss is

$$\mathcal{L}_{\mathrm{rel}} = \mathcal{L}_{\mathrm{dist}} + \mathcal{L}_{\mathrm{dir}} + \mathcal{L}_{\mathrm{normal}} + \mathcal{L}_{\mathrm{feat}}.$$

Final Objective. For each interaction template, the retargeting objective is

$$\mathcal{L}_{\mathrm{ret}} = \mathcal{L}_{\mathrm{cfg}} + \lambda \mathcal{L}_{\mathrm{rel}},$$

matching the objective in the main paper.

Retargeting hyperparameters. Table 4 summarizes the main numerical settings used for object-aware retargeting. The hand-object descriptor is computed by sampling the target object surface and evaluating a soft nearest surface point for each selected hand keypoint.

Figure 12 summarizes the retargeting runtime across object categories. On average, optimization takes about 10 minutes on a single CPU. Waypoints are retargeted independently and can be parallelized.

A.2 Resolving Penetration and Enforcing Contacts

The optimization produces coarse retargeting results that can contain small hand-object penetrations. We therefore apply a lightweight post-processing step that performs sequential line search over finger-joint groups. The line search selects nearby poses that remove robot-object penetration and, when the source template indicates contact at a keypoint, moves that keypoint closer to the object. This step prevents simulation rollouts from starting with interpenetrating geometry.

Table 4: Retargeting hyperparameters for XHand and the Drake/SNOPT solver.

Setting	Value
Solver tolerance	10^{-4}
Object surface samples	512
Hand keypoints in BPS descriptor	39
BPS channels per keypoint	9: 3D displacement, surface normal, normalized object coordinates
Flattened BPS descriptor size	$39 \times 9 = 351$
Relational-loss weight λ	10^{-3}

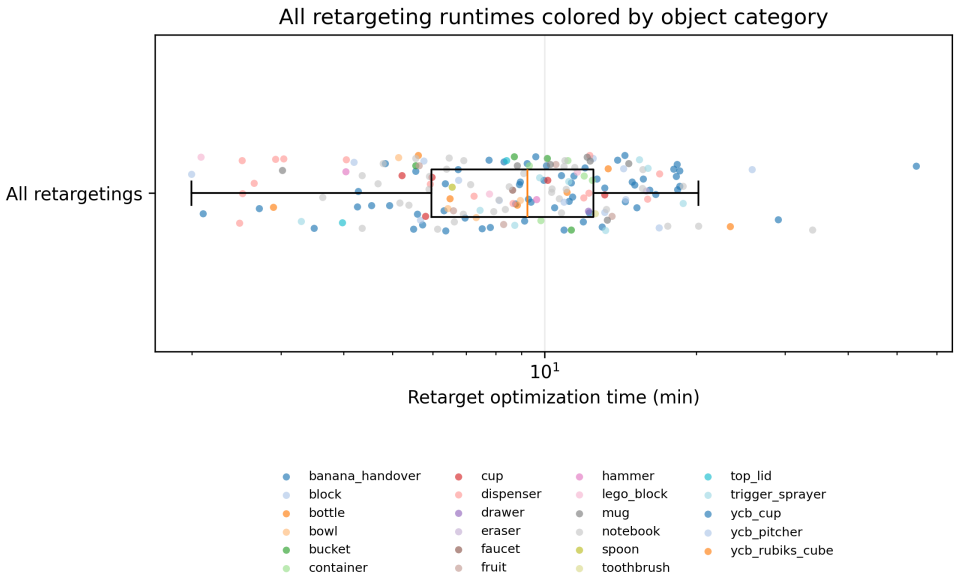

Figure 12: Retargeting runtime by object category. This cost is paid once per task/template before RL training and deployment.

B RL Refinement Details

Table 5: Policy variants and control setup.

Setting	Value
Simulation robot	Floating XHand with 6-DoF wrist
Real robot	Franka arm with XHand
Simulation policy	State based
Real-world policy	Point-cloud based; no privileged simulator state
Controller	Position control with targets interpolated over 8 physics steps

Tables 6 and 7 specify the single-hand state-policy interface.

Table 7: Policy action dimensions for the single-hand state-based experiments.

Action	Dim.
Robot hand action	18
Wrist Cartesian translation delta	3
Wrist rotation-vector delta	3
Controlled hand-joint deltas	12
Total	18

Table 6: Policy observation dimensions for the single-hand state-based experiments.

Observation	Dim.
Robot state	69
Hand joint positions	12
Wrist position and orientation R_x, R_y	9
Hand link positions	48
Reference state	21
Reference wrist position	3
Reference wrist orientation R_x, R_y	6
Reference finger joint positions	12
Object state	10
Object position	3
Object orientation R_x, R_y	6
Object/articulated-part DoFs	1
Task progress: reference timestep	1
Contact state: tactile-link signals	15
Total	116

B.1 Policy Architecture

Table 8: Policy architecture.

Component	**Specification**
Actor/critic	Separate networks; observation-group embeddings are summed and layer-normalized before each MLP
Actor output	Gaussian residual policy
Smoothed target	Correct q_t from reference by Δq_t and blend neighboring corrections with a Gaussian temporal window
Network sizes	Actor: approximately 200k parameters; critic: approximately 25k
Point-cloud policy	PointNet-style hand–object encoder plus non-point-cloud state; state-based critic

B.2 Reward Design

The total reward consists of the terms shown in Table 9. All dense reward values lie in $[0, 1]$, and the dense reward weights are normalized to sum to 1.0. The task-completion reward is added separately.

B.3 Environment Reset

The reset sampler initializes robot and object state from a reference timestep; sampling probability is proportional to the negative exponential of its running visitation density.

Table 9: Reward weights: Dense reward weights are normalized to one; the completion reward is added as a separate sparse bonus.

Reward term	Unnorm. weight	Norm. weight
Dense reward	1.20	1.000
Object tracking	0.75	0.625
Hand-configuration similarity	0.10	0.083
Hand-object relational similarity	0.10	0.083
Retargeted joint-pose closeness	0.20	0.167
Joint smoothness	0.05	0.042
Completion reward	10.00	10.00

Table 10: Components retained in each method variant. A checkmark denotes an enabled component.

	Reference and policy				**Interaction rewards**			
Variant	Optimized robot hand reference	Trajectory-conditioned actor	Reference-relative residual actions	All-phase resets	Hand configuration	Hand–object BPS	Retargeted-joint tracking	Wrist-velocity tracking
Full	✓	✓	✓	✓	✓	✓	✓	✓
No hand trajopt	×	×	×	×	✓*	✓*	×	×
Delta actions	✓	✓	×	✓	✓	✓	✓	✓
Initial-only resets	✓	✓	✓	×	✓	✓	✓	✓
No interaction rewards	✓	✓	✓	✓	×	×	×	×

Variant details. Delta actions predict joint-position changes from the current hand state. Initial-only resets start every episode from the first task state rather than from states spanning the reference trajectory. *Without hand trajopt, the hand-configuration and BPS terms are evaluated against the human templates; robot-reference observations and robot-reference-dependent rewards are removed.

B.4 Ablations

Table 10 describe the ablation variants considered in Table 2, with respect to change in the RL training.

B.5 PPO and Simulator Hyperparameters

Tables 11 and 12 summarize the simulator and optimization settings.

Table 11: Simulation hyperparameters.

Setting	**Value**
Simulation timestep	$1/120$ s
Control decimation	8
Gravity	$-9.81\,\mathrm{m/s^2}$
Default object density	$1200\,\mathrm{kg/m^3}$
Object-mass randomization	None

Table 12: PPO hyperparameters.

Hyperparameter	**Value**
Optimizer	AdamW
Policy learning rate	10^{-4}
Value learning rate	10^{-4}
Discount factor (γ)	0.95
GAE parameter (λ)	0.95
Rollout horizon	8
Minibatch size	1024
Update epochs	4
Clip ratio (ϵ)	0.2
Entropy coefficient	10^{-3}
Value-loss coefficient	0.5
Advantage normalization	No

C DTW METRIC DETAILS

We compute dynamic time warping with the `dtw-python` library, using a normalized Sakoe–Chiba window of 0.10 times the sequence length. We use Euclidean local costs for positions, articulation values, and finger joints, and quaternion geodesic distance for orientations; reported values are the library-normalized distances.

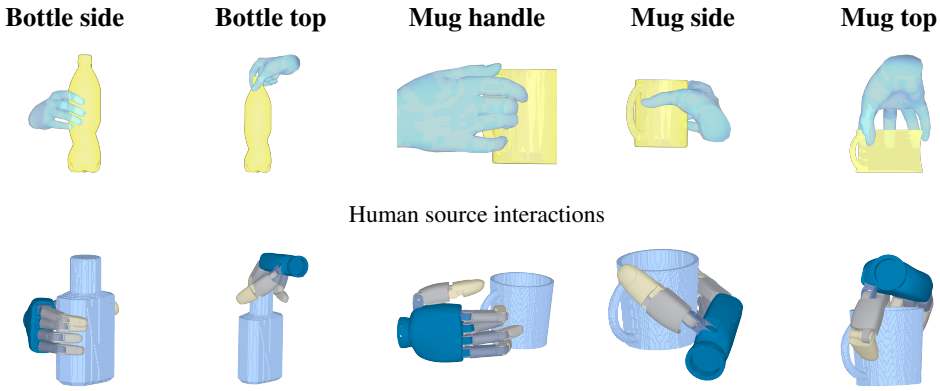

Figure 13: Source and target interactions for scale reuse. Within each row, the same source template is used at all four target-object scales. The $1.00\times$ target is shown for reference.

D Task Details

We use H for the number of human-sourced interaction-frame instances, K for the number of explicit robot keyframes after adding geometrically derived states, and T for the compiled reference horizon. Thus, H/T measures direct human-template coverage, while $(T - K)/T$ is the fraction generated by reach, interpolation, and hold phases.

D.1 Reusability

The scale study fixes one transferred target mesh within each object category and changes only its isotropic scale. The human template object and target mesh are different instances in every scale-reuse family. The transfer study instead compares a cross-instance or cross-category template with an exact-target template while holding the target object and reach–grasp–lift task fixed. Table 13 covers additional details, and Figure 13 and 14 show the templates used visually.

Table 13: Task construction for the reusability experiments. All references contain 20 reach, 5 grasp, 20 lift, and 5 final-hold frames.

Study	**Conditions**	**Interaction families**	H	K	T	**Success criterion**
Scale reuse	$0.70\times$, $0.85\times$, $1.00\times$, $1.25\times$	Bottle side/top; mug handle/side/top. One fixed ShapeNet target per category is rescaled.	1	2	50	Final vertical displacement is within 3 cm of the reference lift; early termination is failure.
Template transfer	Transferred versus exact-target template	Mug top, bottle side, trigger spraying, apple→camera, hammer→wineglass, apple→mouse	1	2	50	Final vertical displacement is within 3 cm of the reference lift; early termination is failure.

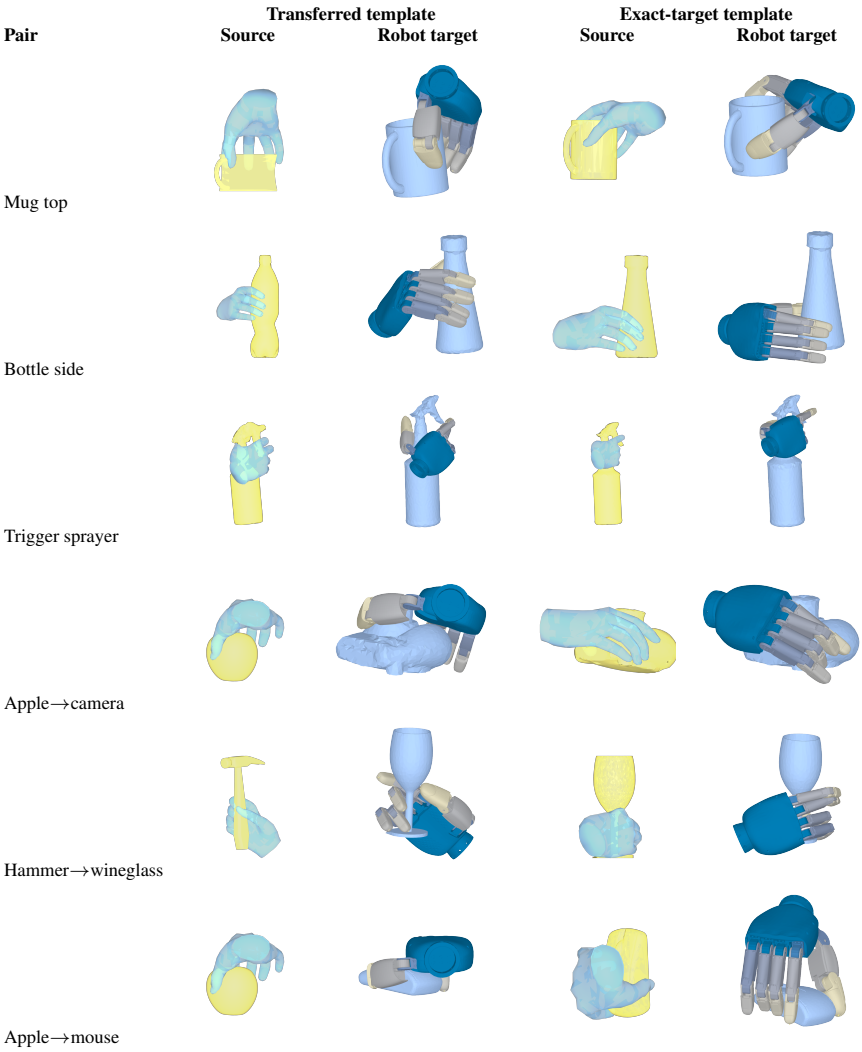

Figure 14: Transferred and exact-target interaction inputs used in the template-transfer comparison. Each row holds the target object and downstream RL task fixed; only the source interaction template changes.

D.2 CONTROLLABILITY

The controllability experiments intervene on one aspect of the task specification at a time. Table 14 records both the direct template budget and the number of derived keyframes used to compile the reference.

Table 14: Task and reference details for the controllability experiments.

Condition	Human source	Robot keyframes	H	K	T	H/T	Success criterion
Cup pose: 45°, 90°, 120°	OakInk cup top grasp	Retargeted grasp; derived lift and terminal tilt	1	3	59	1.7%	Final cup position and orientation are within 3 cm and 15° of the requested pose.
Cube: pick-and-place	OakInk apple top grasp reused on a block	Retargeted grasp; derived lift, hover, place, release	1	5	54	1.9%	Final pose is within 3 cm and 30° of the goal, and maximum center-height increase exceeds 2 cm.
Cube: push	Same source, block, and terminal goal as pick-and-place	Retargeted contact; derived push end and release	1	3	54	1.9%	Final pose is within 3 cm and 30° of the goal, and maximum center-height increase is at most 2 cm.
Hammer: fast	OakInk hammer-handle grasp	Retargeted grasp; lift, ready, three arc points, strike	1	7	78	1.3%	Hammer–peg contact occurs within the authored strike window; early termination is failure.
Hammer: slow	Same grasp and seven robot keyframes as fast	Same keyframes; swing phases expanded	1	7	93	1.1%	Same phase-windowed contact criterion as the fast condition.
Bottle: side/top	OakInk bottle side and top grasps	Retargeted grasp and derived lift	1	2	50	2.0%	Final vertical displacement is within 3 cm of the reference lift. Interaction adherence is reported separately.

D.3 ABLATIONS

The component ablations span both dense demonstration-like references and sparse stage-based references. All policy variants for a task use the same underlying object path, template budget, reference horizon, and evaluation protocol.

Table 15: Task and reference details for the four-task ablation suite.

Task	Human source	Robot keyframes	H	K	T	H/T	Success criterion
Open notebook	ARCTIC notebook-opening sequence	37 consecutive single-hand retargeted frames	37	37	58	63.8%	Final notebook articulation is within 15° of the reference target.
Stack RGB blocks	One OakInk apple top grasp reused for three blocks	Per block: grasp, lift, hover, place, release	3 instances	15	148	2.0%	Complete, aligned, released stack with intended support and 1 s passive stability.
Banana handover	ARCTIC bimanual handover sequence transferred from a ketchup bottle to a banana	30 paired bimanual retargeted frames	30 paired	30 paired	37	81.1%	Banana center is at least 5 cm above the table (not dropped), and left hand is supporting the banana's weight.
Mug grasp change	OakInk top-diagonal and side mug grasps	Two retargeted grasps; derived lift and translation	2	4	75	2.7%	Functional lift: final rise is 0.12 ± 0.03 m. Grasp validity additionally requires an ordered top-to-side grasp similarity.

Stacking success rates and DTW in Table 2 use separate 50-episode-per-seed evaluation batches of the same checkpoints. The batch used for DTW achieved 82.0% success for the full method and 30.0% for initial-only resets. DTW uses strict-success trajectories from that batch and excludes the passive stability window.

D.4 REAL-WORLD ROLLOUTS

Table 16 and Figure 15 provide details on the real-world setup and example rollouts, respectively.

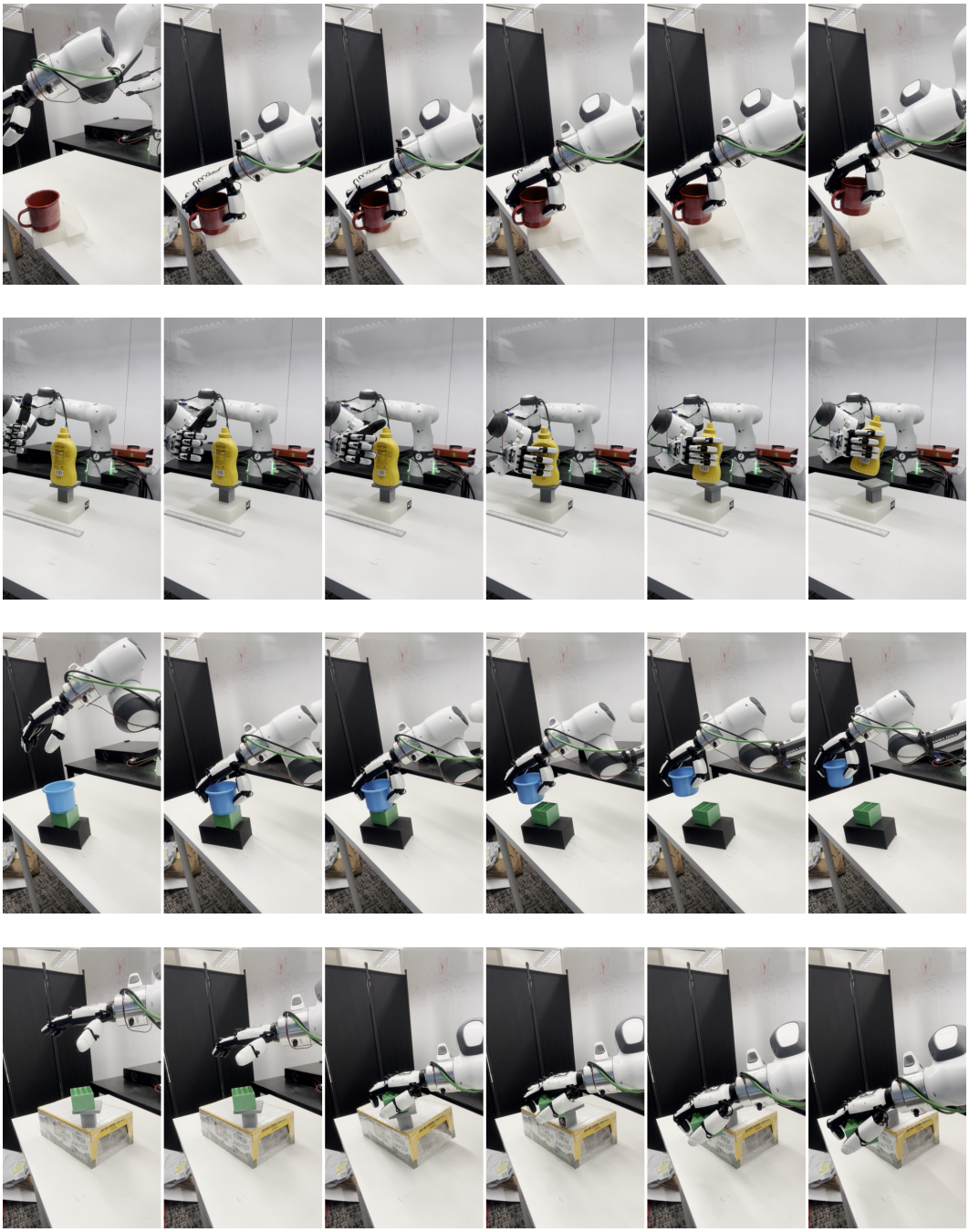

Figure 15: Real-world grasp-and-lift rollouts for the mug, mustard bottle, cup, and block. Each row shows uniformly sampled frames from approach to lift.

Table 16: Real-world deployment and evaluation settings.

Setting	**Value**
Robot	Franka arm with XHand
Sensor	Intel RealSense RGB-D camera
Policy input	Filtered table, object, and robot-hand point clouds
Policy output	delta-wrist pose and finger joints (same as simulation). delta-wrist pose is further processed to 6-DoF wrist actions, and inverse kinematics produces Franka joint targets
Success criterion	Object held in the air for $\sim$ 1 minute.