AdaDexTrack: Dynamic Modulation for Adaptive and Generalizable Dexterous Manipulation Tracking

Jianibieke Adalibieke[*1] Qianwei Han[*1] Xueyi Liu[2] Yuzhe Qin[3] Li Yi[†2,1]
[1]Shanghai Qi Zhi Institute [2]Tsinghua University [3]UC San Diego

https://janebek.github.io/AdaDexTrack

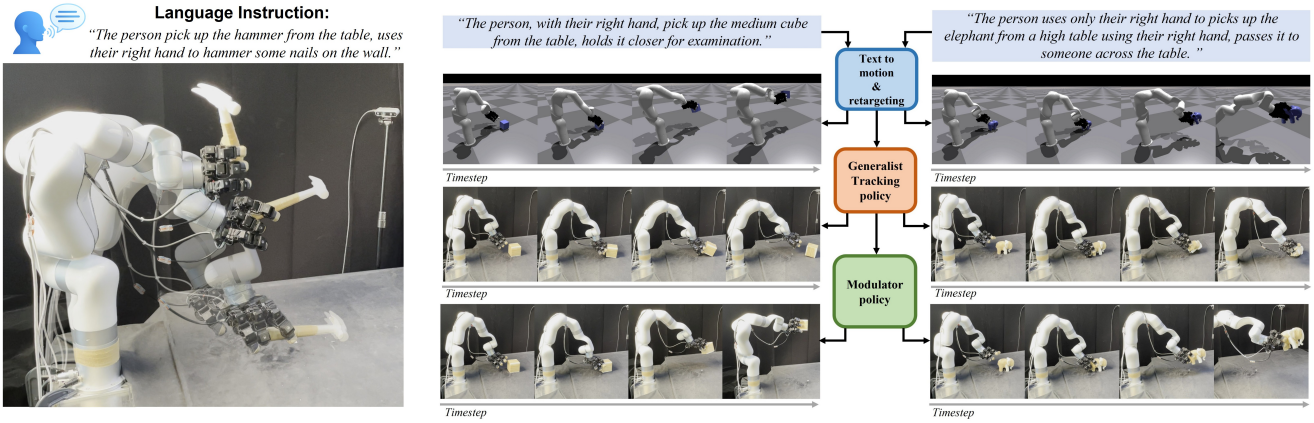


Figure 1. **AdaDexTrack for language-guided hand–object interaction tracking.** Given a noisy text-generated reference retargeted to the robot, the in-loop modulator enables robust long-horizon tracking. **Left:** successful reorientation of a thin object (Hammer). **Right:** generalization to an unseen object (CubeMedium) and an unseen trajectory (Elephant), where the tracker without the modulator fails—highlighting the benefit of in-the-loop modulation. Project website: https://janebek.github.io/AdaDexTrack

Abstract

Language is a natural way to command robots, but converting a single instruction into a long-horizon, contact-rich hand–object interaction remains challenging: synthesized references are noisy, human-to-robot retargeting introduces embodiment bias, and fixed-reference tracking lets small errors snowball. We address this with AdaDexTrack, a modulator-in-the-loop framework for language-conditioned manipulation tracking. A distilled generalist tracker serves as the skill carrier, while a tightly aligned modulator performs three feedback corrections: reference modulation (continual adjustment of what to track), object-latent modulation (online adaptation of the object representation to recruit suitable skills), and positional-target modulation (small state-dependent refinements for execution). The tracker is learned via large-scale specialist to generalist distillation on a corpus of language-conditioned hand–object trajectories; the modulator is trained with RL under the same task objective, ensuring tight coupling. Across large-scale evaluations, AdaDexTrack consistently outperforms prior SOTA on unseen-trajectory and unseen-object sets in both average tracking error and success rate, demonstrating robustness and generalization. We further show zero-shot sim-to-real transfer on real hardware, where adding the modulator yields substantial gains over a tracker-only variant. AdaDexTrack reframes language-conditioned dexterous manipulation as modulated tracking, replacing the open-loop, fixed-reference tracking with in-loop modulation that adjusts the reference, object latent, and positional target, yielding drift-resistant execution from noisy text references.

1. Introduction

Language is a natural interface for commanding robots. We study language-guided hand–object interaction (HOI) tracking: from an instruction and object geometry, a time-indexed hand–object reference is generated and then executed by a multi-fingered hand, requiring stable tracking across contact-rich, long-horizon behaviors.

An intuitive approach is a two-stage system: first, text-to-motion synthesizes an HOI trajectory from the instruction; second, interaction tracking executes that trajectory

on the robot. Although both components have advanced, end-to-end coupling for text-guided manipulation tracking remains nontrivial and rarely demonstrated. Two challenges dominate: (i) noisy references—motion synthesis is imperfect and human-to-robot retargeting introduces embodiment bias; and (ii) open-loop planning under perfect-tracking assumptions—the tracker follows a fixed reference without adapting to how execution unfolds, so small errors accumulate over long horizons and the system eventually loses track. In practice, most trackers treat the reference as fixed [18, 21]. When execution deviates, the controller is forced to chase it. Without conditioning on the robot's current state or capabilities, this chasing degrades control. The policy becomes aggressive, contacts destabilize, and drift accelerates. We therefore advocate modulated tracking: dynamically adjusting the tracking reference, object representations and positional targets—to steer the controller back toward the original reference and maintain stable tracking. By putting these modulations in the feedback loop, the system corrects deviations on the fly and more reliably tracks long-horizon, text-guided trajectories.

We instantiate this idea with AdaDexTrack, a modulator-in-the-loop framework for language-conditioned manipulation tracking. AdaDexTrack has two components: (i) a distilled generalist tracker—the skill carrier—that encodes behaviors learned from diverse demonstrations; and (ii) a tightly aligned modulator that closes the loop via reference, object-latent, and positional-target modulation. Together, they turn noisy, language-conditioned references into drift-resistant executions. The general tracker is trained with large-scale teacher–student distillation: using DiffH2O [9] we synthesize text-guided hand–object trajectories, train a large set of teacher policies with reinforcement learning (RL) to track them, and distill their behaviors into a single general tracking policy. This policy has broad coverage but remains imperfect—exactly where the in-the-loop modulator adds value.

Based on the general tracker, the modulator adapts the tracking process in three ways : (i) reference modulation—adding a small residual to the language-conditioned reference to continually re-align what to track and reduce distribution shift; (ii) object latent modulation—adjusts the object latent to invoke the tracker's most suitable skills, composing behaviors best matched to the current reference; and (iii) positional target modulation—apply a small, state-dependent correction to the tracker's positional targets to refine execution and quickly recover from drift. The modulator shares the same task objective as the tracker and is trained with RL, ensuring tight objective alignment. Together, these modulations make tracking more robust to noise and embodiment mismatch, stabilize long-horizon execution, and improve generalization.

We conduct extensive evaluations in both the simulation and the real world to demonstrate the effectiveness of AdaDexTrack. In simulation, we evaluate on a large-scale text-guided HOI reference dataset and find that AdaDexTrack consistently outperforms all previous state-of-the-art (SOTA) methods on both unseen-trajectory and unseen-object splits in overall average tracking error and success rate, demonstrating robustness and generalization. In the real world, we achieve zero-shot sim-to-real transfer. We show that adding the modulator yields substantial gains over the tracker-only variant on both unseen-trajectory and unseen-object splits, highlighting the benefit of in-the-loop modulation.

To summarize, our contributions are: (i) a re-framing of language-conditioned dexterous manipulation tracking as modulated tracking, augmenting a strong but imperfect distilled tracker with an in-the-loop modulator; (ii) AdaDexTrack, a three-interface modulation scheme—reference, object-latent, and positional-target modulation—where the modulator shares the tracker's task objective, and the tracker is learned via large-scale teacher–student distillation from synthesized text-guided HOI trajectories; and (iii) extensive experiments and ablations showing consistent gains over prior SOTA on unseen-trajectory and unseen-object splits, with zero-shot sim-to-real transfer on hardware further demonstrating the effectiveness of the modulator-in-the-loop design.

2. Related Works

Dexterous Manipulation via HOI Reference. Learning from HOI references is a promising route to human-like dexterity, as such references capture contact timing, coordination synergies, and long-horizon intent. These references can come from (i) teleoperation, which offers high fidelity but is hard to scale [11, 20, 25, 31, 33, 39, 52, 53]; (ii) mocap- or video-based human reconstruction, which is more scalable but labor-intensive, noisy, and subject to embodiment gaps [2, 12, 23, 24, 40, 51]; or (iii) text-to-motion synthesis, which enables language-conditioned HOI generation at higher scale despite synthesis and retargeting noise [4, 9, 10, 13, 30, 46]. A practical way to use these motions is to represent them as time-indexed references and train a policy to track them. Prior RL/IL approaches often learn per-trajectory policies or generalize narrowly [8, 16, 18, 32, 38, 50]. DexTrack [21] broadens coverage by distilling multiple specialists into a single generalist tracker, but still treats tracking as a monolithic controller with a fixed test-time reference. This is fragile in contact-rich settings, where references are noisy and human-biased, embodiment and sensing mismatches introduce estimation error, and small deviations accumulate over long horizons. Classical trajectory modulation methods such as Dynamic Movement Primitives (DMP) [5, 26, 35, 54] and Riemannian Motion Policies (RMP) [34, 43, 47] can adapt motions,

but typically rely on explicit hand-designed dynamical representations. We retain the tracking interface while treating the tracker as a distilled skill carrier and placing a modulator in the loop to adjust the reference, object latent, and positional target online, not to perfectly track the reference, but to correct synthesis noise in the HOI reference and accomplish the overall task.

Skill Hierarchy in Dexterous Manipulation. In dexterous manipulation, a broad class of methods follows a skill hierarchy template: build a library of reusable skills and use a higher-level mechanism to select, sequence, or parameterize them for long-horizon dexterity. Representative instances include open-loop, hand-crafted primitives that compose into longer programs [3], planning-/compliance-based decompositions without learning [28], training-time guidance via simple sub-skill controllers with a learned switch [17], low-level teleoperation skills to scale demonstration collection [19], multi-skill reuse where task transitions are user specified [14], and learned transition-feasibility models that chain dexterous sub-policies and enable long-horizon switching/recovery [6]. ObjDex [7] fits this paradigm with an object-centric planner that proposes wrist trajectories tracked by a low-level controller. While effective, such stacks are loosely coupled: they act primarily via wrist-space proposals, optimize different objectives across levels, and keep the time-indexed reference and the tracker's object representation outside the feedback loop—allowing systematic bias and embodiment mismatch to accumulate into drift. In contrast, we keep the tracking interface and add a tightly aligned in-loop modulator with three channels—reference modulation, object latent modulation, and positional target modulation—yielding more stable, long-horizon tracking.

3. Method

Overview. Fixed text-generated references, together with embodiment mismatch, lead to long-horizon tracking drift; instead of forcing a single controller to track a fixed reference, we close the loop by modulating what the tracker sees and executes. Fig. 2 gives an overview. We split the framework into two stages: (i) a general dexterous manipulation tracking policy π^{track} that serves as the tracking controller and follows the HOI reference; and (ii) a modulation policy π^{modulate} that acts as a tightly aligned modulator, performing reference, object-latent, and positional-target modulation to maximize the tracker's performance under feedback. We first describe how we synthesize a large-scale, text-guided HOI reference dataset and develop a generalizable tracking controller from it (Sec. 3.1), then introduce the modulator and its integration with the tracker (Sec. 3.2), and finally present sim-to-real details (Sec. 3.3).

Task Formulation. From a raw human–object trajectory $\{\hat{\boldsymbol{h}}_t\}_{t=1}^{T}$, we retarget to a robotic hand kinematic sequence $\{\hat{\boldsymbol{s}}_t\}_{t=1}^{T}$ and pose dexterous manipulation tracking as mimicking this sequence. A control policy is learned to mimic the kinematic demonstration while compensating for retargeting errors and motion-generator noise. At each timestep, the reference hand state is defined as $\hat{\boldsymbol{s}}_t^h = \left(\hat{\boldsymbol{q}}_t^h, \hat{\boldsymbol{p}}_t^h\right)$, where $\hat{\boldsymbol{q}}_t^h \in \mathbb{R}^{22}$ denotes the joint positions of the arm–hand chain, and $\hat{\boldsymbol{p}}_t^h \in \mathbb{R}^{5\times 7}$ stacks the rigid-body poses of the tracked links. Specifically, the robotic hand has 16 joints and the robotic arm has 6 joints, and the five tracked links are the four fingertips and the wrist of the robotic hand. The reference object state is defined as $\hat{\boldsymbol{s}}_t^o = \hat{\boldsymbol{p}}_t^o$, where $\hat{\boldsymbol{p}}_t^o \in \mathbb{R}^7$ is the object pose.

We then formulate the dexterous manipulation tracking task as a Markov Decision Process (MDP), $\mathcal{M} = (\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \pi, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}, \gamma)$, where $\boldsymbol{\mathcal{S}}$ is the state space, $\boldsymbol{\mathcal{A}}$ is the action space, π is the policy, $\boldsymbol{\mathcal{T}}$ is the transition dynamics, $\boldsymbol{\mathcal{R}}$ is the reward function, γ is the discount factor. At time step t, the policy maps the reference and current hand–object states to an action, $\boldsymbol{a}_t = \pi(\hat{\boldsymbol{s}}_t, \boldsymbol{s}_t)$, where $\boldsymbol{a}_t \in \mathbb{R}^{22}$ denotes a joint-space action and is controlled by a joint-space proportional–derivative (PD) controller.

3.1. Tracking Controller

Training a generalizable tracker is hard under diverse, noisy language-conditioned references. We address this by (i) synthesizing a large-scale, text-guided HOI reference dataset; (ii) training specialist policies with RL to track these references; and (iii) distilling a large set of RL-trained specialists into a single policy capable of robustly tracking diverse HOI references.

Language-Guided HOI Reference Generation. One of the key challenges in learning a generalizable tracking controller is the scale of the reference dataset. To improve dataset scalability, we leverage the off-the-shelf motion-synthesis method DiffH2O [9] to generate dexterous hand–object interaction trajectories from natural-language prompts. Specifically, we leverage the detailed textual annotations provided by DiffH2O and use GPT-5 [29] for paraphrase augmentation: for each annotated trajectory, we generate multiple semantically equivalent descriptions—following DiffH2O's native annotation format—to increase linguistic variety while preserving intent. We then feed these paraphrases back into DiffH2O to synthesize additional demonstrations. This produces a dataset that is diverse yet plausible: the generated descriptions conform to DiffH2O's annotation style and schema, which keeps the synthesized motions within DiffH2O's generative distribution and reduces distribution shift.

Specialist Policy Training. Based on the human–object references generated by DiffH2O, $\{\hat{\boldsymbol{h}}_n\}_{n=1}^{N}$, we first retarget these motions to the robot arm–hand kinematic chain, yielding $\{\hat{\boldsymbol{s}}_n\}_{n=1}^{N}$. For retargeting, we manually define correspondences between the human-hand and robot-hand

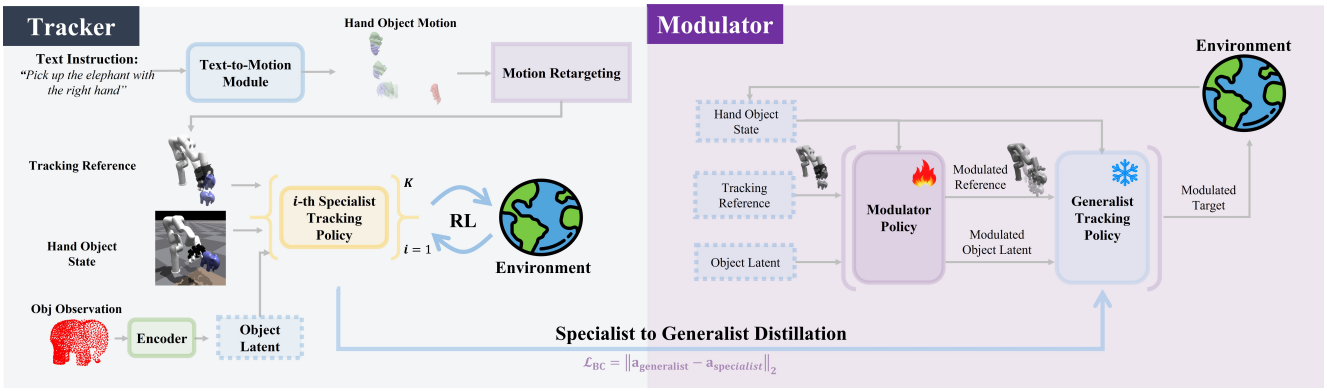


Figure 2. **AdaDexTrack pipeline overview. Left:** From a large-scale set of text-guided HOI references, we train many RL teacher trackers and distill their behaviors into a single generalist tracker. **Right:** On top of this tracker, we insert a modulator in the loop that performs reference modulation, object latent modulation, and positional target modulation, yielding drift-resistant, accurate long-horizon tracking.

meshes and minimize the distances between corresponding points, in a manner similar to DexTrack [21]. For each kinematic reference $\hat{\boldsymbol{s}}_i$, we train a specialized tracking controller π_i^{track} to minimize the discrepancy between the robot state and the reference. We formulate this as a reinforcement learning (RL) problem and train each π_i^{track} with Proximal Policy Optimization (PPO) [36]. For notational simplicity, we drop the subscript i and write π^{track} and $\hat{\boldsymbol{s}}$; unless otherwise stated, they refer to a fixed but arbitrary i.

At timestep t, the observation for π^{track} is defined as $\boldsymbol{o}_t^{\text{track}} = \left(\boldsymbol{s}_t^{\text{track}}, \hat{\boldsymbol{s}}_t^{\text{track}}\right)$, where the current state $\boldsymbol{s}_t^{\text{track}} = \left(\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{s}_t^{\text{o}}, \boldsymbol{f}_t^{\text{o}}\right)$ concatenates robot proprioception $\boldsymbol{s}_t^{\text{prop}}$, the object pose $\boldsymbol{s}_t^{\text{o}}$, and an object latent $\boldsymbol{f}^{\text{o}}$. The latent is produced by a pre-trained point-cloud encoder E_{pc} applied to the object point cloud $\mathcal{P}$, i.e., $\boldsymbol{f}^{\text{o}} = E_{\text{pc}}(\mathcal{P})$. The reference state is defined as $\hat{\boldsymbol{s}}_t^{\text{track}} = \left(\hat{\boldsymbol{s}}_t^{\text{h}}, \hat{\boldsymbol{s}}_t^{\text{o}}\right)$, where $\hat{\boldsymbol{s}}_t^{\text{h}}$ and $\hat{\boldsymbol{s}}_t^{\text{o}}$ denote the reference hand and object states, respectively. We include not only the current reference $\left(\hat{\boldsymbol{s}}_t^{\text{h}}, \hat{\boldsymbol{s}}_t^{\text{o}}\right)$ but also future references $\left(\hat{\boldsymbol{s}}_{t+k}^{\text{h}}, \hat{\boldsymbol{s}}_{t+k}^{\text{o}}\right)$ as observations, with $k \in \{1, 2, 4, 12\}$. All terms are concatenated and treated as the tracking reference state: $\hat{\boldsymbol{s}}_t^{\text{track}} = \text{concat}_{k\in\{0,1,2,4,12\}}\left(\hat{\boldsymbol{s}}_{t+k}^{\text{h}}, \hat{\boldsymbol{s}}_{t+k}^{\text{o}}\right)$. For the action space, the tracking policy π^{track} takes the observation $\boldsymbol{o}_t^{\text{track}}$ as input and outputs an action $\boldsymbol{a}_t \in \mathbb{R}^{22}$ consisting of 22-D relative joint position changes for the hand and arm revolute joints. The reward function is designed to align the transited hand and object states with their respective references and to encourage hand–object proximity [44]. Formally, it is defined as:

$$\begin{aligned} r_t = \omega_1 \|\boldsymbol{q}_t^{\text{h}} - \hat{\boldsymbol{q}}_t^{\text{h}}\|_2^2 + \omega_2 \|\boldsymbol{p}_t^{\text{h}} - \hat{\boldsymbol{p}}_t^{\text{h}}\|_2^2 \\ + \omega_3 \|\boldsymbol{p}_t^{\text{o}} - \hat{\boldsymbol{p}}_t^{\text{o}}\|_2^2 + \omega_4 \|\boldsymbol{p}_t^{\text{h}} - \boldsymbol{p}_t^{\text{o}}\|_2^2 \end{aligned} \tag{1}$$

In Eq. (1), $\|\boldsymbol{q}_t^{\text{h}} - \hat{\boldsymbol{q}}_t^{\text{h}}\|_2^2$ is the joint-position tracking cost for the robot arm–hand chain, where $\boldsymbol{q}_t^{\text{h}}$ and $\hat{\boldsymbol{q}}_t^{\text{h}}$ are the current and reference joint positions, respectively; $\|\boldsymbol{p}_t^{\text{h}} - \hat{\boldsymbol{p}}_t^{\text{h}}\|_2^2$ is the rigid-body tracking cost for the fingers and wrist, where $\boldsymbol{p}_t^{\text{h}}$ and $\hat{\boldsymbol{p}}_t^{\text{h}}$ are the current and reference rigid-body poses; $\|\boldsymbol{p}_t^{\text{o}} - \hat{\boldsymbol{p}}_t^{\text{o}}\|_2^2$ is the object-pose tracking cost, where $\boldsymbol{p}_t^{\text{o}}$ and $\hat{\boldsymbol{p}}_t^{\text{o}}$ are the current and reference object poses; and $\|\boldsymbol{p}_t^{\text{h}} - \boldsymbol{p}_t^{\text{o}}\|_2^2$ is a proximity term that encourages a small hand–object distance. The coefficients $\omega_1, \omega_2, \omega_3, \omega_4$ weight the corresponding terms and are chosen to balance the reward. Please see the supplementary material for details on the observation and action spaces and the reward.

Generalist Distillation. To distill knowledge from multiple specialists into a single general policy $\pi_{\text{general}}^{\text{track}}$, we first collect a set of specialist policies $\{\pi_i^{\text{track}}\}_{i=1}^N$. For each annotated reference trajectory and its generated semantically equivalent trajectories, we train a specialized specialist π_i^{track} to track them. This specialization constrains each specialist to a narrow, semantically consistent behavior, thereby reducing the RL exploration space and accelerating training. Given $\{\pi_i^{\text{track}}\}_{i=1}^N$, we construct a dataset D of successful trajectories generated by the specialists, $D = \bigcup_{i=1}^N \{\tau_m^{(i)}\}_{m=1}^{M_i}$, where each $\tau_m^{(i)}$ is a successful trajectory sampled from π_i^{track}. We then train the general policy $\pi_{\text{general}}^{\text{track}}$ via Behavior Cloning (BC) [22, 49]. Although DAgger-based distillation has proven effective in previous work with a small number of specialists, it requires multiple rounds of interaction with the environment, where the generalist queries specialists for action labels at each visited state. However, with a large set of specialists in our case, this results in substantial increases in both training time and computational cost, making the offline BC approach more efficient in our setting. The observation space of the general policy is the same as that of the specialist policy, $\boldsymbol{o}_t^{\text{track}}$, enabling direct real-world deployment without oracle states.

3.2. Tracking Modulator

The general tracking policy $\pi_{\text{general}}^{\text{track}}$ is trained using a large set of specialist tracking policies via BC. Next, we organize the controller as a hierarchical policy: an RL-based modulator π^{modulate} is placed before the tracker π^{track}. It conditions the tracker in three ways (detailed below) to enhance gener-

alization and reliability.

Reference Modulation. Because the tracking controller is imperfect—long-horizon tracking cannot perfectly follow the reference—errors accumulate over time, causing drift and out-of-distribution (OOD) states. To mitigate this, we exploit the tracker's short-horizon accuracy and dynamically advance the target by using a near-future hand reference $\hat{\boldsymbol{s}}_{t+k}$ with $k \in \{1, 2, 4, 12\}$. Specifically, at each step t, we update the near-future reference with a unified rule, $\hat{\boldsymbol{s}}^x_{t+k} \leftarrow \hat{\boldsymbol{s}}^x_{t+k} + \lambda_x \boldsymbol{a}^{\text{ref},x}_{t+k}$ for $k \in \{1, 2, 4, 12\}$ and $x \in \{\text{h}, \text{o}\}$ ($x = \text{h}$: hand; $x = \text{o}$: object), where $\boldsymbol{a}^{\text{ref},x}_{t+k}$ is the modulator action used to modulate the reference and λ_x controls the adjustment magnitude to keep the modified reference close to the original trajectory. By continuously updating the reference at each timestep t, the modulator leverages the tracker's strong short-horizon accuracy, thereby canceling accumulated error and reducing long-horizon drift and overall tracking error.

Object Latent Modulation. For the general tracker, the observation is defined as $\boldsymbol{o}^{\text{track}}_t = \big(\boldsymbol{s}^{\text{track}}_t, \hat{\boldsymbol{s}}^{\text{track}}_t\big)$, where the current state $\boldsymbol{s}^{\text{track}}_t = \big(\boldsymbol{s}^{\text{prop}}_t, \boldsymbol{s}^{\text{o}}_t, \boldsymbol{f}^{\text{o}}\big)$ concatenates robot proprioception $\boldsymbol{s}^{\text{prop}}_t$, the estimated object pose $\boldsymbol{s}^{\text{o}}_t$, and the object latent $\boldsymbol{f}^{\text{o}}$. During distillation, we log $(\boldsymbol{o}^{\text{track}}_t, \boldsymbol{a}_t)$ pairs from each specialist; with $\boldsymbol{f}^{\text{o}}$ explicitly embedded in $\boldsymbol{o}^{\text{track}}_t$, these data instantiate a skill library of state–action primitives conditioned on the object–latent subspace. At inference time, the general tracker computes a fixed $\boldsymbol{f}^{\text{o}}$ at task start and keeps it constant throughout execution, acting as $\boldsymbol{a}_t = \pi_{\text{track}}\big(\boldsymbol{s}^{\text{prop}}_t, \boldsymbol{s}^{\text{o}}_t, \boldsymbol{f}^{\text{o}}, \hat{\boldsymbol{s}}^{\text{track}}_t\big)$. Our modulator relaxes this restriction by producing a modulated latent $\tilde{\boldsymbol{f}}^{\text{o}}_t$ and forming the modulated observation $\tilde{\boldsymbol{o}}^{\text{track}}_t = \big(\tilde{\boldsymbol{s}}^{\text{track}}_t, \hat{\boldsymbol{s}}^{\text{track}}_t\big)$ with $\tilde{\boldsymbol{s}}^{\text{track}}_t = \big(\boldsymbol{s}^{\text{prop}}_t, \boldsymbol{s}^{\text{o}}_t, \tilde{\boldsymbol{f}}^{\text{o}}_t\big)$. Although trained with BC, the general tracker learns a continuous mapping from $(\boldsymbol{o}^{\text{track}}_t, \boldsymbol{f}^{\text{o}}_t)$ to $\boldsymbol{a}_t$. As $\boldsymbol{f}^{\text{o}}_t$ changes, the general tracker's behavior moves smoothly between specialists, allowing the modulator's object latent modulation to recruit the most suitable primitives at runtime. While the skill library is built from a finite set of object latents $\{\boldsymbol{f}^{\text{o}}_i\}_{i=1}^M$, the modulator treats $\tilde{\boldsymbol{f}}^{\text{o}}_t \in \mathbb{R}^{64}$ as a continuous control variable, effectively turning a discrete ability menu into a continuous manifold that can interpolate between and switch across object anchors.

Positional Target Modulation. We apply a small, bounded action space correction to absorb execution-level discrepancies that the tracker cannot capture alone. Given the tracker's command $\boldsymbol{a}_t = \pi_{\text{track}}(\boldsymbol{o}^{\text{track}}_t)$, the modulator outputs a residual positional target $\Delta\boldsymbol{a}_t$, and the final command is $\boldsymbol{a}'_t = \boldsymbol{a}_t + \Delta\boldsymbol{a}_t$. The residual targets fast, local, high-frequency nonidealities, providing on-the-fly refinement while keeping the tracker in charge. Intuitively, $\boldsymbol{a}_t$ carries the low-frequency structure of the skill learned during distillation, whereas $\Delta\boldsymbol{a}_t$ supplies a small and smooth compensation that improves precision and robustness.

3.3. Sim-to-Real Transfer

To reduce the sim-to-real gap, we perform system identification to tune the simulator's dynamics parameters to best match the real-world system, and apply domain randomization in simulation to span real-world physics variability, thereby improving robustness and transfer.

System Identification. The dynamics parameters we identify in simulation are the joint stiffness P and joint damping D. We first collect real-world joint state–action trajectories by actuating the real robot with sinusoidal action commands at multiple frequencies and amplitudes. After collecting real-world data, we use the black-box optimizer CMA-ES [15] to identify joint stiffness P and joint damping D that best match the real robot's motion. We estimate $(P^\star, D^\star)$ by minimizing the discrepancy between simulated and real joint states, with $L = \sum_{m=1}^{M} \sum_{t=1}^{N} \|\boldsymbol{q}^s_{m,t} - \boldsymbol{q}^r_{m,t}\|_2^2$ and $(P^\star, D^\star) = \arg\min_{P,D} L$, where M is the number of trajectories and N is the length of the trajectory. Please see the supplementary material for additional details.

Domain Randomization. Based on the system identification results, we apply small randomization to robot dynamics parameters (link masses, joint friction, joint stiffness, joint damping) and large randomization to object dynamics (mass, friction, restitution). We also inject Gaussian noise into the robot's proprioceptive states and action commands. In addition, we randomize the object's initial pose relative to the object's initial reference state $\hat{\boldsymbol{s}}^{\text{o}}_0$. Finally, we apply high-friction tape to the fingertips and the object, as we empirically observe that higher friction yields higher success rates and less failures.

4. Experiments

We first describe the experimental setup (Sec. 4.1). We then compare our approach with SOTA baselines, demonstrating its generalization and effectiveness (Sec. 4.2). Next, we conduct ablations of the modulator, isolating the effects of modulating the reference and the object latent, and further analyze how the amount of training data impacts generalization (Sec. 4.3). Finally, we present closed-loop zero-shot sim-to-real results on real hardware (Sec. 4.4).

4.1. Experiment Setup

Datasets. As described in Sec. 3.1, we use DiffH2O [9] to generate human–object motion references. Leveraging the dataset's text annotations, we filter the 1,333 annotated instances to retain only sequences that (i) involve single-hand (predominantly right-hand) manipulation with no bimanual interaction, (ii) exhibit no hand–robot-arm collisions, and (iii) keep the hand pose within the manipulator's reachable workspace at every timestep; this yields 505 reference sequences. We then augment each retained sequence by gen-

Dataset	Method	T_o (cm, ↓)	R_o (rad, ↓)	T_h (cm, ↓)	R_h (rad, ↓)	E_{finger} (rad, ↓)	Succ. (%, ↑)
Unseen Trajectory	ObjDex	4.95	0.6950	*8.02*	0.6296	0.6872	51.72/44.59
	DexTrack	6.93	0.7046	9.17	0.5742	0.2823	76.13/73.89
	Vanilla RL (PPO)	12.10	0.8992	13.30	0.9142	0.3595	43.03/53.94
	Ours (General Tracker)	11.57	0.7607	10.33	0.5501	0.2791	62.20/65.01
	Ours (General Tracker+R)	10.06	0.7445	9.76	*0.5408*	*0.2702*	67.09/68.14
	Ours (General Tracker+R+T)	*4.68*	**0.6272**	**7.99**	**0.5062**	**0.2413**	*84.45*/**80.42**
	Ours (General Tracker+R+O+T)	**4.49**	*0.6720*	8.77	0.5860	0.2714	**88.99**/*77.92*
Unseen Object	ObjDex	20.29	1.0556	11.24	0.9483	0.6833	28.16/37.82
	DexTrack	*18.06*	0.9043	10.32	0.6908	0.2953	*39.59*/47.37
	Vanilla RL (PPO)	23.20	**0.7830**	*9.63*	**0.5828**	**0.2533**	25.71/44.24
	Ours (General Tracker)	21.74	0.9562	11.62	*0.5964*	0.2845	24.08/42.52
	Ours (General Tracker+R)	22.26	0.9415	11.70	0.5995	0.2785	25.71/42.89
	Ours (General Tracker+R+T)	18.26	*0.8966*	11.72	0.6819	*0.2760*	37.55/*47.51*
	Ours (General Tracker+R+O+T)	**15.45**	0.9492	**9.21**	0.6414	0.2922	**46.12**/**53.78**

Table 1. Quantitative results. Best and second-best entries are highlighted in **bold red** and *italic blue*, respectively. Ablations are denoted as "Ours (General Tracker+R)", "Ours (General Tracker+R+T)", "Ours (General Tracker+R+O+T)", where R = reference modulation, O = object latent modulation, and T = positional target modulation. Success rate reported as Mean-error/Completion.

erating nine semantically equivalent texts and synthesizing a motion reference from each, expanding the set to 5,050 sequences. Because the generative process can produce invalid references, we reapply the same feasibility criteria to the augmented set, yielding 2,765 valid sequences spanning 50 objects. We randomly partition these sequences into 2,212 for training and 553 for testing—an unseen-trajectory split. At the object level, we additionally form an unseen-object split by using 45 objects (2,520 sequences) for training and holding out 5 objects (245 sequences) for testing.

Metrics. We evaluate tracking with six metrics: (1) T_o (Object Position Error), the per-frame mean Euclidean distance between the reference and the tracked object positions; (2) R_o (Object Rotation Error), the per-frame mean angular difference (degrees) between the reference and the tracked object orientations; (3) T_h (Wrist Position Error), the per-frame mean Euclidean distance between the reference and the tracked wrist positions; (4) R_h (Wrist Rotation Error), the per-frame mean angular difference (degrees) between the reference and the tracked wrist orientations; (5) E_{finger} (Finger Joint Error), the per-frame mean absolute joint-position error across finger DoFs; and (6) Success Rate (two variants): Mean-error—the fraction of trajectories whose mean errors satisfy $0.5\,T_o + 0.5\,T_h \le 10\,\text{cm}$, $0.5\,R_o + 0.5\,R_h \le 60^\circ$ and $E_{finger} \le 40^\circ$; and Completion—for each trajectory, the fraction of frames t with $0.5\,T_o(t)+0.5\,T_h(t) \le 10\,\text{cm}$, $0.5\,R_o(t)+0.5\,R_h(t) \le 60^\circ$ and $E_{finger}(t) \le 40^\circ$ (the average over trajectories).

Baselines. We compare our approach with three baselines. (1) DexTrack [21]: a single generalist neural tracker distilled from many human–object demonstrations, trained with reinforcement learning and imitation learning (IL). Given the current state and a fixed reference, it outputs tracking actions directly, with no modulator in the loop. Since DexTrack's homotopy-based demo mining and IL+RL training code are unavailable, we re-implemented the IL+RL scheme described in the paper: we execute the teacher on each reference to record its actions, and add a supervised imitation term to the actor's RL objective as a regularizer. Homotopy-based mining is not included in our reproduction. (2) ObjDex [7]: a hierarchical planner-over-controller method learned from human hand motion data. A high-level, object-centric trajectory generator (planner) synthesizes wrist trajectories conditioned on the desired object goal, and a low-level RL controller tracks these trajectories with a dexterous hand. Since the high-level planner code is unavailable, we re-implemented the planner following the paper's description and adapted the released RL controller and public configurations to our setting, retraining all components under our experimental protocol. (3) Vanilla RL (PPO): a policy trained with PPO [36] instead of behavior cloning—using the same reference corpus and task reward—and with no modulator in the loop.

4.2. Generalizable Tracking via Adaptation

We conduct experiments in Isaac Gym [27]. Table 1 shows that AdaDexTrack attains the highest success rates on both unseen-trajectory and unseen-object sets, consistently surpassing all baselines. Compared to a monolithic tracker (DexTrack), which treats the reference as fixed and cannot adapt when synthesis/retargeting noise or embodiment mismatch pushes execution off-script, our in-loop modulator continually adjusts the reference, object latent, and applies a small, bounded positional target correction—preventing long-horizon error accumulation. Compared to a loosely coupled planner-over-controller (ObjDex), which corrects

mainly in action/wrist space and leaves the reference and object encoding outside feedback, AdaDexTrack adapts at all three levels and remains tightly aligned with the tracker's objective, yielding more stable contacts, smoother phase transitions, and better disturbance recovery. Ablations confirm that adding modulation channels improves both mean error and completion, indicating that the gains come from adaptation rather than model scale alone.

4.3. Ablation Experiments

Effect of Modulator Components. We ablate the modulator across its three components reference (R), object latent (O), and positional target (T). We evaluate ablations per split using three variants of our model: Ours (General Tracker+R), Ours (General Tracker+R+T), and Ours (General Tracker+R+O+T). We report results on both the unseen-trajectory and unseen-object sets. As shown in Table 1, on both splits, increasing the number of modulator components yields consistent improvements in tracking performance.

To further demonstrate the effectiveness of each component, we present qualitative results. In the reference modulation (General Tracker+R) setting, Fig. 4 shows that when the tracker drifts, the reference modulator detects the deviation and updates the reference; a clear signature is the updated reference intersecting the executed path in the x–y plane, creating a low-error rejoin point that pulls the policy back onto a feasible trajectory and stabilizes long-horizon tracking. This illustrates the power of reference modulation: by continuously adjusting the tracker's reference, it prevents the error accumulation inherent to a fixed reference and delivers more robust tracking. In the object latent modulation (General Tracker+O) setting, Fig. 5 shows a simple yet strong effect of the object latent modulation: the object latent acts like a "knob" for hand openness—a larger latent makes the hand more open, a smaller latent makes it tighter. The modulator uses this "knob" phase by phase: during reach, it prefers larger latents to keep the fingers open and avoid bumping the object; during grasp, it switches to smaller latents to close the hand and secure contact. This phase-aware selection shows that the object latent is a meaningful, semantically aligned handle for invoking and composing skills from the tracker's library, enabling safer approach and more stable grasping—and improving tracking—without changing the reference or issuing large action corrections.

Data-Scaling Analysis. To study the effect of data scale, we vary only the training set size by downsampling the full dataset (1.00×) to a range of log-spaced fractions and train a separate model from scratch for each scale (each scale is the fraction of the original set used). We evaluate on the unseen-trajectory split (Fig. 6a) and the unseen-object split (Fig. 6b). In both cases, performance increases steadily with

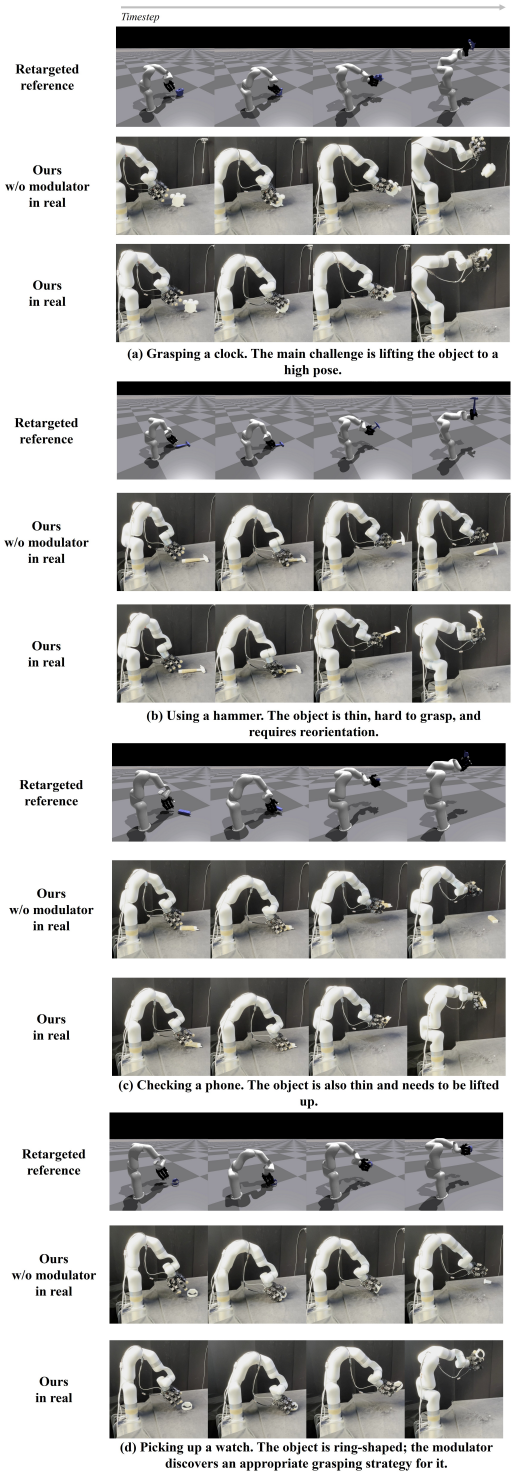


Figure 3. **Qualitative results.** Panels (a–d): given a noisy text-generated reference retargeted to the robot, **AdaDexTrack** successfully tracks the long-horizon interaction, while the tracker-only baseline fails. This is because the in-loop modulator continuously adapts the reference, object latent, and positional target, mitigating sim-to-real noise and preventing error accumulation, whereas the tracker-only baseline accumulates errors over time.

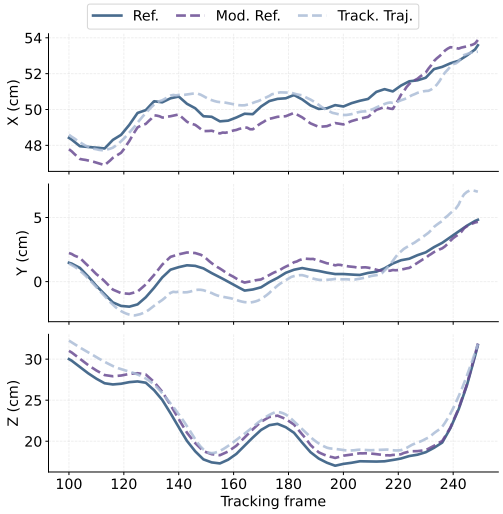


Figure 4. Reference modulation (General Tracker+R). **Ref.**: original object reference; **Mod. Ref.**: reference after modulation by the reference modulator; **Track. Traj.**: executed object trajectory.

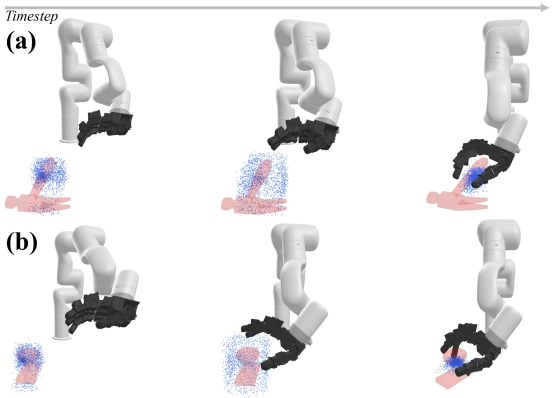


Figure 5. Object latent modulation (General Tracker+O). Red: actual object and its pose. Blue: point cloud decoded from the modulated object latent and its pose. (a) Hammer; (b) Stamp.

data size and surpasses the tracker-only baseline.

4.4. Sim-to-Real Experiments

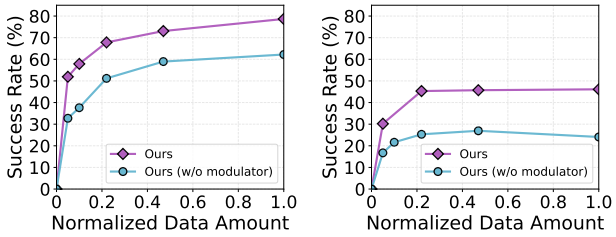
Sim-to-Real Setup. We deploy the policy trained entirely in simulation for closed-loop control in the real world (zero-shot). The hardware mirrors the simulation: an XArm6 [42] paired with a LEAP Hand [37]. Object pose is estimated by FoundationPose [45], and hand–eye calibration is performed via the Tsai–Lenz method [41]. Because FoundationPose exhibits rotational ambiguity on axially symmetric objects (e.g., bottles, where the estimate can freely spin around the symmetry axis), we exclude those objects to avoid confounding perception errors. We evaluate on two real-world sets: (i) an unseen-trajectory set with 60 reference trajectories from the unseen-trajectory split, covering 15 objects; and (ii) an unseen-object set with 20 reference trajectories from the unseen-object split, covering 2 objects.

Methods	Unseen Traj	Unseen Obj
Ours (w/o modulator)	26.63%	22.27%
Ours	**52.36%**	**36.71%**

Table 2. Zero-shot sim-to-real experiments. Completion rate (%).

For real-world evaluation, we report the completion rate, i.e., the Completion variant of Success Rate in Sec. 4.1, using identical thresholds and averaged over trajectories.

Results. As reported in Table 2, our approach outperforms the tracker-only baseline in zero-shot real-world tests on both the unseen-trajectory and unseen-object sets. Qualitative results in Figs. 1 and 3 illustrate the benefits of the modulator compared with the tracker-only baseline. The gains come from the in-the-loop modulator, which performs reference modulation, object latent modulation, and small, bounded positional target corrections—counteracting perception noise, calibration drift, latency, and other sim-to-real mismatches before they accumulate. By contrast, the baseline keeps the reference outside the loop and lacks corrections at the reference, representation, and actuation levels, so errors compound over long horizons and lead to failures. Qualitative results in Fig. 6 show more stable contacts, smoother transitions, and better recovery from disturbances.



(a) Unseen Trjectories. (b) Unseen Objects.

Figure 6. Effect of scaling the amount of reference data.

5. Conclusions and Limitations

We introduced **AdaDexTrack**, a modulator-in-the-loop framework for language-guided HOI tracking. The tracker serves as the skill carrier distilled from many experts, while the modulator provides three tightly coupled interfaces—reference modulation, object-latent modulation, and positional target modulation. Trained under the same task objective as the tracker, these interfaces compose skills on demand, close the loop around the reference, and suppress long-horizon drift. **AdaDexTrack** surpasses baselines on unseen trajectories and unseen objects, improves with dataset scale, and achieves zero-shot sim-to-real transfer.

Limitations. In the real-world setting, our system relies on an explicit 6-DoF pose estimator and thus struggles with symmetric objects, where pose is ambiguous. A pose-free route is to drive tracking and modulation directly from point clouds by learning symmetry-aware, SE(3)-equivariant embeddings for reference matching and latent modulation.

References

[1] AR Code. AR Code. https://ar-code.com/, 2022. Accessed: 2024-09-28. 5

[2] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, et al. Hot3d: Hand and object tracking in 3d from egocentric multi-view videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7061–7071, 2025. 2

[3] Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. Surprisingly robust in-hand manipulation: An empirical study. *arXiv preprint arXiv:2201.11503*, 2022. 3

[4] Junuk Cha, Jihyeon Kim, Jae Shin Yoon, and Seungryul Baek. Text2hoi: Text-guided 3d motion generation for hand-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1577–1585, 2024. 2

[5] Qiguang Chen, Lucas Wan, and Ya-Jun Pan. Object recognition and localization for pick-and-place task using difference-based dynamic movement primitives. *IFAC-PapersOnLine*, 56(2):10004–10009, 2023. 2

[6] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023. 3

[7] Yuanpei Chen, Chen Wang, Yaodong Yang, and C Karen Liu. Object-centric dexterous manipulation from human motion data. *arXiv preprint arXiv:2411.04005*, 2024. 3, 6

[8] Zerui Chen, Shizhe Chen, Etienne Arlaud, Ivan Laptev, and Cordelia Schmid. Vividex: Learning vision-based dexterous manipulation from human videos. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3336–3343. IEEE, 2025. 2

[9] Sammy Christen, Shreyas Hampali, Fadime Sener, Edoardo Remelli, Tomas Hodan, Eric Sauser, Shugao Ma, and Bugra Tekin. Diffh2o: Diffusion-based synthesis of hand-object interactions from textual descriptions. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 2, 3, 5, 1

[10] Sisi Dai, Wenhao Li, Haowen Sun, Haibin Huang, Chongyang Ma, Hui Huang, Kai Xu, and Ruizhen Hu. Interfusion: Text-driven generation of 3d human-object interaction. In *European Conference on Computer Vision*, pages 18–35. Springer, 2024. 2

[11] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan Qi, and Xiaolong Wang. Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning. *arXiv preprint arXiv:2407.03162*, 2024. 2

[12] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12943–12954, 2023. 2

[13] Rao Fu, Dingxi Zhang, Alex Jiang, Wanjia Fu, Austin Funk, Daniel Ritchie, and Srinath Sridhar. Gigahands: A massive annotated dataset of bimanual hand activities. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17461–17474, 2025. 2

[14] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021. 3

[15] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003. 5

[16] Jhen Hsieh, Kuan-Hsun Tu, Kuo-Han Hung, and Tsung-Wei Ke. Dexman: Learning bimanual dexterous manipulation from human and generated videos. *arXiv preprint arXiv:2510.08475*, 2025. 2

[17] Gagan Khandate, Cameron Paul Mehlman, Xingsheng Wei, and Matei Ciocarlie. Dexterous in-hand manipulation by guiding exploration with simple sub-skill controllers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6551–6557. IEEE, 2024. 3

[18] Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6991–7003, 2025. 2

[19] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5637–5643. IEEE, 2025. 3

[20] Yuhao Lin, Yi-Lin Wei, Haoran Liao, Mu Lin, Chengyi Xing, Hao Li, Dandan Zhang, Mark Cutkosky, and Wei-Shi Zheng. Typetele: Releasing dexterity in teleoperation by dexterous manipulation types. *arXiv preprint arXiv:2507.01857*, 2025. 2

[21] Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 4, 6, 1

[22] Xueyi Liu, He Wang, and Li Yi. Dexndm: Closing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model. *arXiv preprint arXiv:2510.08556*, 2025. 4

[23] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022. 2

[24] Yun Liu, Haolin Yang, Xu Si, Ling Liu, Zipeng Li, Yuxiang Zhang, Yebin Liu, and Li Yi. Taco: Benchmarking generalizable bimanual tool-action-object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21740–21751, 2024. 2

[25] Yumeng Liu, Yaxun Yang, Youzhuo Wang, Xiaofei Wu, Jiamin Wang, Yichen Yao, Sören Schwertfeger, Sibei Yang, Wenping Wang, Jingyi Yu, et al. Realdex: Towards human-like grasping for robotic dexterous hand. *arXiv preprint arXiv:2402.13853*, 2024. 2

[26] Zhenyu Lu, Ning Wang, and Chenguang Yang. A dynamic movement primitives-based tool use skill learning and transfer framework for robot manipulation. *IEEE Transactions on Automation Science and Engineering*, 22:1748–1763, 2024. 2

[27] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 6, 3

[28] Andrew S Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M Dollar. Complex in-hand manipulation via compliance-enabled finger gaiting and multi-modal planning. *IEEE Robotics and Automation Letters*, 7(2):4821–4828, 2022. 3

[29] OpenAI. Chatgpt-5. https://chat.openai.com/, 2025. Large language model, accessed October 2025. 3, 1

[30] Xiaogang Peng, Yiming Xie, Zizhao Wu, Varun Jampani, Deqing Sun, and Huaizu Jiang. Hoi-diff: Text-driven synthesis of 3d human-object interactions using diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2878–2888, 2025. 2

[31] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7(4):10873–10881, 2022. 2

[32] 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*, pages 570–587. Springer, 2022. 2

[33] 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. 2

[34] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018. 2

[35] Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13):1133–1184, 2023. 2

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4, 6

[37] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023. 8

[38] Liangzhi Shi, Yulin Liu, Lingqi Zeng, Bo Ai, Zhengdong Hong, and Hao Su. Learning adaptive dexterous grasping from single demonstrations. *arXiv preprint arXiv:2503.20208*, 2025. 2

[39] Zilin Si, Kevin Lee Zhang, Zeynep Temel, and Oliver Kroemer. Tilde: Teleoperation for dexterous in-hand manipulation learning with a deltahand. *arXiv preprint arXiv:2405.18804*, 2024. 2

[40] Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. Grab: A dataset of whole-body human grasping of objects. In *European conference on computer vision*, pages 581–600. Springer, 2020. 2

[41] Roger Y Tsai, Reimar K Lenz, et al. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3): 345–358, 1989. 8

[42] *xArm User Manual*. UFactory, 2023. Accessed 2025-11-07. 8

[43] Karl Van Wyk, Mandy Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, et al. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2): 3202–3209, 2022. 2

[44] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023. 4

[45] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024. 8, 5

[46] Qianyang Wu, Ye Shi, Xiaoshui Huang, Jingyi Yu, Lan Xu, and Jingya Wang. Thor: Text to human-object interaction diffusion via relation intervention. *arXiv preprint arXiv:2403.11208*, 2024. 2

[47] Mandy Xie, Ankur Handa, Stephen Tyree, Dieter Fox, Harish Ravichandar, Nathan D Ratliff, and Karl Van Wyk. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *Conference on Robot Learning*, pages 1355–1367. PMLR, 2023. 2

[48] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 6

[49] Ying Yuan, Haichuan Che, Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Kang-Won Lee, Yi Wu, Soo-Chul Lim, and Xiaolong Wang. Robot synesthesia: In-hand manipulation with visuotactile sensing. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6558–6565. IEEE, 2024. 4

[50] Zhecheng Yuan, Tianming Wei, Langzhe Gu, Pu Hua, Tianhai Liang, Yuanpei Chen, and Huazhe Xu. Hermes: Human-to-robot embodied learning from multi-source motion data for mobile dexterous manipulation. *arXiv preprint arXiv:2508.20085*, 2025. 2

[51] 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*, pages 445–456, 2024. 2

[52] Di Zhang, Chengbo Yuan, Chuan Wen, Hai Zhang, Junqiao Zhao, and Yang Gao. Kinedex: Learning tactile-informed visuomotor policies via kinesthetic teaching for dexterous manipulation. *arXiv preprint arXiv:2505.01974*, 2025. 2

[53] Han Zhang, Songbo Hu, Zhecheng Yuan, and Huazhe Xu. Doglove: Dexterous manipulation with a low-cost open-source haptic force feedback glove. *arXiv preprint arXiv:2502.07730*, 2025. 2

[54] Yan Zhang, Teng Xue, Amirreza Razmjoo, and Sylvain Calinon. Logic dynamic movement primitives for long-horizon manipulation tasks in dynamic environments. *CoRR*, 2024. 2

AdaDexTrack: Dynamic Modulation for Adaptive and Generalizable Dexterous Manipulation Tracking

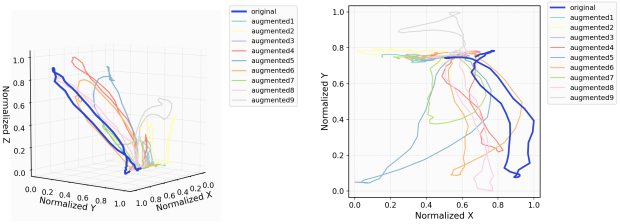
Supplementary Material

Contents

A. Additional Details on Method

A.1. Data Preprocessing

Language-Guided Trajectory Augmentation. We build our language-guided HOI trajectory dataset on top of the text-annotated sequences provided by DiffH2O [9]. Starting from the full set of demonstrations, we retain only sequences that (i) involve single-hand (predominantly right-hand) manipulation without bimanual interaction, (ii) exhibit no hand–robot-arm collisions, and (iii) keep the hand pose within the manipulator's reachable workspace at every timestep, which yields 505 reference trajectories. For each retained trajectory, we then perform paraphrase-based language augmentation. Concretely, we use GPT-5 [29] to generate nine semantically equivalent descriptions that follow DiffH2O's native annotation format and preserve the original task intent. We feed these paraphrased descriptions back into the DiffH2O text-to-motion model to synthesize additional HOI trajectories conditioned on the same object geometry. Fig. 7 shows a qualitative example of our augmentation results, where a single original trajectory is expanded into ten variants. Applying the same feasibility filters to the augmented set results in 2,765 valid text–trajectory pairs spanning 50 objects.



(a) 3D view (b) Top-down XY view

Figure 7. Language-guided HOI reference augmentation in 3D and top-down XY views.

Kinematic Retargeting. Our kinematic retargeting procedure is similar to DexTrack [21]. Given a human hand–object interaction trajectory with MANO hand poses $\{\mathbf{H}_n^{\text{human}}\}$ and object poses $\{\mathbf{O}_n\}$, we retarget it to the robot hand by matching keypoints between the MANO hand and the articulated robot hand. Let $\mathbf{K}_n^{\text{human}}$ be the MANO keypoints at timestep n and $\mathbf{K}_n(\boldsymbol{\theta}_n)$ the corresponding robot hand keypoints obtained via forward kinematics from joint angles $\boldsymbol{\theta}_n$. At each timestep, we solve a least-squares problem $\min_{\boldsymbol{\theta}_n} \|\mathbf{K}_n(\boldsymbol{\theta}_n) - \mathbf{K}_n^{\text{human}}\|$, yielding a sequence of robot hand joint poses $\{\boldsymbol{\theta}_n\}$ that serves as the kinematic reference for our controller.

Object Latent Encoding. We encode object geometry with a latent code extracted by a Chamfer-trained point-cloud autoencoder. Given an object point cloud P, the encoder produces the object latent $f_o = E_{pc}(P)$, which is used as the object representation in our tracker and modulated online during execution.

A.2. Sim-to-Real Transfer

System Identification. We perform system identification on the robot arm and hand by driving each joint with a family of sinusoidal position commands and recording the resulting joint trajectories. Specifically, we apply commands of the form $A(t) = a\sin(2\pi ft)$, where both the frequency f and the amplitude a are swept over a range, and a is chosen such that the commanded motion stays within the joint limits. We then replay the same command sequences in simulation and identify, for each joint independently, the stiffness and damping parameters by minimizing the discrepancy between the simulated and real joint trajectories. As shown in Fig. 8, the identified model produces response curves in simulation that closely match those of the real robot under identical control commands.

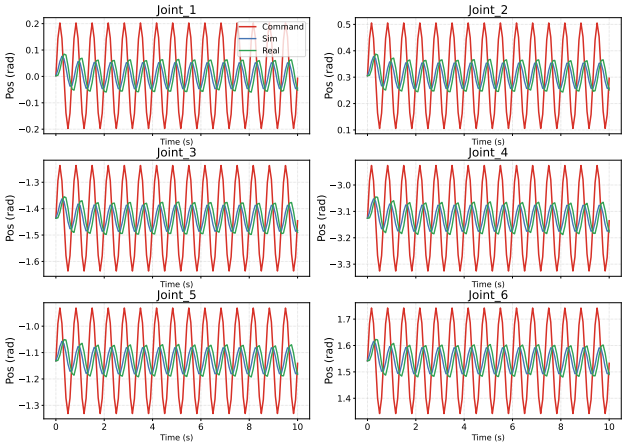


Figure 8. System identification of the arm joints. For each joint, we plot three traces: (1) the commanded input, (2) the simulated response using the identified dynamics model, and (3) the measured real response. The close agreement between (2) and (3) indicates that the identified parameters faithfully capture the joint dynamics.

Parameter	Type	Distribution	Initial Range
Robot			
Mass	Scaling	uniform	$[0.9, 1.1]$
Friction	Scaling	uniform	$[0.9, 1.1]$
Joint stiffness	Scaling	uniform	$[0.9, 1.1]$
Joint damping	Scaling	uniform	$[0.9, 1.1]$
Object			
Mass	Scaling	uniform	$[0.7, 1.3]$
Friction	Scaling	uniform	$[0.7, 1.3]$
Restitution	additive	uniform	$[0.0, 0.4]$
Scale	Scaling	uniform	$[0.95, 1.05]$
Initial Position (x,y)	Additive	Gaussian	$[0.0, 0.05]$
Initial Rotation (yaw)	Additive	Gaussian	$[0.0, 0.5]$
Observation			
Proprioception	Additive	Gaussian	$[0.0, 0.002]$
Action			
Correlated noise	Additive	Gaussian	$[0.0, 0.02]$
Uncorrelated noise	Additive	Gaussian	$[0.0, 0.05]$

Table 3. Domain randomization configuration.

Domain Randomization. To improve robustness and facilitate sim-to-real transfer, we apply domain randomization to both the system dynamics and the observations/actions during training. On the dynamics side, we randomly perturb the arm and hand parameters, including link masses, joint friction, joint stiffness, and joint damping, around their nominal values. For the manipulated objects, we use a larger range of randomization over physical parameters such as mass, friction coefficients, and restitution, and also randomize the initial object pose relative to the reference trajectory. On the sensing and control side, we inject Gaussian noise into the proprioceptive states and add small Gaussian noise to the action commands. The exact ranges used for domain randomization are summarized in Table 3.

A.3. Tracking Controller Training

A.3.1. Specialist Policy

Action Space. The control action at each timestep is defined in joint space. We use a 22-DoF arm–hand system with joint configuration $\mathbf{q} \in \mathbb{R}^{22}$. The action space $\mathcal{A} \subset \mathbb{R}^{22}$ consists of relative joint commands $\Delta\mathbf{q} \in \mathbb{R}^{22}$. Internally, we maintain a vector of joint position targets $\mathbf{q}^{\text{tar}}$. At every control step, the policy outputs $\Delta\mathbf{q}$ and we update the targets by $\mathbf{q}^{\text{tar}} \leftarrow \mathbf{q}^{\text{tar}} + \Delta\mathbf{q}$ (followed by clipping to the joint limits). The updated targets $\mathbf{q}^{\text{tar}}$ are then sent to the low-level joint position controllers as setpoints.

Observation Space. Table 4 details the observation space used for training the specialist policies.

Reward implementation details. In practice, we implement the reward in Eq. (1) as a shaped combination of tracking and proximity terms. The joint and hand-pose terms are realized by a tracking loss $\ell_{\text{track}}(t)$ that aggregates palm position and orientation errors, 22-DoF joint-space errors, and fingertip position errors with respect to their references; this loss enters the reward with a negative weight so that larger deviations from the reference are penalized more strongly. The hand–object proximity term is instantiated using the distances between the palm and the object, and between the fingertips and the object, which encourages the hand to move into and remain in the vicinity of the object. To avoid excessively large gradients, these distances are clipped to fixed upper bounds, and the fingertip penalty is activated only once the palm is sufficiently close to the object.

The object-pose term in Eq. (1) is implemented via a goal distance $d_{\text{goal}}(t)$ that combines translational and rotational discrepancies between the current and target object poses. We represent rotations as unit quaternions and compute the angular component of $d_{\text{goal}}(t)$ from the quaternion difference via the dot product between the normalized goal quaternion and the current quaternion. In addition, we include a sparse success bonus $r_{\text{bonus}}(t) = \mathbb{I}\big[d_{\text{palm}}(t) \le \tau_{\text{palm}},\, d_{\text{fing}}(t) \le \tau_{\text{fing}},\, d_{\text{goal}}(t) \le \tau_{\text{goal}}\big]\,(1 + 10\,d_{\text{goal}}(t))^{-1}$, where $\mathbb{I}[\cdot]$ is the indicator function. This bonus rewards states where the hand remains close to the object and the object pose is very close to the goal, and can be viewed as a shaped realization of the proximity component in Eq. (1).

To improve computational efficiency and obtain a smooth orientation error, we represent all orientations using unit quaternions and compute $d_R(\cdot)$ from the quaternion difference via the dot product between the normalized goal quaternion and the current quaternion.

Semantic Grouping of Specialists. We start from the text-annotated DiffH2O demonstrations and first apply a feasi-

bility filter (single-hand interaction, no collisions, reachable workspace), which leaves 505 valid human–object interaction sequences. On top of these 505 base demonstrations, we then perform paraphrase-based augmentation: for each retained sequence, we use GPT-5 to generate multiple semantically equivalent but textually different captions and feed them into DiffH2O to synthesize additional reference trajectories. This procedure yields at most ten references per base demonstration (one from the original caption and up to nine from paraphrased captions), i.e., 5050 candidates in total. We apply the same feasibility filter again and obtain 2,765 valid reference trajectories overall.

For specialist training, we group references by their underlying semantic intent: all trajectories that originate from the same base demonstration (and its paraphrased captions) are treated as one semantic group. Each such group defines a single specialist tracker π_i^{track}, which is trained jointly on all of its associated reference trajectories by running them in parallel in Isaac Gym. In this way, we obtain one specialist per semantic group, with each specialist responsible for a small cluster of semantically equivalent references rather than a single trajectory. This semantic grouping substantially reduces the number of specialist policies while still exposing each of them to multiple motion and language variants of the same underlying task.

Time consumption. Each specialist is trained in Isaac Gym [27] with 8,192 parallel environments on a single NVIDIA A10 GPU and converges in about 3 hours.

A.3.2. Generalist Policy

The observation space used for behavior cloning of the generalist policy is identical to that of the specialist policy, as summarized in Table 4.

Time consumption. The generalist policy is trained on eight NVIDIA A10 GPUs and converges in about two days.

A.4. Tracking Modulator Training.

During modulator training, we freeze the generalist policy and optimize only the modulator. The modulator uses the same observation space as in Table 4, and its reward is identical to those used in specialist training. As discussed in Sec.3.2, the modulator modifies both the inputs and outputs of the tracker: it modulates the tracking reference and the object latent in the observation, and applies a small residual modulation to the tracker's action.

Time consumption. The modulator policy is trained in Isaac Gym [27] with 8,192 parallel environments on a single NVIDIA A10 GPU and converges in about two days.

B. Simulation Experiments Details

B.1. Environment Setup

We use the Isaac Gym simulator with a simulation frequency of 60 Hz and a control frequency of 12 Hz. For each reference trajectory to be tracked, we define the tracking start frame as the first frame where the minimum distance between the hand mesh and the object center exceeds 20 cm. Starting from this frame, we take the subsequent 300 frames as the full trajectory segment used for training.

Index	Description
0 – 22	arm-hand joint positions
22 – 57	the palm pose and the poses of the four fingertips
57 – 64	object pose
64 – 71	distance between the current object pose and the object reference pose at the current timestep
71 – 99	object reference poses at future timesteps $\{1, 2, 4, 12\}$ relative to the current timestep
99 – 121	distance between the current and reference arm–hand joint positions at the current timestep
121 – 209	reference arm–hand joint positions at future timesteps $\{1, 2, 4, 12\}$ relative to the current timestep
209 – 273	object latent

Table 4. Observation space used for specialist policy training.

Index	Description
0 – 22	arm-hand joint positions
22 – 57	the palm pose and the poses of the four fingertips
57 – 64	object pose
64 – 71	distance between the current object pose and the object reference pose at the current timestep
71 – 141	object reference poses at future ten timesteps relative to the current timestep
141 – 147	distance between the current and reference arm joint positions at the current timestep
147 – 207	reference arm joint positions at future ten timesteps relative to the current timestep
207 – 271	object latent

Table 5. Observation space for the low-level controller in ObjDex.

B.2. Baseline Implementation

B.2.1. DexTrack

We implement a single, generalist neural tracker that takes the current robot/object state and a fixed reference as input and directly outputs actions—without any in-loop modula-

tion (no reference, object latent, or positional target modulation at test time). To ensure a fair comparison, we keep the observation/action spaces, reference format, rewards, training data, and evaluation protocol identical to those used for AdaDexTrack. The object latent is computed once at episode start and held fixed throughout execution, matching the fixed-reference, fixed-representation setting.

Imitation targets (teacher actions). To supply the IL term, we reuse the specialist tracking policies trained in Sec.3.1. For each training reference, we roll out its matched specialist in simulation and log synchronized tuples $(\mathbf{o}_t,\ \hat{\mathbf{s}}_t,\ \mathbf{a}_t^{\text{teacher}})$, where $\mathbf{o}_t$ is the policy observation, $\hat{\mathbf{s}}_t$ is the time-indexed HOI reference, and $\mathbf{a}_t^{\text{teacher}}$ is the specialist's action at time t. These teacher actions form the imitation dataset used by DexTrack's IL term.

Training objective (IL+RL). The combined IL+RL training procedure and the homotopy-based demonstration mining described in DexTrack are not publicly available. We therefore re-implement the IL+RL scheme (without homotopy mining): we train the policy with PPO on the task reward and add a time-indexed imitation regularizer that directly matches the policy's action at timestep t to the logged teacher action $\mathbf{a}_t^{\text{teacher}}$ for that same reference index. In other words, since tracking uses a time-indexed reference, the imitation target is keyed by t (no observation-based matching). The total objective is the standard PPO actor–critic loss plus a weighted imitation term.

B.2.2. ObjDex

We reproduce a hierarchical planner-over-controller baseline: a high-level, object-centric planner synthesizes wrist trajectories conditioned on the task goal, and a low-level RL controller tracks these trajectories with the dexterous hand. The two layers are trained with different objectives (generative trajectory modeling vs. RL) and communicate primarily via wrist-space references.

High-level planner. Since the original planner code is unavailable, we re-implement it as a Transformer-based conditional trajectory generator. The planner takes the object ID and the object reference trajectory as input and outputs a wrist reference trajectory for the low-level controller to track. We train it in a supervised fashion on pairs of object IDs and object reference trajectories, using the wrist reference trajectory as the target. At test time, given an object ID and its reference trajectory, the planner predicts the wrist reference. For unseen objects, following ObjDex, we condition the planner on a seen surrogate object ID from the same category and predict the wrist reference trajectory for the unseen object.

Low-level controller. We implement an RL tracker with the same action space as AdaDexTrack for fairness, but with a different observation design. Following ObjDex, the policy receives a 10-step lookahead reference containing only the object pose and arm joint positions for the next ten timesteps (see Table 5). Following ObjDex to mitigate embodiment gap, the reward removes fingertip tracking and hand joint position tracking terms, and retains only (i) wrist pose tracking and (ii) arm joint position tracking. At test time, the controller takes the current state and the planner's wrist trajectory and tracks it in closed loop; no reference replanning, object latent adjustment, or positional target correction is applied beyond this stack.

Data augmentation loop. Following ObjDex, once the low-level controller is trained, we execute it in simulation under the planner's guidance, collect successful motion trajectories, add them to the planner's training set, and fine-tune the high-level planner on the augmented data.

B.2.3. Vanilla RL (PPO)

We train a single PPO policy across all training references, using the same action space, observation space, and reward as in the specialist training (Sec. 3.1).

B.3. Qualitative Results

We provide qualitative comparisons with the baselines on the unseen trajectory set in Figure 13. Across tasks, AdaDexTrack achieves the most faithful hand–object tracking, while Vanilla RL (PPO) and DexTrack often drift or fail. ObjDex typically follows the object trajectory but, lacking hand-shape tracking, produces unnatural, non-humanlike hand motions.

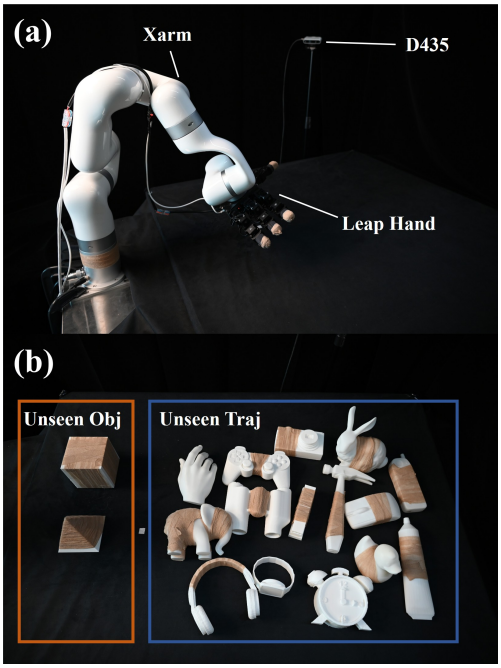


Figure 9. Real-world setup. (a) Hardware. (b) Test objects. The blue box marks objects used for the real-world unseen-trajectory set; the orange box marks objects used for the real-world unseen-object set.

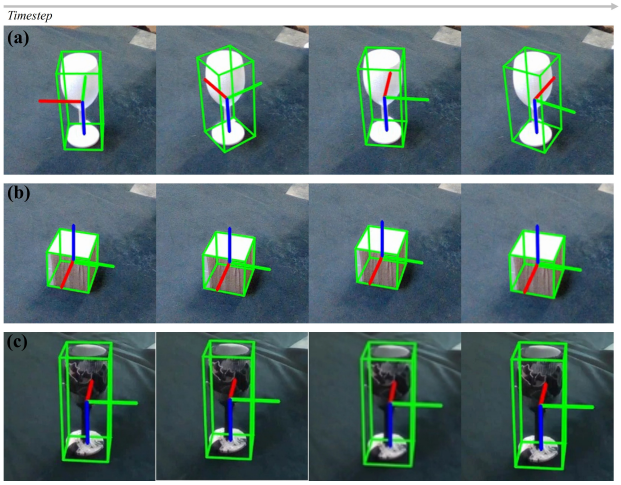


Figure 10. FoundationPose estimation behavior. (a) Wineglass: although the object is static, the estimate continuously spins around the blue symmetry axis due to axial symmetry. (b) Cube: despite discrete symmetry, the pose remains stable (no continuous spin). (c) Textured wineglass: asymmetric texture breaks the symmetry and yields a stable object frame.

C. Real-World Experiments Details

C.1. Environment Setup

Our real-world setup is shown in Fig. 11. We use an Intel RealSense D435 for RGB-D sensing and FoundationPose [45] for 6-DoF object pose estimation. In the main real-world evaluation, we exclude axially symmetric objects (e.g., wineglass) because FoundationPose exhibits continuous rotational ambiguity on them (Fig. 10a), which would confound the manipulation results. By contrast, objects with discrete symmetries (e.g., cube with 90° rotational invariance; Fig. 10b) do not induce continuous spin; edges or mild texture typically yield stable poses up to the object's symmetry group. Accordingly, we include discretely symmetric objects when the estimator is stable under this equivalence and exclude objects that exhibit free spinning about a symmetry axis. We further study axially symmetric objects with added asymmetric texture in Sec. C.3, where the texture breaks the visual symmetry and enables a stable object frame (Fig. 10c).

C.2. Metrics

In real-world experiments, we report Success Rate as the completion metric defined in Sec. 4.1, identical to the simulation setting. All thresholds match Sec. 4.1, and we report the average completion over trajectories.

C.3. Textured Axially Symmetric Objects

To study axially symmetric objects, we add asymmetric texture to break the rotational symmetry and obtain stable FoundationPose estimates (Fig. 10c). Fig. 11 shows the 8 textured axially symmetric objects used in this study. Since FoundationPose also requires a known object mesh, we obtain the object meshes using AR Code [1]. We evaluate these 8 objects in the real world, with 5 unseen trajectories per object. Table 6 reports the average completion rate over all trials. With stable pose estimates, our full method substantially outperforms the variant without the modulator, showing that the modulator remains effective on rotationally symmetric geometries.

Figure 11. Eight textured axially symmetric objects used in our real-world experiments.

Method	Completion (%, ↑)
Ours (w/o Modulator)	29.72%
Ours	**58.61%**

Table 6. Real-world results on textured axially symmetric objects.

C.4. Real-World Evaluation Without Object Tape

To address the concern that tape on the objects may artificially simplify grasping, we conduct an additional real-world evaluation after removing all tape from the objects. We keep the rest of the setup unchanged and re-run both the unseen-trajectory and unseen-object settings. Table 7 reports the resulting completion rates.

Removing the tape does not degrade performance; instead, it leads to higher success rates for both methods. We attribute this to the fact that excessive friction can hinder the small relative motions needed for grasp adjustment, whereas untaped objects allow more natural micro-sliding during contact. Our full method still substantially outperforms the variant without the modulator, indicating that the improvement does not rely on artificial adhesion, but on more effective handling of natural contact dynamics.

Method	Unseen Traj.	Unseen Obj.
Ours (w/o Modulator)	36.42%	18.75%
Ours	**65.71%**	**32.50%**

Table 7. Real-world results without grip tape on the objects.

C.5. Real-World Evaluation with Practical Object Geometry

To further assess the practicality of our method, we replace the ground-truth object geometry used in the main

experiments with inputs obtainable from lightweight perception tools. Specifically, we use InstantMesh [48] to reconstruct object geometry from a single RGB image, and use the resulting geometry for both policy inference and FoundationPose-based pose tracking. We evaluate this setting on the unseen-trajectory and unseen-object sets from Sec. 4.4, without object tape. As shown in Table 8, using reconstructed geometry yields performance comparable to ground-truth geometry, indicating that our method does not rely on carefully prepared CAD models.

We further consider a stricter setting where the object input is a segmented partial point cloud from a single-view depth observation. This setting is more challenging due to incomplete geometry and self-occlusion. Nevertheless, our method remains robust, achieving strong performance on both evaluation sets. These results support the practical applicability of our approach under realistic geometry acquisition pipelines.

Input	Method	Unseen Traj.	Unseen Obj.
GT Pointcloud	Ours	65.71%	32.50%
Recon. Pointcloud	Ours	59.28%	34.73%
Partial Pointcloud	Ours	49.46%	25.72%

Table 8. Real-world results under practical object-geometry inputs. GT Pointcloud uses ground-truth object geometry; Recon. Pointcloud is reconstructed from a single RGB image using InstantMesh [48]; Partial Pointcloud is obtained from segmented single-view depth.

C.6. Qualitative Results

Figure 14 shows additional qualitative comparisons on the unseen-trajectory set against the tracker-only baseline, highlighting AdaDexTrack's robustness.

D. Additional Analysis

D.1. Object Latent Smoothness

In our framework, object latent modulation is intended to act as a continuous interface for skill composition, rather than a discrete object identifier. The general tracker is conditioned on the object latent, and the modulator adjusts this latent online to recruit the most suitable behavior primitives for the current interaction phase. A key question, however, is whether such a mechanism is meaningful when the tracker is distilled from behavior cloning on a finite set of training objects. If the latent-to-action mapping were highly discontinuous, then interpolating in the latent space would not correspond to smooth interpolation between skills.

To examine this, we perform a Lipschitz-style perturbation analysis on the learned object latent. Specifically, we sample 200 reference observations from the unseen-trajectory set, and for each one we randomly select 5 timesteps, yielding 1,000 observations in total. For each observation, we perturb only the object latent by a small amount ϵ, while keeping the robot state, object pose, and tracking reference fixed. We then measure the change in the tracker output using

$$\rho = \frac{\|\Delta a\|_2}{\epsilon},$$

where Δa denotes the induced change in the predicted action.

As shown in Fig. 12, the response ratio ρ remains bounded across samples and perturbation directions. This indicates that the tracker responds smoothly to local changes in the object latent, rather than switching erratically between unrelated behaviors. Such local smoothness is precisely the property needed for object latent modulation to function as a continuous control variable: nearby latent codes induce nearby actions, making interpolation between latent anchors meaningful at execution time.

This observation is consistent with the qualitative behavior shown in Fig. 5 of the main paper, where the object latent acts like a phase-dependent "knob" for hand openness. The modulator can therefore use small latent adjustments to continuously recruit different grasping behaviors—e.g., preferring a more open hand during approach and a tighter hand during grasp—without changing the reference or relying on large action corrections. Overall, this analysis supports our claim that object latent modulation enables smooth interpolation and switching across the skill library learned by the distilled general tracker.

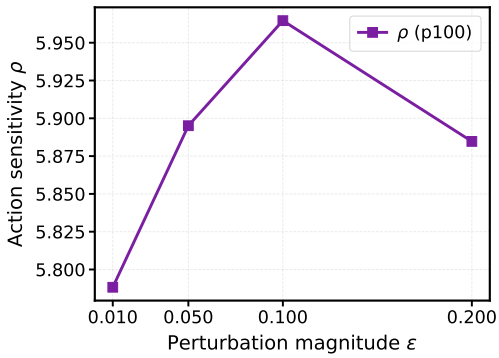
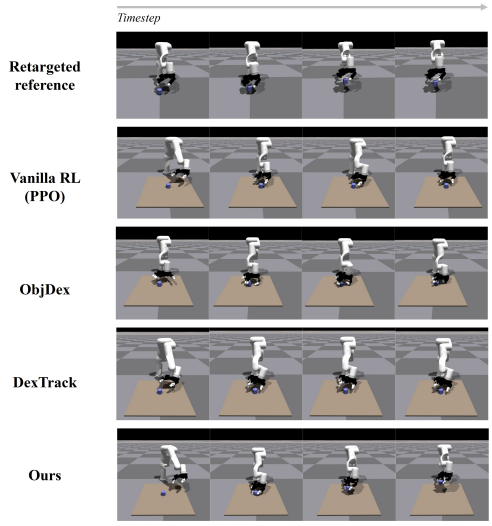
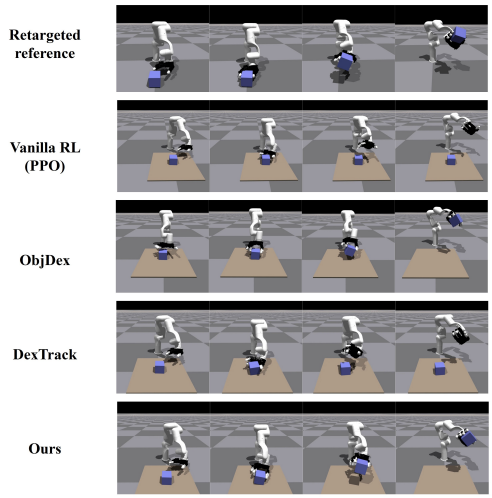


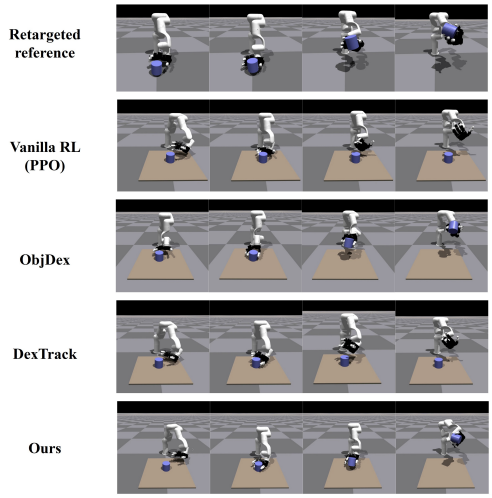
Figure 12. Lipschitz-style perturbation analysis of object latent modulation. The response ratio $\rho = \|\Delta a\|_2/\epsilon$ remains bounded under small latent perturbations, supporting a locally smooth latent-to-action mapping and meaningful interpolation between object anchors.



(a) Lifting a small cylinder. Severe reference noise and tiny geometry make all baselines fail; only AdaDexTrack maintains stable tracking.

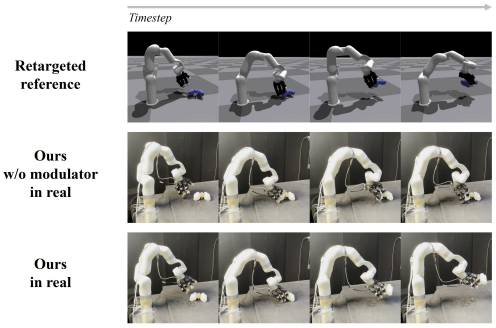


(b) Inspecting a medium cube. With a noisy reference, grasp timing and pose are critical. Vanilla RL (PPO) and DexTrack fail. ObjDex tracks the object but yields unnatural, non-human hand shapes. AdaDexTrack tracks both the hand and the object reliably.

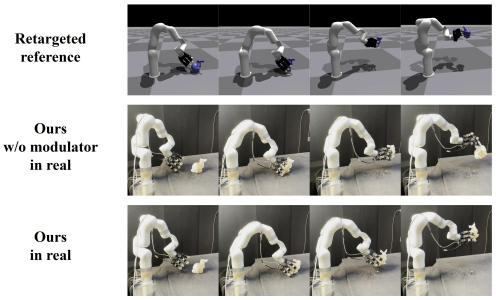


(c) Inspecting a medium cylinder. The reference induces a low-contact grasp, increasing difficulty. Vanilla RL (PPO) and DexTrack fail. ObjDex tracks the object but again produces unnatural hand shapes. AdaDexTrack tracks both the hand and the object and achieves stable grasps.

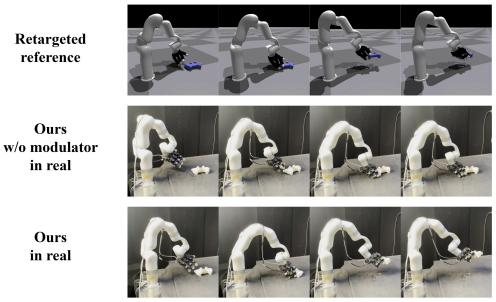
Figure 13. Qualitative results in simulation.



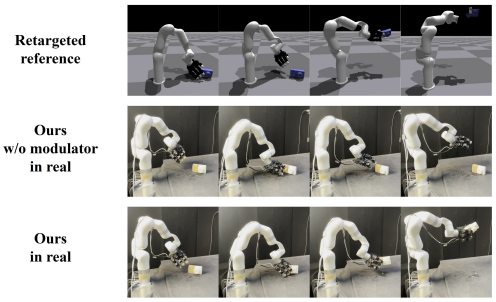
(a) Picking up binoculars. Stable grasp lies on a narrow central bridge, requiring precise finger placement and alignment.



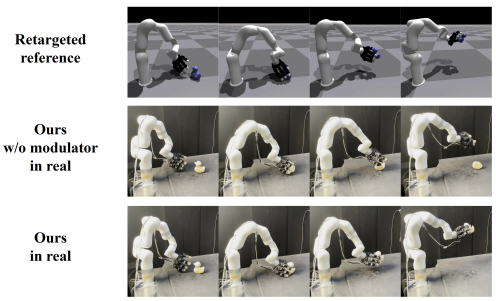
(b) Inspecting a bunny. Irregular geometry makes contact placement sensitive; small errors lead to slips.



(c) Grasping a game controller. Reliable grasp points cluster around a slim central neck, demanding accurate alignment.



(d) Using a camera. Asymmetric shape can cause tilt if grasp points are poorly chosen; careful placement is needed to maintain pose.



(e) Playing with a duck. Small size and curved surfaces tighten tolerances on contact location, calling for careful approach and closure.

Figure 14. Additional qualitative results in real-world.