Towards Human-level Dexterous Teleoperation

Puhao Li[1,2,*], Zeyuan Chen[2,3,*], Yingying Wu[1,2,*], Pengkun Wei[2], Yuyang Li[2,3], Tianyu Wang[2,3], Jiaxiao Shi[2], Mingrui Yu[1], Baoxiong Jia[2], Song-Chun Zhu[1,2,3], Tengyu Liu[2,†], Siyuan Huang[2,†]
[1]Tsinghua University [2]State Key Lab of General Artificial Intelligence, BIGAI
[3]Peking University [*]Equal Contribution [†]Corresponding author

***Abstract*—Humans routinely wield tools, swap grasps, and reposition objects within a single hand, seamlessly orchestrating contact transitions that span in-hand translation, reorientation, and finger gaiting. Endowing robot hands with this level of in-hand dexterity through teleoperation requires precise control of object motion via dynamic hand–object contact, yet current teleoperation systems remain far from this capability. We take a major step toward human-level dexterous teleoperation by introducing TELEDEXTER, a hand–object co-tracking controller that maps operator intent into learned low-level contact execution. The controller is trained on consecutive co-tracking subgoals derived from human reference motions, using a hybrid reward that couples sparse subgoal objectives with dense tracking signals to learn diverse interaction modalities rather than rigid frame-wise imitation. The entire pipeline requires only single-stage Reinforcement Learning (RL) and, through random action masking and domain randomization, transfers zero-shot to the real robot. Across seven challenging dexterous tasks spanning object reorientation and long-horizon tool use on two dexterous hands, TELEDEXTER achieves a 75% average success rate where all baselines consistently fail. The collected demonstrations further train autonomous policies via behavioral cloning, marking a concrete step toward human-level dexterous teleoperation. The project page is available at https://bigai-dex.github.io/blog/teledexter.**

I. INTRODUCTION

Everyday manipulation demands human-level dexterity: the ability to dynamically reorient, translate, and regrasp objects within a single hand by continuously coordinating complex finger–object contacts [3, 4]. While such agility is effortless for humans, it remains far beyond the reach of current robots. Teleoperation lets a human teach the robot in the loop [25, 30, 42]. *Yet achieving human-level in-hand dexterity through dexterous teleoperation remains an open challenge.*

Existing dexterous teleoperation systems fall short of this goal. One dominant line employs kinematic retargeting to map human hand motion directly onto robot topologies, using vision-based tracking [7, 9, 13, 30] or wearable gloves [11, 39, 41, 43]. While this provides an intuitive, low-latency interface that faithfully captures the operator's kinematic intent, it ignores hand–object contact forces and object inertia. Consequently, high-acceleration maneuvers, non-prehensile interactions, and continuous finger gaiting remain highly unstable, frequently causing object slippage or drops.

An alternative paradigm [40] learns a dexterous action prior in simulation to map coarse teleoperation commands to contact-rich hand actions, improving local contact robustness. However, it relies on synthetically generated grasp transitions that often lack physical feasibility, and encoding the prior into a generative model introduces compounding trajectory drift and covariate shift during closed-loop execution, severely degrading long-horizon real-world performance.

To overcome these limitations, we introduce **TELEDEXTER**, *a hand–object co-tracking controller designed for human-level dexterous teleoperation*. Instead of mapping hand joints in isolation, the operator specifies explicit, synchronized geometric targets for both the fingertip positions and the object pose; a low-level controller, trained entirely in simulation with RL, then resolves the multi-contact physics needed to realize these dual goals in real time. This separates *what* the operator wants from *how* the robot makes and switches contacts. The key technical designs of **TELEDEXTER** are three-fold:

- **Consecutive subgoal co-tracking.** Rather than enforcing frame-by-frame imitation, we decompose human reference motions into a sequence of synchronized fingertip and object-pose subgoals. Training the policy to reach these consecutive targets, instead of copying exact configurations, grants the flexibility to discover feasible contact-switching strategies, all optimized in a single-stage RL pipeline that couples sparse subgoal rewards with dense tracking rewards and removes the need for task-specific reward engineering.
- **Geometry-aware co-tracking construction.** We translate unscripted human hand–object demonstrations into physically grounded reference motions via a two-stage retargeting that, beyond pure kinematic mapping, incorporates object meshes to enforce contact-surface attraction and penalize interpenetration, yielding synchronized fingertip-and-object subgoal sequences for the low-level controller.
- **Sim-to-real robustness.** We introduce random action masking as a strong action-space regularizer that, combined with systematic domain randomization, prevents the policy from overfitting to perfectly synchronized simulated actuation and enables zero-shot deployment on real robots.

We evaluate **TELEDEXTER** on seven challenging dexterous teleoperation tasks across two hand embodiments, ranging from in-hand reorientation to long-horizon tool use with a hammer, screwdriver, brush, and light bulb. **TELEDEXTER achieves a 75.2% average success rate, whereas baselines consistently fail.** The demonstrations it collects directly train fully autonomous policies via behavioral cloning, executing long-horizon dexterous tasks without a human in the loop. Together, these results establish **TELEDEXTER** as a scalable foundation for collecting rich in-hand manipulation data and a viable path toward human-level robotic dexterity.

II. Related Work

Dexterous Teleoperation transfer human dexterity to robotic hands [10, 15, 17, 26, 42]. The dominant approach, kinematic retargeting, translates the human hand configuration to robot joint positions via vision-based tracking [7, 9, 13, 30, 38], wearable or exoskeleton gloves [11, 34, 41, 43], or learned neural mappings [39]. It offers an intuitive, low-latency interface but lacks a dynamics prior, making contact-rich actions such as in-hand reorientation, finger gaiting, and tool use unreliable. To address this, DexGen [40] learns a generative dexterous action prior that maps coarse commands to fine hand actions, but its synthetic grasp transitions lack physically feasible and its generative rollout accumulates compounding error during real execution. In contrast, we directly train an RL controller guided by human hand–object reference motions, providing physically grounded goals that cover diverse in-hand modalities; the learned controller directly serves as a robust low-level teleoperation policy for diverse in-hand dexterity.

Learning Dexterous Manipulation via RL in simulation has driven rapid progress, from grasping [18–20, 36] and in-hand reorientation [1, 2, 5, 14, 23, 28, 37] to dynamic finger-gaiting skills [29, 35], typically via task-specific reward engineering. Recent work mitigates this using human hand–object data as reference motions to guide RL, enabling diverse skills without per-task reward design [6, 18, 21, 22]. However, these methods mostly learn one policy per trajectory and rarely achieve in-hand skills such as reorientation or finger gaiting. We attribute this to their dense frame-wise tracking formulation, which prevents exploring dynamic contact strategies beyond basic grasping. We instead introduce a consecutive subgoal tracking formulation that learns from human references, yielding diverse, dynamic in-hand skills within a single RL stage; combined with random action masking and systematic domain randomization [5, 27], the policy transfers zero-shot to the real robot as a dexterous teleoperation controller.

III. TeleDexter

We formulate dexterous teleoperation as hand–object co-tracking: the operator specifies hand and object pose targets, while a learned low-level controller resolves the multi-contact dynamics that physically realize them. As shown in Fig. 1, we first formulate the co-tracking problem over a set of hand–object reference motions and present a single-stage RL framework for training the controller (Sec. III-A), then describe how the reference motions are constructed (Sec. III-B) and how the learned policy is deployed for real-world teleoperation (Sec. III-C). Full implementation details in Appendix C.

Problem Formulation All quantities are expressed in the wrist frame, and the arm tracks the human wrist independently via inverse kinematics (IK). Given a set of hand–object reference motions (Sec. III-B), our goal is to learn a co-tracking policy $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{o}_t, g_t)$ that drives the robot hand and the manipulated object toward poses prescribed by a co-tracking goal g_t. Here $\boldsymbol{o}_t$ encodes the current hand–object state and previous action, $\boldsymbol{a}_t \in \mathbb{R}^{n_{\text{dof}}}$ are target joint positions, and the goal $g_t = (\hat{\boldsymbol{p}}_t^{\text{tip}}, \hat{T}_t^o)$ specifies target fingertip positions $\hat{\boldsymbol{p}}_t^{\text{tip}} \in \mathbb{R}^{N_f \times 3}$ and a target object pose $\hat{T}_t^o \in SE(3)$. During teleoperation, g_t is built from real-time captured hand–object poses, casting teleoperation as co-tracking: the goal prescribes *what* the hand and object should achieve, while the learned controller decides *how* to make and switch contacts.

A. Learning a Co-tracking Controller

We train the controller π_θ in simulation via RL so that it acquires a dynamic hand–object action prior through simulated contact, rather than copying kinematic trajectories.

Consecutive Subgoal Co-tracking Each reference trajectory is converted into a sequence of co-tracking subgoals sampled at varying intervals. The policy must reach each subgoal in order before advancing to the next, while freely discovering its own contact strategy in between. For a subgoal $g_k = (\hat{\boldsymbol{p}}_k^{\text{tip}}, \hat{T}_k^o)$, the tracking errors are $e_{t,k}^f = \left\| \boldsymbol{p}_{t,f}^{\text{tip}} - \hat{\boldsymbol{p}}_{k,f}^{\text{tip}} \right\|_2$, $e_{t,k}^{\text{pos}} = \|\boldsymbol{x}_t^o - \hat{\boldsymbol{x}}_k^o\|_2$, and $e_{t,k}^{\text{rot}} = \left\| \text{Log}\left(\hat{R}_k^{o\top} R_t^o \right) \right\|_2$. A subgoal is reached once $\max_f e_{t,k}^f < \epsilon_{\text{tip}}$, $e_{t,k}^{\text{pos}} < \epsilon_{\text{pos}}$, and $e_{t,k}^{\text{rot}} < \epsilon_{\text{rot}}$ hold for N_{stay} consecutive frames, after which the policy advances. This preserves the order of the human demonstration but leaves the contact strategy to the policy.

Hybrid Reward Design The dominant learning signal is consecutive goal reaching, augmented with a small dense tracking reward for early exploration and a time penalty:

$$r_t = \underbrace{\mathbb{1}_{\text{reach}}(t)\, w_{\text{step}}(t)\, r_{\text{score}}(t)}_{\text{sparse subgoal}} + \underbrace{\alpha_{\text{dense}}\, r_{\text{dense}}(t)}_{\text{dense tracking}} - \underbrace{c_{\text{time}}}_{\text{time}}, \tag{1}$$

where the indicator $\mathbb{1}_{\text{reach}}(t)$ fires when the active subgoal is reached, w_{step} weights the bonus by the inter-subgoal step size, and the subgoal score r_{score} measures how well the hand and object match the active subgoal. The dense term r_{dense} applies the same per-finger and object tracking kernels at every step, providing a small dense signal during early training.

Curriculum Learning We progressively increase three dimensions of difficulty during training: gravity is annealed from reduced to full, easing initial contact establishment; the subgoal tolerances $(\epsilon_{\text{tip}}, \epsilon_{\text{pos}}, \epsilon_{\text{rot}})$ are tightened from permissive to strict; and the inter-subgoal step size grows from small to large, so the policy first masters fine-grained local tracking and later handles longer-horizon goal jumps. Episodes are initialized at random frames, and successful traversal of one trajectory resets the environment to another (cross-trajectory reset), enabling continuous learning across the full set.

Sim-to-Real Robustness To transfer zero-shot, we apply domain randomization over object and hand dynamics, external perturbations, and sensing noise and latency [5], and additionally introduce *random action masking* as a strong action-space regularizer. Given the policy action $\boldsymbol{a}_t$, we sample a binary mask $\boldsymbol{m}_t \in \{0,1\}^{n_{\text{dof}}}$ and execute $\tilde{\boldsymbol{a}}_t = \boldsymbol{m}_t \odot \boldsymbol{a}_t + (1 - \boldsymbol{m}_t) \odot \tilde{\boldsymbol{a}}_{t-1}$, freezing the masked dimensions at their previous command for a randomly sampled duration. By forcing the policy to succeed even when subsets of joints retain stale commands, this regularization prevents overfitting to simulation actuation.

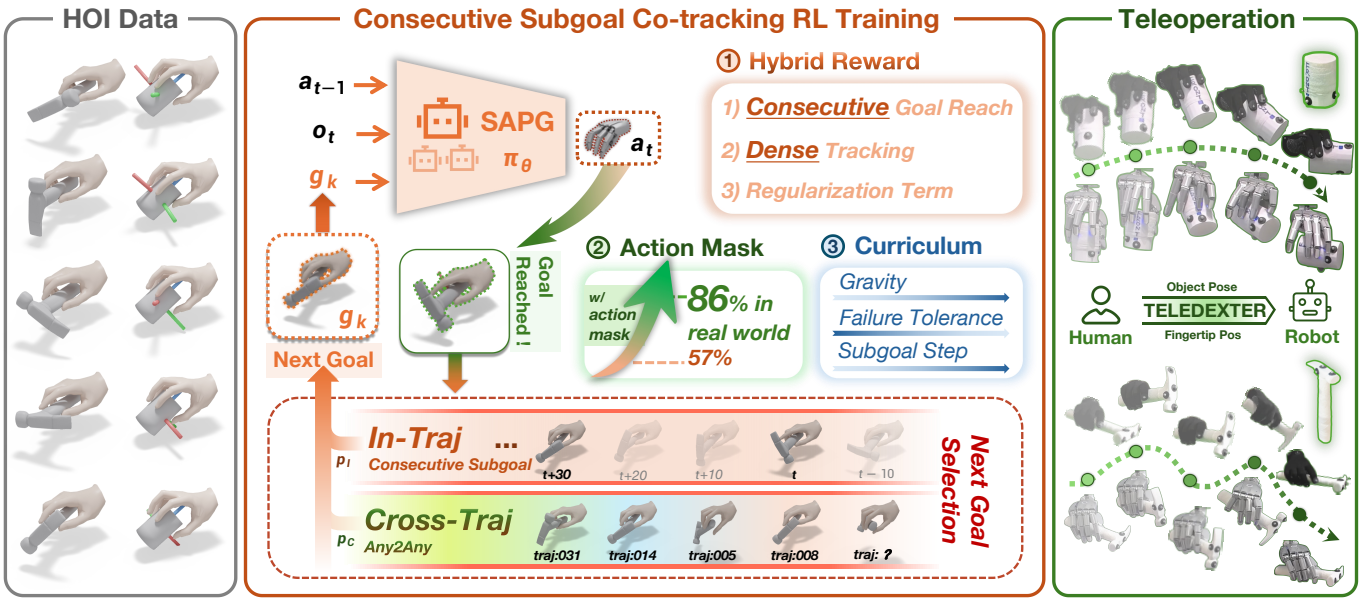


Fig. 1. **Method overview of TELEDEXTER.** Human hand–object references define consecutive co-tracking goals; a single-stage RL controller learns to reach them and deploys zero-shot for real teleoperation.

Single-stage RL Training Each controller is trained in a single RL stage, with no skill decomposition or task-specific reward engineering. We use Isaac Gym [24] with all reference motions for one object loaded simultaneously, and optimize with SAPG [32] over ∼62,000 parallel environments; training converges within $\sim 10^{10}$ environment steps. Being reference-driven, the framework scales to new objects and hands.

B. Hand–Object Reference Motion Construction

We record human hand–object interaction trajectories with a MoCap system, spanning in-hand translation, in-hand rotation, and free-play motions, and retarget them into robot hand references through geometry-aware retargeting. The first stage follows vector-based retargeting [13, 30], optimizing robot joint angles to match human hand pose. A second geometry-aware stage refines the result using the object mesh, adding contact-surface attraction, mesh-interpenetration and self-collision penalties, and a Curobo [33] temporal-smoothness term. The resulting trajectories $\{(\boldsymbol{q}_t^*, T_t^o)\}_{t=1}^T$ yield the co-tracking goals used above, with fingertip targets $\hat{\boldsymbol{p}}_t^{\text{tip}} = \text{FK}_{\text{tip}}(\boldsymbol{q}_t^*)$ and object targets $\hat{T}_t^o = T_t^o$.

C. Real-world Teleoperation Deployment

The learned co-tracking controller deploys zero-shot as the teleoperation policy. A real-time system captures the operator's wrist, fingertip, and object poses; the arm tracks the wrist via IK, while the fingertip and object poses form the co-tracking goal. For contact initialization, kinematic retargeting [13] handles pre-grasp positioning, after which the operator switches to the co-tracking controller for dexterous in-hand manipulation.

IV. EXPERIMENTS

We conduct a systematic real-world evaluation of **TELEDEXTER**, asking whether it enables dexterous in-hand teleoperation beyond representative baselines (Sec. IV-A), whether the demonstrations it collects can train autonomous policies (Sec. IV-B), and how its key designs contribute (Sec. IV-C). All experiments run on a Franka FR3 arm with two hand embodiments—the 16-DoF four-finger LeapHand [31] and the 22-DoF five-finger SharpaWave—using a NOKOV motion-capture system to track the operator's hand and the object's 6D pose in real time at 30 Hz. Detailed implementations are provided in Appendices B and D.

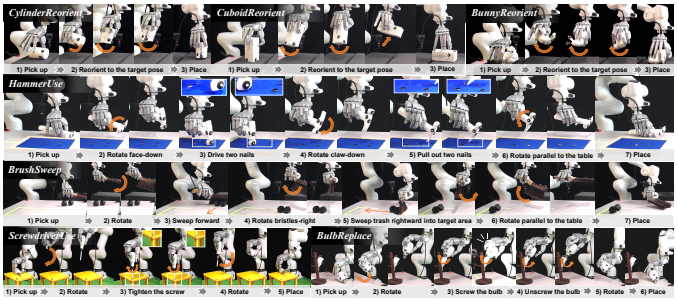


Fig. 2. **Task descriptions.** Zoom in for stage details.

A. Dexterous Teleoperation Evaluation

We compare **TELEDEXTER** against *DexRT* [13, 30], which maps hand motion to the robot hand by kinematic retargeting; *GeoRT* [39], a learned neural retargeter with geometric objectives; *DexGen* [40], a learned generative action prior; and *SimToolReal* [16], an object-centric sim-to-real tool-use policy that, while not a teleoperation method, serves as a strong reference for learned dexterous tool use. The seven tasks in Fig. 2 span three in-hand reorientation tasks over symmetric, corner-rich, and irregular geometries, and four long-horizon tool-use tasks requiring functional grasp transitions, finger gaiting, and tool use. We run 15 trials per task and report success rate (**SR**),

Tab. I. **Dexterous teleoperation results on SharpaWave.** Each cell reports **SR** / **TP** (%; higher is better). *SimToolReal* is not a teleoperation method († category-specific, ‡ all categories).

Task	*DexRT*	*GeoRT*	*DexGen*	*SimToolReal*†	*SimToolReal*‡	**TELEDEXTER**
`CylinderReorient`	6.7 / 37.8	0.0 / 24.4	0.0 / 31.1	—	—	**80.0 / 86.7**
`CuboidReorient`	26.7 / 51.1	0.0 / 33.3	0.0 / 28.9	—	—	**80.0 / 86.7**
`BunnyReorient`	0.0 / 35.6	0.0 / 31.1	0.0 / 26.7	—	—	**66.7 / 77.8**
`HammerUse`	0.0 / 26.7	0.0 / 30.5	0.0 / 26.7	0.0 / 27.6	20.0 / 36.2	**66.7 / 86.7**
`BrushSweep`	0.0 / 39.0	0.0 / 29.5	0.0 / 8.6	26.7 / 41.9	0.0 / 5.7	**73.3 / 89.5**
`ScrewdriverUse`	6.7 / 37.3	0.0 / 25.3	0.0 / 33.3	0.0 / 17.3	0.0 / 20.0	**73.3 / 86.7**
`BulbReplace`	0.0 / 35.6	0.0 / 25.6	0.0 / 20.0	—	—	**86.7 / 95.6**
Average	5.7 / 37.6	0.0 / 28.5	0.0 / 25.0	8.9 / 28.9	6.7 / 20.6	**75.2 / 87.1**

the fraction of trials completing all stages, and task progress (**TP**), the average fraction of stages completed.

As shown in Tab. I, **TELEDEXTER** achieves 75.2% average **SR** and 87.1% average **TP** on SharpaWave, while all teleoperation baselines near-uniformly fail. On the reorientation tasks, **TELEDEXTER** obtains 66.7–80.0% **SR** by executing dynamic in-hand contact strategies that kinematic retargeting cannot stabilize, whereas *DexRT* and *GeoRT* rarely progress beyond pick-up and *DexGen* fails as generative-model drift degrades contact execution. The gap widens on tool-use tasks, which demand long-horizon coordination of functional grasp transitions and finger gaiting: **TELEDEXTER** reaches 66.7–86.7% **SR** while baselines collapse at the first contact-intensive stage. The narrow **SR**-to-**TP** gap shows that **TELEDEXTER**'s few failures occur at late stages: on `BulbReplace` it clears both rotation stages without losing a trial (15/15 through stage 4 of 6), and on `ScrewdriverUse` 13/15 trials sustain finger gaiting through tightening, a contact mode no baseline executes. *SimToolReal*, despite task-specific training, reaches at most 26.7% **SR** on a single tool, showing that even dedicated tool policies fail to generalize across diverse contact transitions. The same pipeline applied to LeapHand—adapting only the retargeting stage to its 16-DoF morphology—yields 60.0–73.3% **SR** on reorientation (Tab. III), confirming cross-embodiment generalization without re-collecting references. More analysis are in Appendix A.

B. From Teleoperation to Autonomy

A key advantage of **TELEDEXTER** is its ability to collect dexterous manipulation data beyond the reach of existing teleoperation systems, providing high-quality expert demonstrations for autonomous policy learning. We train Diffusion Policies [8] on three tasks—`BulbInstall`, `HammerDriver`, and `BrushForward`—each retaining the most contact-intensive stages of its teleoperation counterpart. With only 50 demonstrations per task, the policies reach non-trivial real-world success (Tab. II): 73.3% on hammer driving, 46.7% on bulb installation, and 40.0% on brush sweeping. The stage-wise breakdown reveals a distinct per-task bottleneck—nail driving for the hammer, precision socket alignment for the bulb, and the thin-handle grasp for the brush—while the remaining stages transfer reliably from demonstrations. Because no baseline in Tab. I can reliably complete these tasks, comparable data cannot be collected with existing methods, validating **TELEDEXTER** as both a teleoperation interface and a scalable data engine for autonomous dexterous manipulation.

Tab. II. **Autonomous policy stage-wise success.** Entries are reached trials; 50 **TELEDEXTER** demonstrations per task.

Policy	S1	S2	S3	S4	**SR**
Bulb	13/15	12/13	8/12	7/8	46.7%
Hammer	15/15	15/15	11/15	—	73.3%
Brush	7/15	7/7	6/7	—	40.0%

C. Ablation Studies

We ablate four design choices central to **TELEDEXTER**'s performance, with detailed analysis in appendix Sec. A-C. Replacing consecutive subgoal tracking with dense frame-wise tracking drastically shortens episodes and reduces subgoals reached by orders of magnitude, confirming that frame-wise imitation leaves no room to discover feasible contact strategies. Removing random action masking sharply degrades real-world success across all tasks, as the policy exploits simulation-specific actuation that does not transfer. Disabling the curriculum yields faster initial progress but lower final performance, and replacing SAPG with PPO substantially reduces reward and subgoal coverage due to insufficient exploration diversity for the varied contact modes in co-tracking.

V. CONCLUSION AND LIMITATIONS

We presented **TELEDEXTER**, a hand–object co-tracking controller that takes a concrete step toward human-level dexterous teleoperation. Built on consecutive subgoal tracking, a hybrid reward, and random action masking, it learns diverse in-hand skills in a single RL stage with no task-specific reward and transfers zero-shot to real robots, achieving 75.2% average success across seven challenging dexterous tasks where baselines uniformly fail. The demonstrations it collects further train autonomous policies, establishing a scalable foundation for rich in-hand manipulation data and a viable path toward human-level robotic dexterity.

Limitations **TELEDEXTER** learns an object-specific controller, so a new object requires collecting interaction data and training a dedicated policy, and deployment relies on motion capture for real-time pose estimation. Scaling to a unified, object-conditioned controller and to markerless vision-based tracking are promising next steps.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1), 2020. doi: 10.1177/0278364919887447.

[3] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.

[4] Ian M. Bullock, Raymond R. Ma, and Aaron M. Dollar. A hand-centric classification of human and robot dexterous manipulation. *IEEE Transactions on Haptics*, 6(2): 129–144, 2013. doi: 10.1109/TOH.2012.53.

[5] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[6] Yuanpei Chen, Chen Wang, Yaodong Yang, and Karen Liu. Object-centric dexterous manipulation from human motion data. In *8th Annual Conference on Robot Learning*, 2024.

[7] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. In *8th Annual Conference on Robot Learning*, 2024.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[9] 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. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12248–12255. IEEE, 2025.

[10] Guanglong Du, Ping Zhang, Jianhua Mai, and Zeling Li. Markerless kinect-based hand tracking for robot teleoperation. *International Journal of Advanced Robotic Systems*, 9(2):36, 2012.

[11] Hao-Shu Fang, Branden Romero, Yichen Xie, Arthur Hu, Bo-Ruei Huang, Juan Alvarez, Matthew Kim, Gabriel Margolis, Kavya Anbarasu, Masayoshi Tomizuka, et al. Dexop: A device for robotic transfer of dexterous human manipulation. *arXiv preprint arXiv:2509.04441*, 2025.

[12] Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Or Perel, Charles Loop, Towaki Takikawa, Vismay Modi, Alexander Zook, Jiehan Wang, Wenzheng Chen, Tianchang Shen, Jun Gao, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Sanja Fidler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. Kaolin: A pytorch library for accelerating 3d deep learning research, 2024. URL https://github.com/NVIDIAGameWorks/kaolin.

[13] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[14] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023.

[15] Hooman Hedayati, Michael Walker, and Daniel Szafir. Improving collocated robot teleoperation with augmented reality. In *Proceedings of the 2018 ACM/IEEE international conference on human-robot interaction*, pages 78–86, 2018.

[16] Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. *arXiv preprint arXiv:2602.16863*, 2026.

[17] Jonathan Kofman, Siddharth Verma, and Xianghai Wu. Robot-manipulator teleoperation by markerless vision-based hand-arm tracking. *International Journal of Optomechatronics*, 1(3):331–357, 2007.

[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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6991–7003, 2025.

[19] Puhao Li, Tengyu Liu, Yuyang Li, Yiran Geng, Yixin Zhu, Yaodong Yang, and Siyuan Huang. Gendexgrasp: Generalizable dexterous grasping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8068–8074. IEEE, 2023.

[20] Yuyang Li, Bo Liu, Yiran Geng, Puhao Li, Yaodong Yang, Yixin Zhu, Tengyu Liu, and Siyuan Huang. Grasp multiple objects with one hand. *IEEE Robotics and Automation Letters*, 9(5):4027–4034, 2024.

[21] Xueyi Liu, Kangbo Lyu, Jieqiong Zhang, Tao Du, and Li Yi. Parameterized quasi-physical simulators for dexterous manipulations transfer. In *European Conference on Computer Vision*, pages 164–182. Springer, 2024.

[22] Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. *arXiv preprint arXiv:2502.09614*, 2025.

[23] Xueyi Liu, He Wang, and Li Yi. Dexndm: Clos-

ing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model. *arXiv preprint arXiv:2510.08556*, 2025.

[24] 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.

[25] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[26] Günter Niemeyer, Carsten Preusche, Stefano Stramigioli, and Dongjun Lee. Telerobotics. In *Springer handbook of robotics*, pages 1085–1108. Springer, 2016.

[27] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[28] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1722–1732. PMLR, 2023. URL https://proceedings.mlr.press/v205/qi23a.html.

[29] Haozhi Qi, Brent Yi, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. From simple to complex skills: The case of in-hand object reorientation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14291–14298. IEEE, 2025.

[30] 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.

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

[32] Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: split and aggregate policy gradients. In *Proceedings of the 41st International Conference on Machine Learning*, pages 45759–45772, 2024.

[33] Balakumar Sundaralingam, Adithyavairavan Murali, and Stan Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots, 2026.

[34] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[35] Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin" pens". *arXiv preprint arXiv:2407.18902*, 2024.

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

[37] Max Yang, Alex Church, Yijiong Lin, Christopher J Ford, Haoran Li, Efi Psomopoulou, David AW Barton, Nathan F Lepora, et al. Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch. In *8th Annual Conference on Robot Learning*, 2024.

[38] Shiqi Yang, Minghuan Liu, Yuzhe Qin, Runyu Ding, Jialong Li, Xuxin Cheng, Ruihan Yang, Sha Yi, and Xiaolong Wang. Ace: A cross-platform and visual-exoskeletons system for low-cost dexterous teleoperation. In *8th Annual Conference on Robot Learning*, 2024.

[39] 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. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 17376–17382. IEEE, 2025.

[40] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[41] 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.

[42] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[43] Alvin Zhu, Mingzhang Zhu, Beom Jun Kim, Jose Victor SH Ramos, Yike Shi, Yufeng Wu, Raayan Dhar, Fuyi Yang, Ruochen Hou, Hanzhang Fang, et al. Dexexo: A wearability-first dexterous exoskeleton for operator-agnostic demonstration and learning. *arXiv preprint arXiv:2603.17323*, 2026.

APPENDIX A
EXTENDED RESULTS AND ANALYSIS

A. Cross-Embodiment Teleoperation Results

Tab. III reports the regular teleoperation evaluation on LeapHand. These trials use the same human hand–object references as the SharpaWave experiments; only the geometry-aware retargeting stage is adapted to the four-finger, 16-DoF LeapHand morphology.

Tab. III. **Teleoperation on LeapHand.** Each cell reports **SR / TP** (%).

Task	TELEDEXTER
`CylinderReorient`	60.0 / 73.3
`CuboidReorient`	73.3 / 82.2

B. Any-to-Any Reposition Evaluation

Overview This evaluation directly tests whether our co-tracking controller, deployed zero-shot to the real robot, can handle arbitrary in-hand hand-object tracking goal transitions. During teleoperation the operator's goal can jump to any feasible hand-object configuration at any time; the any-to-any reposition test isolates this capability by streaming a sequence of randomly sampled targets and measuring how many the controller reaches consecutively before failure.

Protocol and Metrics We evaluate the co-tracking controller on `Cylinder` and `Cuboid` on LeapHand. We choose LeapHand for this stress test. Despite LeapHand's more limited dexterity (4 fingers, 16 DoFs vs. 5 fingers, 22 DoFs), the controller trained by our framework still demonstrates non-trivial in-hand repositioning capability and robustness, as shown below.

For each object, we construct a hand-object targets pool consisting of all frames from our HOI reference set (Sec. C-F), which covers the full feasible workspace of in-hand configurations. We run 15 trials per object. Each trial begins by sampling an initial grasp pose from the pool; the tester places the object into the hand accordingly and starts the controller. The controller is then queried with a stream of targets, each freshly sampled from the same pool as soon as the previous one is reached. A target is counted as reached when the object position error falls below 2 cm and the object rotation error falls below 20° simultaneously. A trial terminates when (i) the object slips out of the hand, or (ii) the active target has not been reached for 20 s. We report two metrics: consecutive successes, the number of targets reached in sequence before failure (averaged over trials), and per-target success rate, the fraction of all attempted targets that are successfully reached.

Results and Analysis As shown in Tab. IV, the controller sustains long sequences of arbitrary goal transitions on both objects, confirming that it generalizes well beyond the training reference trajectories. On `Cylinder` the controller reaches 41.1 consecutive targets on average with a per-target success rate of 97.6%, substantially outperforming `Cuboid` (12.1 consecutive successes, 92.3%). We attribute this gap to object geometry: the cylinder's continuous surface allows fingers to slide and roll the object fluidly during transitions without encountering abrupt geometric changes, and its rotational symmetry reduces the effective distance between arbitrary target poses. The cuboid's edges and corners, by contrast, can obstruct finger motion during rapid regrasps — fingers must navigate around these features to reach certain target orientations, increasing the chance of contact transition jams and failed repositioning attempts.

Tab. IV. **Any-to-any results on LeapHand.** Each object is evaluated over 15 trials.

Object	Consecutive Successes (↑)	Per-target SR (↑)
`Cylinder`	41.1	97.6%
`Cuboid`	12.1	92.3%

C. Ablation Studies

We ablate four design choices that drive **TELEDEXTER**'s performance: consecutive subgoal tracking, random action masking, the curriculum schedule, and the policy optimizer. All ablations share the same reference motions and remaining training settings unless stated otherwise. The curriculum and optimizer ablations are conducted on `Hammer`.

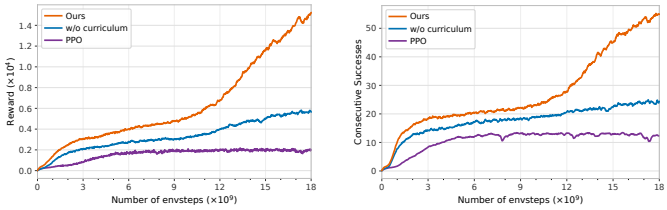
Consecutive Subgoal vs. Dense Tracking We replace our sparse consecutive subgoal formulation with dense frame-wise tracking, where the policy must match every reference frame in sequence, and evaluate both variants in simulation on held-out references. As shown in Tab. V, sparse subgoal tracking yields far longer episodes (373–379 vs. 89–131 frames) and reaches orders of magnitude more consecutive subgoals (33–187 vs. 2.6–2.7) across all three objects. Dense tracking forces step-by-step replication of the reference, leaving no tolerance for the agent to discover alternative contact strategies; the policy commits to a narrow trajectory corridor and terminates quickly once any deviation accumulates. The subgoal formulation, by contrast, prescribes only the waypoints and lets the policy freely explore the contact sequences between them, enabling the diverse dynamic behaviors (finger gaiting, regrasping, continuous rotation) observed at deployment.

Random Action Masking We train an otherwise identical controller without random action masking and deploy it on the real robot. As shown in Tab. VI, removing masking sharply degrades success across all tasks: **SR** drops from 66.7–80.0% to 0.0–33.3%, with the largest collapse on `ScrewdriverUse` (73.3% → 0.0%), where sustained axial rotation demands robust per-joint recovery from stale commands. Without masking, the policy overfits to the perfectly synchronized actuation of simulation, exploiting tight inter-joint coordination patterns that break under the actuator compliance, backlash, and occasional missed commands present on real hardware. Masking forces the policy to succeed even when subsets of joints retain stale commands for variable durations, closely matching these dominant real-world failure modes.

Curriculum Schedule We disable the three-dimensional curriculum (Sec. C-C), training at full deployment difficulty

throughout (full gravity, maximum subgoal step size, maximum action-mask duration). As shown in Fig. 3, the no-curriculum variant achieves faster initial reward growth but plateaus at a substantially lower final performance in both reward and consecutive successes. The curriculum completes its annealing within the first 2×10^9 environment steps, yet the advantage it provides continues to grow well beyond that point. We attribute this to the quality of the early-training foundation: the curriculum first exposes the policy to small subgoal steps, permissive tolerances, and reduced gravity, allowing it to discover a diverse repertoire of stable contact primitives before difficulty ramps up. Without this scaffolding, the policy must simultaneously learn basic contact strategies and cope with full-difficulty dynamics, converging to a narrower set of behaviors that limits its ability to compose longer manipulation sequences later in training.

SAPG vs. PPO We replace SAPG [32] with vanilla PPO while keeping all other settings fixed. As shown in Fig. 3, PPO causes a substantial drop in both reward and consecutive successes. SAPG maintains multiple independently exploring policy blocks whose gradients are aggregated, promoting diverse strategy discovery within a single training run. This exploration diversity is critical for co-tracking, where the policy must master qualitatively different contact modes (translation, continuous rotation, finger gaiting, regrasping) within a single network. PPO's unimodal gradient updates tend to commit early to a limited strategy set, under-covering the full spectrum of in-hand manipulation modalities present in our reference motions.



(a) Reward. (b) Consecutive successes.

Fig. 3. **Training-recipe ablation curves on `Hammer`.** (a) Reward and (b) consecutive subgoals reached vs. environment steps for the full method, without curriculum, and with PPO replacing SAPG.

D. Stage-Wise Teleoperation Success Analysis

For each of the four long-horizon tool-use tasks, we plot the number of trials (out of 15) that survive past each task stage for **TELEDEXTER** and all baselines (Fig. 4). The main paper highlights key stage-wise numbers for **TELEDEXTER**; here we provide the complete per-stage breakdown across all methods and analyze where each baseline breaks down. We additionally include *SimToolReal* [16], an object-centric sim-to-real tool-use policy that is not a teleoperation method but provides a strong reference for learned dexterous tool manipulation.

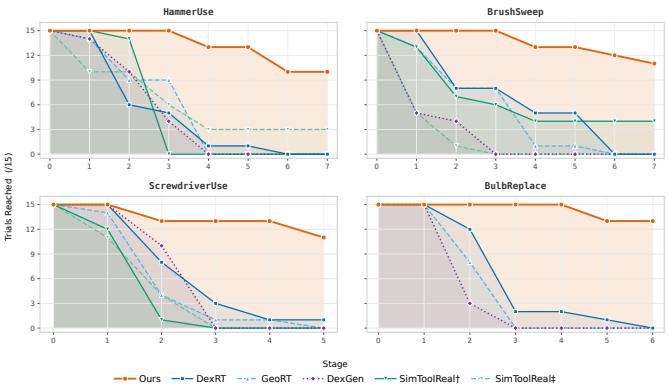


Fig. 4. **Stage-wise success on four long-horizon tool-use tasks.** For each task, the horizontal axis indexes the task stage and the vertical axis is the number of trials (out of 15) that reach that stage. Each curve corresponds to one method (**TELEDEXTER** and baselines from Tab. I in the main paper). *SimToolReal* is not a teleoperation method but is included as a strong baseline for learned tool manipulation.

Across all four tasks, a consistent pattern emerges. **TELEDEXTER** retains 13–15 out of 15 trials through the demanding mid-task stages and only drops modestly at the final placement stages, whereas every baseline suffers a sharp collapse at or shortly after the first stage that requires in-hand reorientation or functional grasp transition.

In `HammerUse`, all methods pick up the hammer (stage 1), but baselines diverge sharply at stage 2 (rotate face-down): *DexRT* drops from 15 to 6 and *DexGen* from 14 to 10. By stage 4 (rotate claw-down), no teleoperation baseline retains any trial. *SimToolReal*† (category-specific) leverages task-specific training to reach $14/15$ at stage 2 but collapses entirely at stage 3 (drive nails), unable to sustain the repeated contact force required for hammering. *SimToolReal*‡ (all-category) maintains $3/15$ through completion at the cost of weaker early-stage performance. **TELEDEXTER** retains all 15 trials through stage 3 and 10 through final placement.

In `BrushSweep`, a similar bottleneck emerges at grasp-transition stages. *DexGen* collapses earliest, losing most trials at pick-up ($5/15$) due to compounding generative-model errors. *DexRT* and *GeoRT* survive the initial sweep (stage 3, $8/15$ each) but fail at stage 4 (rotate bristles-right), which demands controlled in-hand reorientation while maintaining grasp. *SimToolReal*† is the strongest baseline on this task with $4/15$ completions, while **TELEDEXTER** reaches $11/15$.

In `ScrewdriverUse`, *DexGen* maintains $10/15$ through stage 2 (rotate to align) but drops to 0 at stage 3 (tighten the screw), as its action prior cannot sustain continuous axial rotation. Both *SimToolReal* variants also fail entirely at stage 3. *DexRT* retains $1/15$ to completion, the only teleoperation baseline trial to finish this task. **TELEDEXTER** maintains $13/15$ through tightening and finishes with $11/15$.

In `BulbReplace`, evaluated without *SimToolReal* as the task falls outside its object-centric formulation, all methods

Tab. V. **Sparse subgoal vs. dense** (sim). EpLen: episode length (↑). Goals: subgoals reached (↑).

	Dense: EpLen		Sparse: Goals	
Object	Ours	Dense	Ours	Dense
Cuboid	**378.6**	115.8	**32.6**	2.6
Hammer	**376.9**	131.2	**186.6**	2.7
Screwdriver	**373.2**	88.7	**178.5**	2.7

Tab. VI. **Action masking** (real). **SR/ TP** (%).

Task	w/ AM	w/o AM
`HammerUse`	**66.7/86.7**	33.3/57.1
`ScrewdriverUse`	**73.3/86.7**	0.0/36.0
`CuboidReorient`	**80.0/86.7**	26.7/51.1

pick up the bulb ($15/15$), but the critical drop occurs at stage 3 (screw in), where *DexRT* falls from 12 to 2 and both *GeoRT* and *DexGen* reach 0. These stages require precise bidirectional rotation about the bulb axis under sustained contact, which kinematic retargeting cannot achieve. **TELEDEXTER** completes both rotational stages without trial loss ($15/15$ through stage 4) and finishes with $13/15$.

Taken together, these results show that all baselines fail not at grasping or gross positioning, but at contact-rich in-hand manipulation: reorientation, finger gaiting, and sustained tool application. Kinematic methods (*DexRT*, *GeoRT*) lack a dynamics prior and collapse at the first contact-intensive stage. *DexGen* suffers compounding trajectory drift that causes abrupt failure at contact transitions. *SimToolReal* achieves partial success on trained tool categories but cannot generalize across the diverse contact modes within a single long-horizon task. **TELEDEXTER** sustains high trial survival precisely at these stages, confirming that the co-tracking controller provides the in-hand dexterity needed for long-horizon tool use.

E. TELEDEXTER *Failure Analysis*

We identify three dominant failure modes of our teleoperation controller, illustrated in Fig. 5.

- **Interaction perturbation (Fig. 5a).** Forceful tool–environment contact, such as striking a nail during `HammerUse`, generates impulsive reaction forces that shift the object's in-hand pose by a large, instantaneous amount. The controller, trained exclusively on free-space hand–object interaction, has never encountered such impact dynamics and cannot recover from the resulting out-of-distribution grasp configuration. This mode does not appear in reorientation-only tasks.
- **Contact transition jam (Fig. 5b).** During finger gaiting or in-hand reorientation, a finger that should lift away occasionally remains wedged against the object due to actuator compliance or geometric interlocking. As the remaining fingers continue to move, the jam generates unbalanced forces that eject the object from the hand. This failure is most frequent on irregular objects (`BunnyReorient`) where concavities increase the chance of interlocking.
- **Tracking stall (Fig. 5c).** The controller attempts a regrasp but fails to establish contacts that can move the object toward the target pose. Unlike the contact transition jam, the object is not lost—the hand simply cannot make progress. We attribute this to the absence of tactile observations (Tab. VII): the policy cannot distinguish between a finger pressing against the object and one sliding past it, and thus cannot adapt when a regrasp attempt fails.

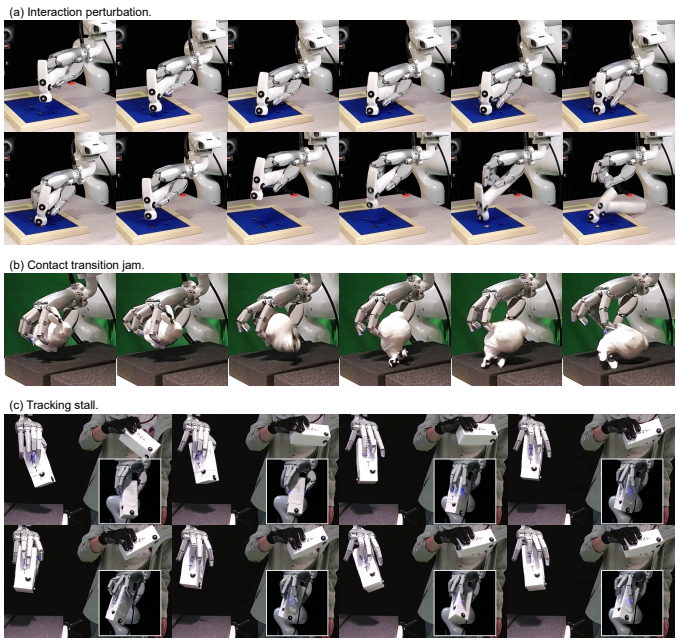


Fig. 5. **Real-world failure cases.** Each panel shows a snapshot at the point of failure together with the representative failure mode: (a) interaction perturbation, (b) contact transition jam, (c) tracking stall.

The three modes point to distinct limitation of our current system. Interaction perturbation reflects a *training distribution* gap, as tool–environment impact dynamics are absent from the simulation training. Contact transition jams reveal an *actuation compliance* mismatch between rigid-body simulation and real direct-drive fingers with passive compliance. Tracking stalls expose the lack of *tactile observation*, without which the policy cannot close the loop on contact state during regrasping. Addressing these limits through interaction-aware training, compliant-contact simulation, and tactile-rich observation spaces is a promising direction for future work.

APPENDIX B
EXPERIMENTAL SETUP DETAILS

A. Teleoperation Task Definitions

We provide detailed descriptions of the seven real-world tasks shown in Fig. 2, together with their stage decompositions. Each task is decomposed into N well-defined stages

that the trial must reach in order. The task progress (**TP**) reported in the main paper is the average k/N of the furthest stage k reached across trials, and the success rate (**SR**) is the fraction of trials that reach all N stages. For all tasks, stage $0/N$ denotes failure to pick up the object. We wrap the tested objects with medical bandage tape to increase surface friction.

`CylinderReorient` /`CuboidReorient` /`BunnyReorient` (3 stages each). The three reorientation tasks share an identical stage decomposition and differ only in object geometry, testing continuous in-hand pose control on rotationally symmetric (`CylinderReorient`), corner-rich (`CuboidReorient`), and irregular freeform (`BunnyReorient`) geometries. *Stages:*

- $1/3$: picked up the object.
- $2/3$: reoriented to the target pose in-hand.
- $3/3$: placed back on the table.

`HammerUse` (7 stages). A long-horizon task that exercises bidirectional functional grasp transitions (face-down vs. claw-down) and repeated striking and pulling. *Stages:*

- $1/7$: picked up the hammer.
- $2/7$: rotated face-down for hammering.
- $3/7$: drove the two nails into the board.
- $4/7$: rotated claw-down.
- $5/7$: pulled out the two nails with the claw.
- $6/7$: rotated parallel to the table.
- $7/7$: placed back on the table.

`BrushSweep` (7 stages). A long-horizon task requiring transitions between two functional brush orientations while sweeping across an extended workspace. *Stages:*

- $1/7$: picked up the brush.
- $2/7$: rotated for forward sweeping.
- $3/7$: swept the debris forward.
- $4/7$: rotated so the bristles face rightward.
- $5/7$: swept the debris rightward into the target area.
- $6/7$: rotated parallel to the table.
- $7/7$: placed back on the table.

`ScrewdriverUse` (5 stages). A precision task that tests axial alignment with the screw and sustained in-hand rotation about the tool axis. *Stages:*

- $1/5$: picked up the screwdriver.
- $2/5$: rotate the screwdriver tip downward.
- $3/5$: tightened the screw until fully seated.
- $4/5$: rotated the screwdriver for placement.
- $5/5$: placed back on the table.

`BulbReplace` (6 stages). A precision insertion task that combines functional in-hand reorientation with bidirectional rotation about the bulb axis. *Stages:*

- $1/6$: picked up the bulb.
- $2/6$: rotated to align with the socket.
- $3/6$: screwed the bulb in until it lit up.
- $4/6$: unscrewed it until fully removed.
- $5/6$: rotated for placement.
- $6/6$: placed back on the table.

B. Autonomous Policy Task Definitions

The autonomous Diffusion Policies are evaluated on simplified subsets of three teleoperation tasks, each retaining the most dexterous stages while removing the return-and-place phases. Stage decompositions follow the same protocol as the teleoperation tasks (Sec. B-A); a trial is terminated when an unrecoverable failure occurs.

`BulbInstall` (4 stages). A subset of `BulbReplace` that covers bulb installation only (no unscrewing or return). *Stages:*

- $1/4$: picked up the bulb.
- $2/4$: reoriented to the installation pose.
- $3/4$: aligned with the socket.
- $4/4$: screwed in until the bulb lit up.

`HammerDriver` (3 stages). A simplified variant of `HammerUse` with larger nails and a foam target, retaining the core in-hand rotation to a functional hammering grasp. *Stages:*

- $1/3$: picked up the hammer.
- $2/3$: rotated face-down for hammering.
- $3/3$: drove the nails into foam.

`BrushForward` (3 stages). A subset of `BrushSweep` that covers a single forward sweep only (no bristle-reorientation or return sweep). *Stages:*

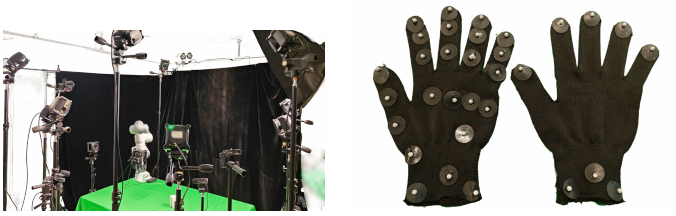
- $1/3$: picked up the brush.
- $2/3$: rotated for forward sweeping.
- $3/3$: swept forward across the workspace.

C. Hardware and Teleoperation System

1) Motion Capture System: We use a NOKOV optical motion capture system for both offline reference-motion collection and real-time teleoperation, with a different glove configuration for each setting. For offline collection (Fig. 6a), the operator stands in a dedicated capture volume equipped with tripod-mounted infrared cameras and wears a full-coverage glove instrumented with retro-reflective markers on the wrist, palm, and all finger joints (Fig. 6b, left). This dense marker layout enables the system to output, per frame, (i) 24 hand joint 3-D positions, (ii) 21 joint pose matrices, (iii) the wrist $\mathrm{SE}(3)$ pose, and (iv) the object's 6-D pose. For real-time teleoperation, the operator wears a compact glove with markers placed only on the wrist and fingertips (Fig. 6b, right) at the deployment workstation. This lightweight configuration provides the wrist pose and fingertip positions needed by the control loop. The MoCap stream is fed directly into the teleoperation loop at 30 Hz. In both settings, each rigid object is tagged with a marker rigid body that yields its 6-D pose.

2) Teleoperation Interface: At runtime, the NOKOV system (Sec. B-C1) streams the operator's wrist pose and fingertip positions together with the manipulated object's 6-D pose at 30 Hz. Each frame triggers two parallel control paths that together close the teleoperation loop at the same rate:

- **Arm.** The operator's wrist pose (world frame) is mapped to a 7-DoF Franka FR3 joint target via inverse kinematics.
- **Hand.** The operator's fingertip positions and the object's 6-D pose are converted to the wrist frame and assembled



(a) Motion capture setup. (b) Marker-instrumented gloves.

Fig. 6. **Hand–object motion capture setup.** (a) The dedicated capture volume equipped with tripod-mounted NOKOV infrared cameras. (b) Two glove configurations: the left glove, used for offline reference-motion collection, has dense markers on the wrist, palm, and all finger joints; the right glove, used for real-time teleoperation, has markers only on the wrist and fingertips.

into the co-tracking goal g_t. The learned policy then maps the current hand–object state and g_t to joint position targets sent to the dexterous-hand SDK via position control.

The co-tracking policy is trained on in-hand HOI references (Sec. C-F) and does not cover the pre-grasp approach. We therefore split each trial into two phases. In the *reaching and grasping phase*, the hand is driven by kinematic vector retargeting [13, 30], which maps the operator's fingertip positions to robot joints via vector alignment and handles the pre-grasp approach and initial contact acquisition. Once the hand reaches an approximate grasp pose near the object, the operator switches to the *in-hand phase*, where the co-tracking policy drives all subsequent in-hand reorientation, regrasping, and finger gaiting.

APPENDIX C
TELEDEXTER IMPLEMENTATION DETAILS

This section provides the full implementation details of **TELEDEXTER**. We follow the same notation as Sec. III unless stated otherwise.

A. Observation & Action Space

Observation Space The observation $\boldsymbol{o}_t$ concatenates the elements listed in Tab. VII (SharpaWave: $N_f = 5$, $n_{\text{dof}} = 22$; LeapHand: $N_f = 4$, $n_{\text{dof}} = 16$). Two design choices are worth highlighting: (i) since every other entry is expressed in $\mathcal{F}_{\text{wrist}}$, the gravity direction $\hat{\boldsymbol{g}}_{\text{wrist}}$ is the *only* signal that anchors the policy to the world frame and tells it the absolute orientation of the wrist; and (ii) we deliberately exclude joint velocities and contact forces, since both are noisy or unavailable on hardware and the policy can implicitly recover them from the kinematic state, previous action, and gravity cue.

Action Space Extending the residual joint-target parameterization of HORA [28] with a soft deadzone, the policy output $\boldsymbol{a}_t \in [-1, 1]^{n_{\text{dof}}}$ is converted into a joint command by

$$\begin{aligned} \boldsymbol{a}_t^{\text{cmd}} &= \text{clip}_{\text{joint}}\Big(\boldsymbol{a}_{t-1}^{\text{cmd}} + \alpha_a \cdot \text{deadzone}_\tau(\boldsymbol{a}_t)\Big), \\ \text{deadzone}_\tau(a) &= \text{sign}(a)\ \max\big(|a| - \tau,\, 0\big), \end{aligned} \tag{2}$$

with action scale $\alpha_a = 0.1$, deadzone threshold $\tau = 0.1$. Under this residual formulation, exactly outputting zero is hard for a Gaussian policy; the deadzone gives an explicit "hold-still" region in action space (any $\boldsymbol{a}_t$ within $\pm\tau$ produces zero delta), letting the policy actively choose to keep the current command rather than having to emit exactly zero.

B. Complete Reward Design

We give the complete form of the reward used to train the co-tracking controller. The reward couples a sparse *consecutive subgoal-reaching* term, a dense *tracking* term, and a small time penalty:

$$r_t \;=\; \underbrace{\mathbb{1}_{\text{reach}}(t)\, w_{\text{step}}(t)\, r_{\text{score}}(t)}_{\text{sparse subgoal}} \;+\; \underbrace{\alpha_{\text{dense}}\, r_{\text{dense}}(t)}_{\text{dense tracking}} \;-\; \underbrace{c_{\text{time}}}_{\text{time}}, \tag{3}$$

Subgoal Indicator $\mathbb{1}_{\text{reach}}(t)$ At each step, the active subgoal g_k is considered *instantaneously reached* when all tracking errors fall below their tolerances simultaneously: $e_{t,k}^{\text{pos}} < \epsilon_{\text{pos}}$, $e_{t,k}^{f} < \epsilon_{\text{tip}}$ for every fingertip $f \in \{\text{thumb}, \text{index}, \text{middle}, \text{ring}, \text{pinky}\}$, and $e_{t,k}^{\text{rot}} < \epsilon_{\text{rot}}$. The indicator $\mathbb{1}_{\text{reach}}(t)$ fires only when this condition is held for N_{stay} consecutive frames, where N_{stay} is resampled after every successful subgoal hit.

Step Weighting $w_{\text{step}}(t)$ After a subgoal g_k is hit, the next subgoal is drawn from the same reference trajectory at index $k' = k + \Delta k$, where the jump $\Delta k \in \mathbb{Z}$ is drawn from a uniform distribution whose range expands over training according to the curriculum (Sec. C-C); $|\Delta k|$ is the number of reference frames skipped between two consecutive subgoals. The step-weighting factor w_{step} scales the sparse bonus by this temporal gap so that larger jumps yield proportionally larger rewards and the policy is not biased toward exploiting trivially close subgoals. When the environment performs a cross-trajectory switch (Sec. C-D), w_{step} is set to a much larger fixed value for the reward computation at that step instead. Both values are listed in Tab. VIII.

Subgoal Score $r_{\text{score}}(t)$ The score blends per-fingertip and object tracking terms with fixed weights:

$$r_{\text{score}} = \alpha_s \Big[\; w_{\text{tip}}^{s} \sum_{f \in \mathcal{F}} \rho_f \;+\; w_{\text{obj}}^{s}\, (\rho_{\text{pos}} + \rho_{\text{rot}}) \;\Big], \tag{4}$$

where the fingertip set $\mathcal{F}$ consists of thumb, index, middle, ring, and pinky for SharpaWave, with the ring finger omitted for LeapHand. Each $\rho_\cdot$ is an exponential kernel applied to the corresponding tracking error—the per-fingertip distance $e_{t,k}^{f}$, the object position error $e_{t,k}^{\text{pos}}$, and the object rotation error $e_{t,k}^{\text{rot}}$ in radians:

$$\begin{aligned} \rho_f &= \exp\big(-\beta_f\, e_{t,k}^{f}\big), \\ \rho_{\text{pos}} &= \exp\big(-\beta_{\text{pos}}\, e_{t,k}^{\text{pos}}\big), \\ \rho_{\text{rot}} &= \exp\big(-\beta_{\text{rot}}\, |e_{t,k}^{\text{rot}}|\big). \end{aligned} \tag{5}$$

Outer scale $\alpha_s = 1.5$. The remaining blend weights and decay rates β (following Li et al. [18]) are listed in Tab. VIII.

Tab. VII. **Observation o_t: per-element dimensions.** $\hat{\cdot}$ denotes a target quantity; Δ denotes target $-$ current.

Element	Description	Formula	SharpaWave	LeapHand
$\boldsymbol{q}_t$	joint positions	n_{dof}	22	16
$\cos \boldsymbol{q}_t$	cosine of joint positions	n_{dof}	22	16
$\sin \boldsymbol{q}_t$	sine of joint positions	n_{dof}	22	16
$\boldsymbol{x}_t^o$	object position in $\mathcal{F}_{\text{wrist}}$	3	3	3
$\boldsymbol{q}_t^o$	object quaternion in $\mathcal{F}_{\text{wrist}}$	4	4	4
$\hat{\boldsymbol{g}}_{\text{wrist}}$	gravity direction in $\mathcal{F}_{\text{wrist}}$	3	3	3
$\hat{\boldsymbol{p}}_t^{\text{tip}}$	target fingertip positions	$3N_f$	15	12
$\Delta\hat{\boldsymbol{p}}_t^{\text{tip}}$	fingertip target $-$ current	$3N_f$	15	12
$\hat{\boldsymbol{x}}_t^o$	target object position	3	3	3
$\Delta\hat{\boldsymbol{x}}_t^o$	object-position target $-$ current	3	3	3
$\hat{\boldsymbol{q}}_t^o$	target object quaternion	4	4	4
$\Delta\hat{\boldsymbol{q}}_t^o$	object-quaternion target $-$ current	4	4	4
$\tilde{\boldsymbol{a}}_{t-1}$	previous low-level command	n_{dof}	22	16
Total $\lvert\boldsymbol{o}_t\rvert$			**142**	**112**

Dense Tracking $r_{\text{dense}}(t)$ Following Li et al. [18], a small dense signal shapes early exploration before any subgoal is reached:

$$r_{\text{dense}} = \sum_{i\in\mathcal{I}} w_i^d\,\rho_i + (1-\sigma_t)\left(w_{\text{pos}}^d\,\rho_{\text{pos}} + w_{\text{rot}}^d\,\rho_{\text{rot}}\right), \quad (6)$$

where $\mathcal{I}$ = {thumb, index, middle, ring, pinky, lvl1, lvl2}, with *lvl1* the MCP (root) knuckles and *lvl2* the medial (proximal) knuckles, each averaged across fingers, sharing the same exponential kernel form as the per-finger ρ_f. The $(1-\sigma_t)$ factor turns the dense object signal off early in training and ramps it in as the curriculum hardens (Sec. C-C). Weight values w_i^d and decay rates β are listed in Tab. VIII; the overall dense-reward scale in Eq. (3) is $\alpha_{\text{dense}} = 0.1$.

Time Penalty A constant per-step cost $c_{\text{time}} = 0.1$ pressures the policy to complete subgoals quickly and prevents it from lingering in locally stable but unproductive configurations.

C. Curriculum Schedule

Following the curriculum design of Li et al. [18], we progressively harden four difficulty knobs over training. Three of them are jointly controlled by a scalar progress factor σ_t that decays from 1 to $\sigma_{\min} = 0.7$, and simulator gravity follows its own schedule. The four knobs are:

- **Inter-subgoal step size** (driven by σ_t). After each successful subgoal hit (Sec. C-D), the next subgoal is drawn $|\Delta k|$ reference frames ahead, with the upper bound on $|\Delta k|$ growing from 40 frames ($\sim$ 0.67 s at the 60 Hz reference rate) to $\sim$ 80 frames ($\sim$ 1.33 s) over training. Closer subgoals are easier to reach, so the policy first learns to reach nearby subgoals and only later is asked to reach farther ones.
- **Action-masking duration** (driven by σ_t). The freeze duration of the random action mask (Sec. C-H) is sampled uniformly from $\{1,\ldots,d_t^{\max}\}$, with the upper bound $d_t^{\max}$ growing from 1 to 10 frames over training. Shorter freezes are easier to tolerate, so the policy first faces brief freezes and only later is exposed to longer ones.
- **Dense object reward** (driven by σ_t). The object-tracking term $(\rho_{\text{pos}} + \rho_{\text{rot}})$ in r_{dense} (Eq. (6)) is scaled by $(1-\sigma_t)$, growing from zero early on to its full value late, so the policy first learns to track the fingertips and only later learns to track the object pose.
- **Simulator gravity** (independent schedule). Gravity is annealed linearly from 0 to $-9.8\,\text{m/s}^2$ over the first 32 K environment frames, so the policy first learns to manipulate under reduced object weight and only later has to support the full deployment load.

Schedules The progress factor σ_t decays linearly over an annealing window of $T_\sigma = 25{,}600$ environment frames (counted per env), where t is the per-env frame counter:

$$\sigma_t = 1 - (1-\sigma_{\min})\cdot\min\!\left(t/T_\sigma,\,1\right).$$

Each σ_t-driven upper bound expands cubically toward its saturation value $u_{\max}$ from its initial value $u_{\min}$,

$$u_t^{\max} = u_{\min} + \frac{u_{\max}-u_{\min}}{1-\sigma_{\min}^3}\left(1-\sigma_t^3\right),$$

instantiated with $(u_{\min}, u_{\max}) = (40, 80)$ for the inter-subgoal step bound $k_t^{\max}$ and $(1, 10)$ for the action-mask duration bound $d_t^{\max}$.

D. Reset Conditions

Goal Reset Whenever a subgoal g_k is hit, the next reference trajectory τ_{next} and subgoal index k_{next} are sampled as

$$(\tau_{\text{next}},\,k_{\text{next}}) = \begin{cases} (\tau,\; k+\Delta k) & \text{with probability } 0.9, \\ (\tau' \sim \text{Unif}(\mathcal{D}_o),\; k) & \text{with probability } 0.1, \end{cases} \quad (7)$$

where the second branch is a cross-trajectory switch. where τ is the current trajectory, $\mathcal{D}_o$ is the reference set for the current object o, and Δk is drawn from a uniform distribution whose range expands over training according to the curriculum (Sec. C-C). The next subgoal is then read as the k_{next}-th frame of τ_{next}. A cross-trajectory switch swaps the trajectory while preserving the frame index. Its purpose is to train the policy

Tab. VIII. **Reward parameters.** Full set of constants used in the reward function (Eq. (3)), grouped by reward component.

Parameter	Value
Subgoal indicator	
ϵ_{pos} (object pos. tolerance)	1 cm
ϵ_{tip} (fingertip tolerance)	3 cm
ϵ_{rot} (object rot. tolerance)	10°
N_{stay} (dwell duration)	$\mathcal{U}\{5, 15\}$
Step weighting w_{step}	
in-traj	$\lvert\Delta k\rvert + 5$
cross-traj	100
Subgoal score r_{score}	
α_s (outer scale)	1.5
w^s_{tip} (per-fingertip)	0.5
w^s_{obj} (pos / rot)	2.0
Time penalty	
c_{time}	0.1

Parameter	Value
Kernel decay rates β	
β_{thumb}	100
$\beta_{\text{index}} = \beta_{\text{middle}} = \beta_{\text{ring}} = \beta_{\text{pinky}}$	90
β_{lvl1}	50
β_{lvl2}	40
β_{pos}	80
β_{rot} (rad)	3
Dense tracking r_{dense}	
w^d_{thumb}	1.0
$w^d_{\text{index}} = w^d_{\text{middle}} = w^d_{\text{ring}} = w^d_{\text{pinky}}$	0.8
w^d_{lvl1}	0.6
w^d_{lvl2}	0.4
$w^d_{\text{pos}} = w^d_{\text{rot}}$	1.5
α_{dense} (outer scale)	0.1

to handle transitions between different demonstrations rather than overfitting to any single one. This generality is required at deployment, where the operator's motion will not stay within any recorded trajectory.

Episode Reset An episode ends if (i) any joint velocity or object linear or angular velocity exceeds its safety bound, (ii) the object position error exceeds 15 cm, or (iii) the policy accumulates too many out-of-tolerance frames before reaching the next subgoal:

$$n_{\text{fail}} \leftarrow \begin{cases} n_{\text{fail}} + 1, & \text{frame out of tolerance,} \\ 0, & \text{subgoal hit,} \end{cases}$$

terminate if $n_{\text{fail}} > n^{\max}_{\text{fail}} = \eta_{\text{fail}} \lvert\Delta k\rvert$.

where "out of tolerance" means *any* of the five fingertip, object position, or object rotation errors is above threshold, and η_{fail} is the failure-tolerance scale, meaning the policy is allowed η_{fail} out-of-tolerance frames per unit of subgoal step before the episode terminates. We use $\eta_{\text{fail}} = 1.5$. Immediately after a cross-trajectory switch, $n^{\max}_{\text{fail}}$ is overridden by a flat 300 frames to accommodate the longer regrasp transition. On termination, the environment is re-initialized via *reference state initialization* (RSI) [18]: a random frame in the first 90% of the assigned trajectory sets hand DoFs from the retargeted $\boldsymbol{q}^*$ and the object pose from the reference.

E. Geometry-Aware Retargeting Details

The full geometry-aware retargeting objective, summarized in the main paper (Sec. III-B), is

$$\boldsymbol{q}^*_{1:T} = \arg\min_{\boldsymbol{q}_{1:T}} \sum_{t=1}^{T} \Big(\mathcal{L}^t_{\text{vec}} + \lambda_{\text{surf}} \mathcal{L}^t_{\text{surf}} + \lambda_{\text{pen}} \mathcal{L}^t_{\text{pen}} + \lambda_{\text{col}} \mathcal{L}^t_{\text{col}} \Big) + \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}}. \tag{8}$$

We define each loss term in Eq. (8) and specify the two-stage optimization procedure and loss weights below. Let $\mathcal{H}_t$ denote the robot hand model at frame t with joint configuration $\boldsymbol{q}_t$ and wrist pose $(\boldsymbol{p}^w_t, R^w_t)$, and $\mathcal{O}_t$ the manipulated object mesh transformed by the recorded object pose T^o_t. Each link of $\mathcal{H}_t$ carries a precomputed surface point cloud and, for self-collision, a set of body-attached collision spheres. All signed distances $\text{sdf}_{\mathcal{O}_t}(\cdot)$ from a point on the hand surface to the object (positive outside) are computed with NVIDIA Kaolin [12], which provides a differentiable point-to-mesh SDF used by both $\mathcal{L}^t_{\text{surf}}$ and $\mathcal{L}^t_{\text{pen}}$ below.

Vector Alignment $\mathcal{L}^t_{\text{vec}}$ We use the standard vector-retargeting loss of Handa et al. [13], Qin et al. [30] without modification: a weighted Huber on per-vector errors between the captured operator hand keypoints (Sec. C-F) and the corresponding robot vectors. We refer readers to the original papers for the exact loss form, keypoint vector set, and per-vector weights.

Surface Attraction $\mathcal{L}^t_{\text{surf}}$ In the second stage we incorporate the object mesh: points on the hand surface that fall within a threshold τ_{surf} of the object are pulled onto the surface,

$$\mathcal{L}^t_{\text{surf}} = \frac{1}{\lvert\mathcal{S}_t\rvert} \sum_{p \in \mathcal{S}_t} \text{ReLU}\big(\text{sdf}_{\mathcal{O}_t}(p)\big),$$

where $\mathcal{S}_t = \{p : \text{sdf}_{\mathcal{O}_t}(p) < \tau_{\text{surf}}\}$. This glues the contact-side of the hand to the object without forcing non-contacting links onto it.

Mesh Interpenetration $\mathcal{L}^t_{\text{pen}}$ A symmetric penetration penalty pulls any hand surface point that has entered the object back out:

$$\mathcal{L}^t_{\text{pen}} = \sum_{p \in \mathcal{H}_t} \text{ReLU}\big(-\text{sdf}_{\mathcal{O}_t}(p)\big).$$

The weights λ_{pen} and λ_{surf} are listed in the optimization paragraph below.

Self-Collision $\mathcal{L}^t_{\text{col}}$ Self-collision is computed between collision spheres on *different* fingers: for every pair of spheres (i, j) with finger indices $f(i) \neq f(j)$, radii r_i, r_j and centers

$\boldsymbol{c}_i, \boldsymbol{c}_j$,

$$\mathcal{L}_{\text{col}}^t = \tfrac{1}{2} \sum_{f(i)\neq f(j)} \text{ReLU}\big((r_i + r_j) - \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2\big).$$

Intra-finger sphere pairs are ignored because adjacent links are designed to touch.

Trajectory Smoothness $\mathcal{L}_{\text{smooth}}$ We adopt the temporal smoothness energy of Curobo [33] without modification: a log-cosh penalty on the velocity, a squared ℓ_2 penalty on the acceleration, and a squared ℓ_2 penalty on the jerk of the per-frame joint configuration $\boldsymbol{q}_t$, summed across the trajectory with internal coefficients fixed to the Curobo defaults.

Optimization The two stages are run sequentially per trajectory:

- **Stage 1 (vector retargeting).** We minimize $\sum_t \mathcal{L}_{\text{vec}}^t$ jointly over all frames $\boldsymbol{q}_{1:T}$ with Adam, using a GPU-batched FK implementation that follows the dex-retargeting design [30]. The learning rate is 10^{-3} and we run 6,000 iterations. This gives a strong but contact-blind initialization.
- **Stage 2 (geometry-aware post-optimization).** From the Stage 1 solution we minimize the remaining loss terms of Eq. (8) with Adam. The learning rate is 3×10^{-3}, we run 160 iterations, and the gradient norm is clipped to 1.0. The surface-attraction mask $\mathcal{S}_t$ is built once at the start of Stage 2 with $\tau_{\text{surf}} = 2$ and frozen for all iterations.

We set $\lambda_{\text{surf}} = 10$, $\lambda_{\text{pen}} = 2$, $\lambda_{\text{col}} = 10$, $\lambda_{\text{smooth}} = 0.1$ for both SharpaWave and LeapHand. The final per-frame robot configuration $\boldsymbol{q}_t^*$, together with the recorded object pose T_t^o, forms the reference motion consumed by RL.

F. Hand–Object Reference Motion Construction

For each object, we record 150 reference trajectories of unscripted hand–object interactions using the NOKOV MoCap system (Sec. B-C1) at 30 Hz. Each trajectory lasts 20 s (600 frames), giving a total of $\sim$ 50 minutes of interaction data per object. The interactions span three categories: (1) in-hand translation, (2) in-hand rotation, and (3) free-play combining arbitrary grasps, finger gaiting, and tool-use sequences. This diversity ensures the reference set covers the full range of contact modes the controller may encounter at deployment.

Fig. 7 visualizes short reference-motion clips for `Cylinder` and `Cuboid`, each showing the source MANO hand alongside the retargeted LeapHand and SharpaWave configurations. The retargeted robot hands accurately reproduce the operator's grasp poses and contact transitions across both embodiments, confirming that the two-stage retargeting pipeline (Sec. C-E) transfers contact-rich interaction structure despite the significant kinematic differences between the human hand and the two robot platforms.

G. Domain Randomization

We randomize hand and object dynamics, external perturbations, sensing noise, and observation latency to tolerate the sim-to-real gap. Each parameter is sampled independently at the start of every episode and held fixed throughout. The

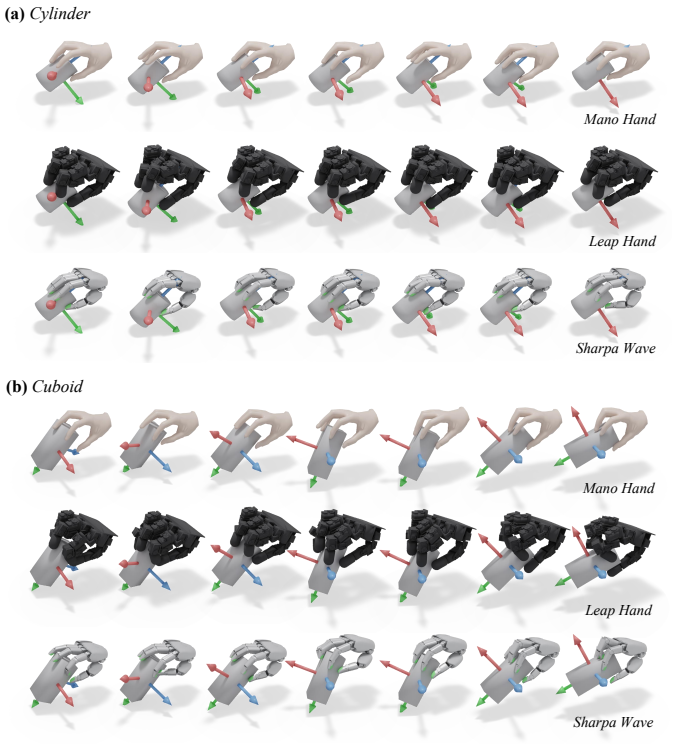


Fig. 7. **Hand–object reference motion visualization.** Retargeted motion clips for (a) `Cylinder` and (b) `Cuboid`. Each clip shows the source MANO hand and the corresponding retargeted LeapHand and SharpaWave sequences. Coordinate axes indicate the object 6-D pose.

implementation of random force perturbation follows VisualDexterity [5]. The full set of ranges is listed in Tab. IX.

H. Random Action Masking Details

Random action masking is the strong action-space regularizer introduced in Sec. III-A. It prevents the policy from overfitting to the perfectly synchronized actuation of simulation, which is unrealistic on hardware where actuator compliance, backlash, and PD response vary across DoFs.

Mechanism At each environment step, with probability $p_{\text{mask}} = 0.15$ *and only when no mask is currently active*, we sample a fresh mask: $n_m = 3$ DoF indices are drawn uniformly without replacement, and a freeze duration $d \sim \text{Uniform}\{1, d_t^{\max}\}$ is drawn (where $d_t^{\max}$ ramps from 1 to 10 following the curriculum schedule in Sec. C-C). For the next d control steps, the executed action $\tilde{\boldsymbol{a}}_t$ on the masked DoFs is overwritten with the previously commanded action, while the unmasked DoFs receive the current policy output:

$$\tilde{\boldsymbol{a}}_t[j] = \begin{cases} \tilde{\boldsymbol{a}}_{t-1}[j] & j \in \mathcal{M}_t \\ \boldsymbol{a}_t[j] & \text{otherwise} \end{cases}.$$

After d steps the mask deactivates, and a new mask can be sampled on the next step. Because the mask refreshes asynchronously across the $\sim$62 K parallel environments, training sees a wide spectrum of partially-stale joint commands.

Tab. IX. **Domain randomization ranges.**

Group	Parameter	Operation	Range
Hand dynamics	body mass	scaling	$\mathcal{U}[0.9, 1.2]$
	shape friction	absolute	$\mathcal{U}[1.0, 4.0]$
	DoF stiffness K_p	scaling	$\mathcal{U}[0.8, 1.2]$
	DoF damping K_d	scaling	$\mathcal{U}[0.8, 1.2]$
Object physics	body mass	scaling	$\mathcal{U}[0.5, 2.0]$
	surface friction	absolute	$\mathcal{U}[0.5, 4.0]$
	rolling / torsional friction	absolute	$\mathcal{U}[0, 0.05]$
	restitution	additive	$\mathcal{U}[0, 1.0]$
	mesh scale	scaling	$\mathcal{U}[0.95, 1.05]$
External force on object	trigger probability per step	—	$\mathcal{U}[0.01, 0.25]$
	force scale	constant	1.0
	exponential decay	constant	0.99
Sensing noise	joint position $\boldsymbol{q}_t$	additive Gaussian	$\sigma = 0.1$
	fingertip position	additive uniform	± 5 mm
	object position	additive uniform	± 5 mm
	object orientation	additive uniform	$\pm 2^\circ$
Observation latency	queue size	constant	2 frames
	queue sampling probability	constant	0.5
Initial state	wrist orientation	additive	$\pm 30^\circ$
	reference frame index	uniform	first 90% of clip

Sim-to-Real Effect Random action masking effectively augments the training distribution with desynchronized, partially stale joint commands. The policy is therefore forced to recover useful contact configurations even when some joints respond late or not at all, which closely matches the dominant failure modes we observe on the real hardware (motor lag, backlash, occasional missed commands on the SharpaWave SDK). Empirically, masking is the single most impactful sim-to-real intervention we tested (Tab. VI in the main paper; further analysis in Sec. A-E).

I. RL Training Hyperparameters and Compute

Tab. X provides the full set of network, PPO, and SAPG-specific hyperparameters used to train the controller. All our controllers are trained on 4 NVIDIA RTX 5090 GPUs running 15,600 parallel environments per GPU (62,400 environments in total). Each controller is trained for $\sim 10^{10}$ environment steps in a single RL stage, which takes approximately 1 day on this setup.

APPENDIX D
BASELINE IMPLEMENTATION DETAILS

This section describes how each baseline in Tab. I is implemented and what we change relative to the original release. All baselines are deployed on the same hardware platform and share the same teleoperation interface (Sec. B-C).

DexRT [13, 30] We use the open-source dex-retargeting codebase[1] with vector-alignment retargeting. The key parameter is the fingertip scaling factor, set to 1.0 for SharpaWave and 1.2 for LeapHand.

GeoRT [39] We follow the original paper and official implementation[2] to train and deploy the neural retargeter. The fingertip workspace data used for training is collected with our own inference glove to match the operator's hand kinematics at deployment.

DexGen [40] No official implementation is available for DexGen. We re-implement its foundation dexterity controller following the training and deployment recipe described in the original paper. Because the key intermediate step (the AnyGrasp-to-AnyGrasp RL policy) lacks sufficient detail for faithful reproduction, we substitute it with our co-tracking controller to generate the simulation rollouts, matching the data scale reported in the original paper. All subsequent stages (diffusion-based action prior training and deployment) follow the original design.

SimToolReal [16] SimToolReal is not a teleoperation method but an object-centric sim-to-real tool-use policy; we include it as a strong reference for learned dexterous manipulation. We follow the official implementation[3] and re-train on our hardware (Franka FR3 + SharpaWave right hand), adapting the simulation workspace and robot embodiment to match our deployment setup. The three tool categories (hammer, brush, screwdriver) and all other training details follow the original paper. We evaluate both the category-specific variant (SimToolReal†, one policy per tool) and the all-categories variant (SimToolReal‡, a single policy across tools), as reported in Tab. I.

[1] https://github.com/dexsuite/dex-retargeting

[2] https://github.com/facebookresearch/GeoRT

[3] https://github.com/tylerlum/simtoolreal

Tab. X. **Hyperparameters of SAPG.**

Hyperparameter	Value
Network	
LSTM hidden units	512
LSTM Layer Normalization	enabled
LSTM sequence length	4
MLP hidden sizes	[512, 1024, 1024, 512, 512]
Activation	ELU
PPO	
Learning rate	2×10^{-4}
LR schedule	adaptive
KL threshold	0.008
Num opt-epochs	4
Minibatch size (per GPU)	31,200
Horizon length	32
Discount (γ)	0.99
GAE λ	0.95
Clip range (ϵ)	0.2
Max grad norm	1.0
Bounds-loss coef.	10^{-4}
Parallel envs (per GPU)	15,600
SAPG	
Num blocks	6
Entropy Bonus Scale	0.005
Off-policy ratio	1.0
Mix ratio	0.5

APPENDIX E
AUTONOMOUS POLICY DETAILS

We adopt the Conv-UNet Diffusion Policy of Chi et al. [8] (DDPM noise predictor with 1-D temporal convolutions). The policy is conditioned on RGB observations from one third-person and one wrist-mounted camera, each encoded by a separate frozen DINOv2 ViT-S/14 encoder (Fig. 8). At each step the policy observes the last $T_o = 2$ frames and predicts an action chunk of length $T_p = 16$. Tab. II reports the stage-wise success of the three autonomous policies evaluated in the main paper (Sec. IV-B).

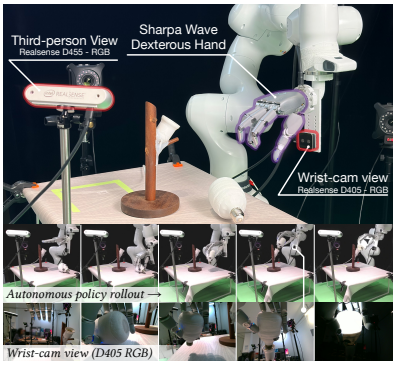


Fig. 8. **Autonomous policy setup and rollout.**