GENERATIVE NEURAL RETARGETING FOR HUMAN-TO-ROBOT DEXTEROUS MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Human demonstrations are a scalable data source for learning dexterous manipulation, but the embodiment gap prevents human motion from being executed directly on robots. Inverse kinematics (IK) retargets human motion to robots efficiently but ignores dynamics, often producing infeasible motions. Reinforcement learning (RL) and sampling-based model predictive control (MPC) are commonly employed to yield dynamically feasible motions, but both are sample-inefficient and sensitive to hyperparameters. RL suffers from costly and unstable training and tedious reward engineering; MPC avoids policy optimization, yet retargets each trajectory in isolation, and solving one does not make the next easier. Sampling cost grows rapidly with dataset size and task difficulty. We hypothesize that dynamically feasible trajectories concentrate near a low-dimensional manifold shared across demonstrations, so that retargeting can be reduced to sampling from that manifold, conditioned on human motion, rather than solving a fresh optimization problem for every demonstration. We propose **Generative Neural Retargeting** (GNR), which uses a flow matching model to sample feasible trajectories conditioned on human motion. GNR outperforms MPC with only 8.5% of the samples required by MPC, achieving a success rate of 56.20% compared to 27.20% for MPC. GNR can be used for scalable and efficient retargeting of large-scale and high-precision human demonstrations: by applying GNR within a real-to-sim data engine, we produce a dexterous manipulation dataset with dense contact-force labels, spanning 195k demonstrations and 3.2k object geometries.

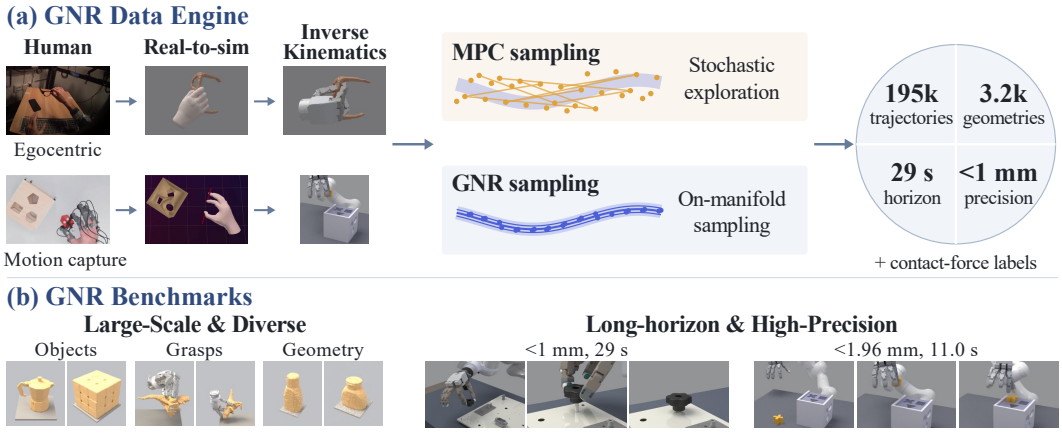


Figure 1: We propose GNR, a scalable and generalizable retargeting method that retargets large-scale and high-precision human demonstrations for five-fingered dexterous manipulation. We apply GNR with a real-to-sim data engine that reconstructs human motion from egocentric videos or a wearable exoskeleton with motion capture to scale up both diverse and difficult high-precision tasks.

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

1 INTRODUCTION

Progress in dexterous manipulation has been driven largely by the growing scale of demonstration data (O'Neill et al., 2024; Jiang et al., 2025; Hoque et al., 2026; Goel et al., 2023). Collecting such data directly on robots, e.g., through teleoperation, is prohibitively expensive: it demands extensive human labor and is constrained by hardware availability and embodiment-specific interfaces (Handa et al., 2020; Qin et al., 2023). A promising and scalable alternative is to record human hand manipulation from videos or wearable devices and transfer it to robots (Xu et al., 2025a; Banerjee et al., 2025; Zheng et al., 2026; Ma et al., 2026). However, a fundamental challenge in utilizing such data is that human demonstrations do not provide robot action labels, while the *embodiment gap* prevents human motions from being transferred directly to robots (Pan et al., 2025).

Prior work retargets human motions to robots through *kinematic* or *physics-based retargeting* (Pan et al., 2025; Mandi et al., 2025; Xu et al., 2025b;c; Allshire et al., 2025; Li et al., 2025; Qiu et al., 2026). Kinematic methods, e.g., fingertip IK, align human and robot poses geometrically. They are fast but ignore dynamics and contact, so the resulting trajectories are often dynamically infeasible (Handa et al., 2020; Qin et al., 2023). *Physics-based retargeting* instead optimizes for dynamically feasible motion through reinforcement learning (RL) (Qiu et al., 2026; Li et al., 2025; Allshire et al., 2025; Liao et al., 2026; Liu et al., 2025b; Adalibieke et al., 2026; Mandi et al., 2025; Tang et al., 2025) or sampling-based optimization methods such as model predictive control (MPC) (Pan et al., 2025; Williams et al., 2017; Xue et al., 2024). The retargeting objective is highly non-convex and discontinuous, which makes each trajectory expensive to solve. RL suffers from unstable and expensive training and tedious reward engineering (Liu et al., 2025b; Zhu et al., 2026; Mandi et al., 2025). Sampling-based MPC avoids costly policy optimization and directly optimizes the control sequence, but it can be sample-inefficient and sensitive to hyperparameters as every trajectory is solved with highly stochastic sampling and hand-designed heuristics (Pan et al., 2025; Paliwal et al., 2026; Mandi et al., 2025). The sample inefficiency and engineering burden create a scaling bottleneck for robot learning with human data, which motivates our research question: *how can we generate dynamically feasible robot trajectories from human demonstrations in a scalable and efficient way?*

A key observation is that conventional sampling-based retargeting repeatedly solves a non-convex optimization problem for every trajectory in isolation, and solving one does not make the next cheaper (Mandi et al., 2025; Liu et al., 2025b). However, solved trajectories can provide useful knowledge about feasible motion and contact when retargeting other demonstrations. Thus inspired, we propose **Generative Neural Retargeting** (GNR), a conditional flow matching model that learns from a small seed dataset generated by MPC to guide sampling toward dynamically feasible trajectories, conditioned on human motion. Scaling the training dataset can improve model performance (Yang et al., 2023), but generating the seed dataset can itself be expensive. GNR complements data scaling with *test-time scaling*: increasing the sampling budget improves retargeting success while reducing reliance on costly seed data. GNR generalizes to unseen motions, geometries, and objects, and even solves demonstrations on which MPC fails under the sampling budget used to generate GNR's training data.

We apply GNR within a real-to-sim data engine to address the scaling bottleneck of retargeting along two axes, diversity and difficulty, for dexterous manipulation. Along the **diversity** axis, the goal is a dataset spanning many distinct geometries and motions, where sampling cost grows linearly to solve more human trajectories, while each trajectory can take hours to optimize (Pan et al., 2025; Mandi et al., 2025; Zhu et al., 2026). Therefore, state-of-the-art methods often demonstrate a few thousand sequences derived from sources such as egocentric videos (Liu et al., 2025b; Adalibieke et al., 2026; Zhu et al., 2026; Paliwal et al., 2026), far short of the scale needed for large-scale, diverse dataset generation. Along the **difficulty** axis, the goal is solving long-horizon and high-precision multi-fingered dexterous manipulation. Errors compound over time, and millimeter-level precision leaves little tolerance for error, so the number of samples needed to discover a feasible trajectory grows rapidly (Ross et al., 2011; Yang et al., 2025). For example, one complex industrial assembly task inspired by NIST (Lian et al., 2021), a multi-stage, long-horizon task that requires 1 mm precision to screw a knob onto its seat (see Figure 21, App. K), requires 221 CPU core-h to optimize a trajectory averaging 29 seconds. Along the difficulty axis, existing works are evaluated primarily on tasks with 10 to 100 millimeter precision (Pan et al., 2025; Liu et al., 2025b; Mandi et al., 2025), a consequence of both the lack of high-precision data sources and the massive sampling required for retargeting. We further discuss existing retargeting datasets in App. A.1.

As illustrated in Figure 1, the data engine takes human demonstrations from large-scale egocentric video and high-precision motion capture to scale along both axes. A real-to-sim pipeline transfers them into SuperDex (The Mochi Team, 2026) and applies domain randomization to object pose, physical properties, and geometry. Along the **diversity** axis, GNR outperforms state-of-the-art sampling-based methods such as MPC (Xue et al., 2024; Pinneri et al., 2021) with about 8.5% of their sampling budget on HOT3D. On out-of-distribution (OOD) objects, motions, and geometries, it reaches 56.2% success versus around 30% for sampling-based methods, and solves 42.6% of demonstrations on which MPC fails. GNR reduces CPU cost per trajectory by 7.6–15.1×. Along the **difficulty** axis, GNR solves long-horizon, millimeter-precision tasks open-loop or by supplying MPC with on-manifold initializations. CPU cost per trajectory falls by $14.9\times$ on ShapeFilter and $3.0\times$ on NIST. Test-time scaling complements data scaling: with only 50 training demonstrations on ShapeFilter, GNR success rises from 11.5% to 99.5% as the sample budget grows to 256. Our contributions are:

Retargeting as generative modeling with test-time scaling. We recast physics-based retargeting from per-trajectory sampling-based optimization to generative modeling, GNR, which allows for efficient and generalizable robot action sampling conditioned on human motion. GNR demonstrates test-time scaling and generalizes to unseen geometries and motions, making it a scalable and cost-efficient method for retargeting large-scale, long-horizon, and high-precision human demonstrations to feasible robot trajectories.

A scalable and generalizable data engine. We develop a real-to-sim pipeline that extracts and retargets human demonstrations from egocentric videos and motion capture at scale. From HOT3D, it produces 195k paired human and feasible robot trajectories across 3.2k object geometries with dense contact-force labels. We also produce 3.4k retargeted trajectories for two long-horizon tasks requiring millimeter precision.

2 Related Work

Dexterous retargeting converts human demonstrations into robot trajectories (Pan et al., 2025; Mandi et al., 2025; Liu et al., 2025b; Zhu et al., 2026; Paliwal et al., 2026; Handa et al., 2020; Qin et al., 2023; Bahety et al., 2024; 2025; Abbatematteo et al., 2020; Abdul-Rashid et al., 2022). Kinematic retargeting aligns human and robot keypoints (Dariush et al., 2008a; Rohling & Hollerbach, 1993; Antotsiou et al., 2018; Dariush et al., 2008b; Yin et al., 2025), but does not enforce dynamic feasibility. Physics-based retargeting instead aims at generating feasible trajectories through reinforcement learning (RL) (Liu et al., 2025b; Zhu et al., 2026; Mandi et al., 2025) or sampling-based optimization (Pan et al., 2025; Paliwal et al., 2026; Xue et al., 2024). RL suffers from tedious reward engineering and unstable, costly policy optimization. Sampling-based methods such as MPC avoid costly policy training but optimize each trajectory in isolation and fail to utilize knowledge in solved trajectories. GNR amortizes this optimization through a learned generative model for efficient trajectory sampling. Additional related work is discussed in App. A.

Real-to-sim methods reconstruct simulation environments from real data (Torne et al., 2024; Jiang et al., 2022; Dai et al., 2024; Chen et al., 2024; Fang et al., 2025), and have been utilized for robot data generation from human demonstrations (Zhang et al., 2026; Qin et al., 2022). These datasets typically omit contact-force labels, with DexCanvas (Xu et al., 2025c) as an exception, but its labels describe interactions with an actuated MANO human-hand model rather than a robot hand. Our data engine labels contact forces on the robot hand and also uses GNR for large-scale and high-precision retargeting, scaling to 195k physically feasible trajectories across 3.2k geometries.

3 Generative Neural Retargeting

3.1 Retargeting as a Generative Problem

Physics-based retargeting problem. Retargeting takes a human demonstration, represented either as human hand motion or as a robot kinematic reference obtained by IK, and asks for a dynamically feasible control sequence $\boldsymbol{u}_{0:T-1}$ over T timesteps that reproduces the demonstrated behavior under physics. We denote the reference motion as $\bar{\boldsymbol{x}}_{0:T} = (\bar{\boldsymbol{x}}_0, \ldots, \bar{\boldsymbol{x}}_T)$, where each $\bar{\boldsymbol{x}}_t$ contains the robot and object states at timestep $0 \leq t \leq T$. We follow the physics-based retargeting formulation that casts retargeting as a constrained optimization problem (Pan et al., 2025), $\min_{\boldsymbol{u}_{0:T-1}} J(\boldsymbol{u}_{0:T-1})$,

subject to $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ for $0 \leq t < T$, $\boldsymbol{x}_{0:T} \in \mathcal{X}$, and $\boldsymbol{u}_{0:T-1} \in \mathcal{U}$, where $\boldsymbol{x}_t$ is the simulated robot and object state, f is the dynamics, $\mathcal{X}$ and $\mathcal{U}$ are the sets of admissible state and control sequences, and J is the retargeting cost (e.g., object tracking errors and force regularization).

The scaling bottleneck of sampling-based optimization. The retargeting problem is highly non-convex and discontinuous due to the contact-rich nature of dexterous manipulation (Pan et al., 2025). Sampling-based optimizers solve it by numerous stochastic rollouts (Pan et al., 2025; Mandi et al., 2025; Xue et al., 2024) but exploring with task-agnostic noise makes it sample-inefficient to discover feasible trajectories. Thus, several methods hand-design heuristics to mitigate the issue through, e.g., contact guidance, annealed noise schedules (Pan et al., 2025), and virtual spring-damper constraints (Mandi et al., 2025). Sample inefficiency and heuristics limit scalability. Furthermore, each trajectory is optimized independently, which fails to transfer knowledge across demonstrations.

Generative modeling with test-time scaling. We posit that feasible trajectories concentrate on a low-dimensional manifold that can be learned by a model that samples from it. Given a set $\mathcal{D}$ of human demonstrations, GNR learns from a seed dataset of N demonstrations retargeted by sampling-based methods such as DIAL-MPC (Xue et al., 2024), then retargets the remaining $|\mathcal{D}| - N$ demonstrations. Increasing N can improve model performance but requires costly seed generation. Test-time scaling complements data scaling by improving success through sampling, reducing reliance on seed datasets. Notably, GNR can outperform the seed generator and solve demonstrations on which it fails, expanding dataset coverage while reducing generation cost.

Flow-matching (Lipman et al., 2022). We model GNR as a conditional flow matching generator $G_\theta(\cdot \mid \boldsymbol{c})$ of residual corrections $\boldsymbol{\Delta} = \boldsymbol{u}_{0:T-1} - \bar{\boldsymbol{u}}_{0:T-1}$ to the IK reference controls $\bar{\boldsymbol{u}}_{0:T-1}$, conditioned on $\boldsymbol{c} = (\bar{\boldsymbol{x}}_{0:T}, \boldsymbol{o})$, where $\boldsymbol{o}$ encodes object geometry or pose, depending on the task. For training pairs $(\boldsymbol{\Delta}, \boldsymbol{c})$ from the optimizer-generated data, we independently sample standard Gaussian noise $\boldsymbol{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and flow time $\tau \sim \mathcal{U}[0, 1]$, set $\boldsymbol{z}_\tau = (1-\tau)\boldsymbol{z}_0 + \tau\boldsymbol{\Delta}$, and train the velocity field $\boldsymbol{v}_\theta$ with the flow matching loss $\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{(\boldsymbol{\Delta},\boldsymbol{c}),\boldsymbol{z}_0,\tau}\left[\|\boldsymbol{v}_\theta(\boldsymbol{z}_\tau, \tau, \boldsymbol{c}) - (\boldsymbol{\Delta} - \boldsymbol{z}_0)\|_2^2\right]$. At inference, we draw Gaussian noise $\boldsymbol{z}_0$ and integrate $d\hat{\boldsymbol{z}}_\tau/d\tau = \boldsymbol{v}_\theta(\hat{\boldsymbol{z}}_\tau, \tau, \boldsymbol{c})$ from $\hat{\boldsymbol{z}}_0 = \boldsymbol{z}_0$ to $\tau = 1$, yielding the residual $\hat{\boldsymbol{\Delta}} = G_\theta(\boldsymbol{z}_0 \mid \boldsymbol{c}) = \hat{\boldsymbol{z}}_1$ and retargeted trajectory $\boldsymbol{u}_{0:T-1} = \bar{\boldsymbol{u}}_{0:T-1} + \hat{\boldsymbol{\Delta}}$.

3.2 GNR Deployment

Open-Loop GNR Retargeting. GNR enables test-time scaling through its generative formulation: for a condition $\boldsymbol{c}$, we can draw S latents $\{\boldsymbol{z}_0^{(k)}\}_{k=1}^S$ and decode S candidate trajectories $\{\boldsymbol{u}_{0:T-1}^{(k)}\}_{k=1}^S$. Each candidate is replayed in simulation and checked against the task success criterion (Table 15, App. G). Retargeting succeeds if any of the S candidates passes, which we report as pass@S.

GNR for MPC Initialization. Closed-loop feedback remains valuable for long-horizon and high-precision tasks, where small errors compound over time and lead to catastrophic failure. GNR integrates naturally with MPC by supplying a high-quality on-manifold initialization, $\boldsymbol{u}_{0:T-1} = \bar{\boldsymbol{u}}_{0:T-1} + \hat{\boldsymbol{\Delta}}$, which MPC then refines (App. **??**).

4 A Human-to-Robot Data Engine

GNR accelerates physics-based retargeting and drives a human-to-robot data engine (Figure 2) which produces feasible robot trajectories with simulation-derived contact-force annotations across both difficulty and diversity axes. We describe the retargeting pipeline in Section 4.1 and the discovery and scaling phases for expanding dataset diversity in Section 4.2.

4.1 Human-to-Robot Retargeting

The data engine takes per-frame human hand motion, object poses, and an object surface mesh $\mathcal{M}$ as input. We segment multi-skill recordings using hand–object contact estimated from finger-to-surface distances, while using single-skill recordings directly. Object meshes are repaired to obtain watertight surfaces, and their signed distance fields Φ are precomputed for penetration checks. We transfer the captured motion to a simulated human hand, then apply keypoint-based inverse kinematics to retarget it to the robot. Together with the recorded object poses, the resulting robot configurations form the kinematic reference trajectory $\bar{\boldsymbol{x}}_{0:T}$. For segments starting with an object already grasped,

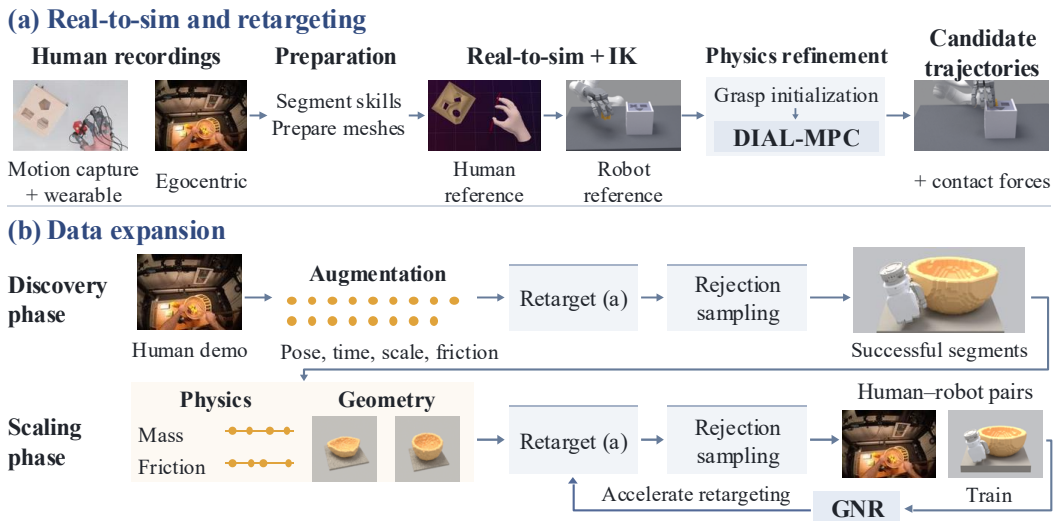


Figure 2: **The human-to-robot data engine.** (a) A pipeline retargets motion-capture recordings for the difficulty axis and egocentric demonstrations for the diversity axis into candidate robot trajectories with simulation-derived contact-force labels. (b) The discovery and scaling phases repeatedly apply (a) under varied conditions and filter candidates using the same success criterion Γ (Section 4.2), first identifying retargetable segments and then expanding their physical and geometric coverage. The resulting pairs train GNR to accelerate subsequent retargeting.

we re-establish the grasp by opening the robot hand, closing its fingers around the temporarily fixed object, and then releasing the object to move freely in simulation. To construct the initial training pairs for GNR, we use DIAL-MPC to optimize robot controls that track the reference object motion, recording simulation-derived contact forces during replay. Once trained, GNR accelerates subsequent retargeting in open loop or by initializing MPC for refinement. Implementation details are provided in App. B. For the difficulty axis, individually recorded motion-capture demonstrations are converted into paired training data through this pipeline without skill segmentation. For the diversity axis, egocentric demonstrations undergo the additional stages below.

4.2 Two-Phase Data Expansion

To expand coverage across diverse interactions, we apply the shared pipeline to egocentric demonstrations through two phases (Figure 2b). The discovery phase identifies retargetable segments, while the scaling phase expands them across object geometry and physical properties.

Discovery Phase. To avoid costly expansion of unsuccessful segments, we first search for feasible retargeting under a small set of candidate conditions. Reconstruction errors and human–robot morphology differences can hinder retargeting under the recorded conditions. We therefore search for feasible solutions by optimizing each segment under the nominal condition and sixteen single-axis perturbations of object position, orientation, start time, scale, or surface friction using the shared pipeline. However, low object-tracking error alone does not ensure valid physical interaction. Grasp initialization reduces initial penetration but does not prevent it during subsequent motion, and optimized controls may still produce excessive contact forces. For demonstrations with little object motion, low tracking error can also occur without a sustained grasp. We therefore retain only simulated trajectories $\xi = (\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T-1})$ with $\Gamma(\xi) = 1$, where $\Gamma(\xi) = \gamma_{\text{track}}(\xi) \wedge \gamma_{\text{pen}}(\xi) \wedge \gamma_{\text{grasp}}(\xi) \wedge \gamma_{\text{force}}(\xi)$. Here, γ_{track} bounds object position and orientation errors relative to the reference; γ_{pen} bounds finger-capsule penetration using Φ; γ_{grasp} requires a minimum sustained squeeze force; and γ_{force} limits per-finger contact force. Thresholds are provided in Table 15 (App. G). Accepted trajectories form the discovery dataset $\mathcal{D}_1$; only segments with at least one accepted trajectory proceed to the scaling phase. We also use Γ to evaluate GNR trajectories on HOT3D.

Scaling Phase. We expand successfully retargeted segments by varying object physics and geometry. For physical variation, we resample surface friction and compute object mass from mesh volume

and a density sampled from an object-specific range estimated by a large language model. Geometric variation includes mesh regeneration from the same image with different random seeds, anisotropic deformation, completion from partially masked images, regeneration after changing the object's orientation about its vertical axis, and single-axis stretching. We use SAM3D for image-based mesh generation and apply deformation and stretching directly to the meshes. For each variant, we optimize control sequences under the modified conditions using multiple optimizer seeds and retain trajectories satisfying the same criterion Γ used in discovery. The additional accepted trajectories form $\mathcal{D}_2$, yielding the MPC-generated robot dataset $\mathcal{D}_{\text{robot}} = \mathcal{D}_1 \cup \mathcal{D}_2$.

5 EXPERIMENTS

We evaluate GNR along the diversity and difficulty axes. We first describe the experimental settings (Section 5.1), then evaluate retargeting performance and efficiency using GNR on large-scale, diverse egocentric demonstrations (Section 5.2) and long-horizon, high-precision demonstrations (Section 5.3).

5.1 EXPERIMENTAL SETUP

Simulator & Embodiment. We use SuperDex Physics (The Mochi Team, 2026) as the simulator, which supports non-convex collision geometry and spatially distributed 3-D contact forces, enabling dense contact-force annotations. The target embodiment is a five-fingered Tesollo DG-5F hand for HOT3D, and a DG-5F hand mounted on a Franka FR3 arm for ShapeFilter and NIST.

GNR Training and Sampling. We implement GNR as a conditional flow matching model, which predicts the residual to correct the IK reference. The backbone is a temporal U-Net (Ronneberger et al., 2015) with feature-wise linear modulation (FiLM) (Perez et al., 2018) to inject the IK reference and object conditioning. For geometry conditioning, we use a Point Transformer V3 (Wu et al., 2024) encoder over 1,024 points. We sample from GNR with an Euler ordinary differential equation (ODE) solver (Lipman et al., 2022); model-specific settings are in App. G.1. We write S for the number of samples drawn per demonstration. Multiple samples can be drawn concurrently in a batch.

MPC Sampling for Seed Dataset Generation. We denote samples per MPC iteration by n_s and the maximum number of MPC iterations per call by M. Following the retargeting workflow in Section 4.1, we use DIAL-MPC (Xue et al., 2024) as both a retargeting baseline and the seed dataset generator for all tasks. We use 300 samples for 10 MPC iterations on HOT3D and 500 samples for 6 iterations on ShapeFilter. For NIST, the three stages use 86, 86, and 172 samples, respectively, for 10 iterations per stage.

Baselines. On HOT3D, we compare against the sampling-based methods MPPI (Williams et al., 2017), DIAL-MPC (Xue et al., 2024), iCEM (Pinneri et al., 2021), and CMA-ES (Hansen & Ostermeier, 2001), which use the cost function J defined in Section 3.1. Baseline results for ShapeFilter are discussed in App. C.6. We also compare GNR against baselines like SPIDER (Pan et al., 2025) on their public benchmarks (see App. F.2).

Metrics. The primary metric is success rate (SR), measured using task-specific criteria (Table 15, App. G). We additionally report object position error E_{pos}, object orientation error E_{rot}, and finger-joint angle error E_{finger} relative to the reference trajectory (Mandi et al., 2025; Liu et al., 2025b; Zhu et al., 2026). We compare CPU cost per trajectory in core-hours (core-h) or core-seconds (core-s), including inference and simulation. GPU training time is reported separately.

5.2 SCALING TO LARGE-SCALE AND DIVERSE RETARGETING

This section demonstrates GNR's sample efficiency and generalization advantages that enable scalable and efficient retargeting of large, diverse collections of human demonstrations.

Collecting demonstrations from egocentric data. Egocentric data has emerged as a cost-efficient source to expand data diversity and quantity for robot learning (Zheng et al., 2026; Ma et al., 2026). We use human demonstrations from the egocentric HOT3D dataset (Banerjee et al., 2025), extracting 780 source motions covering 33 objects with an average duration of approximately 2 s. The data en-

gine in Section 4 transfers these motions into simulation and constructs 361,920 augmented human demonstrations by varying object pose, geometry, physical properties, and motion timing.

Retargeting coarse motion references. However, reconstructed human motions from videos are often infeasible due to reconstruction errors. IK exacerbates infeasibility due to the embodiment gap. We therefore use physics-based retargeting. We employ DIAL-MPC (Xue et al., 2024), which succeeds on 95k of 361k demonstrations (26.43%) and requires 390k CPU core-h, equivalent to 31.7 days on 512 CPU cores. We refer to the successful and unsuccessful demonstrations as the MPC-success (95k) and MPC-failure (266k) datasets, respectively.

GNR training and evaluation. The costly MPC generation motivates the key questions: *does GNR outperform MPC while reducing sampling, and does GNR generalize to OOD or MPC-failure demonstrations?* To examine this, we randomly sample a small training dataset for GNR using 28k demonstrations (about 30% of the 95k MPC-success) covering 23 of 33 objects. Because running sampling-based baselines is prohibitively expensive, we randomly sample 1,000 MPC-success and MPC-failure demonstrations for evaluation. The test set comprises three OOD subsets: object (203), grasp (338), and geometry (459).

We compare GNR with state-of-the-art retargeting methods in Table 1, using SR and CPU cores as the primary metrics. Fingertip IK directly matches the fingertip reference and thus has the lowest E_{finger}, but achieves only 18.3% success. MPPI, DIAL-MPC, iCEM, and CMA-ES optimize costs that include object tracking errors and generally achieve lower E_{pos} and E_{rot}. However, optimization can get trapped in local minima with low tracking errors that still violate grasp or contact constraints, so lower tracking errors alone do not ensure success. These methods achieve only 27.2–34.7% success with 3,000 samples, whereas open-loop GNR reaches 52.2% and 56.2% with 128 and 256 samples, respectively. Compared with its seed generator DIAL-MPC, GNR reduces CPU cost per trajectory by 7.6–15.1×.

Table 1: Performance comparison on HOT3D, split into MPC-success and MPC-failure datasets.

			MPC-success				MPC-failure			
Method	S	core-s ↓	SR (%) ↑	E_{pos} (mm) ↓	E_{rot} (°) ↓	E_{finger} (°) ↓	SR (%) ↑	E_{pos} (mm) ↓	E_{rot} (°) ↓	E_{finger} (°) ↓
IK	*1*	*39.0*	*50.00*	*34.92*	*16.38*	*0.10*	*6.46*	*54.52*	*34.59*	*0.13*
DIAL-MPC	*3,000*	*2689.3*	*100.00*	*25.33*	*10.98*	*1.15*	*0.00*	*36.29*	*21.73*	*1.49*
MPPI	3,000	1849.6	71.32	28.04	11.96	1.22	11.40	36.92	20.59	2.07
iCEM	3,000	3862.1	78.68	21.96	7.07	1.74	18.27	26.04	10.92	2.14
CMA-ES	3,000	2912.7	72.79	23.21	8.55	1.02	16.76	32.89	14.55	1.08
GNR	1	48.9	48.16	34.96	16.26	1.22	6.87	54.48	33.07	1.24
GNR	64	90.9	88.97	27.82	13.74	1.05	34.89	37.10	18.29	0.97
GNR	256	354.1	**92.65**	28.04	13.99	1.05	**42.58**	36.18	17.58	0.98

* $E_{\text{pos}}, E_{\text{rot}}, E_{\text{finger}}$: object position, orientation, and finger-joint angle errors (medians); CPU core-s: mean per demonstration.

GNR's generalization. Figure 3 reports GNR's performance on three OOD subsets. Test-time scaling improves SR across all three: on MPC-success demonstrations, the mean SR across subsets rises from 46.2% at S=1 to 93.2% at S=256. With S=256, GNR also solves 31.9% to 58.4% of demonstrations that MPC fails to solve with 3,000 samples, expanding coverage beyond the seed optimizer with fewer samples.

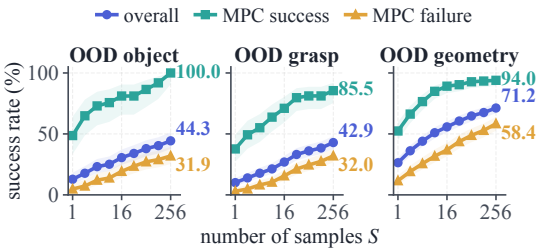
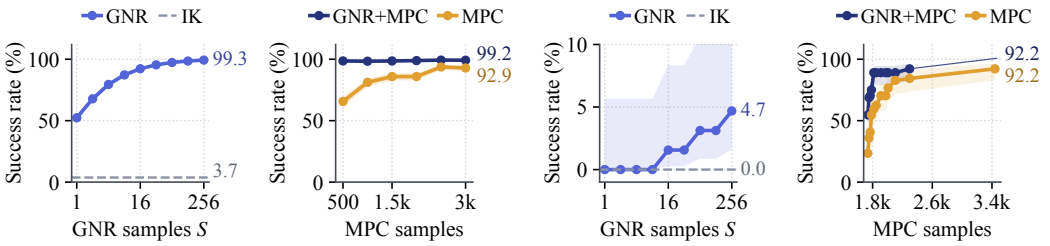


Figure 3: GNR performance on HOT3D across three OOD subsets. For each subset, we report GNR's SRs on MPC-success, MPC-failure, and all demonstrations.

On the 266k MPC-failure dataset, GNR solves 99,710 demonstrations (37.47%) with up to S=64 samples, using 2.1% of MPC's sampling budget. Together with the 95k MPC-success trajectories, this expands the dataset to 195k feasible trajectories across 3.2k geometries. App. E.4 reports the scaling curves for $S \leq 64$.

GNR's efficiency. Retargeting 361k augmented HOT3D demonstrations with DIAL-MPC requires 390k CPU core-h and yields 95k successful trajectories (26.43% coverage). Applying GNR to the



(a) ShapeFilter (b) NIST

Figure 4: High-precision, long-horizon retargeting. On ShapeFilter, open-loop GNR improves with the number of samples S, and GNR warm starts reduce MPC sampling requirements. On NIST, open-loop sampling alone is insufficient, while GNR+MPC outperforms MPC from scratch at matched optimizer budget. Shading shows 95% confidence intervals, approximated from reported SRs for the ShapeFilter MPC comparison. Evaluation settings are in App. G.1.

266k MPC-failure demonstrations with S=64 costs an 6.7k additional CPU core-h. This expands the dataset by 2.04× for 1.7% additional generation compute, reducing CPU cost per successful trajectory by 2.0×. Details are in App. H.

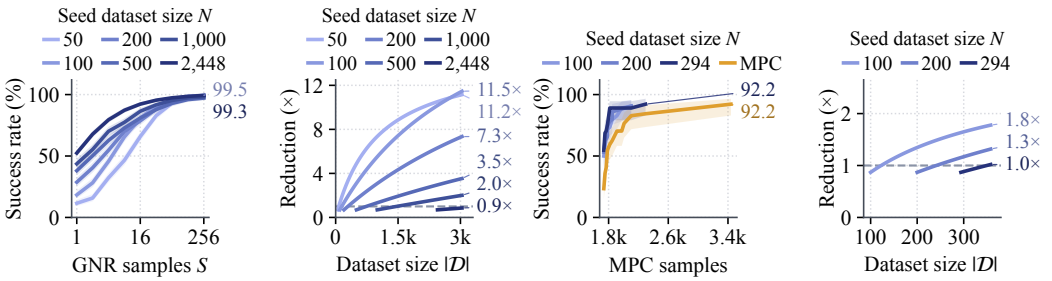
5.3 Solving High-Precision and Long-Horizon Retargeting

This section examines the scaling challenges along the difficulty axis and how high-precision human demonstrations and GNR improve dataset-generation efficiency. Furthermore, we apply GNR-generated data for policy learning using behavioral cloning and results are discussed in App. I.

Scaling challenges for high-precision and long-horizon retargeting. Existing dexterous retargeting methods primarily focus on shorter tasks that tolerate centimeter-level errors (Pan et al., 2025; Liu et al., 2025b; Xu et al., 2025b; Zhu et al., 2025) (Table 3, App. A), whereas precise manipulation often demands millimeter precision over long horizons. Scaling along this axis presents two challenges. *First*, human motion recovered from perception, as in HOT3D, may lack the precision needed for effective optimization. *Second*, obtaining feasible long-horizon and high-precision trajectories is expensive using RL and MPC. Addressing these challenges requires higher-quality human motion references and more sample-efficient retargeting.

Therefore, we collect human demonstrations using 15 OptiTrack cameras and an instrumented wearable exoskeleton to improve human motion precision (Furtado et al., 2019). We consider two tasks: 1) ShapeFilter inserts a cross-shaped part into a matching hole with a tolerance of 1.96 mm; its dataset contains 3,037 demonstrations augmented from 605 human demonstrations. 2) NIST, an industrial assembly task (Lian et al., 2021), involves grasping, transporting, aligning, and screwing a knob onto a threaded post; success requires horizontal error below 1 mm, screwing down by at least 8 mm, and axis error below 0.003 rad. Its dataset contains 358 demonstrations. Our real-to-sim pipeline reconstructs both tasks in SuperDex, where DIAL-MPC optimizes each reference trajectory for feasibility, providing the seed datasets and the baseline. Dataset and optimization details are in App. C and App. D; demonstrations are in App. K.1 and App. K.2.

GNR training and evaluation. We use 2,448/589 training/test demonstrations for ShapeFilter and 294/64 for NIST. Following Section 3.2, we evaluate GNR open-loop and as an on-manifold initialization for MPC. On ShapeFilter, IK achieves only 3.7% SR, while open-loop success rises from 52.4% at S=1 to 97.4% at S=64 and 99.3% at S=256. At S=64, GNR exceeds MPC's 92.9% SR with 3,000 samples (M=6, n_s=500), using only 1.6% of its CPU cost. Combined with MPC, GNR achieves 98.4% to 99.3% SR across M=1 to M=6 iterations and consistently outperforms vanilla MPC initialized from IK at each M, showing the benefit of on-manifold initialization (Figure 4a). On NIST, open-loop GNR reaches only 4.7% SR at S=256, compared with 0% for IK (Figure 4b), motivating closed-loop feedback for timely corrections over long horizons. With GNR initialization, MPC reaches 92.2% SR using 2,296 samples instead of 3,440 of vanilla MPC, a 33.3% reduction (Figure 4b).



(a) `ShapeFilter` (b) `NIST`

Figure 5: `GNR`'s efficiency in data generation. For each task, left: test-time scaling using `GNR` trained with different seed dataset sizes N. Right: CPU cost reduction as dataset size $|\mathcal{D}|$ grows. Costs include MPC seed generation and GNR generation.

`GNR` data and test-time scaling. Larger seed datasets can improve `GNR`'s performance but cost more to generate with MPC. We ablate the seed dataset size N in Figures 5a and 5b. On `ShapeFilter`, success at S=1 increases from 11.5% with 50 training demonstrations to 52.4% with 2,448. With only 50 demonstrations, test-time scaling raises success to 99.5% at S=256. On `NIST`, increasing the MPC sampling budget raises success from 54.7% to 92.2% with 294 training demonstrations and from 50.0% to 90.6% with 100. These endpoints use 33.3% and 38.8% fewer MPC samples, respectively, than MPC initialized from IK, which reaches 92.2%. Test-time scaling thus complements data scaling, reducing the need for costly MPC-generated seeds. App. H provides details.

Table 2: Comparing efficiency and SR using MPC and `GNR`.

Method	Samples	SR (%) ↑	core-s ↓
`ShapeFilter`			
MPC	2,500	93.7	10,124
`GNR`	77	94.7	678
`NIST`			
MPC	3,440	92.2	795,777
`GNR`+MPC	1,976	89.1	266,199

`GNR`'s efficiency. Table 2 reports per-trajectory success and CPU cost: `GNR` reduces cost by 14.9× on `ShapeFilter` and 3.0× on `NIST`. We then compute the total cost of generating each dataset, including MPC generation of N successful seeds and retargeting the remaining $|\mathcal{D}| - N$ demonstrations with `GNR` or `GNR`+MPC. For 3,037 `ShapeFilter` demonstrations, N=50, S=77 gives an 94.8% dataset success at 765.6 CPU core-h, versus 93.7% at 8,541.0 CPU core-h for MPC, an 11.2× compute reduction (Figure 5a). For 358 `NIST` demonstrations, N=100 with a total MPC sample budget of 1,976 gives an 92.1% dataset success at 44,516.7 CPU core-h, versus 92.2% at 79,135.6 CPU core-h for MPC, a 1.8× reduction (Figure 5b).

6 Conclusion

We propose Generative Neural Retargeting (`GNR`), which learns from dynamically feasible trajectories and amortizes physics-based retargeting across demonstrations. Increasing test-time sampling boosts `GNR`'s success, complementing data scaling when feasible training trajectories are costly. `GNR` generalizes across motions and geometries, achieves higher success than MPC with far fewer samples, and retargets demonstrations on which MPC fails. Our real-to-sim data engine scales along two axes: diversity, producing a retargeting dataset with 195k robot trajectories spanning 3.2k geometries; and difficulty, producing 3.4k robot trajectories for two challenging millimeter-precision, long-horizon tasks. Behavioral cloning policies trained on `GNR`-generated data succeed in simulation; future work will study their sim-to-real transfer.

AI USE STATEMENT

In this work, we used generative AI tools to design and provide feedback on research methodology and experiments, to clean and reformat datasets, and to support qualitative and thematic data analysis. Our data engine uses generative models as pipeline components (Section 4): SAM3D generates object geometry variants, a large language model estimates object density ranges, and vision-language models can segment skills in RGB-only video. Additionally, we used generative AI tools to create and modify scientific figures and images, to suggest experimental parameters, to create and edit software code, to produce supporting artifacts, to summarize and analyze existing literature, to discover research topics and identify gaps, to identify relevant literature, to format references, and to edit the paper for readability. We have reviewed all AI-assisted work. Specifically: (i) literature surveyed by AI was verified by the authors against the original publications; (ii) all generated artifacts, including figures and video demonstrations, were inspected to confirm that the underlying data are accurately and faithfully represented; and (iii) all AI-generated or AI-edited code and configurations were read, tested, and validated by the authors, and we additionally swept multiple parameter settings to confirm that our reported claims are robust and well supported. Any AI-assisted text was revised and approved by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments were conducted in simulation or on standard robotic platforms in controlled lab settings, and did not involve sensitive user data, or personally identifiable information. The authors declare no conflicts of interest, and no external sponsorship influenced the design, execution, or reporting of this work.

REPRODUCIBILITY STATEMENT

We take several steps to ensure the reproducibility of our results. The main text provides a detailed description of our model architecture, training objectives, and algorithmic design choices. Full hyperparameters, training configurations, hardware details, and ablation studies are reported in App. G.1.

REFERENCES

Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, Proceedings of Machine Learning Research, pp. 1289–1299. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/abbatematteo20a.html.

Hameed Abdul-Rashid, Miles Freeman, Ben Abbatematteo, George Konidaris, and Daniel Ritchie. Learning to infer kinematic hierarchies for novel object instances. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8461–8467. IEEE, 2022.

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

Arthur Allshire, Hongsuk Choi, Junyi Zhang, David McAllister, Anthony Zhang, Chung Min Kim, Trevor Darrell, Pieter Abbeel, Jitendra Malik, and Angjoo Kanazawa. Visual imitation enables contextual humanoid control. *arXiv preprint arXiv:2505.03729*, 2025.

Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Task-oriented hand motion retargeting for dexterous manipulation imitation, 2018. URL https://arxiv.org/abs/1810.01845.

Arpit Bahety, Priyanka Mandikal, Ben Abbatematteo, and Roberto Martín-Martín. Screwmimic: Bimanual imitation from human videos with screw space projection. *arXiv preprint arXiv:2405.03666*, 2024.

Arpit Bahety, Arnav Balaji, Ben Abbatematteo, and Roberto Martín-Martín. Safemimic: Towards safe and autonomous human-to-robot imitation for mobile manipulation. *arXiv preprint arXiv:2506.15847*, 2025.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7061–7071. IEEE, 2025.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pp. 306–324. Springer, 2024.

Boyu Chen, Yi Chen, Lu Qiu, Jerry Bai, Yuying Ge, and Yixiao Ge. Unit: Toward a unified physical language for human-to-humanoid policy learning and world modeling. *arXiv preprint arXiv:2604.19734*, 2026a.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7220–7232, 2026b.

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

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*, 2024.

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

Behzad Dariush, Michael Gienger, Arjun Arumbakkam, Christian Goerick, Youding Zhu, and Kikuo Fujimura. Online and markerless motion retargeting with kinematic constraints. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 191–198. IEEE, 2008a.

Behzad Dariush, Michael Gienger, Bing Jian, Christian Goerick, and Kikuo Fujimura. Whole body humanoid control from human motion descriptors. In *2008 IEEE International Conference on Robotics and Automation*, pp. 2677–2684. IEEE, 2008b.

Yu Fang, Yue Yang, Xinghao Zhu, Kaiyuan Zheng, Gedas Bertasius, Daniel Szafir, and Mingyu Ding. Rebot: Scaling robot learning with real-to-sim-to-real robotic video synthesis. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11351–11358. IEEE, 2025.

Yunhai Feng, Natalie Leung, Jiaxuan Wang, Lujie Yang, Haozhi Qi, and Preston Culbertson. A minimalist retargeting-guided reinforcement learning recipe for dexterous manipulation. *arXiv preprint arXiv:2607.11874*, 2026.

Harry Freeman, Chung Hee Kim, and George Kantor. Warped: Wrist-aligned rendering for robot policy learning from egocentric human demonstrations. *arXiv preprint arXiv:2604.10809*, 2026.

Joshua S Furtado, Hugh HT Liu, Gilbert Lai, Herve Lacheray, and Jason Desouza-Coelho. Comparative analysis of optitrack motion capture systems. In *Advances in Motion Sensing and Control for Robotic Applications: Selected Papers from the Symposium on Mechatronics, Robotics, and Control (SMRC'18)-CSME International Congress 2018, May 27-30, 2018 Toronto, Canada*, pp. 15–31. Springer, 2019.

Shubham Goel, Georgios Pavlakos, Jathushan Rajasegaran, Angjoo Kanazawa, and Jitendra Malik. Humans in 4d: Reconstructing and tracking humans with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14737–14748. IEEE, 2023.

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

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Ryan Hoque, Peide Huang, David Yoon, Mouli sivapurapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 4218–4237, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/07fcc6e2b89439d3ee5ab60939aaa6a0-Paper-Conference.pdf.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5606–5616. IEEE, 2022.

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

Jinhan Li, Yifeng Zhu, Yuqi Xie, Zhenyu Jiang, Mingyo Seo, Georgios Pavlakos, and Yuke Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. *arXiv preprint arXiv:2410.11792*, 2024.

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

Wenzhao Lian, Tim Kelch, Dirk Holz, Adam Norton, and Stefan Schaal. Benchmarking off-the-shelf solutions to robotic assembly tasks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1046–1053. IEEE, 2021.

Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Vincent Liu, Ademi Adeniji, Haotian Zhan, Siddhant Haldar, Raunaq Bhirangi, Pieter Abbeel, and Lerrel Pinto. Egozero: Robot learning from smart glasses, 2025a. URL https://arxiv.org/abs/2505.20290.

Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references, 2025b. URL https://arxiv.org/abs/2502.09614.

Juncheng Ma, Jianxin Bi, Yufan Deng, Xuanran Zhai, Kewei Zhang, Ye Huang, Bo Liang, Shukai Gong, Jiankai Tu, Xiaotian Tang, et al. Humanscale: Egocentric human video can outperform real-robot data for embodied pretraining. *arXiv preprint arXiv:2606.20521*, 2026.

Robert Jomar Malate, Erik Bauer, Norica Bacuieti, Stefanos Charalambous, Elvis Nava, Robert K Katzschmann, and Benedek Forrai. Smooth operator: A real-time sampling-based algorithm for kinematic hand retargeting. *arXiv preprint arXiv:2607.07491*, 2026.

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

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Bhawna Paliwal, Haritheja Etukuru, William Liang, Pieter Abbeel, Nur Muhammad Mahi Shafiullah, and Jitendra Malik. Do as i do: Dexterous manipulation data from everyday human videos. *arXiv preprint arXiv:2606.19333*, 2026.

Chaoyi Pan, Changhao Wang, Haozhi Qi, Zixi Liu, Homanga Bharadhwaj, Akash Sharma, Tingfan Wu, Guanya Shi, Jitendra Malik, and Francois Hogan. Spider: Scalable physics-informed dexterous retargeting, 2025. URL https://arxiv.org/abs/2511.09484.

Georgios Papagiannis, Norman Di Palo, Pietro Vitiello, and Edward Johns. R+ x: Retrieval and execution from everyday human videos. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8284–8290. IEEE, 2025.

Sungjae Park, Homanga Bharadhwaj, and Shubham Tulsiani. Demodiffusion: One-shot human imitation using pre-trained diffusion policy. *arXiv preprint arXiv:2506.20668*, 2025.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021.

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

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

Bin Qiu, Yanming Shao, Guanyu Cai, and Yao Mu. Enforcing human-like kinematics in dexterous piano playing via adversarial posture regularization. *arXiv preprint arXiv:2606.23848*, 2026.

Robert N Rohling and John M Hollerbach. Optimized fingertip mapping for teleoperation of dextrous robot hands. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 769–775. IEEE, 1993.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, pp. 654–665. PMLR, 2023.

Zhenhao Shen, Jiaqi Liang, Jasper Lu, Feng Jiang, Yuran Wang, Chuanbo Wei, Jiayi Liu, Jianchun Yang, Qize Yu, Jiadi You, et al. Ld4wam: Learning latent dynamics from human videos for world action models. *arXiv preprint arXiv:2608.22403*, 2026.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(1):153–188, 2025.

The Mochi Team. An implicit physics engine for contact-rich simulation, 2026.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

Hongwu Wang, Chenhao Yu, Youhao Hu, Jiachen Zhang, Yuanyuan Li, and Shaqi Luo. Bifrostumi: Bridging robot-free demonstrations and humanoid whole-body manipulation. *arXiv preprint arXiv:2605.03452*, 2026a.

Zhi Wang, Botao He, Kelin Yu, Seungjae Lee, Ruohan Gao, Furong Huang, and Yiannis Aloimonos. Humanego: Zero-shot robot learning from minutes of human egocentric videos. *arXiv preprint arXiv:2605.24934*, 2026b.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1714–1721. IEEE, 2017.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4851. IEEE, 2024.

Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025a.

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

Xinyue Xu, Jieqiang Sun, Jing, Dai, Siyuan Chen, Lanjie Ma, Ke Sun, Bin Zhao, Jianbo Yuan, Sheng Yi, Haohua Zhu, and Yiwen Lu. Dexcanvas: Bridging human demonstrations and robot learning for dexterous manipulation, 2025c. URL https://arxiv.org/abs/2510.15786.

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing, 2024. URL https://arxiv.org/abs/2409.15610.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

Yue Yang, Linfeng Zhao, Mingyu Ding, Gedas Bertasius, and Daniel Szafir. Boss: Benchmark for observation space shift in long-horizon task. *IEEE Robotics and Automation Letters*, 10(9): 8882–8889, 2025.

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)*, pp. 17376–17382. IEEE, 2025.

Yuhong Zhang, Zihan Gao, Shengpeng Li, Ling-Hao Chen, Kaisheng Liu, Runqing Cheng, Xiao Lin, Junjia Liu, Zhuoheng Li, Jingyi Feng, et al. Robowheel: A data engine from real-world human demonstrations for cross-embodiment robotic learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6664–6674, 2026.

Weiguang Zhao, Tian Liang, Xihao Guo, Rui Zhang, Irwin King, and Kaizhu Huang. Towards robotic dexterous hand intelligence: A survey. *arXiv preprint arXiv:2605.13925*, 2026.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

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

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

Appendix

A Extended Related Work **17**
A.1 Comparing Retargeting Datasets . . . 17
A.2 Dexterous Retargeting . . . 17

B Data Engine Details **18**
B.1 Real-to-Sim Transfer and Kinematic Retargeting . . . 18
B.2 Skill Segmentation . . . 18
B.3 Grasp Establishment . . . 19
B.4 Contact-Force Labels . . . 19
B.5 Geometry Variants . . . 19

C `ShapeFilter` **19**
C.1 Settings . . . 19
C.2 Evaluation . . . 20
C.3 Ablation of Conditioning . . . 20
C.4 Domain Randomization Settings . . . 21
C.5 Ablation of Residual Prediction . . . 21
C.6 Baselines . . . 22
C.7 Efficiency . . . 22
C.8 Trajectory Analysis . . . 23

D `NIST` **23**
D.1 Settings . . . 23
D.2 Optimization . . . 24
D.3 Efficiency . . . 24

E `HOT3D` **25**
E.1 Settings . . . 25
E.2 Data Generation . . . 25
E.3 GNR Training and Test Set . . . 25
E.4 Dataset Expansion . . . 26
E.5 Evaluation . . . 26
E.6 Trajectory Analysis . . . 28
E.7 Efficiency . . . 28

F Multi-Embodiment Evaluation **28**
F.1 Setup and Results . . . 28
F.2 Efficiency Comparison . . . 29

G Modeling and Training **30**
G.1 Evaluation and Optimization . . . 30
G.2 GNR Architecture and Training . . . 31

H Efficiency **31**
H.1 Compute Cost . . . 31
H.2 Cost versus Dataset Coverage . . . 32

I Behavior Cloning **32**

J Data Flywheels **33**

K Retargeting Demos **34**
K.1 `ShapeFilter` . . . 34
K.2 `NIST` . . . 35
K.3 `HOT3D` . . . 37

A Extended Related Work

A.1 Comparing Retargeting Datasets

Table 3 compares datasets used by recent retargeting works along the diversity and difficulty axes. As discussed in Section 1, the computational cost and engineering burden of physics-based retargeting limit scaling along both axes.

Along the diversity axis, most works evaluate on a few thousand trajectories with limited object geometries. We apply `GNR` within our real-to-sim data engine (Section 4) to scale both the number of trajectories and the range of geometries from egocentric videos in `HOT3D` (Banerjee et al., 2025). This produces 195k paired human and feasible robot trajectories.

Along the difficulty axis, existing works primarily evaluate on short-horizon tasks lasting a few seconds (Pan et al., 2025; Liu et al., 2025b), or use longer demonstrations lasting 10 to 30 seconds with success criteria that tolerate centimeter-level errors (Zhu et al., 2026; Mandi et al., 2025). Long-horizon tasks requiring millimeter-level precision remain underexplored in existing retargeting work. Such tasks require accurate human motion references, but motion recovered from egocentric videos may lack the precision needed for successful retargeting. We therefore use motion capture with a wearable exoskeleton to collect high-precision human demonstrations for `ShapeFilter` and `NIST`. These long-horizon, multi-stage tasks require millimeter-level precision and are, to our knowledge, among the *most challenging* tasks studied in dexterous retargeting.

Table 3: Recent physics-based retargeting methods on the two scaling axes: *diversity* (number of demonstrations and distinct geometries) and *difficulty* (task precision and average demonstration length).

Method	Demonstrations	Geometries	Precision (mm)	Horizon
DexTrack (Liu et al., 2025b)	3,585	257	100	short
AdaDexTrack (Adalibieke et al., 2026)	2,765	50	100	short
ManipTrans (Li et al., 2025)	3,300	1,200	30-80	short
DexMachina (Mandi et al., 2025)	7	5	10 to 90	short to long
SPIDER (Pan et al., 2025)	2,885	103	100	short
Do-As-I-Do (Paliwal et al., 2026)	500	N/A	100	short
DexCanvas (Xu et al., 2025c)	12,000	30	50	short to long
DexFlyWheel (Zhu et al., 2025)	2,000	80	20	short
CHORD (Zhu et al., 2026)	4,739	10	150	short to long
`GNR` (`HOT3D`)	**195k**	**3.2k**	150	short (2 s)
`GNR` (`ShapeFilter`)	3,037	1	**1.96**	long (11.0 s)
`GNR` (`NIST`)	358	1	**1.0**	long (**29 s**)

A.2 Dexterous Retargeting

Robot Learning from Human Data. Human video provides a scalable source of manipulation data (Shaw et al., 2023; Bharadhwaj et al., 2024; Papagiannis et al., 2025; Liu et al., 2025a; Ma et al., 2026). Robot systems increasingly learn from internet and egocentric video (Shaw et al., 2023; Wang et al., 2023; Li et al., 2024; Wang et al., 2026b; Freeman et al., 2026; Wang et al., 2026a; Zhao et al., 2026), but differences in morphology and actuation create an embodiment gap that prevents human motions from being transferred directly to robots (Chen et al., 2026a; Zhao et al., 2026; Wu et al., 2026). Retargeting has emerged as a promising approach to transfer human motions to robots (Pan et al., 2025; Mandi et al., 2025; Dariush et al., 2008a).

Kinematic Retargeting. Kinematic approaches match human and robot fingertips or keypoints and recover robot configurations with IK (Dariush et al., 2008a; Rohling & Hollerbach, 1993; Antotsiou et al., 2018; Dariush et al., 2008b; Yin et al., 2025; Handa et al., 2020; Qin et al., 2023; Malate et al., 2026). Source motion may come from motion capture systems, wearable hardware, or video (Bahety et al., 2024; 2025; Li et al., 2024; Park et al., 2025; Ma et al., 2026; Wu et al., 2026). Existing works have explored object-aware retargeting (Li et al., 2024), policy refinement for retargeting (Park et al., 2025), and visual synthesis to reduce geometric or visual mismatch (Freeman et al., 2026; Shen et al., 2026; Wu et al., 2026), but kinematic constraints alone do not ensure dynamically feasible trajectories (Malate et al., 2026).

Physics-based Retargeting. Physics-based methods produce dynamically feasible trajectories through RL (Liu et al., 2025b; Zhu et al., 2026; Mandi et al., 2025; Feng et al., 2026; Adalibieke et al., 2026) or sampling-based control (Pan et al., 2025; Paliwal et al., 2026; Xue et al., 2024; Malate et al., 2026). RL can suffer from costly and unstable policy optimization and often requires careful reward engineering (Feng et al., 2026; Pan et al., 2025). Sampling-based control avoids policy training but still optimizes each trajectory independently. GNR learns a generative model of dynamically feasible trajectories to improve sample-efficiency for retargeting new demonstrations. We deploy GNR open-loop, or use GNR to provide MPC with on-manifold initialization for contact-rich tasks. GNR generalizes to OOD demonstrations and solves demonstrations on which MPC fails, thus expanding the dataset while boosting efficiency.

B Data Engine Details

Optimizer hyperparameters are listed in App. G.1; HOT3D success thresholds and augmentation magnitudes appear in App. E.

B.1 Real-to-Sim Transfer and Kinematic Retargeting

Capture to simulated human hand. The capture system and simulator use different joint parameterizations, rest poses, and bone proportions. We transfer motion through world-space bone frames to account for these differences. After transforming the captured motion into the simulator's coordinate frame, we apply forward kinematics to recover 21 anatomical landmarks and construct 19 world-space bone frames. The simulator's IK maps these frames to a 33-degree-of-freedom configuration with the simulated hand's bone proportions. An orientation offset aligns the two rest poses. Mirroring is applied to both the source parameterization and target asset. Joint positions come from the anatomical landmarks, since the bone-frame origins coincide with the wrist in the rest pose.

Simulated human hand to robot. The DG-5F's limited thumb opposition and rigid palm prevent it from reproducing all human hand poses. We use the wrist pose and five fingertip positions as IK targets, initializing each frame with the previous solution to encourage temporal continuity. Our simulator uses a right-handed DG-5F asset, so we mirror the complete scene across the sagittal plane for left-handed segments. The resulting robot configurations and recorded object poses form the kinematic reference $\bar{\boldsymbol{x}}_{0:T}$.

B.2 Skill Segmentation

We divide multi-skill recordings into single-skill segments before retargeting; individually recorded single-skill demonstrations are used directly.

Contact heuristic. When 3D hand and object states are available, as in HOT3D, we estimate hand-object contact from their relative positions. A segment is a maximal run of frames in which at least two fingers lie within 20 mm of the object surface, measured with a k-d tree over the forward-kinematics landmarks. A set-down longer than 5 s breaks a segment, gaps shorter than 0.5 s are merged, and very short segments are discarded. Each segment starts at contact onset and extends past release. Segments are labeled left-handed, right-handed, or bimanual. Applied to HOT3D, this yields roughly 5,000 segments over 33 objects: about 40% bimanual, 40% right-handed, and 20% left-handed. The subset used by the data engine is reported in Table 11.

Vision-language segmenter. When only RGB video is available, Qwen3-VL labels the contact state of each hand in sampled frames. We convert these labels into contact and separation intervals, discard isolated single-frame transitions, and merge gaps shorter than 0.7 s. Claude reviews montages of the resulting segments and corrects hand and object assignments. The HOT3D dataset uses the contact heuristic.

B.3 Grasp Establishment

Segments may begin with the object already grasped. IK aligns the fingertips but can leave the DG-5F palm or knuckles intersecting the object. This initial penetration can eject the object or trap it against the hand, producing low tracking error without a valid grasp.

We therefore re-establish the grasp in simulation. The hand starts open and closes around the object while the object is fixed in place with gravity disabled. After roughly 12 frames, the constraint is gradually released and gravity is restored. Force and penetration are evaluated after release.

B.4 Contact-Force Labels

The object collider is a signed distance field. A penalty contact model computes the normal force from penetration depth and the surface normal from the field gradient. We replay the optimized control sequence open loop with the same proportional-derivative (PD) controller, recording the position, normal, penetration depth, and force at every contact point.

Contact stiffness affects the magnitude of the recorded forces. We use a value an order of magnitude below the simulator default, with lower finger joint gains.

We repair object meshes to be watertight so the signed distance field is well defined. Physics runs at a higher rate than control to reduce penetration between control updates. A short temporal median filter, applied per finger and force component, removes isolated outliers.

B.5 Geometry Variants

The physics branch resamples object density and surface friction; the geometry branch modifies the object mesh.

Table 4 lists the five geometry augmentations. Deformation and stretching preserve vertex correspondence, allowing grasp contact points to transfer to the variant. The other three regenerate the mesh from a reference image with SAM3D (Chen et al., 2026b) and do not preserve this correspondence. Together, they define 200 variants per object, or 6,600 possible meshes across the 33 source objects.

We replace the object mesh in a scene that passed discovery, then re-optimize the control sequence and evaluate it with the same criterion Γ. Sampling 10 geometry variants for each of 728 source motions, with 24 optimizer seeds per pair, gives 174,720 trials. The 7,280 motion-variant pairs involve 3,976 distinct meshes, since motions involving the same object draw from a shared variant pool. Of these, 2,723 yield at least one accepted trajectory. Together with the 33 original meshes, they give the 2,756 geometries in the MPC-success dataset (Table 11).

Table 4: Geometry augmentations and variants per object.

Edit	Implementation	Variants per object
Seed	Regenerate from the reference image with a new random seed	20
Deformation	Anisotropic deformation of the original mesh	60
Occlusion completion	Mask part of the reference image and complete the shape	20
Rotation	Rotate about the vertical axis, then regenerate	20
Stretch	Scale the original mesh along a single axis	80
Total		200

C ShapeFilter

C.1 Settings

Task. A Tesollo DG-5F hand on a Franka FR3 grasps a cross-shaped part and inserts it through a shape-sorter aperture. The 29-dimensional action comprises end-effector translation, a continuous 6-D rotation, and 20 hand joints; control runs at 20 Hz and physics at 60 Hz.

Dataset. Pose augmentation of 605 human demonstrations produces 3,037 demonstrations and 665,770 frames. Unless otherwise stated, we use the motion split with 2,448 training and 589 test episodes from 484 and 121 base trials, respectively.

Success criterion. The object centroid enters the box-local region $[-0.08, 0.08] \times [0, 0.16] \times [-0.08, 0.08]$ m, indicating insertion completion. The insertion tolerance is 1.96 mm.

C.2 Evaluation

The spatial split assigns complete human trials by initial object position, with 2,348 training and 689 test episodes from 453 and 152 base trials, respectively. The test quadrant is separated from training support by $7.14\times$ the spatial scale within the training set.

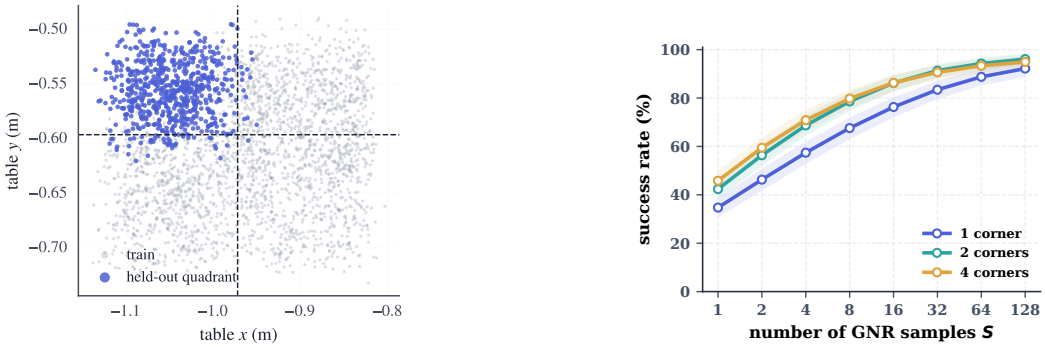


Figure 6: `ShapeFilter` spatial generalization. The test quadrant is shown at left. At right, GNR models trained with residual prediction and one, two, or four spatial corners are compared on the same 256-example test set; bands are 95% bootstrap intervals.

Figure 6 compares performance of one-, two-, and four-corner trained `GNR`. Table 5 reports motion-OOD scaling; IK succeeds on 3.74% of the motion test set.

Table 5: `ShapeFilter` motion-OOD test-time scaling.

Samples S	1	2	4	8	16	32	64	128	256
Success (%)	52.36	67.79	79.47	87.23	92.26	95.44	97.37	98.55	99.32

C.3 Ablation of Conditioning

We compare three conditioning inputs. State uses the IK reference and simulator object state; geometry replaces the object state with a 1,024-point scene cloud; geometry + pose provides both.

Table 6: `ShapeFilter` conditioning ablation. We report success rates (%) across different numbers of `GNR` samples S using different conditioning.

Samples S	State	Geometry	Geometry + pose
1	50.42	50.93	**53.48**
2	67.06	66.38	**69.10**
4	78.44	**80.31**	78.61
8	87.10	**88.62**	86.59
16	92.36	93.72	**94.06**
32	95.93	96.77	**97.11**
64	97.45	98.13	**98.81**
128	98.47	98.81	**99.15**
256	99.32	**99.49**	99.32

C.4 Domain Randomization Settings

We apply domain randomization to augment 605 human demonstrations to 3,037 demonstrations. Perturbations are sampled once per episode. The receptacle body and lid move rigidly together, while vertical translation applies to the complete scene. We randomize object and scene poses over the ranges in Table 7.

Table 7: `ShapeFilter` domain-randomization settings.

Scene element	Perturbation	Sampling range
Cross-shaped part	horizontal position	$[-30, 30]$ mm per axis
Cross-shaped part	yaw	$[-10, 10]^\circ$
Receptacle and lid	horizontal position	$[-30, 30]$ mm per axis
Receptacle and lid	yaw	$[-2.5, 2.5]^\circ$
Complete scene	vertical position	$[-20, 20]$ mm

Captured demonstrations add variation beyond these ranges. For example, initial cross positions span approximately ± 167 mm and ± 121 mm along the two table axes.

C.5 Ablation of Residual Prediction

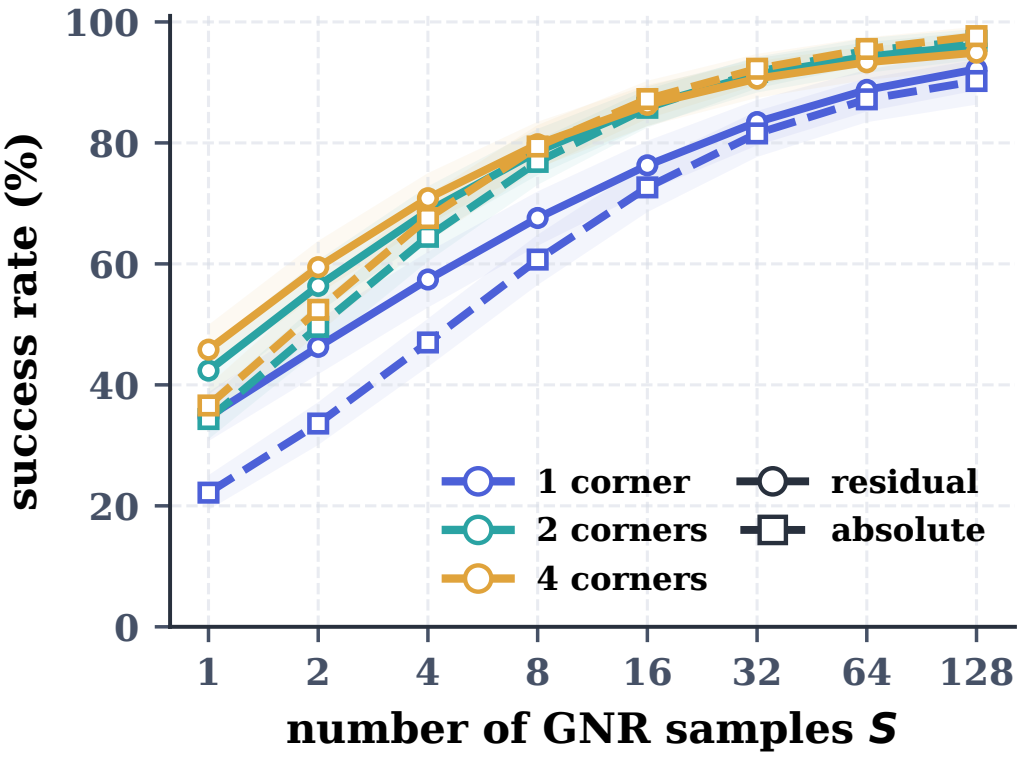


Figure 7: Residual and absolute prediction with one, two, or four training corners on 256 test episodes. Curves show expected best-of-S success from 128-candidate pools; bands are 95% bootstrap intervals.

We compare absolute and residual prediction on `ShapeFilter` with one, two, or four training corners.

With one, two, and four training corners, residual prediction reaches 34.72%, 42.35%, and 45.82% success at S=1, and 92.19%, 96.09%, and 94.92% at S=128 (Figure 7), showing that the residual prediction is beneficial especially when S is small.

C.6 BASELINES

Table 8: `ShapeFilter` success and per-trajectory compute cost. Optimizer budgets are samples per iteration n_s × MPC iterations M; GNR budgets are complete trajectory samples S. GNR cost includes inference and replay at 37.96 core-s per evaluated candidate, with early stopping on success, as in Table 9.

Method	Samples per iteration × MPC iterations M, or Samples S	Test episodes n	Success (%)	Core-s
DIAL-MPC	32×6	64	48.4	639
DIAL-MPC	500×6	589	92.9	12,939
CEM	32×6	64	50.0	1,045
CEM	500×6	64	82.8	18,737
iCEM	32×6	64	65.6	729
CMA-ES	32×6	64	40.6	1,045
GNR	S=8	589	87.23	103
GNR	S=16	589	92.26	134
GNR	S=32	589	95.44	170
GNR	S=64	589	97.37	212

C.7 EFFICIENCY

Tables 9 and 19 report success and generation cost. Figure 8 compares total generation cost as dataset size grows; App. K.1 shows qualitative comparisons.

Table 9: `ShapeFilter` dataset-generation efficiency. Success rates combine the seed set with test-set success for the remaining demonstrations. Costs include MPC seed generation, GNR inference, and simulation, in thousands of core-h.

Seed set size N	Quantity	Samples S								
		1	2	4	8	16	32	64	128	256
50	Success (%)	13.0	17.2	33.4	47.4	66.4	83.1	93.2	96.7	99.5
	Cost (k core-h)	0.235	0.263	0.313	0.390	0.501	0.627	0.740	0.832	0.890
100	Success (%)	21.0	30.5	47.5	66.7	81.3	91.1	96.4	98.7	99.3
	Cost (k core-h)	0.438	0.463	0.504	0.560	0.626	0.688	0.745	0.789	0.828
200	Success (%)	33.4	44.3	57.7	71.9	80.0	89.1	94.9	96.8	99.0
	Cost (k core-h)	0.843	0.865	0.898	0.944	1.005	1.081	1.163	1.243	1.311
500	Success (%)	48.2	58.4	68.8	77.3	84.1	89.8	94.0	96.6	97.6
	Cost (k core-h)	2.061	2.077	2.102	2.137	2.186	2.251	2.330	2.417	2.526
1,000	Success (%)	62.2	70.1	80.0	84.9	90.9	94.4	96.8	98.2	98.7
	Cost (k core-h)	4.089	4.101	4.118	4.141	4.172	4.209	4.252	4.299	4.362
2,448	Success (%)	90.8	93.8	96.0	97.5	98.5	99.1	99.5	99.7	99.9
	Cost (k core-h)	9.964	9.967	9.970	9.974	9.979	9.985	9.992	10.000	10.008

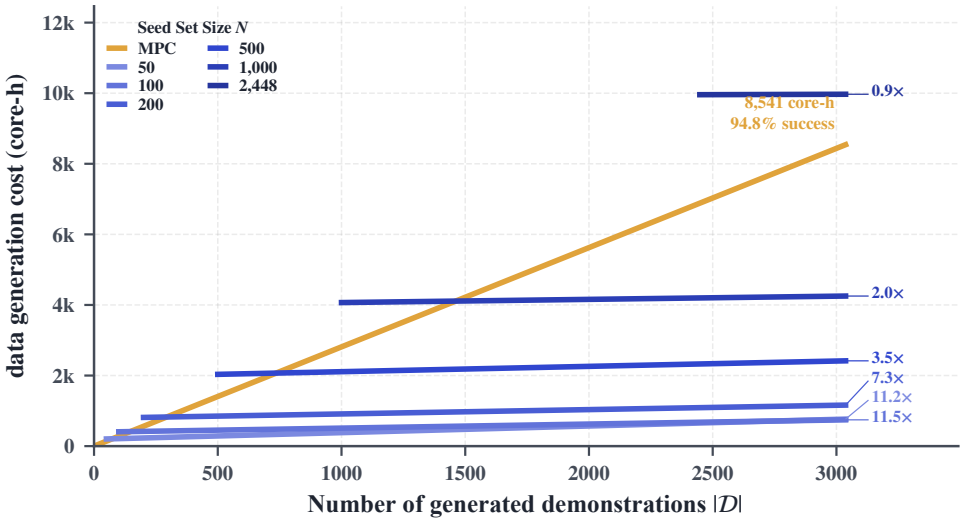


Figure 8: `ShapeFilter` generation cost versus dataset size. MPC generates all $|\mathcal{D}|$ demonstrations; GNR uses MPC for N successful seeds and generates the remaining $|\mathcal{D}| - N$ with configurations that meet or exceed the best MPC success rate. Numbers show the reduction in total cost.

C.8 Trajectory Analysis

Figure 9 compares tracking error and smoothness.

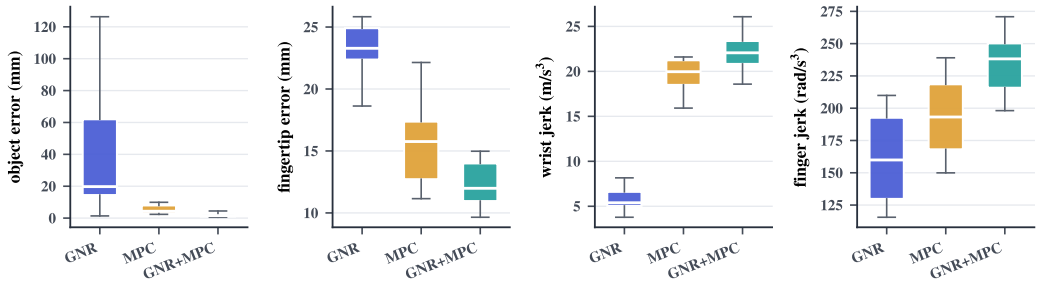


Figure 9: `ShapeFilter` trajectory errors and jerk over 11 paired episodes. GNR has lower jerk; GNR+MPC has the lowest object and fingertip tracking errors.

D NIST

D.1 Settings

Task. In a bimanual Franka/DG-5F system, one arm grasps and screws an M8 knob onto a threaded stud while the other supports the board. The 29-dimensional operational-space action contains translation, continuous 6-D rotation, and 20 hand joints.

Dataset. The dataset contains 358 successful demonstrations and 206,946 frames from 412 IK seeds. At 20 Hz, the mean trajectory duration is 29 s.

Success criterion. Success requires horizontal error below 1 mm, screwing down by at least 8 mm, and axis error below 0.003 rad.

D.2 Optimization

Optimization has three phases. Wrist alignment uses 86 samples for ten iterations over seven steps at stride two. Finger closure uses 86 samples for ten iterations over six steps. The final stage uses 172 samples for ten iterations over 13 steps at stride four and 200 Hz physics, using approximately 83% of the compute.

D.3 Efficiency

MPC sample budgets sum all three stages: $86 \times 10 + 86 \times 10 + n_s M$. The first two stages contribute 37.28 core-h per attempt. The complete three-stage optimization are 221.05 core-h (6.32 wall-hours) for MPC and 73.94 core-h (about 2.11 wall-hours) for GNR+MPC in Table 2, including `GNR` inference. Qualitative results are in App. K.2. Table 10 lists the configurations used for cost calculation; Figure 10 compares methods at approximately matched dataset success.

Table 10: `NIST` dataset-generation efficiency. Success rates combine the seed set generation and `GNR` generation for the remaining demonstrations. Each MPC budget uses the schedule with the highest success rate. `GNR` generation costs include MPC seed generation, GNR inference, and GNR-initialized MPC, in thousands of CPU core-h (App. D.3).

N=100			N=200			N=294		
MPC samples	Success (%)	Cost (k core-h)	MPC samples	Success (%)	Cost (k core-h)	MPC samples	Success (%)	Cost (k core-h)
1,736	64.0	35.90	1,736	79.3	57.34	1,736	91.9	77.41
1,752	70.7	36.79	1,752	83.4	57.89	1,752	93.3	77.61
1,768	74.1	37.42	1,784	89.0	58.91	1,784	95.5	77.97
1,784	79.7	38.17	1,816	90.3	59.08	1,816	97.2	78.37
1,816	80.9	39.81	1,848	93.8	60.10	1,912	97.8	78.95
1,848	85.4	41.02	2,104	95.9	64.46	–	–	–
1,912	87.6	42.51	–	–	–	–	–	–
1,976	92.1	44.52	–	–	–	–	–	–
2,104	93.2	45.91	–	–	–	–	–	–

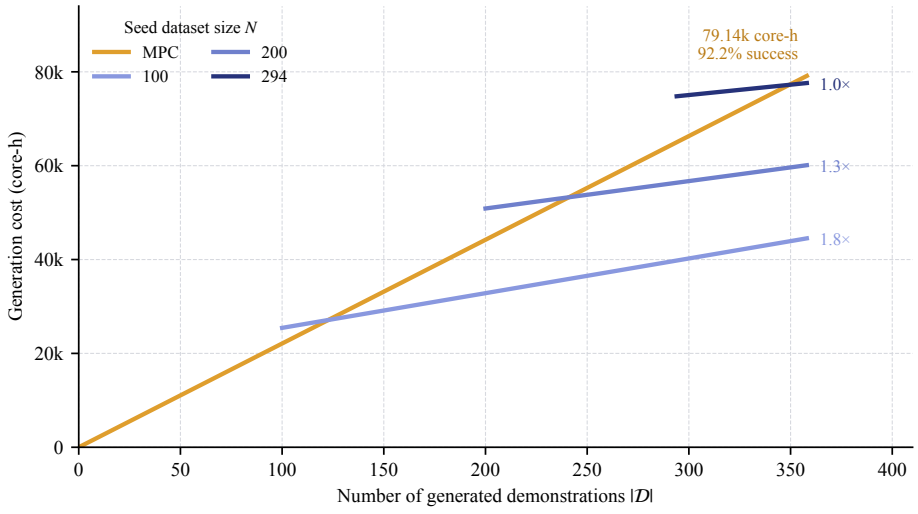


Figure 10: `NIST` generation cost versus dataset size. MPC generates all $|\mathcal{D}|$ demonstrations; GNR+MPC uses MPC for N successful seeds and generates the remaining $|\mathcal{D}| - N$ with measured configurations at approximately matched dataset success. Numbers show the reduction in total cost. Cost computation follows App. D.3.

E HOT3D

E.1 SETTINGS

Task. Egocentric hand-object recordings are retargeted to the DG-5F hand with 26-dimensional joint control.

E.2 DATA GENERATION

The data engine augments 780 human motions from 210 recordings and 33 objects by varying pose, timing, rotation, scale, friction, and geometry. DIAL-MPC succeeds on 95,670 of 361,920 attempted trajectories (26.43%). Adding 99,710 trajectories solved by GNR with up to $S{=}64$ expands the dataset to 195,380 trajectories (195k; App. E.4). They add 428 geometries, expanding coverage from 2,756 to 3,184 (3.2k). Table 11 reports the data-generation statistics.

Table 11: HOT3D data-generation statistics. The first branch augments object pose, motion timing, mass, and friction; the second changes object geometry.

Quantity	Pose, timing, and physics	Object geometry	Total
Attempted trajectories	187,200	174,720	361,920
DIAL-MPC successes	47,621	48,049	95,670
DIAL-MPC failures	139,579	126,541	266,120
Technical omissions	0	130	130
Source motions	780	728	780
Object geometries	33	3,976	4,009
MPC-success geometries	33	2,723	2,756
GNR successes ($S{=}64$)	58,318	41,392	99,710
Combined successes	105,939	89,441	195,380
Combined successful geometries	33	3,151	3,184

Augmentation. During discovery, the data engine evaluates the nominal condition and 16 perturbations of object translation, start time, scale, rotation, and friction. Translation is shifted by 15 mm in either direction along the palm normal or grasp axis; time is shifted by -5, -3, $+3$, or $+5$ frames; scale is multiplied by 1.15 or 1.30; rotation is shifted by 10° in either direction about the up or grasp axis; and friction is multiplied by 0.5 or 2.0. Geometry augmentation adds reconstruction variants, deformations, occlusion completions, yaw rotations, and stretches. Every variant is re-optimized and re-evaluated under physics.

Success criterion. Success uses a position-error threshold of 150 mm (Table 15), rotation error below 40°, mean capsule penetration at most 5 mm, median post-weld squeeze force at least 1 N, and per-finger 95th-percentile force at most 100 N.

E.3 GNR TRAINING AND TEST SET

GNR is trained on 28,480 DIAL-MPC successes from pose, timing, mass, and friction augmentations. The training set contains 1,838,731 frames (17.03 hours; 64.56 frames, or 2.15 s at 30 Hz, per episode), 23 objects and 353 source motions.

The randomly sampled test set contains 1,000 episodes. Relative to this training set, it comprises 203 OOD-object, 338 OOD-grasp, and 459 OOD-geometry episodes. Their DIAL-MPC outcome composition and GNR results are reported in App. **??**.

E.4 Dataset Expansion

We retarget demonstrations from the 266k MPC-failure dataset with up to $S = 64$ samples per demonstration. A demonstration succeeds if any of its first S candidates succeeds; sampling stops upon success. Successes increase from 19,384 (7.28%) at S=1 to 99,710 (37.47%) at S=64 (Figure 11). Combined with the 95,670 MPC successes, this yields 195,380 trajectories (195k) and adds 428 geometries, increasing the total from 2,756 to 3,184 (3.2k).

At S=64, SR on the three OOD subsets reaches 29.88%, 25.83%, and 45.06%, respectively.

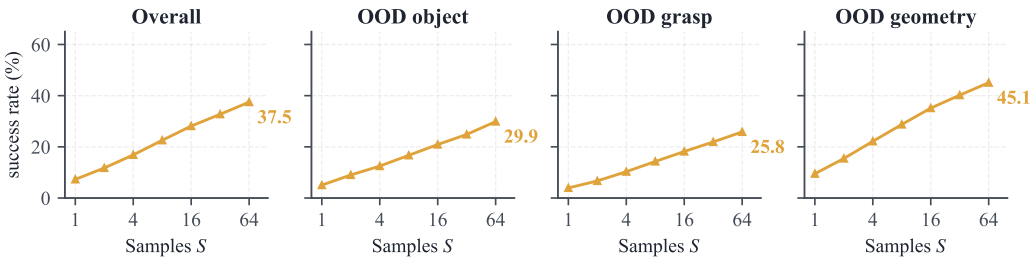


Figure 11: Test-time scaling on HOT3D MPC-failure demonstrations, overall and across three OOD subsets of the 266,120 demonstrations.

E.5 Evaluation

E.5.1 Objects, Geometry, and Grasps

The dataset contains 33 scanned objects in total; GNR used in Section 5 is trained on 23 objects and 353 source motions. Each test episode belongs to exactly one OOD subset, as summarized in Table 12.

Table 12: HOT3D OOD subsets.

OOD axis	Test episodes
Object	203
Grasp	338
Geometry	459

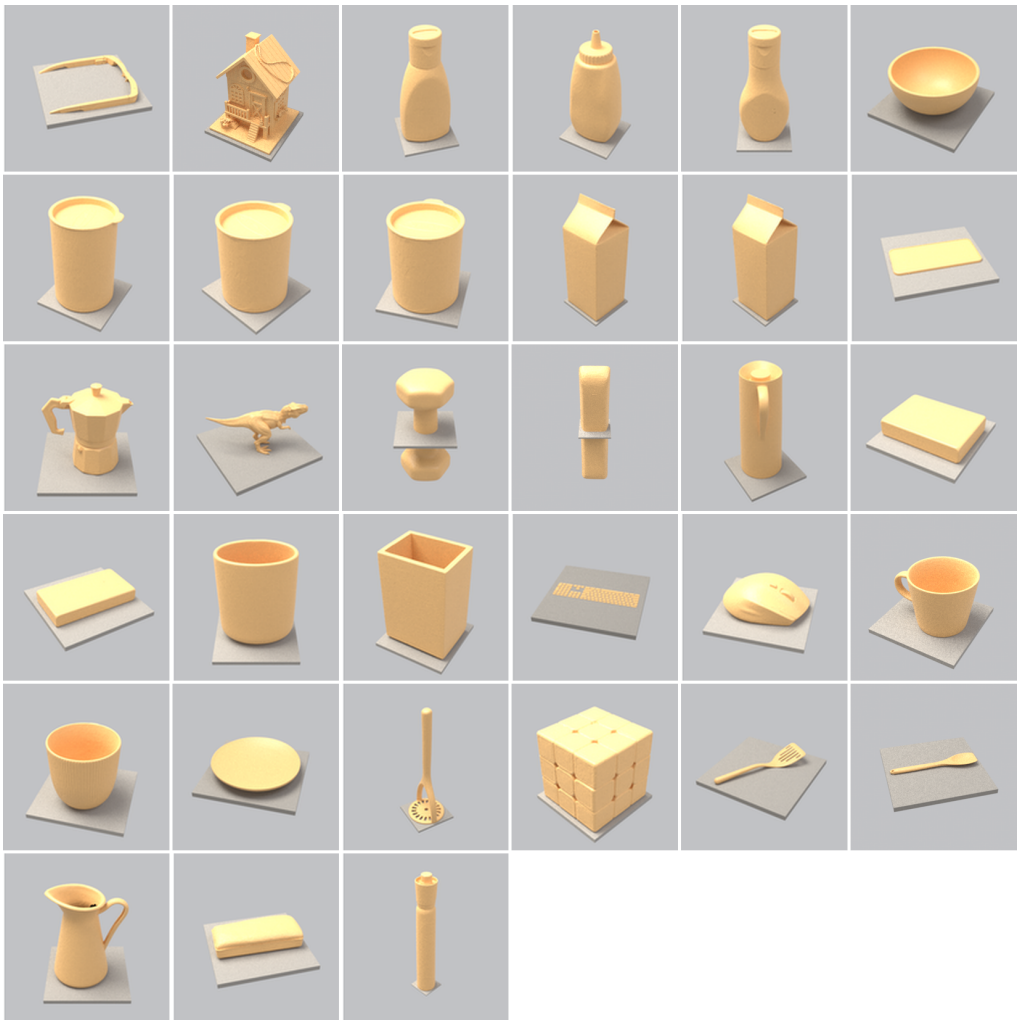


Figure 12: All 33 HOT3D object instances.



Figure 13: Geometry variants across six objects. Each row uses a shared camera scale.

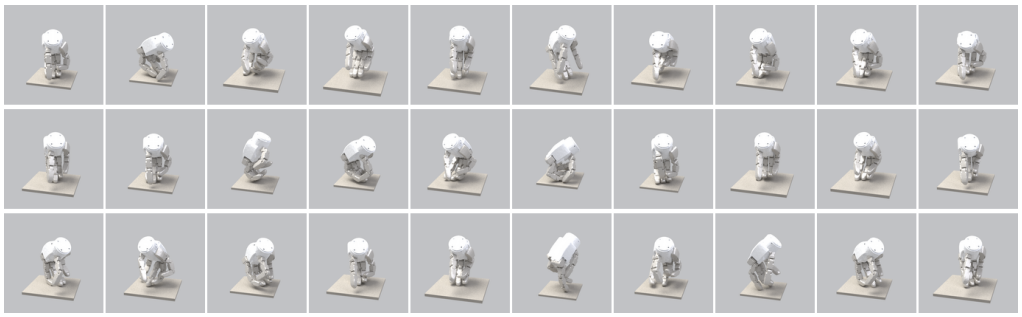


Figure 14: DG-5F grasp configurations at a shared camera view and scale.

On test set, GNR success rises from 12.81% to 44.33% on OOD objects, from 10.06% to 42.90% on OOD grasps, and from 26.36% to 71.24% on OOD geometry as S increases from 1 to 256.

E.6 Trajectory Analysis

Figure 15 compares object and finger-joint tracking errors and finger-control jerk.

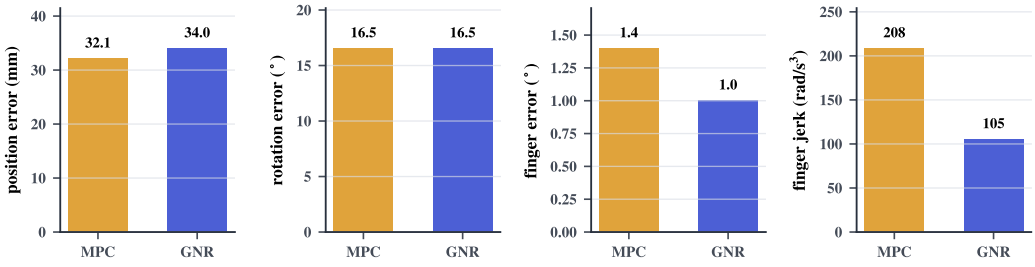


Figure 15: HOT3D trajectory quality. From left to right: median object-position, object-orientation, and finger-joint angular errors on the randomly sampled 1,000-episode test set, followed by median finger-control jerk on a separate matched 10-episode, 256-sample study. GNR approximately halves jerk while preserving comparable object tracking.

E.7 Efficiency

Per-episode and full-dataset costs are reported in Table 18 and Table 19. Real-to-sim, physically optimized simulation, and robot-rendered comparisons are shown in App. K.3.

F Multi-Embodiment Evaluation

F.1 Setup and Results

One GNR model is trained on 147 trajectories spanning five hands, and tested with 100 trajectories, with 20 per hand.

Each hand is represented by a five-dimensional one-hot identity and a binary actuator mask over a padded 52-coordinate action tensor. The mask applies to normalization, flow noise, intermediate states, outputs, and training loss.

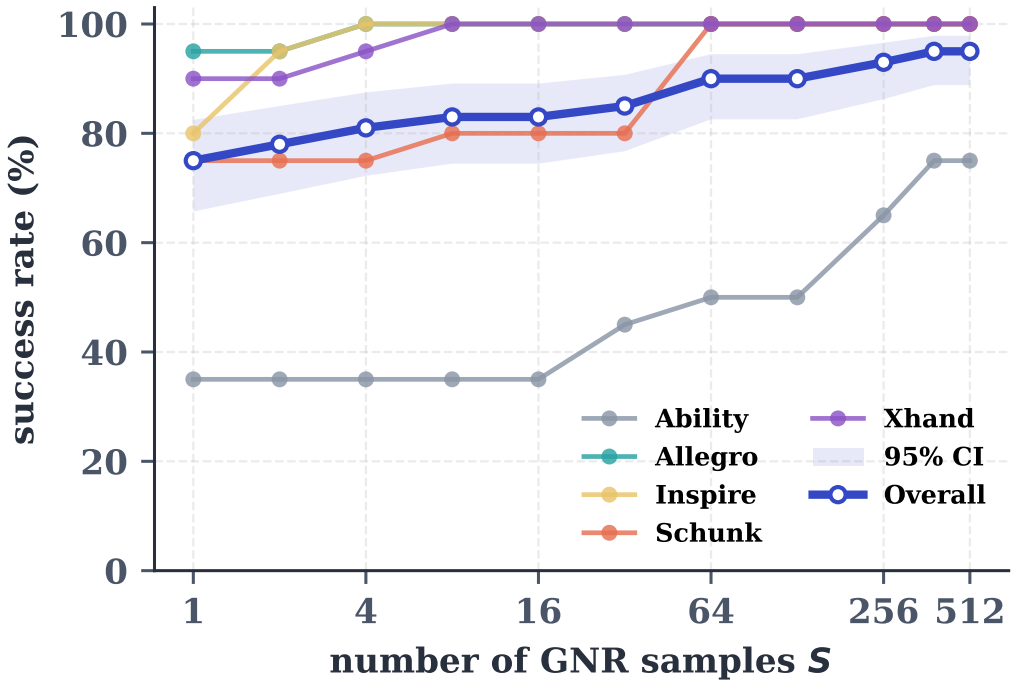


Figure 16: GNR test-time scaling on the 100-trajectory test set. Thin curves report the five embodiments.

Table 13: Table 14.

Method	Samples S	Successes	Success (%, 95% CI)
GNR	128	90/100	90.0 [82.6, 94.5]
GNR	256	93/100	93.0 [86.3, 96.6]
GNR	384	95/100	95.0 [88.8, 97.8]
SPIDER	–	96/100	96.0 [90.2, 98.4]

F.2 Efficiency Comparison

Table 14: Sampling budgets of `GNR` and SPIDER. GNR draws S complete trajectories once per test episode. SPIDER draws 1,024 candidates per optimizer iteration inside a receding-horizon MPC loop.

Method	Test subset	Episodes	Samples / iter.	Iters.	Cost
GNR, S=128	all	100	128	1	12.8
GNR, S=256	all	100	256	1	25.6
GNR, S=384	all	100	384	1	38.4
SPIDER	`OakInk`	50	1,024	16	819.2
SPIDER	`GigaHands`	50	1,024	32	1638.4
SPIDER	combined	100	–	–	2457.6

SPIDER's per-episode sampling depends on early stopping within each optimizer call and the number of receding-horizon replanning calls. Table 14 reports its configured per-call caps.

G MODELING AND TRAINING

G.1 EVALUATION AND OPTIMIZATION

We use S for the number of complete GNR trajectories, n_s for samples per optimizer iteration, and M for the maximum number of MPC iterations per call, giving a sample budget of $n_s M$. Within each comparison, methods share the simulator, task reference, and success criterion (Table 15). `HOT3D` uses the trajectory-level criterion Γ in Section 4.2.

Table 15: Success criteria. Each benchmark requires all listed criteria.

Benchmark	Success criterion
`ShapeFilter`	Object centroid lies in the box-local insertion region $[-80, 80] \times [0, 160] \times [-80, 80]$ mm.
`NIST`	Horizontal knob-stud error < 1 mm, seating displacement ≤ -8 mm, and axis error < 0.003 rad.
`HOT3D`	Object position error < 150 mm.
`HOT3D`	Object orientation error $< 40^\circ$.
`HOT3D`	Mean capsule penetration ≤ 5 mm.
`HOT3D`	Median post-weld squeeze force ≥ 1 N and per-finger force 95th percentile ≤ 100 N.
SPIDER benchmark	Mean object-position error ≤ 0.10 m and mean object-orientation error ≤ 0.50 rad.

DIAL-MPC. DIAL-MPC perturbs the current control sequence, scores the resulting simulator rollouts, and updates the sequence with an annealed sampling kernel. Table 16 lists the settings for training-data generation and matched MPC evaluation. Each iteration uses the task's per-degree-of-freedom noise map; "full episode" denotes optimization over the complete reference. On `ShapeFilter`, the optimizer returns the best sampled trajectory and uses exploitation fraction 0.15, exploitation noise 0.5, temperature 0.08, and trajectory-noise decay 0.7. On `HOT3D`, it uses temperature 0.03, trajectory-noise decay 0.99, and knot noise that decreases log-linearly from 0.10 to 0.04.

Table 16: DIAL-MPC settings. Maximum budget per call: n_s samples per iteration $\times$ MPC iterations M.

Benchmark	Phase	n_s	MPC iterations M	Samples per call	Horizon	Control stride	Active coordinates
`ShapeFilter`	insertion	32	6	192	20	5	arm and hand
`ShapeFilter`	insertion	500	6	3,000	20	5	arm and hand
`NIST`	wrist alignment	86	10	860	7	2	end effector
`NIST`	finger closure	86	10	860	6	1	thumb and index
`NIST`	final refinement	172	10	1,720	13	4	arm and hand
`HOT3D`	tracking	300	10	3,000	full episode	full episode	hand

MPPI and SPIDER. The 100-trajectory multi-embodiment comparison uses the authors' released annealed SPIDER configuration, with MPPI as a secondary reference. SPIDER combines iterative physics optimization with an annealed search distribution and optional contact guidance.

MPPI and SPIDER draw n_s=1,024 candidates per optimizer iteration, with at most 16 MPC iterations (M=16) on `OakInk` and 32 (M=32) on `GigaHands`. A two-step improvement test may stop an optimizer call early. MPPI uses a fixed final noise scale of 1.0, whereas SPIDER anneals it to 0.01. The controller invokes the optimizer repeatedly per trajectory; we report configured per-call budgets.

On the 100 test episodes, SPIDER takes 54.7 s per episode on average (median 52.3 s) on a GPU worker.

CEM, iCEM, and CMA-ES. We use published implementations of CEM, iCEM, and CMA-ES. They optimize the full control perturbation within twice DIAL-MPC's noise scale, and their rollout counts include one replay of the selected control. Classical CEM removes iCEM's colored noise, sample-count decay, and elite reuse. On `ShapeFilter`, CEM is evaluated at both 32 and 500 samples for six MPC iterations; iCEM and CMA-ES use 32 samples for six iterations. All use horizon 20. The larger CEM run therefore totals 3,000 optimizer samples per optimization call.

Table 17: Sampling-baseline settings. HOT3D uses full-episode optimization; CEM is the matched ShapeFilter ablation.

Method	Benchmark	Samples per iteration	MPC iterations M	Samples per call	Sampling rule and schedule
MPPI	HOT3D	300	10	3,000	Softmax-weighted update under the task's DIAL-MPC noise map; no elite truncation.
CEM	ShapeFilter	32	6	192	Elite ratio 0.10, mean/variance smoothing 0.10, white Gaussian noise, fixed sample count, fresh optimizer state at each optimization call.
CEM	ShapeFilter	500	6	3,000	Elite ratio 0.10, mean/variance smoothing 0.10, white Gaussian noise, fixed sample count, fresh optimizer state at each optimization call.
iCEM	ShapeFilter	32	6	192	Elite ratio 0.10; smoothing 0.10; colored-noise exponent 2.0; fresh sample set at each optimization call.
CMA-ES	ShapeFilter	32	6	192	Full covariance over the active 29-dimensional control, initialized from the DIAL-MPC noise scale.
iCEM	HOT3D	300	10	3,000	Elite ratio 0.10; smoothing 0.10; fixed matched sample count; colored-noise exponent 2.0; retain 0.30 of elites; fresh optimizer state; return the best trajectory rather than the elite mean.
CMA-ES	HOT3D	300	10	3,000	Full covariance over active hand coordinates; initial global scale 0.5; per-coordinate scale from the DIAL-MPC noise map; arm coordinates frozen; one final replay (3,001 rollouts total).

These baselines share scene construction, physics, rewards, and success criteria. They run on CPU workers; core-s sums process-tree user and system time.

G.2 GNR Architecture and Training

The HOT3D model has 19.57M trainable parameters: 560k in its Point Transformer V3 geometry encoder without sparse convolutions over 1,024 scene points and 19.01M in the temporal U-Net. It uses widths $(128, 256, 512)$, geometry width 96, conditioning width 256, flow-step width 128, and ten Euler steps. We train it with bfloat16 mixed precision and batch size 128 on 28,480 episodes on one NVIDIA A100.

The ShapeFilter and NIST models use the same temporal U-Net widths $(128, 256, 512)$, kernel size 5, eight normalization groups, and a 128-dimensional flow-step embedding. ShapeFilter conditions on the IK reference, object pose, and arm state, using 64-frame sequences with a 10-frame stride. It is trained for 100 epochs. NIST conditions on the reference and knob pose and is trained for 200 epochs on complete episodes with a constant learning rate of 4×10^{-4}. Both use AdamW, weight decay 10^{-6}, gradient clipping at 1.0, EMA decay 0.999, batch size 128, and mixed precision on one GPU.

H Efficiency

H.1 Compute Cost

We report both wall time and CPU compute (core-h).

Table 18: Measured per-episode generation cost.

Task	Method	Sample budget	Core-h / episode	Wall h / episode
HOT3D	DIAL-MPC	3,000	1.0777	0.2427
HOT3D	iCEM	3,000	1.4213	0.3282
HOT3D	CMA-ES	3,000	1.2272	0.2877
HOT3D	GNR, S=256	256	0.1094	0.0383
ShapeFilter	MPC	3,000	3.2001	0.0472

GPU training time. On one H100-class GPU, ShapeFilter training takes 0.0108 to 0.5022 GPU-h over 100 epochs as the seed set grows from 50 demonstrations to the full training set. NIST training takes 0.0202 to 0.0406 GPU-h over 200 epochs with 100 to 294 demonstrations.

H.2 Cost versus Dataset Coverage

Table 19 sums CPU generation costs for retained MPC data and GNR generation; GPU training is separate. `HOT3D` reports the collected dataset expanded at S=64; `ShapeFilter` and `NIST` compare generation cost and success. `NIST` costs follow App. D.3.

Table 19: Dataset-generation cost in thousands of core-h at similar SRs on the same task.

Task	Construction strategy	Generation compute (10^3 core-h)			Coverage
		MPC data	GNR	Total	
`HOT3D`	all DIAL-MPC	390.043	–	390.043	26.4%
`HOT3D`	retained MPC data + `GNR` S=64	390.043	6.720	396.763	54.0%
`ShapeFilter`	all MPC	8.541	–	8.541	93.70%
`ShapeFilter`	50 MPC seeds + `GNR`	0.203	0.562	0.766	94.82%
`NIST`	all MPC	79.136	–	79.136	92.19%
`NIST`	100 MPC seeds + `GNR`+MPC	25.439	19.078	44.517	92.12%

For `HOT3D`, the 95k MPC-success trajectories cost 390k core-h to generate, including unsuccessful attempts. Applying `GNR` adds 99k MPC-failure successes (App. E.4), giving 195k trajectories and 53.98% coverage of the original dataset. Using 90.9 core-s per demonstration at S=64 (Table 1), the additional cost is estimated as $266{,}120 \times 90.9/3600 = 6{,}720$ core-h, for a total of 396h core-h. This estimate uses the full sampling budget for every demonstration. The dataset grows by $2.04\times$ for 1.72% additional compute, reducing CPU cost per successful trajectory from 4.08 to 2.03 core-h ($2.01\times$).

For `ShapeFilter`, using the 50 seed dataset reduces generation from 8.54k to 766 core-h while increasing dataset success from 93.7% to 94.8%. For `NIST`, using 100 seed dataset reduces generation from 79.14k to 44.52k core-h at about 92.2% success. The configurations and estimated costs are listed in Table 9 and Table 10.

I Behavior Cloning

We evaluate policy learning from retargeted trajectories (Pan et al., 2025; Chi et al., 2024; Zheng et al., 2026) using equally sized MPC, `GNR`+MPC, and `GNR` datasets with the same policy architecture and training hyperparameters.

The state-based policy is a deterministic residual multilayer perceptron with width 4096, six residual blocks, about 203 million parameters, and a masked regression loss. Each dataset contains 500 episodes with $108{,}825$ frames and a mean episode length of 217.7 frames (10.9 s) at 20 Hz. We evaluate ten training checkpoints using 32 test episodes with a 400-step time limit.

The best SR achieved by the policy trained using MPC data is 37.5%, 43.75% using `GNR`+MPC, and 43.75% using `GNR` (Figure 17), demonstrating that `GNR`-generated data is applicable to policy learning.

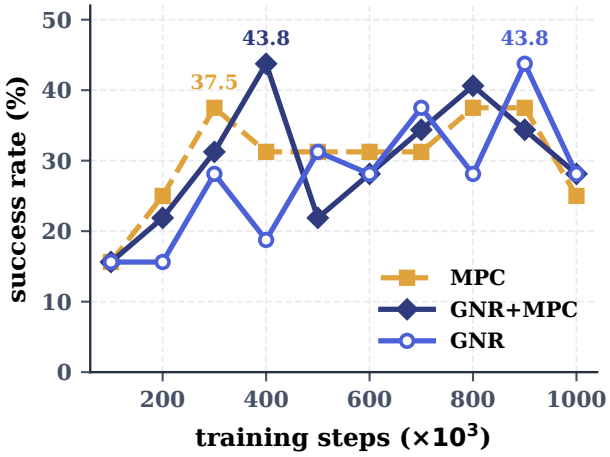


Figure 17: SR curves of behavior cloning policy training using data generated by different retargeters.

J Data Flywheels

Newly retargeted demonstrations can support subsequent rounds of `GNR` training, forming a data flywheel (Liu et al., 2025b; Zhu et al., 2025). We test this on `ShapeFilter`, starting with 200 MPC-generated seed trajectories. Each round uses MPC to solve demonstrations on which `GNR` fails and adds the successful trajectories to the training set. Four rounds add 220, 104, 120, and 105 trajectories, respectively, yielding 749 training trajectories. On a fixed 256-episode test set, best-of-8 success increases from 55.2% to 74.4%, while best-of-16 success increases from 68.4% to 84.8%.

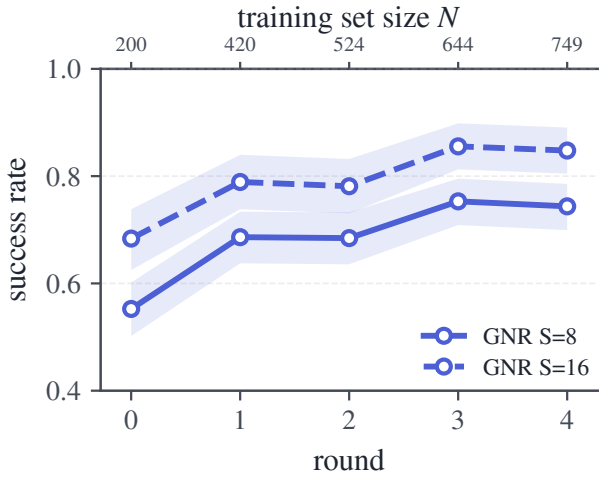


Figure 18: `ShapeFilter` success across four data-flywheel rounds.

K RETARGETING DEMOS

This section presents trajectories retargeted by different methods on `ShapeFilter`, `NIST`, and `HOT3D`.

K.1 SHAPEFILTER

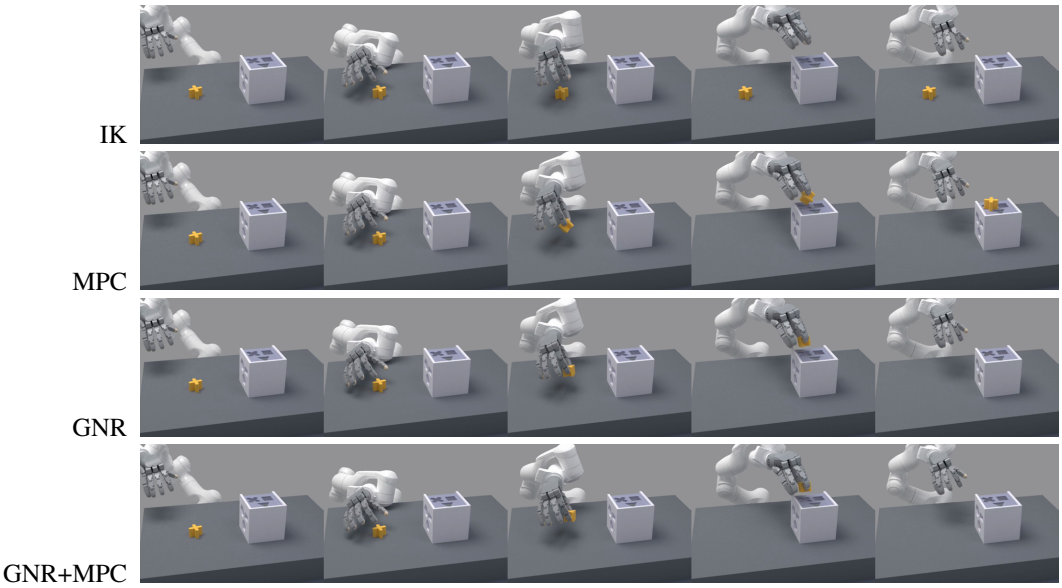


Figure 19: `ShapeFilter` example in which MPC fails while GNR and GNR+MPC succeed.

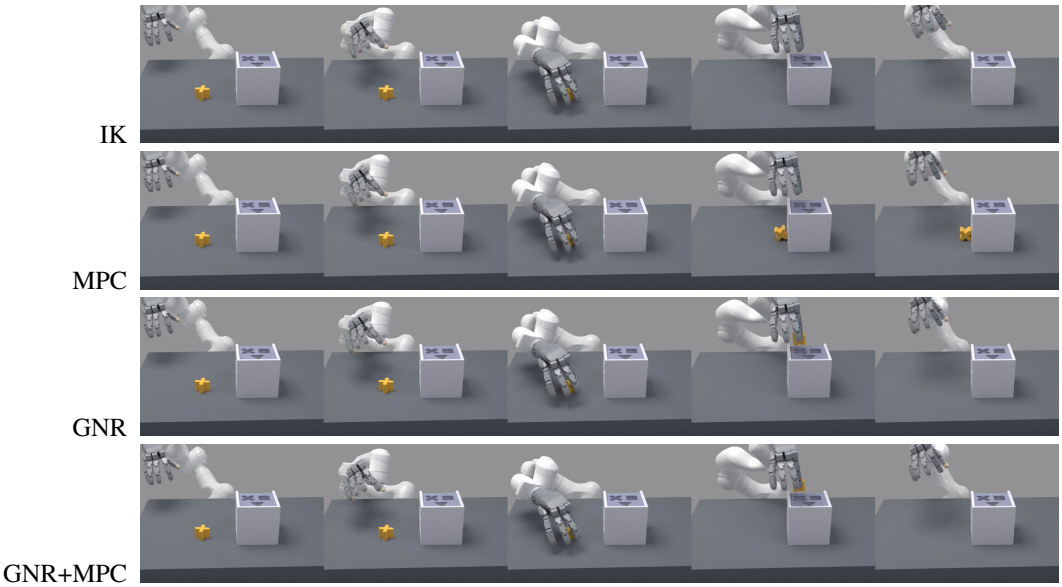


Figure 20: `ShapeFilter` example in which MPC fails while GNR and GNR+MPC succeed.

K.2 NIST

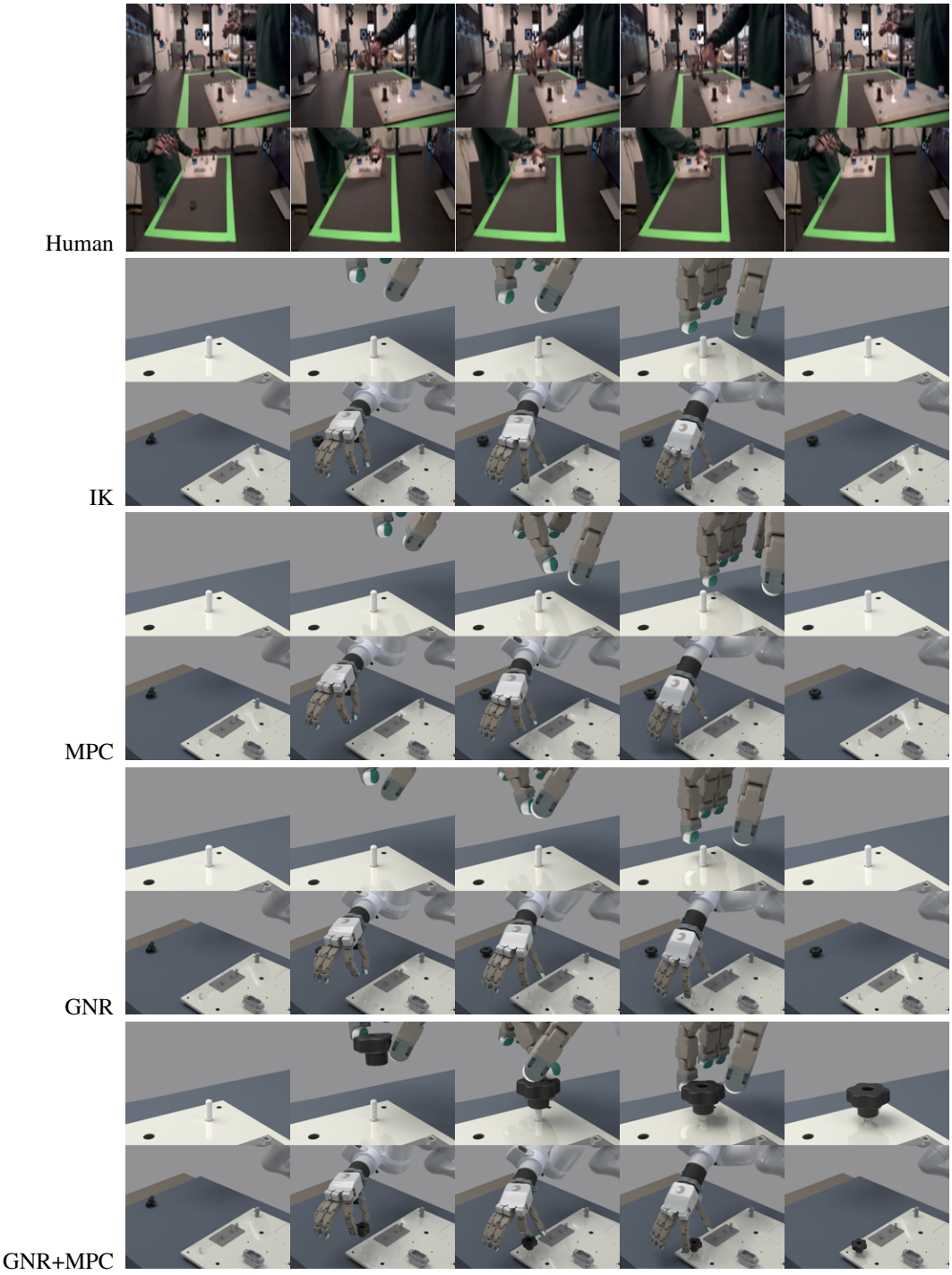


Figure 21: A NIST example. GNR+MPC succeeds where IK, MPC, and open-loop GNR fail.

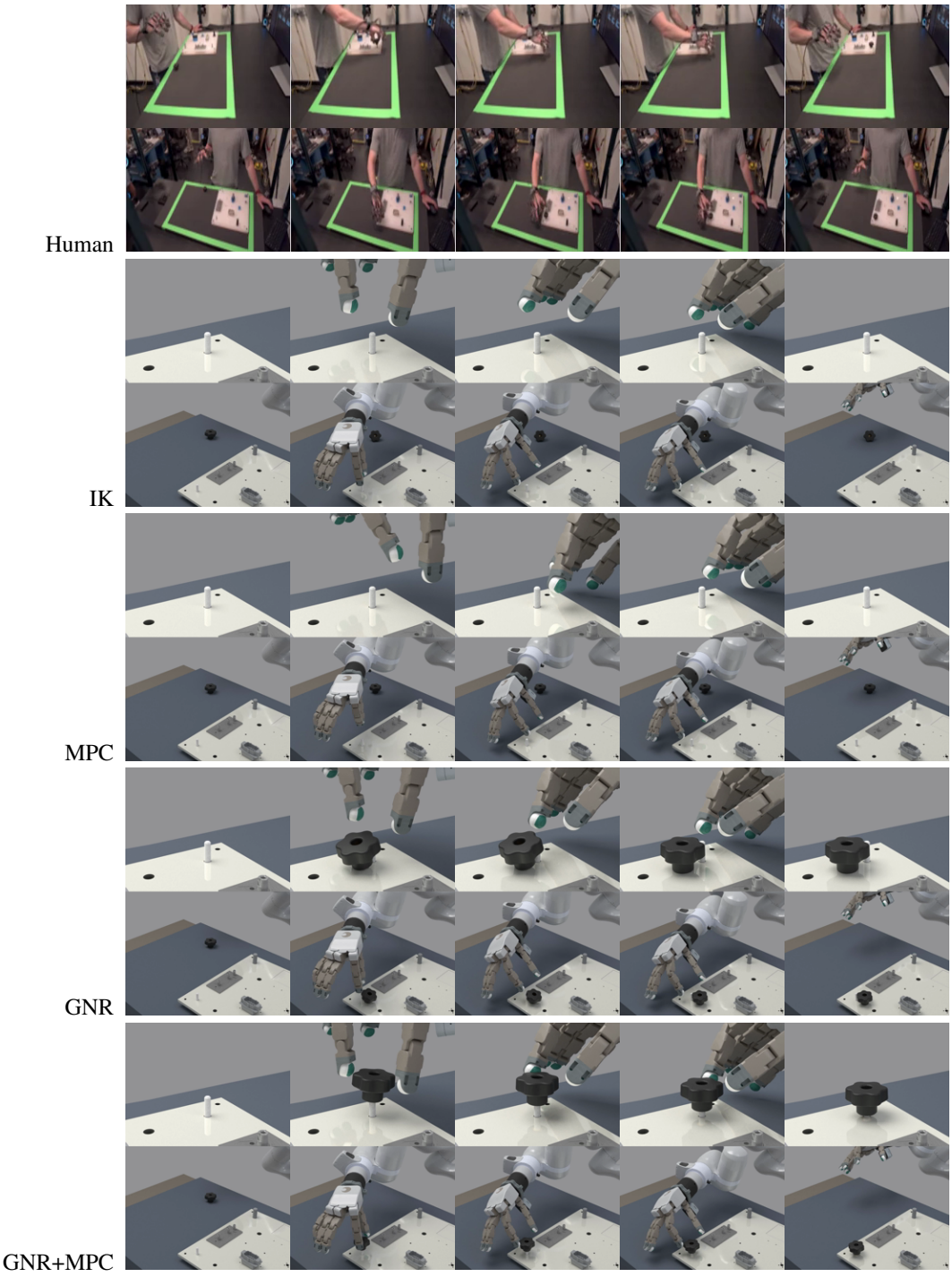


Figure 22: A `NIST` example. GNR+MPC succeeds where IK, MPC, and open-loop GNR fail.

K.3 HOT3D

We present egocentric recordings and trajectories generated by MPC and GNR across different objects.

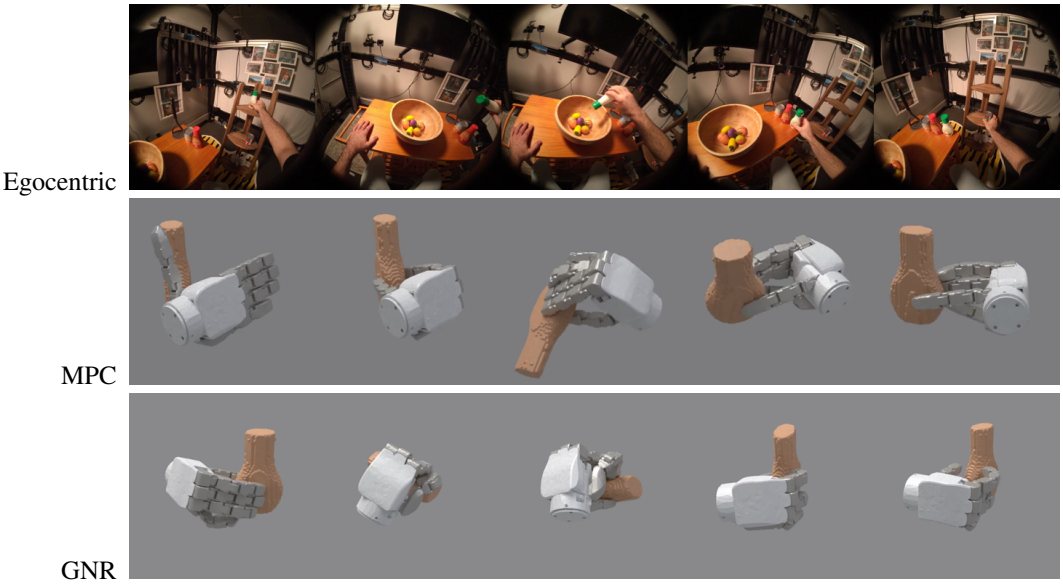


Figure 23: HOT3D ranch bottle with a stretched OOD geometry.

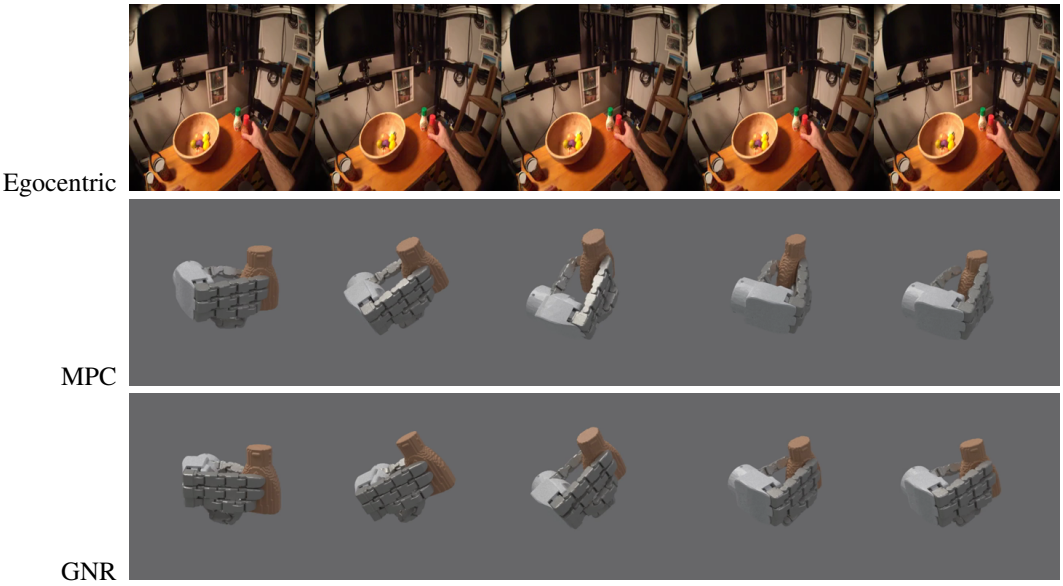


Figure 24: HOT3D barbecue sauce bottle with a stretched OOD geometry.

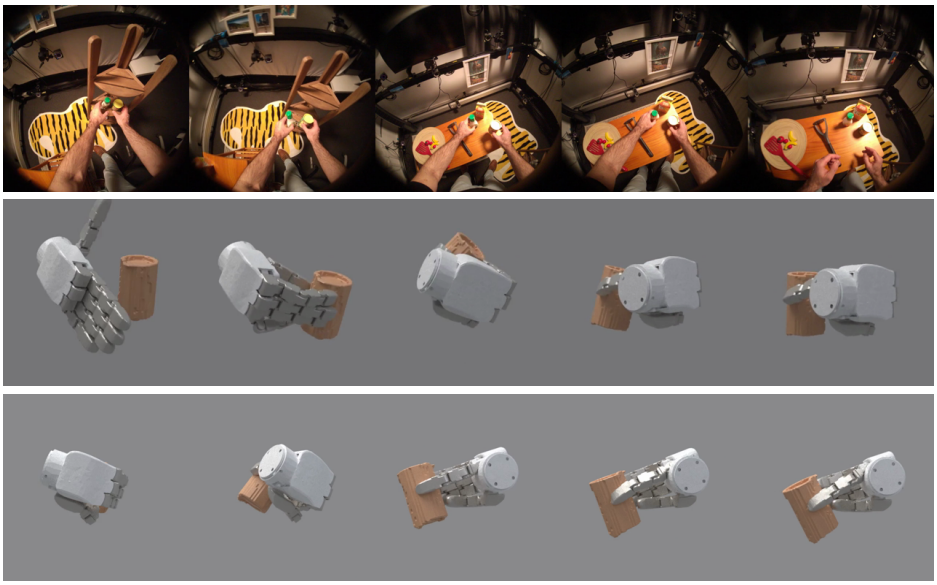


Figure 25: HOT3D parmesan can with an OOD deformation.

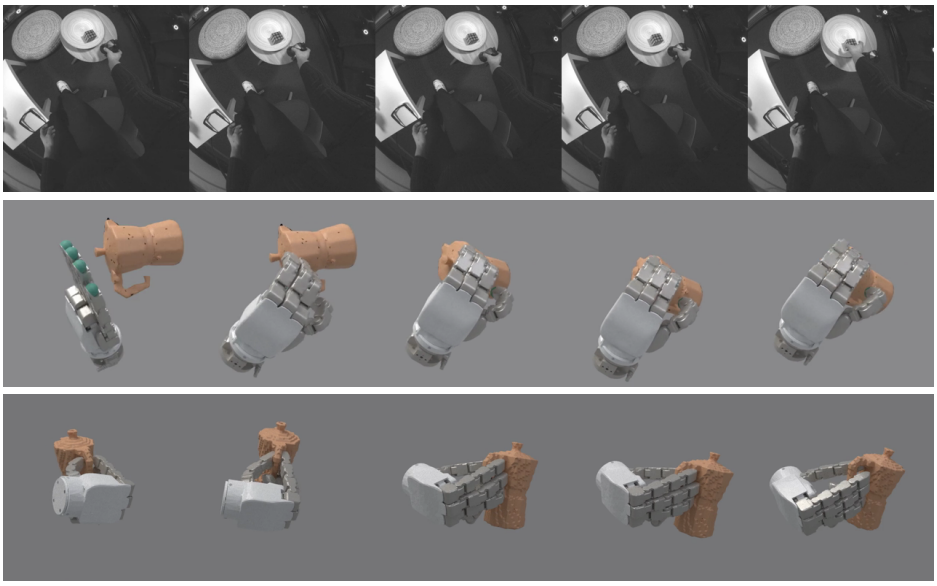


Figure 26: HOT3D coffee pot. GNR succeeds where MPC fails.

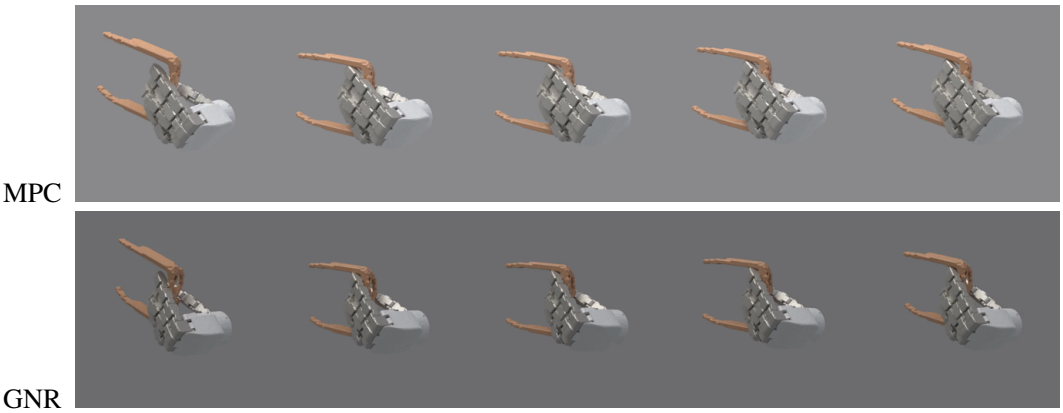


Figure 27: HOT3D Aria glasses with an OOD deformation.

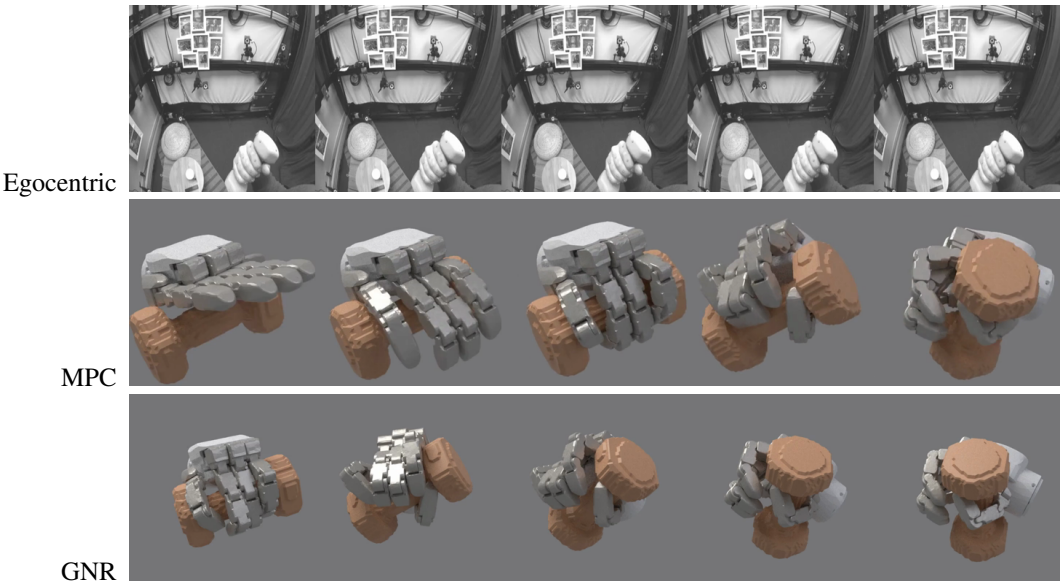


Figure 28: HOT3D dumbbell manipulation.

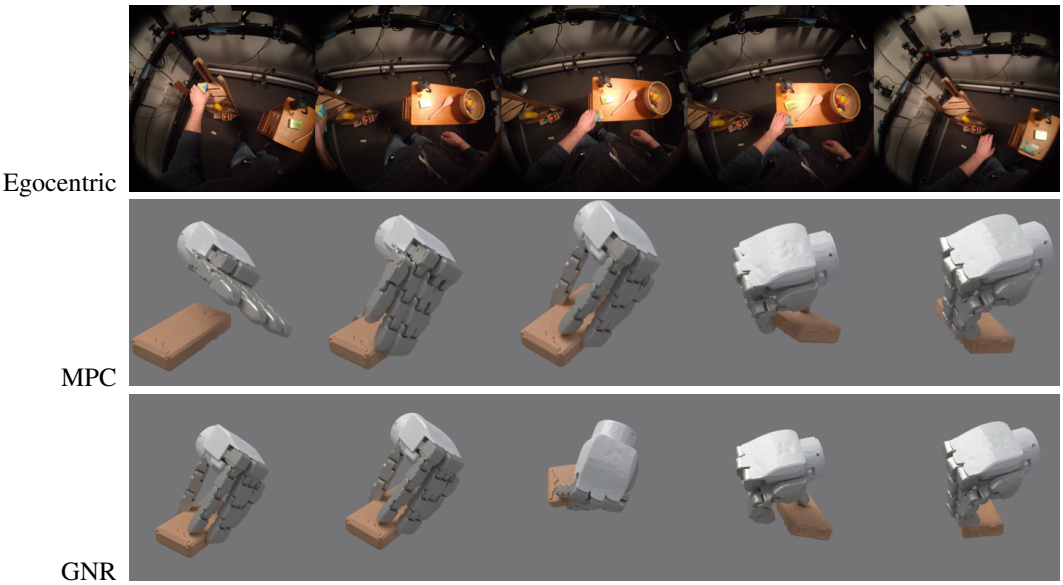


Figure 29: HOT3D waffle pick-and-place with an OOD rotation.

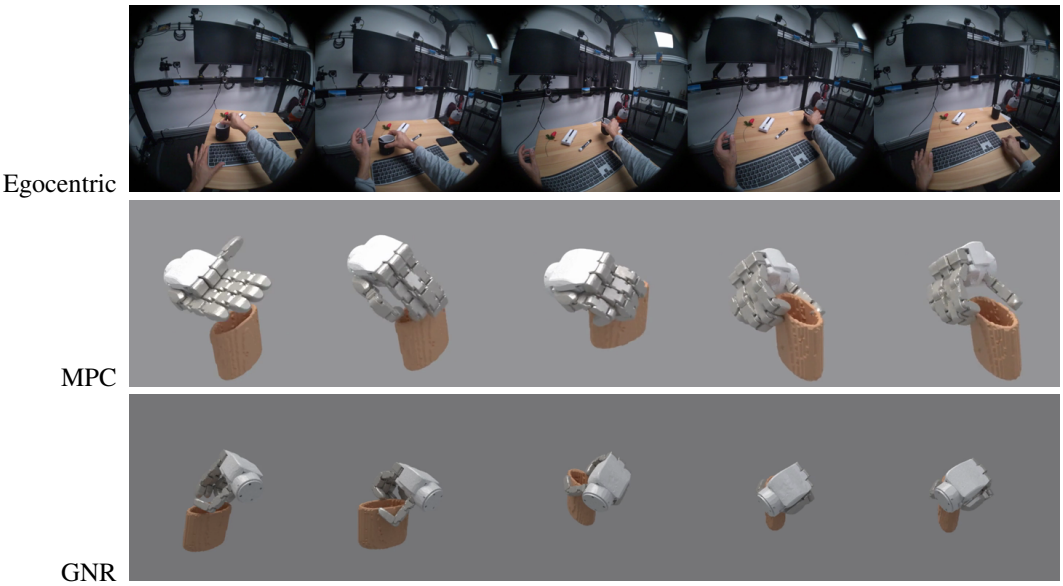


Figure 30: HOT3D holder with a stretched OOD geometry.

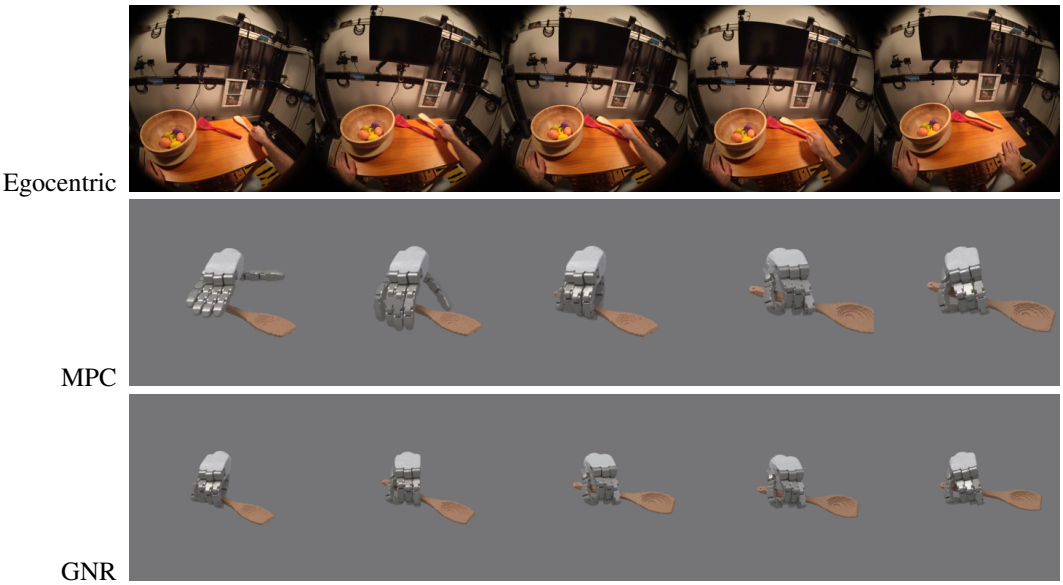


Figure 31: HOT3D wooden spoon with an OOD deformation.

K.3.1 Human Motion Optimization

Aside from optimizing robot trajectories, we also generated 37,468 successful MPC-optimized human-motion rollouts. The examples below show MPC making reconstructed human motion physically feasible. Our robot-retargeting comparisons use the original real-to-sim reference.

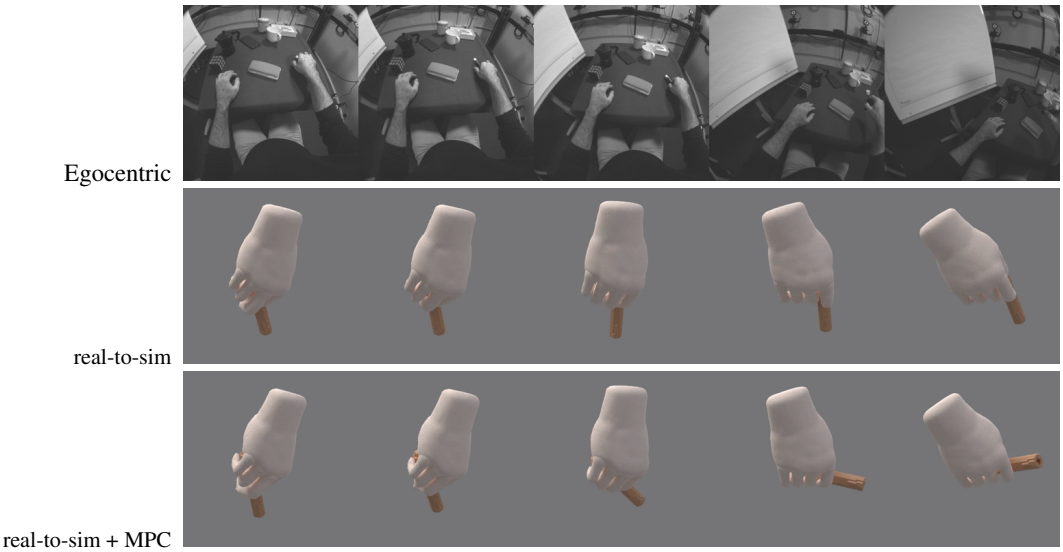


Figure 32: HOT3D whiteboard marker. MPC makes the reconstructed human motion physically feasible.

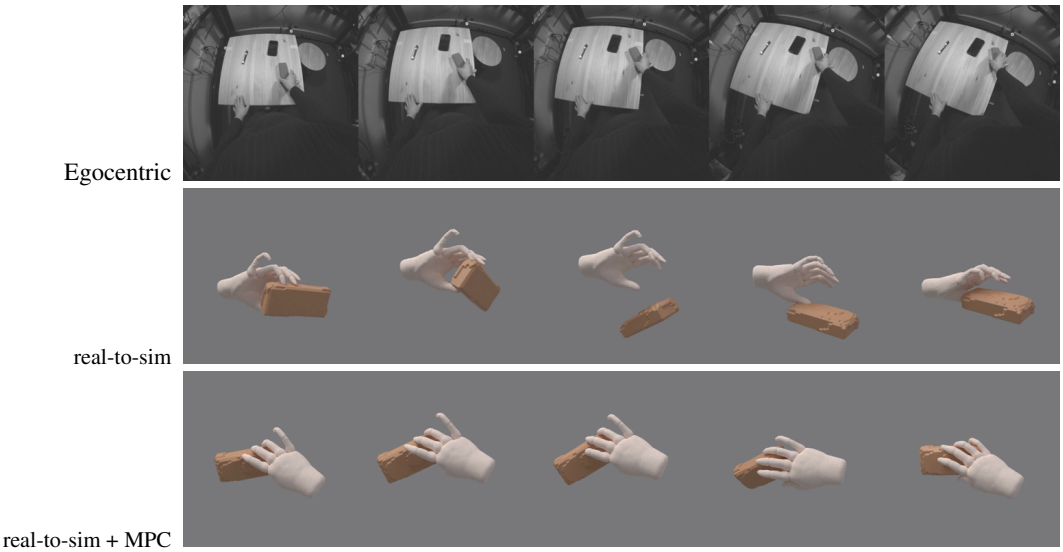


Figure 33: HOT3D whiteboard eraser. MPC makes the reconstructed human motion physically feasible.



Figure 34: HOT3D dinosaur toy. MPC makes the reconstructed human motion physically feasible.