DEXSEED: SCALABLE PHYSICS-GROUNDED DEMONSTRATION GENERATION FOR SIM-TO-REAL DEXTEROUS MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning visuomotor policies for contact-rich dexterous manipulation requires demonstrations that are diverse and physically valid, yet teleoperation is costly and embodiment-specific, while human videos are not directly executable by robot hands. Existing data-generation methods either replay kinematically transformed demonstrations, which break under physics variation, or track reference trajectories in simulation, which inherit the source timing and contact schedule and limit diversity. We introduce DEXSEED, a physics-grounded pipeline that converts a handful of human RGB-D videos into large-scale dexterous manipulation datasets under joint randomization of layout, geometry, visual, and physics. Rather than treating a human demonstration as a trajectory to copy, DEXSEED uses it as an object-centric, phase-aligned prior over task-relevant contact structure, combining temporal style augmentation, task-conditioned objective, and sampling-based optimization through full physical rollouts. Across four contact-rich tasks, DEXSEED achieves the highest data-generation success from both teleoperated demonstrations and human videos, improves simulated downstream policy success by 33.2% on average, and achieves a **45% zero-shot real-world success** rate using only **3 human videos** per task. Videos are available at dexseed.github.io.

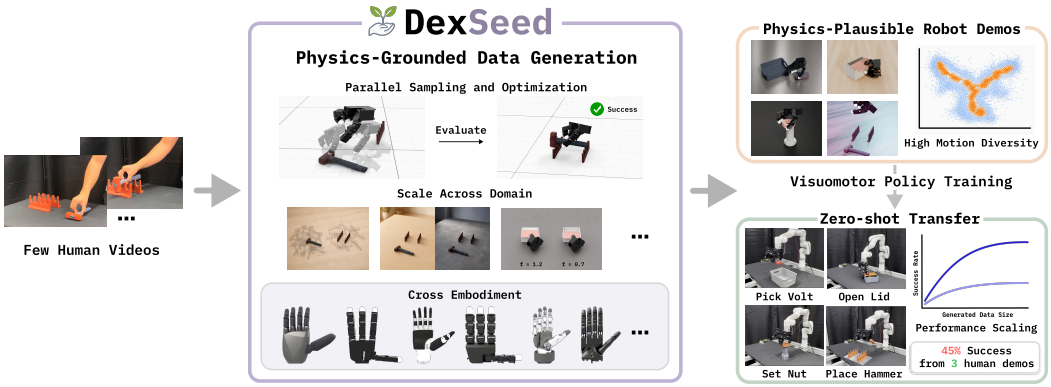

Figure 1: DEXSEED turns a few human RGB-D videos into large-scale, physics-plausible robot demos across randomized domains and embodiments, enabling zero-shot sim-to-real transfer.

1 INTRODUCTION

Contact-rich dexterous manipulation remains difficult to scale because learning robust visuomotor policies requires large collections of physically valid demonstrations covering variations in layout, geometry, and contact dynamics. Teleoperation provides executable robot data, but is costly, embodiment-specific, and hard to repeat for every new task or hand. Human RGB-D videos offer a cheaper, embodiment-agnostic source of supervision, yet they are not directly executable by robots: recovered human motion is noisy, violates robot kinematics, and ignores the contact and actuation constraints of the target hand.

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.

A natural solution is to use a few human videos as seeds and generate the remaining data in simulation, yet useful demonstrations must be simultaneously *diverse*, *physically valid*, and *task-successful*. Kinematic transform-and-replay methods (Mandlekar et al., 2023; Garrett et al., 2024; Jiang et al., 2025) preserve object-relative motion under new layouts but do not optimize through contact dynamics, breaking when mass, friction, compliance, or contact timing changes. Physics-based retargeting methods (Pan et al., 2025; Li et al., 2025b) improve feasibility through control search, but still track a source trajectory frame by frame, tying the generated behavior to the source horizon, contact schedule, and layout.

We propose DEXSEED, a physics-grounded data-generation pipeline that converts a handful of human RGB-D videos into large, diverse, and physically valid datasets for dexterous manipulation under joint randomization of layout, geometry, and physics (Fig. 1). The key idea is to treat a human demonstration not as a trajectory to copy, but as an object-centric, phase-aligned *prior over task-relevant contact structure*: it specifies *where* and *when* contact is useful, while the simulator determines *how* the target robot realizes that contact in each randomized instance. Concretely, DEXSEED builds a layout-conditioned phase reference and contact mask from each video, then optimizes robot controls through full physical rollouts. Temporal style augmentation varies rollout horizon and phase timing; a task-conditioned objective prioritizes terminal task completion while anchoring hand-object relations only during relevant contact phases; and a sampling-based optimizer accepts only executable, task-successful rollouts, which are then rendered with independent visual domain randomization for policy training.

We evaluate DEXSEED on four contact-rich dexterous tasks (Pick Volt, Open Lid, Set Nut, Place Hammer) in simulation and the real world. From three source demonstrations per task, DEXSEED attains the highest data-generation success on teleoperated and human video sources, reaching 65.5% and 50.0% under joint randomization (+11.5 and +9.0 percentage points over the strongest baseline), with wider wrist and finger-joint coverage. The same human videos retarget to four dexterous hands (XHand, LEAP, Allegro, Inspire), and policies trained on DEXSEED data achieve up to $2\times$ the success rate of the strongest baseline in simulation, with a 33.2% average improvement across the four tasks, and transfer zero-shot to a LEAP Hand at 45% average real-world success. Policy performance further improves with more synthesized demonstrations, showing that DEXSEED provides scalable supervision beyond the small-data regime.

In summary, our key contributions are: (i) a physics-grounded pipeline for generating large dexterous-manipulation datasets from a few human RGB-D videos under layout, geometry, visual, and physics randomization; (ii) a formulation that uses human demos as object-centric, phase-aligned priors rather than frame-aligned tracking targets, enabling adaptation of timing, intermediate motion, and contact forces; and (iii) empirical gains in data-generation success, trajectory diversity, downstream policy learning, cross-embodiment retargeting, and zero-shot sim-to-real transfer.

2 RELATED WORK

2.1 DATA GENERATION FOR ROBOT LEARNING

Data is the central bottleneck for robot policy learning: human teleoperation yields only a few hundred demonstrations per task after hours of effort (Khazatsky et al., 2024; O'Neill et al., 2024). Offline augmentation broadens visual or semantic diversity through generative edits (Chen et al., 2024; Tian et al., 2024; Mandi et al., 2022; Wang et al., 2024b; Yu et al., 2023; Chen et al., 2023b), but leaves the underlying layout, geometry, and physics unchanged. A more ambitious line synthesizes new state-action demonstrations from a small seed set by kinematic transformation. MimicGen (Mandlekar et al., 2023) and its extensions (Garrett et al., 2024; Jiang et al., 2025; Xue et al., 2025; Lin et al., 2025) replay object-centric subtasks under new layouts via $\mathrm{SE}(3)$-equivariant relative-pose transforms, while MoMaGen (Li et al., 2025a) treats task success as a hard constraint and the demonstration as a soft prior. However, these methods remain largely kinematic and are brittle under physics variation (Yang et al., 2025), especially for contact-rich dexterous manipulation. Closer to our setting, trajectory-optimization (Yang et al., 2025; Reda et al., 2023; Lakshmipathy et al., 2025; Pan et al., 2025) and RL-based retargeting (Singh et al., 2025; Lum et al., 2025; Li et al., 2025b; Mandi et al., 2025) search for robot controls whose simulated rollouts track a reference under joint, contact, and actuation constraints. Yet their frame-aligned tracking objectives inherit the source horizon and contact schedule, so rollouts cluster near the source and fail when new layouts or physical parameters require different timing, intermediate motion, or contact forces. DEXSEED

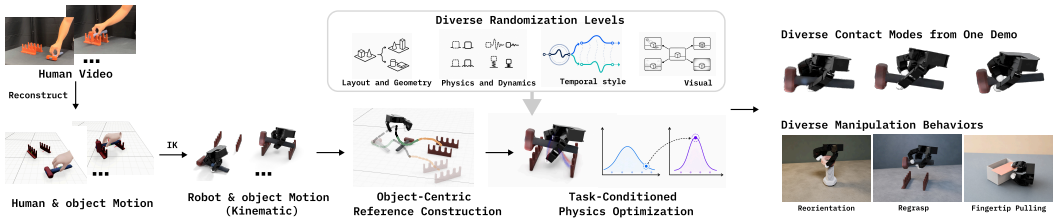

Figure 2: **Pipeline of DEXSEED.** From a few human RGB-D videos, DEXSEED builds an object-centric, phase-aligned manipulation prior, optimizes robot controls through task-conditioned physical rollouts under layout, geometry, physics, and temporal randomization, and renders successful demonstrations with visual randomization for policy learning, yielding diverse contact strategies and manipulation behaviors from a small seed set.

instead performs task-conditioned, time-warped generation through physical rollouts, using human videos as phase-aligned contact priors rather than trajectories to copy.

2.2 POLICY LEARNING FOR DEXTEROUS MANIPULATION

Dexterous manipulation remains challenging due to high-dimensional actions and contact-rich dynamics (Salisbury & Craig, 1982; Mordatch et al., 2012). Reinforcement learning has shown strong sim-to-real results through large-scale domain randomization (Akkaya et al., 2019; Singh et al., 2024; Chen et al., 2023a; 2022; Yin et al., 2023; Qin et al., 2022; Wan et al., 2023), but often relies on task-specific dense rewards and yields unnatural motions. Imitation learning avoids reward design by training from teleoperated or wearable-glove demonstrations (Arunachalam et al., 2022; Qin et al., 2023; Wang et al., 2024a; Fu et al., 2024; Wan et al., 2025; Guzey et al., 2025; Xu et al., 2025), yet real-robot data collection remains costly, embodiment-specific, and hard to scale. DEXSEED instead synthesizes large, diverse, physically valid datasets from a few human RGB-D videos across embodiments and task settings.

3 METHOD

DEXSEED synthesizes a large, diverse, physically valid dataset from a small set of human RGB-D videos $\mathcal{D}_H$ under joint randomization of layout, geometry, and physics. Unlike physics-based retargeting, which treats the source as a frame-aligned trajectory to track, DEXSEED treats each human demonstration as an *object-centric, phase-aligned prior over task-relevant contact structure* and optimizes robot behavior directly through physical rollouts. The generator combines temporal style augmentation, which makes rollout timing an explicit data-generation axis, with a task-conditioned objective that pairs terminal task completion with a contact-mask-weighted reference prior, optimized by a sampling-based inner loop over full physical rollouts.

Each generated demo is conditioned on simulation variables $\xi_{\text{sim}} = (\xi_{\text{layout}}, \xi_{\text{geom}}, \xi_{\text{phys}}) \sim p_{\text{sim}}$ that govern the simulator dynamics f_ξ, and visual variables $\xi_{\text{vis}} \sim p_{\text{vis}}$ that affect only rendering. The resulting dataset is $\mathcal{D}_{\text{gen}} = \{(\tau_R, o_R, \xi_{\text{sim}}, \xi_{\text{vis}})\}$, where τ_R is a robot rollout and o_R its rendered observations. The pipeline (Fig. 2) proceeds as follows: Sec. 3.1 reviews the tracking formulation we modify; Sec. 3.2 converts each video into a phase-aligned reference $\bar{\phi}_k(\sigma)$ and contact mask $\bar{c}_k(\sigma)$; Sec. 3.3 generates robot rollouts with temporal augmentation and task-conditioned physical optimization; and Sec. 3.4 filters, renders, and packages accepted rollouts for policy training. For brevity, we write ξ for ξ_{sim} when unambiguous.

3.1 PRELIMINARIES

We build on two prior formulations. *Kinematic object-centric transformation* (Mandlekar et al., 2023; Garrett et al., 2024; Jiang et al., 2025) segments each source demo into object-centric subtasks, applies a constant SE(3) transform $T_o \bar{T}_o^{-1}$ per subtask (with $\bar{T}_o$ and T_o the object poses in the source and new scenes; bars throughout denote source-scene quantities), and connects subtasks via free-space interpolation; this preserves action–object relative poses across scenes but never optimizes through contact dynamics. *Physics-based retargeting* (Pan et al., 2025; Li et al., 2025b), originally developed to retarget human motion onto a robot under contact and actuation constraints,

searches for controls $U = \{u_t\}_{t=0}^{T-1}$ whose simulated rollout follows a reference trajectory $x_{0:T}^{\mathrm{ref}}$: $\min_U \; J_{\mathrm{ret}}(U) = \sum_{t=0}^{T-1} \big[\, \ell\big(x_{t+1}, x_{t+1}^{\mathrm{ref}}\big) + \ell_u(u_t) \,\big]$ s.t. $x_{t+1} = f_\xi(x_t, u_t)$. Although this enforces physical feasibility, it remains a frame-aligned tracking objective: the horizon and contact schedule are inherited from the source, and task success is encouraged only indirectly through reference matching. DEXSEED keeps physics-in-the-loop optimization but repurposes it for data generation. We replace per-frame tracking with a contact-mask-weighted prior on a phase-aligned reference (Secs. 3.2, 3.3.2), add terminal task completion as the primary signal (Sec. 3.3.2), and decouple the rollout horizon from the source via temporal style augmentation (Sec. 3.3.1).

3.2 OBJECT-CENTRIC REFERENCE PREPARATION

From each human RGB-D video, we construct two artifacts for the physics optimizer: a layout-conditioned, phase-aligned reference $\bar{\phi}_k(\sigma)$ and a contact mask $\bar{c}_k(\sigma)$ for each subtask.

Source reconstruction and parsing. We recover per-frame object pose $\bar{T}_{o,t} \in SE(3)$ with FoundationPose (Wen et al., 2024), MANO hand parameters with HAMER (Pavlakos et al., 2024), and object meshes using ARCode (AR Code, 2022). The MANO reconstruction provides wrist pose $\bar{T}_{h,t} \in SE(3)$ and fingertip positions $\{\bar{p}_{f_i,t} \in \mathbb{R}^3\}_{i=1}^5$; we also retarget the MANO hand to the target robot by fingertip IK to initialize the physics optimizer. The source trajectory is split into object-centric *subtasks* separated by free-space *transits*. Each subtask has an anchor frame b_k chosen from the task scene, such as the manipulated object, target, or support object, with boundaries obtained from annotations or proximity/contact heuristics on $\bar{T}_{o,t}$ and $\bar{p}_{f_i,t}$.

Layout-conditioned phase reference and contact mask. For subtask k, let $\bar{T}_{b_k}$ and T_{b_k} be the anchor poses in the source and sampled scenes. We apply the per-subtask transform $A_k = T_{b_k} \bar{T}_{b_k}^{-1}$ to the source object, wrist, and fingertip quantities $(\bar{T}_{o,t}, \bar{T}_{h,t}, \bar{p}_{f_i,t})$, then resample them over a normalized phase $\sigma \in [0,1]$ to obtain $\bar{\phi}_k(\sigma)$. This reference stacks the layout-conditioned object state, wrist pose, fingertip keypoints, and task-relevant relational quantities used by the physics objective. We also extract a smoothed contact mask $\bar{c}_k(\sigma) \in [0,1]$ from source hand–object distance: it is high when any fingertip is near the manipulated-object surface and smoothly decays during free-space transits. Transits are connected by world-frame interpolation, holding or interpolating finger posture according to the grasp state. Finally, per-subtask references and masks are concatenated into trajectory-level $\bar{\phi}(\sigma)$ and $\bar{c}(\sigma)$ on $[0,1]$. This stage produces only a kinematic prior; physical feasibility is enforced by the optimizer in Sec. 3.3.

3.3 PHYSICS-INFORMED CONTACT-RICH GENERATION

Given the kinematic reference and contact mask $(\bar{\phi}_k(\sigma), \bar{c}_k(\sigma))$, the generator samples a task instance ξ_{sim} and temporal style ζ (Sec. 3.3.1), then optimizes robot controls with a task-conditioned objective whose candidates are evaluated by full physical rollout (Sec. 3.3.2).

3.3.1 TEMPORAL STYLE AUGMENTATION

Contact-rich manipulation may require different timing under randomized geometry and dynamics, making rigid frame alignment to the source overly restrictive. We therefore treat timing as an explicit data-generation axis. Before each rollout, we sample a temporal style $\zeta = (\rho, w)$, where ρ sets the rollout length $T \approx \rho T_{\mathrm{src}}$ and $w : [0,1] \to [0,1]$ maps normalized time to reference phase $\sigma_t = w(t/T)$. We draw ρ from a clipped log-normal distribution with median 1, and construct w by sampling a smooth local-speed profile on cubic B-spline bases, clipping speeds to $[r_{\min}, r_{\max}]$, and integrating to obtain a monotone warp with $w(0) = 0$ and $w(1) = 1$. The sampled ζ is held fixed during inner-loop optimization, so the sampler searches only over robot controls U rather than exploiting the timing variables. Thus, temporal variation becomes a deliberate randomization axis alongside ξ_{sim}, not a degree of freedom optimized away.

3.3.2 TASK-CONDITIONED OBJECTIVE AND OPTIMIZATION

With temporal style ζ fixed, we optimize controls using a task-conditioned objective that replaces frame-wise tracking with a contact-mask-weighted reference prior and uses terminal task completion

as the primary signal:

$$U^* = \arg\min_U J(U;\xi,\zeta), \quad x_{t+1} = f_\xi(x_t, u_t),$$
$$J = \lambda_{\text{task}} L_{\text{task}} + \lambda_{\text{ref}} L_{\text{ref}} + \lambda_{\text{reg}} L_{\text{reg}}. \tag{1}$$

Task term L_{task}. We define task completion as a terminal cost, $L_{\text{task}} = \ell_{\text{term}}(x_T;\xi)$, where ℓ_{term} scores task-specific completion quantities, e.g., grasp stability, lift height, or receptacle alignment. The term is terminal-only: intermediate guidance is supplied by L_{ref}, while dense progress costs would either be redundant or bias rollouts toward source-like states. The same terminal thresholds are used as hard success predicates in the post-rollout filter (Sec. 3.4).

Contact-mask–weighted reference prior L_{ref}. The reference term encourages the rollout to follow the phase reference only when contact is task-relevant: $L_{\text{ref}} = \frac{1}{T+1}\sum_{t=0}^{T} \alpha(\sigma_t) \left\|\phi(x_t;\xi) - \bar{\phi}(\sigma_t)\right\|^2$. Here $\phi(x_t;\xi)$ extracts the same relational quantities stacked in $\bar{\phi}$, and $\bar{c}(\sigma)$ is the trajectory-level contact mask from Sec. 3.2. The weight $\alpha(\sigma)$ is low during free-space transits and high during contact establishment, manipulation, and release. This phase-dependent weighting makes $\bar{\phi}$ a *prior* rather than a tracking target: free-space motion can adapt to the sampled ξ and ζ, while contact phases preserve the demonstrated hand–object relation. By contrast, physics-based retargeting (Pan et al., 2025; Li et al., 2025b) applies contact-phase-agnostic tracking weights and therefore remains tied to the source trajectory.

Regularization L_{reg}. L_{reg} penalizes control effort and action smoothness, normalized by horizon length to avoid biasing the sampler toward shorter rollouts under variable $T \approx \rho T_{\text{src}}$. Hard feasibility constraints, including joint and actuator limits, object loss or penetration, and persistent unstable contact, are enforced by the post-rollout filter rather than folded into L_{reg}, ensuring that retained demonstrations are executable rather than merely low-cost.

Optimization. We minimize J with an MPPI-style sampler initialized from the kinematic retargeting in Sec. 3.2. At each iteration, the sampler perturbs the current control sequence, evaluates each candidate by full physical rollout under f_ξ, and reweights candidates to update U, so the search operates directly over executable robot behavior. Following SPIDER (Pan et al., 2025), we anneal the proposal covariance and temperature across iterations, enabling broad contact-mode exploration early and local refinement later. We also use robust evaluation, where the candidate cost becomes $\widetilde{J}(U;\xi,\zeta) = \max_{\delta\in\Delta(\xi)} J(U;\xi\oplus\delta,\zeta)$, where $\Delta(\xi)$ contains local layout, geometry, and physics perturbations, encouraging rollouts that remain feasible under nearby variations.

3.4 Dataset Construction

Generation and filtering. Each generation attempt samples a source demonstration and simulation variables $\xi_{\text{sim}} = (\xi_{\text{layout}}, \xi_{\text{geom}}, \xi_{\text{phys}})$ spanning object layout, geometry, and physical parameters. The pipeline builds the layout-conditioned reference (Sec. 3.2), optimizes controls through physical rollout (Sec. 3.3), and accepts the rollout only if it satisfies terminal task success under the thresholds of Sec. 3.3.2. We further reject rollouts that violate joint or actuator limits, lose or penetrate objects, or exhibit persistent unstable contact.

Visual rendering and policy training. Accepted rollouts are rendered with visual variables $\xi_{\text{vis}} \sim p_{\text{vis}}$ sampled independently of ξ_{sim}, covering lighting, material appearance, scene textures, and image-level perturbations, with camera extrinsics fixed to the deployment calibration. This rendering stage is decoupled from physical generation and can be paired with any data-generation method. The final dataset $\mathcal{D}_{\text{gen}}$ stores states, actions, RGB observations, sampled randomization variables, and success labels. We train standard imitation policies on $\mathcal{D}_{\text{gen}}$: each policy takes robot proprioception and two RGB views as input and predicts a chunk of wrist poses and target finger-joint positions, instantiated as a flow-matching (Lipman et al., 2022) policy.

4 Experimental Evaluation

We evaluate DexSeed along two axes: data-generation quality and downstream policy learning. First, we measure whether generated demonstrations are successful and diverse under randomized layouts, geometries and physics. Second, we train visuomotor policies on the generated data and evaluate both sim-to-sim and zero-shot sim-to-real transfer. Using only three human videos per task, DexSeed yields policies that transfer to challenging real-world dexterous manipulation.

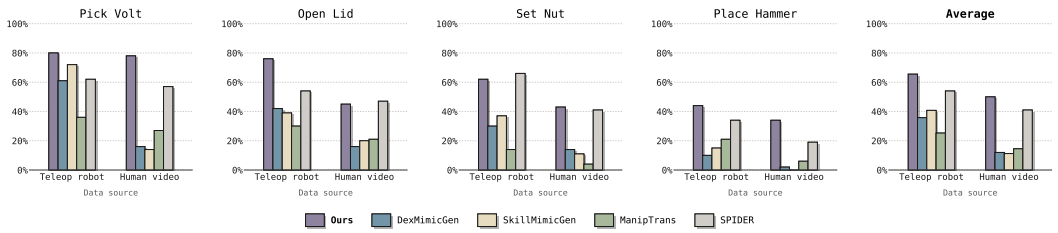

Figure 3: **Data-generation success rate (↑) under D2.** Per-task success of each method on the four tasks under the most challenging randomization regime (joint layout, geometry, and physics). Each bar is computed over 1000 generation attempts for the corresponding method, task, and source type, using three teleoperated robot trajectories or three human videos as source demonstrations.

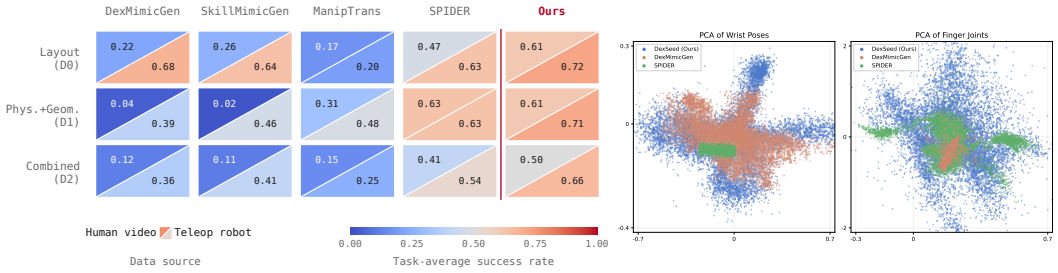

Figure 4: **Coverage and diversity of generated data. Left:** average data-generation success rate (↑) across randomization regimes (D0/D1/D2) and source types (teleop, human video), averaged over the four tasks. **Right:** 2D PCA of wrist poses and finger-joint actions on Place Hammer under D2, where wider spread indicates more diverse generated trajectories.

4.1 Experimental Setup

Tasks setting. We evaluate on four contact-rich tasks spanning functional grasping, in-hand regrasping, and fingertip pushing: **Pick Volt**—pick up a handheld voltmeter and place it into a target basket; **Open Lid**—slide open the lid of a box; **Set Nut**—pick up a nut and mount it onto a bolt; and **Place Hammer**—grasp a hammer, reorient it in hand, and insert it into a matching holder. All data-generation experiments are conducted in MuJoCo. We consider three randomization regimes: **D0**, layout randomization under nominal physics, following MimicGen (Mandlekar et al., 2023); **D1**, physics and object geometry randomization under the source layout; and **D2**, joint layout, geometry, and physics randomization. Randomization ranges and task details are provided in the supplementary material.

Baselines. We compare with four representative data-generation methods: **DexMimicGen** (Jiang et al., 2025) and **SkillMimicGen** (Garrett et al., 2024), which generate demonstrations by object-centric replay and skill composition; **SPIDER** (Pan et al., 2025), which performs physics-based retargeting; and **ManipTrans** (Li et al., 2025b), which uses RL-based tracking to convert human demonstrations into robot trajectories.

4.2 Data Generation Comparison

We compare DexSeed with four baselines under two source regimes: *teleoperated robot sources*, which are physically feasible but embodiment-specific, and *human video sources*, which are embodiment-agnostic but require retargeting before use. For each task and source type, we use only three source demonstrations and adopt XHand as the default embodiment. To isolate the generation algorithm, all methods share the same kinematic retargeting when starting from human videos.

DexSeed attains the highest data-generation success. Fig. 3 reports per-task success rates under D2, the most challenging regime with joint layout, geometry, and physics randomization. From teleoperated sources, DexSeed reaches 65.5% average success, outperforming the strongest baseline by **11.5** percentage points; from human videos, it reaches 50.0%, improving over the strongest baseline by **9.0** points. Fig. 4 (left) shows consistent gains across D0, D1, and D2 for both source types. Replay-based methods are competitive on layout-only randomization but degrade under ge-

Data Gen. Method	Pick Volt	Open Lid	Set Nut	Place Hammer
DexMimicGen (Jiang et al., 2025)	**0.91**	0.45	0.39	0.33
SkillMimicGen (Garrett et al., 2024)	0.86	0.47	0.45	0.31
ManipTrans (Li et al., 2025b)	0.72	0.51	0.22	0.21
SPIDER (Pan et al., 2025)	0.76	0.60	0.60	0.36
Ours	0.89	**0.71**	**0.77**	**0.72**

Table 1: **Policy success rate (↑) across four tasks.** All policies use 1000 successful D2 demonstrations and identical training and evaluation.

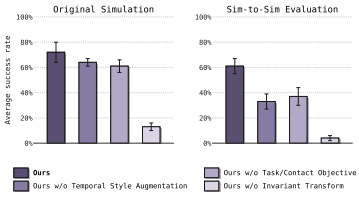

Figure 5: **Component ablations on Place Hammer.** Policy success (↑) in the source simulator and after transfer to Isaac Lab.

ometry/physics changes and human-video retargeting, while physics-based tracking methods handle imperfect sources better but remain tied to the source layout and contact schedule. DEXSEED is the only method that remains strong across all regimes and source modalities.

DEXSEED produces substantially more diverse trajectories than prior methods. Fig. 4 (right) shows PCA projections of wrist poses and finger-joint actions for generated trajectories on Place Hammer under D2. DEXSEED expands the three source demonstrations into a markedly wider region in both spaces, while baseline trajectories remain clustered near the source. Compared with DexMimicGen, which mainly interpolates and replays finger-joint actions, DEXSEED produces different finger behaviors under different physical parameters because contact strategies are re-optimized rather than copied. Compared with SPIDER, which tracks the source wrist trajectory, DEXSEED broadens wrist-pose coverage through layout-conditioned generation and motion planning between manipulation segments. Thus, DEXSEED combines the spatial coverage of kinematic generation with the physical validity of rollout-based optimization.

Cross-embodiment. Due to the cost of running full optimization across hands, we use Place Hammer as the cross-embodiment testbed and evaluate DEXSEED with four dexterous hands under D2 using the same three human videos. It achieves data-generation success rates of 50% on XHand, 39% on LEAP, 67% on Allegro, and 54% on Inspire. This shows that embodiment-conditioned optimization can reuse the same human videos across robot hands.

4.3 POLICY LEARNING WITH GENERATED DEMONSTRATIONS

To isolate the effect of data generation, we train the same flow-matching visuomotor policy on every generated dataset. The policy takes robot proprioception and two RGB views as input and predicts a chunk of wrist poses and target finger-joint positions; architecture and training details are in the Supp. We generate data under D2 with both XHand and LEAP Hand, evaluate sim-to-sim transfer in Isaac Lab, and deploy LEAP-Hand policies zero-shot in the real world with randomized initial layouts over 20 trials per task.

Does the choice of data-generation method affect downstream policy performance? We generate 1000 successful demonstrations per task with each method and train the same policy on each dataset (Tab. 1). On the relatively simple Pick Volt task, DEXSEED matches DexMimicGen and SkillMimicGen, suggesting that all methods provide sufficient coverage. On Open Lid, which requires precise fingertip contact, DEXSEED slightly outperforms SPIDER and ManipTrans. The gap becomes much larger on Set Nut and Place Hammer, where complex in-hand manipulation and layout generalization are both required: DEXSEED reaches up to 2× the success rate of the strongest baseline and improves average success by **33.2%** across all tasks.

Which components of DEXSEED drive this gain? We ablate three design choices on Place Hammer and report average policy success in the original simulator and after sim-to-sim transfer (Fig. 5). Removing the invariant transform causes the largest drop in both settings, as the policy loses layout-aligned demonstrations. Removing temporal style augmentation or task and contact objective causes only modest drops in the original simulator (8.0 % and 11.0 %), but much larger drops under sim-to-sim transfer (28.0 % and 24.0 %), showing that temporal-style diversity and task and contact objective are critical for transfer rather than for in-distribution performance.

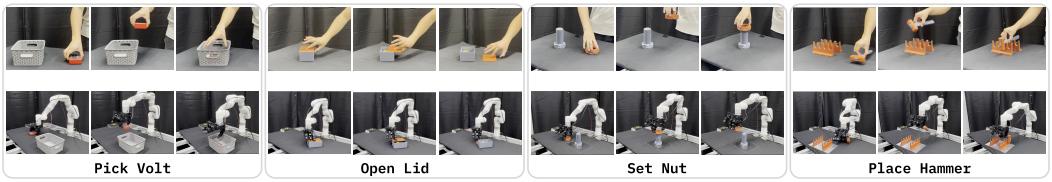

Figure 6: **Zero-shot real-world deployment.** Human RGB-D source videos (top) and policy rollouts (bottom) across four dexterous tasks. Each task is evaluated over 20 trials.

Can policies trained on DEXSEED data be deployed zero-shot in the real world? Starting from only three human videos per task, we evaluate zero-shot sim-to-real transfer on all four tasks (Fig. 6). The resulting policies attain real-world success rates of 60% (Pick Volt), 45% (Open Lid), 40% (Set Nut), and 35% (Place Hammer). Despite a non-trivial sim-to-real gap, these numbers show that DEXSEED converts a handful of human videos into synthetic data diverse enough to support zero-shot transfer to challenging dexterous manipulation, highlighting its practical utility for real-world deployment.

How does policy performance scale with the amount of generated data? We train policies on $\{100, 500, 1000, 3000\}$ demonstrations for Pick Volt and Place Hammer (Fig. 7). Success improves consistently with data scale in both sim-to-sim and real-world evaluation, showing that randomized demonstrations from DEXSEED continue to provide useful supervision beyond the small-data regime.

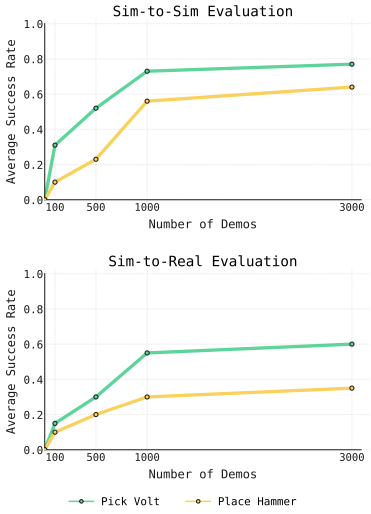

Figure 7: **Scaling policy with generated data.** Policy success rate (↑) as the number of generated demos grows.

4.4 FAILURE AND COMPUTATIONAL COST ANALYSIS

Generation failures mainly occur when noisy hand–object reconstruction, poor initial retargeting, or difficult layout/geometry/physics samples cause missed contacts, object loss, or terminal task failure; real-world failures are mostly due to residual sim-to-real contact mismatch and object slip. DEXSEED is more expensive than kinematic replay because it evaluates candidates through physical rollouts, but generation is offline and parallelizable across sources, randomized instances, and optimizer samples. On a single NVIDIA RTX 4090 GPU, the pipeline takes approximately 2.2 seconds per output frame on average: 1.9 seconds for physics-based optimization and 0.3 seconds for visual rendering, corresponding to an overall throughput of approximately 0.45 FPS. This cost is incurred offline; deployed policies require no test-time trajectory optimization. Further details are provided in the supplementary material.

5 CONCLUSIONS

We presented DEXSEED, a physics-grounded pipeline that uses a few human RGB-D videos as object-centric contact priors to generate diverse, physically valid dexterous demonstrations through task-conditioned physical rollouts. Across contact-rich tasks and robot hands, DEXSEED improves data generation, policy learning, and zero-shot real-world transfer, offering a scalable alternative to task- and embodiment-specific teleoperation.

Limitations and future work. DEXSEED still assumes task-specific subtask structure and success predicates, which are provided by annotations or heuristics rather than learned automatically. Because generated rollouts are filtered mainly by task success, the accepted dataset may be biased toward easier contact modes; in addition, human demonstrations may not fully exploit robot-specific capabilities such as continuous joints or specialized dexterous maneuvers. Future work will study learned subtask/contact/success models, faster proposal-based generation, and embodiment-aware extensions to broader deformable, multi-object, and open-ended manipulation settings.

AI USE STATEMENT

Generative AI tools were used to assist with LaTeX format conversion and language editing. The authors reviewed all AI-assisted changes and take responsibility for the final content, claims, and artifacts in this work.

REPRODUCIBILITY STATEMENT

The method formulation and optimization procedure are described in Sections 3–4. The appendix provides the complete generation algorithm, implementation and policy-training details, randomization ranges, task and evaluation protocols, ablation definitions, and computational-cost analysis.

REFERENCES

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

AR Code. AR Code. `https://ar-code.com/`, 2022.

Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

Lawrence Yunliang Chen, Chenfeng Xu, Karthik Dharmarajan, Muhammad Zubair Irshad, Richard Cheng, Kurt Keutzer, Masayoshi Tomizuka, Quan Vuong, and Ken Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. *arXiv preprint arXiv:2409.03403*, 2024.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on robot learning*, pp. 297–307. PMLR, 2022.

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

Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv:2302.06671*, 2023b.

Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024.

Caelan Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. *arXiv preprint arXiv:2410.18907*, 2024.

Irmak Guzey, Haozhi Qi, Julen Urain, Changhao Wang, Jessica Yin, Krishna Bodduluri, Mike Lambeta, Lerrel Pinto, Akshara Rai, Jitendra Malik, et al. Dexterity from smart lenses: Multi-fingered robot manipulation with in-the-wild human demonstrations. *arXiv preprint arXiv:2511.16661*, 2025.

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.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Arjun S Lakshmipathy, Jessica K Hodgins, and Nancy S Pollard. Kinematic motion retargeting for contact-rich anthropomorphic manipulations. *ACM Transactions on Graphics*, 44(2):1–20, 2025.

Chengshu Li, Mengdi Xu, Arpit Bahety, Hang Yin, Yunfan Jiang, Huang Huang, Josiah Wong, Sujay Garlanka, Cem Gokmen, Ruohan Zhang, et al. Momagen: Generating demonstrations under soft and hard constraints for multi-step bimanual mobile manipulation. *arXiv preprint arXiv:2510.18316*, 2025a.

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

Kevin Lin, Varun Ragunath, Andrew McAlinden, Aaditya Prasad, Jimmy Wu, Yuke Zhu, and Jeannette Bohg. Constraint-preserving data generation for visuomotor policy learning. *arXiv preprint arXiv:2508.03944*, 2025.

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

Tyler Ga Wei Lum, Olivia Y Lee, C Karen Liu, and Jeannette Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. *arXiv preprint arXiv:2504.12609*, 2025.

Zhao Mandi, Homanga Bharadhwaj, Vincent Moens, Shuran Song, Aravind Rajeswaran, and Vikash Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning. *arXiv preprint arXiv:2212.05711*, 2022.

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

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 137–144, 2012.

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.

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

Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3d with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9826–9836, 2024.

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.

Daniele Reda, Jungdam Won, Yuting Ye, Michiel Van De Panne, and Alexander Winkler. Physics-based motion retargeting from sparse inputs. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3):1–19, 2023.

J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

Himanshu Gaurav Singh, Antonio Loquercio, Carmelo Sferrazza, Jane Wu, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Hand-object interaction pretraining from videos. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3352–3360. IEEE, 2025.

Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

Stephen Tian, Blake Wulfe, Kyle Sargent, Katherine Liu, Sergey Zakharov, Vitor Guizilini, and Jiajun Wu. View-invariant policy learning via zero-shot novel view synthesis. *arXiv preprint arXiv:2409.03685*, 2024.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3891–3902, 2023.

Weikang Wan, Jiawei Fu, Xiaodi Yuan, Yifeng Zhu, and Hao Su. Lodestar: long-horizon dexterity via synthetic data augmentation from human demonstrations. In *9th Annual Conference on Robot Learning*, 2025.

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

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4890–4924, 2024b.

Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17868–17879, 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*, 2025.

Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.

Lujie Yang, HJ Suh, Tong Zhao, Bernhard Paus Graesdal, Tarik Kelestemur, Jiuguang Wang, Tao Pang, and Russ Tedrake. Physics-driven data generation for contact-rich manipulation via trajectory optimization. *arXiv preprint arXiv:2502.20382*, 2025.

Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. *arXiv preprint arXiv:2303.10880*, 2023.

Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023.

A Appendix Overview

This appendix provides additional implementation details, randomization ranges, task definitions, policy-learning settings, extended experimental results, and failure analysis. Sec. B gives additional details of the generation pipeline. Sec. C reports simulation and visual randomization. Sec. D describes tasks, robot hands, source demonstrations, and baselines. Sec. E provides policy-learning and evaluation details. Sec. F includes extended results and ablations. Sec. G reports failure cases and runtime analysis.

B Additional Method Details

B.1 Full Demonstration Generation Algorithm

Alg. 1 summarizes the full data-generation pipeline. Each attempt samples a source demonstration, simulation randomization, and temporal style, then optimizes robot controls through full physical rollouts. A rollout is accepted only if it satisfies the task predicate and quality filter; accepted rollouts are rendered under visual randomization for policy training.

Algorithm 1 Physics-Grounded Demonstration Generation

Require: Source demos $\mathcal{D}_{\text{src}}$ (human videos $\mathcal{D}_H$ or teleoperated demonstrations), simulation randomization p_{sim}, visual randomization p_{vis}, temporal-style distribution p_{time}, simulator f_ξ, success predicate Φ, quality filter Ψ, target number of accepted rollouts N, renderings per rollout R.

Initialize $\mathcal{D}_{\text{gen}} \leftarrow \emptyset$, $n_{\text{acc}} \leftarrow 0$.
while $n_{\text{acc}} < N$ **do**
  Sample source demo $\tau_{\text{src}} \sim \mathcal{D}_{\text{src}}$ and simulation variables $\xi_{\text{sim}} \sim p_{\text{sim}}$.
  Reset the scene, robot, and objects under ξ_{sim}.
  Build layout-conditioned reference and contact mask: $(\bar{\phi}, \bar{c}) \leftarrow \text{BuildReference}(\tau_{\text{src}}, \xi_{\text{sim}})$.
  Sample temporal style $\zeta = (\rho, w) \sim p_{\text{time}}$.
  Optimize controls $U^* \leftarrow \text{PhysicsOptimize}(\bar{\phi}, \bar{c}, \xi_{\text{sim}}, \zeta)$.
  Roll out $\tau_R \leftarrow \text{Rollout}(f_{\xi_{\text{sim}}}, U^*)$.
  if $\Phi(\tau_R; \xi_{\text{sim}}) = 0$ or $\Psi(\tau_R; \xi_{\text{sim}}) = 0$ **then**
    Reject τ_R and continue.
  end if
  $n_{\text{acc}} \leftarrow n_{\text{acc}} + 1$
  for $r = 1, \ldots, R$ **do**
    Sample $\xi_{\text{vis}}^{(r)} \sim p_{\text{vis}}$ and render $o_R^{(r)} \leftarrow \text{Render}(\tau_R, \xi_{\text{vis}}^{(r)})$.
    Add $(\tau_R, o_R^{(r)}, \xi_{\text{sim}}, \xi_{\text{vis}}^{(r)})$ to $\mathcal{D}_{\text{gen}}$.
  end for
end while
return $\mathcal{D}_{\text{gen}}$

The optimizer searches only over robot controls U. The simulation variables ξ_{sim}, visual variables ξ_{vis}, and temporal style ζ are sampled before optimization and held fixed during each attempt. Thus, layout, geometry, physics, appearance, and timing are treated as data-generation axes rather than variables optimized to simplify the task. Unless otherwise specified, we collect N=3000 accepted physical rollouts per task, each rendered R=3 times for policy learning.

B.2 Source Reconstruction and Robot Retargeting

For human-video sources, we capture RGB-D videos of a human hand manipulating the task objects from a fixed calibrated camera. We recover object poses using FoundationPose (Wen et al., 2024), reconstruct MANO hand parameters with HAMER (Pavlakos et al., 2024), and obtain object meshes from ARCode (AR Code, 2022) or scanned/CAD assets. The MANO reconstruction provides wrist pose and five fingertip positions for each frame. All reconstructed quantities are transformed into the simulator world frame using the calibrated camera-to-world transform.

We retarget the MANO fingertips to the robot hand by solving a fingertip inverse-kinematics problem with joint-limit penalties. This retargeted sequence is used only to initialize the physics optimizer and define reference features. It is not directly accepted as a robot demonstration unless the subsequent physical rollout satisfies the task predicate and quality filter.

B.3 Subtask Parsing, Phase Reference, and Contact Mask

Each source demonstration is split into object-centric subtasks separated by free-space transits. Subtask boundaries are obtained from annotations and simple proximity/contact heuristics, and each subtask is assigned an anchor frame b_k, such as the manipulated object, target object, or support object. In our tasks, anchors correspond to the voltmeter and basket for Pick Volt, the box and lid for Open Lid, the nut and bolt for Set Nut, and the hammer and holder for Place Hammer.

For subtask k, let $\bar{T}_{b_k}$ and T_{b_k} denote the anchor poses in the source and sampled scenes. We apply the object-centric transform

$$A_k = T_{b_k} \bar{T}_{b_k}^{-1}$$

to source object poses, wrist poses, and fingertip keypoints. The transformed quantities are resampled over normalized phase $\sigma \in [0, 1]$ to obtain the phase reference $\bar{\phi}_k(\sigma)$, which stacks anchor-relative object state, wrist pose, fingertip keypoints, and task-relevant relational quantities.

We compute a soft contact mask $\bar{c}_k(\sigma) \in [0, 1]$ from fingertip-to-object distances in the source demonstration:

$$c_t = \exp\left(-d_t^2 / 2\sigma_c^2\right), \qquad d_t = \min_i \operatorname{dist}(\bar{p}_{f_i,t}, \mathcal{S}_o),$$

where $\mathcal{S}_o$ denotes the manipulated-object surface. We use $\sigma_c = 2.5$ cm, smooth c_t with a 5-frame window, and resample it over phase to obtain $\bar{c}_k(\sigma)$. Per-subtask references and masks are concatenated into trajectory-level $\bar{\phi}(\sigma)$ and $\bar{c}(\sigma)$. This stage produces only a kinematic, object-centric prior; physical feasibility is enforced by rollout optimization.

B.4 Temporal Style Augmentation

Temporal style augmentation decouples generated rollouts from the source horizon and frame-level contact schedule. For each generation attempt, we sample a temporal style $\zeta = (\rho, w)$. The horizon multiplier ρ sets the rollout length

$$T = \lfloor \rho T_{\mathrm{src}} \rfloor,$$

where T_{src} is the source horizon. We sample ρ from a clipped log-normal distribution with median 1 and range $[0.8, 1.2]$.

The phase warp $w : [0, 1] \to [0, 1]$ maps rollout time to reference phase. We construct w by sampling a smooth local-speed profile on cubic B-spline bases, clipping local speeds to $[r_{\min}, r_{\max}] = [0.5, 2.0]$, and integrating the profile so that $w(0) = 0$ and $w(1) = 1$. During optimization, ζ is held fixed, so the optimizer adapts robot controls to the sampled timing rather than optimizing the timing itself.

B.5 Task-Conditioned Objective and Sampling-Based Optimization

Given a sampled simulation instance ξ and temporal style ζ, we optimize robot controls using

$$U^* = \arg\min_U J(U; \xi, \zeta), \qquad x_{t+1} = f_\xi(x_t, u_t),$$

$$J = \lambda_{\mathrm{task}} L_{\mathrm{task}} + \lambda_{\mathrm{ref}} L_{\mathrm{ref}} + \lambda_{\mathrm{reg}} L_{\mathrm{reg}}.$$

The task term $L_{\mathrm{task}} = \ell_{\mathrm{term}}(x_T; \xi)$ is a shaped terminal completion cost rather than a binary success indicator. For the four tasks, it scores the placement of the voltmeter into the basket, the opening displacement of the lid, the seating of the nut on the bolt, and the insertion of the hammer into the holder. The same terminal thresholds are used as hard success predicates in the post-rollout filter.

The reference term encourages consistency with the phase reference while remaining weak during free-space motion:

$$L_{\mathrm{ref}} = \frac{1}{T+1} \sum_{t=0}^{T} \alpha(\sigma_t) \left\| \phi(x_t; \xi) - \bar{\phi}(\sigma_t) \right\|^2, \qquad \sigma_t = w(t/T),$$

Table 2: Optimization hyperparameters used by DEXSEED.

Hyperparameter	Value / range
Control frequency	20 Hz
Candidates per iteration	256
Optimization iterations	8
Initial temperature η	1.0
Annealing factor	0.7
Action perturbation std.	0.05–0.15 of action range
Horizon multiplier ρ	clipped log-normal in $[0.8, 1.2]$
Local phase speed range $[r_{\min}, r_{\max}]$	$[0.5, 2.0]$
Robust layout perturbation	object xy ± 1 cm, yaw $\pm 3^\circ$
Robust physics perturbation	mass $\pm 10\%$, friction $\pm 10\%$
Renderings per accepted rollout R	3

where

$$\alpha(\sigma) = \alpha_{\text{free}} + (\alpha_{\text{contact}} - \alpha_{\text{free}})\bar{c}(\sigma).$$

We use $\alpha_{\text{free}} = 0.1$ and $\alpha_{\text{contact}} = 1.0$. Thus, the reference acts as a contact-aware prior rather than a frame-aligned tracking target. The regularization term L_{reg} penalizes control effort and action smoothness, normalized by horizon length.

We optimize U with an MPPI-style sampler initialized from the retargeted source trajectory. At each iteration, we perturb the current control sequence, roll out all candidates in simulation, and update the sequence by cost-weighted averaging:

$$\beta^{(m)} = \frac{\exp(-\widetilde{J}^{(m)}/\eta)}{\sum_j \exp(-\widetilde{J}^{(j)}/\eta)}, \qquad U \leftarrow \sum_m \beta^{(m)} U^{(m)},$$

where $\widetilde{J}^{(m)}$ is the robust candidate cost defined below. We use 256 candidates per iteration and 8 optimization iterations. The proposal covariance and temperature are annealed by a factor of 0.7 after each iteration. For robust evaluation, each candidate is scored under small local perturbations of object pose, friction, and mass using

$$\widetilde{J}(U; \xi, \zeta) = \max_{\delta \in \Delta(\xi)} J(U; \xi \oplus \delta, \zeta).$$

Tab. 2 summarizes all optimizer hyperparameters.

C DOMAIN RANDOMIZATION DETAILS

C.1 LAYOUT, GEOMETRY, AND PHYSICS RANDOMIZATION

We use three randomization regimes in simulation. D0 randomizes object and target layout while keeping nominal geometry and physics. D1 keeps the source layout fixed but randomizes object geometry and physics. D2 jointly randomizes layout, geometry, and physics, and is the main setting used for policy-learning experiments.

For each generation attempt, we sample simulation variables $\xi_{\text{sim}} = (\xi_{\text{layout}}, \xi_{\text{geom}}, \xi_{\text{phys}})$ before optimization. These variables remain fixed during the inner-loop control optimization. Tab. 3 lists the ranges of all randomized simulation variables. If a sampled task instance is physically invalid, e.g., due to initial object collision or unreachable target placement, it is rejected before optimization and resampled. The exact feasible workspace is task-dependent: Pick Volt randomizes the voltmeter and basket, Open Lid randomizes the box and lid geometry while preserving the sliding direction, Set Nut randomizes the nut and bolt within a reachable insertion region, and Place Hammer randomizes the hammer and holder while discarding unreachable insertion poses.

C.2 VISUAL DOMAIN RANDOMIZATION

For RGB policy learning, successful physical rollouts are rendered with visual domain randomization. This stage is applied after the robot and object trajectory has been generated and accepted.

Table 3: Simulation randomization ranges for data generation.

Category	Variable	Range / distribution
Layout	Manipulated-object position	xy ± 10 cm around nominal pose
	Manipulated-object yaw	$\pm 30^\circ$
	Target / receptacle position	xy ± 10 cm around nominal pose
	Target / receptacle yaw	$\pm 30^\circ$
	Initial robot joints	finger joints $\pm 5^\circ$
Geometry	Manipulated-object scale	uniform in $[0.9, 1.1]$
	Target / receptacle scale	uniform in $[0.95, 1.05]$
	Collision margin	uniform in $[0, 2]$ mm
	Object asset variant	sampled from task-specific asset set
Physics	Object mass / density	scale uniform in $[0.7, 1.3]$
	Object–finger friction	uniform in $[0.6, 1.4]$
	Object–table friction	uniform in $[0.6, 1.4]$
	Joint damping	scale uniform in $[0.8, 1.2]$
	Actuator gain	scale uniform in $[0.8, 1.2]$

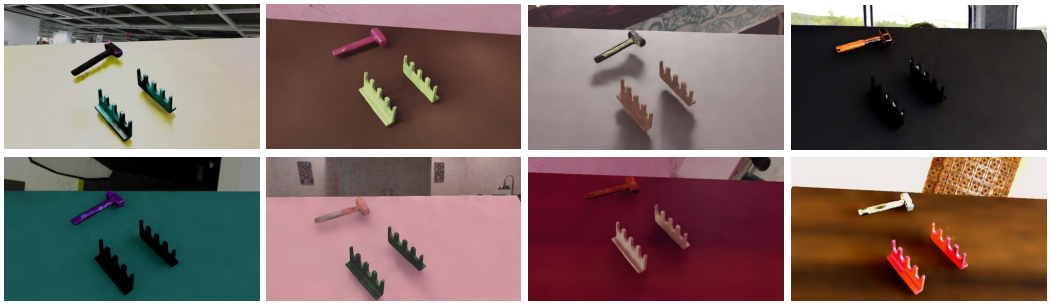

Figure 8: **Visual domain randomization.** Eight renderings of the same accepted Place Hammer initial state under different visual randomization samples. The physical state and camera geometry are unchanged, while lighting, object and robot materials, table/background appearance, and image-level statistics vary across renderings.

Thus, visual randomization changes the rendered observations, but does not change the underlying contact dynamics, object motion, or optimized robot controls.

Fig. 8 visualizes eight randomized renderings of the same Place Hammer initial state. The physical state, camera geometry, and optimized trajectory are fixed across renderings, while lighting, object appearance, robot material, table/background appearance, and image statistics are randomized. Tab. 4 lists all visual randomization variables and ranges. Visual variables ξ_{vis} are sampled independently from ξ_{sim}, so the same accepted physical rollout can be rendered under multiple visual conditions. Unless otherwise specified, camera extrinsics are kept fixed to match the calibrated deployment setup.

In the policy-learning experiments, we render $R = 3$ RGB variants for each accepted physical rollout. All methods use the same rendering pipeline and the same visual randomization distribution, so visual appearance is not a method-specific advantage.

D Tasks, Robots, Sources, and Baselines

D.1 Tasks, Robot Hands, and Source Demonstrations

We evaluate DexSeed on four contact-rich dexterous manipulation tasks. **Pick Volt** requires grasping a handheld voltmeter, lifting it from the table, and placing it into a target basket. **Open Lid** requires establishing precise fingertip contact and sliding open the lid of a box. **Set Nut** requires

Table 4: Visual domain randomization used for RGB demonstration rendering.

Category	Variable	Range / distribution
Lighting	HDRI map	sampled from environment-map library
	HDRI rotation	uniform in $[0, 2\pi]$
	Intensity scale	uniform in $[0.5, 1.5]$
Object	Texture map	sampled from task-specific texture set
	Diffuse tint	RGB multiplier in $[0.7, 1.3]$
	Roughness	uniform in $[0.2, 0.9]$
	Metallic	uniform in $[0.0, 0.3]$
	Specular	uniform in $[0.2, 0.8]$
Robot	Diffuse tint	RGB multiplier in $[0.8, 1.2]$
	Roughness	uniform in $[0.3, 0.8]$
	Metallic	uniform in $[0.0, 0.5]$
	Specular	uniform in $[0.2, 0.8]$
Scene	Table texture	sampled from texture library
	Background image	sampled from background library
	Table roughness	uniform in $[0.3, 0.9]$
	Table specular	uniform in $[0.1, 0.6]$
Image	Color jitter	brightness/contrast/saturation $\pm 20\%$
	Gaussian blur	kernel size 3, probability 0.2
	Motion blur	probability 0.1
	Gaussian noise	std. uniform in $[0, 0.02]$

grasping a nut, aligning it with a bolt, and setting it onto the bolt. **Place Hammer** requires grasping a hammer, reorienting it in hand, and inserting it into a holder. These tasks cover functional grasping, fingertip pushing, in-hand regrasping, and precise placement under randomized layouts and physical parameters. Each task uses a task-specific terminal success predicate for both data-generation filtering and policy evaluation.

All data-generation comparisons are conducted in MuJoCo with XHand as the default robot hand. For cross-embodiment generation, we additionally evaluate LEAP, Allegro, and Inspire hands using the same source demonstrations. For sim-to-sim transfer, policies trained from MuJoCo-generated demonstrations are evaluated in Isaac Lab. For real-world deployment, we use a LEAP Hand with the same action interface as in simulation. All simulated hands use position-controlled finger joints and a Cartesian wrist controller; the policy action contains target wrist poses and target finger-joint positions. Joint limits, action bounds, and actuator limits are enforced during rollout and by the post-rollout quality filter.

We consider two source-demonstration modalities: teleoperated robot trajectories and human RGB-D videos. For each task and source modality, we use three source demonstrations. Teleoperated sources are executable robot trajectories collected in the nominal scene. Human-video sources are RGB-D recordings of a human hand performing the task and are processed as described in Sec. B.2. For human-video experiments, all methods use the same reconstructed object poses, MANO hand trajectories, and fingertip-IK retargeting outputs. For teleoperated-source experiments, all methods start from the same robot demonstrations and subtask annotations. This shared preprocessing isolates differences in the generation algorithms rather than differences in source reconstruction, retargeting, or task specification.

D.2 Baseline Implementations and Fairness Protocol

We compare DexSeed with four representative data-generation baselines: DexMimicGen (Jiang et al., 2025), SkillMimicGen (Garrett et al., 2024), SPIDER (Pan et al., 2025), and ManipTrans (Li et al., 2025b). DexMimicGen and SkillMimicGen are kinematic transform-and-replay methods: they segment the source trajectory, transform object-centric source motions to new task layouts, solve IK for the target hand, and connect subtasks by motion stitching or skill composition. SPIDER

and ManipTrans are physics-based retargeting methods: they start from a retargeted source trajectory and optimize or learn robot behavior that tracks the source motion under robot contact and actuation constraints. Unlike these baselines, DEXSEED does not track the source trajectory frame by frame; it uses the source as a phase-aligned contact prior and optimizes task-conditioned controls through physical rollout.

All methods use the same three source demonstrations per task, the same task assets, and the same randomization regimes D0–D2. For human-video sources, all methods share the same reconstruction and fingertip-IK retargeting pipeline before generation. Each generated rollout is evaluated with the same task predicate and quality filter. For downstream policy learning, each method contributes the same number of successful demonstrations, and all policies are trained with the same architecture, optimizer, training schedule, and evaluation protocol.

E POLICY LEARNING AND EVALUATION DETAILS

E.1 POLICY ARCHITECTURE AND TRAINING

We train visuomotor policies by imitation learning on successful generated demonstrations. Each policy observes RGB images from two fixed external cameras and robot proprioception, including wrist pose, finger-joint positions, finger-joint velocities, and previous actions. The policy predicts an action chunk of target wrist poses and target finger-joint positions. Wrist actions are represented as 6-DoF delta poses in the robot base frame, and finger actions are represented as absolute joint-position targets clipped to robot joint limits.

We use a transformer flow-matching policy. Each RGB view is encoded by a shared ViT-B/14 DINOv2 visual backbone, and proprioceptive history is encoded by an MLP. The visual and proprioceptive embeddings are fused and passed to a transformer action head conditioned on the flow time. Given a demonstration action chunk a and Gaussian noise ϵ, we sample $s \sim \mathcal{U}(0,1)$, form $a_s = (1-s)\epsilon + sa$, and train the policy to predict the flow vector $a - \epsilon$:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{a,\epsilon,s}\left[\left\|v_\theta(a_s, s, o) - (a-\epsilon)\right\|^2\right].$$

At inference, we generate actions by integrating the learned flow for 10 steps and execute only the first action prefix before replanning. Tab. 5 lists all policy-learning hyperparameters.

Unless otherwise specified, we train one policy per task and per data-generation method. For policy-comparison experiments, each policy is trained on 1000 successful demonstrations generated under D2. All methods use the same policy architecture, optimizer, training duration, data augmentation, and checkpoint-selection rule. During training, RGB images are augmented with random crop, color jitter, and mild Gaussian noise. No privileged object pose, simulator state, or randomization parameter is provided to the policy; the policy observes only RGB images and robot proprioception, matching the real-world deployment setting.

E.2 SIMULATION AND REAL-WORLD EVALUATION PROTOCOLS

We evaluate policies in both the original MuJoCo simulator and a separate Isaac Lab environment. MuJoCo evaluation measures performance in the simulator used for data generation, while Isaac Lab evaluation measures sim-to-sim transfer under different contact modeling, rendering, and solver behavior. For each task and policy, we evaluate 100 episodes with randomized initial layouts sampled from the D2 evaluation distribution. A trial is successful if the task-specific predicate is satisfied at the end of the episode and remains stable for at least 0.5s. No test-time optimization, online adaptation, privileged object pose, or simulator randomization parameter is used during evaluation.

For real-world evaluation, we deploy the learned policy zero-shot on the platform shown in Fig. 9, without any real-world fine-tuning. The main real-world results use 3000 successful generated physical rollouts per task; the scaling experiments vary this number as specified in Sec. F.2. The system uses a 16-DoF LEAP Hand mounted on a 7-DoF xArm7, together with two RGB cameras calibrated to approximately match the simulation camera geometry. The policy outputs target wrist poses and finger-joint positions, which are executed by the low-level controller. Each task is evaluated over 20 trials with randomized object and target poses within the same workspace region used in simulation,

Table 5: Visuomotor policy-learning hyperparameters.

Hyperparameter	**Value**
Successful demos per task	3000 unless otherwise specified
RGB cameras	2 fixed external views
Image resolution	$2 \times 224 \times 224$
Visual observation horizon	1 frame
Proprioceptive observation horizon	3 policy steps
Visual stream frequency	20 Hz
Policy query frequency	12.5 Hz
Low-level control frequency	50 Hz
Action horizon	16 low-level steps
Action horizon duration	320 ms
Executed prefix length	4 low-level steps
Executed prefix duration	80 ms
Vision backbone	ViT-B/14 DINOv2
Policy backbone	transformer flow-matching policy
Flow sampling steps	10
Optimizer	AdamW
Policy learning rate	3×10^{-4}
Vision backbone learning rate	3×10^{-5}
Batch size	256
Training epochs	200
Weight decay	1×10^{-4}
EMA decay	0.999

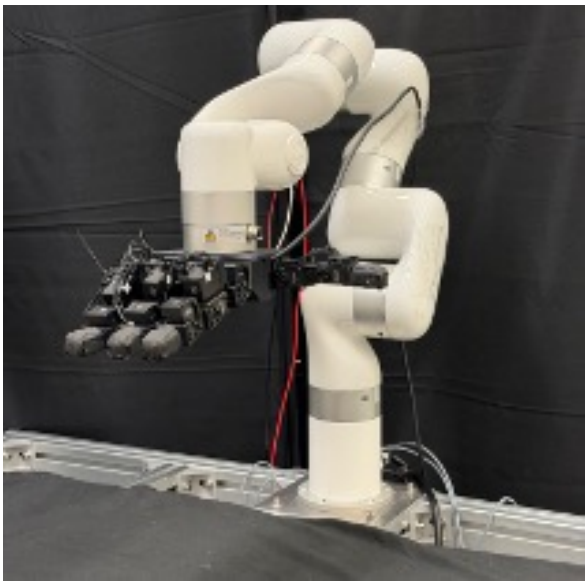

Figure 9: **Real-world platform.** Our real-world setup uses a 16-DoF LEAP Hand mounted on a 7-DoF xArm7, with two fixed RGB cameras for zero-shot policy evaluation.

and the scene is reset by a human between trials. Success is judged from the final object configuration and stability using the same task-level definitions as in simulation. No test-time trajectory optimization, online adaptation, or manual correction is used during policy execution.

F Additional Experimental Results

The main paper reports the primary data-generation, policy-learning, sim-to-real, cross-embodiment, and scaling results. Here we provide additional details for interpreting these results,

including ablation definitions, diversity-analysis protocol, and evaluation details for scaling and cross-embodiment experiments.

F.1 Ablation and Diversity Analyses

We ablate three core components of DexSeed on Place Hammer, the task with the most complex combination of grasping, reorientation, and insertion. In *w/o invariant transform*, we remove the per-subtask object-centric anchor transform and use a world-frame reference, so demonstrations no longer align consistently across randomized layouts. In *w/o temporal style augmentation*, we set $\rho = 1$ and $w(\tau) = \tau$, forcing generated rollouts to inherit the source horizon and phase timing. In *w/o task and contact objective*, we replace the terminal task objective and contact-mask-weighted prior with a tracking-style objective, making the generation closer to physics-based retargeting. The results in Fig. 5 show that all three components contribute to transfer, with the largest drop caused by removing the invariant transform.

For the diversity analysis in Fig. 4, we collect successful generated trajectories on Place Hammer under D2. Wrist poses are represented by translation and axis-angle rotation, and finger actions are represented by target joint positions. Features are normalized using the mean and standard deviation computed over the union of all methods before fitting PCA. The same PCA projection is then applied to each method. A wider spread in the projected space indicates broader coverage of wrist motions and finger-joint strategies. This analysis shows that DexSeed expands the three source demonstrations into diverse contact-rich behaviors rather than simply replaying or tracking the source trajectories.

F.2 Cross-Embodiment and Scaling Details

For cross-embodiment evaluation, we use Place Hammer as the testbed and reuse the same three human videos across XHand, LEAP, Allegro, and Inspire hands. Only the target robot model, kinematics, actuation limits, and contact parameters are changed. The subtask structure, source reconstruction, temporal-style distribution, task predicate, and D2 randomization protocol are kept fixed. Data-generation success for each hand is computed over 1000 generation attempts. This evaluates whether the same human-video prior can seed physics-grounded generation for different dexterous hands when a kinematic and dynamic model is available.

For scaling experiments, we train policies with $\{100, 500, 1000, 3000\}$ successful demonstrations generated by DexSeed under D2 on Pick Volt and Place Hammer. All policies use the same observation/action space, flow-matching architecture, training procedure, and evaluation protocol as in Sec. E. Unless otherwise specified, demonstrations are sampled from the same generated pool without changing the source videos or randomization distribution. The scaling results in Fig. 7 show that increasing the number of generated demonstrations improves both sim-to-sim and real-world performance, indicating that the generated data remains useful beyond the small-data regime.

G Failure Cases and Computational Cost

Failure cases. Generation failures mainly occur when source reconstruction noise, poor retargeting initialization, or difficult randomized instances lead to missed contact, object loss, unstable contact, or terminal task failure. These failures are filtered before policy training by the task predicate and quality filter, which reject rollouts with task failure, joint or actuator limit violations, object loss, severe penetration, or persistent unstable contact. Real-world failures are mostly caused by residual sim-to-real gaps in contact dynamics, such as object slip, slightly misplaced fingertip contact, visual calibration error, or long-horizon compounding. These failures are more common in Set Nut and Place Hammer, where small alignment errors can prevent final insertion.

Computational cost. DexSeed is more expensive than kinematic replay because each candidate control sequence is evaluated through full physical rollout. However, generation is offline and parallelizable across source demonstrations, randomized task instances, and optimizer samples; in practice, we distribute generation across multiple GPUs. Deployed policies do not require test-time trajectory optimization.

Table 6: Average processing time per output frame on a single NVIDIA RTX 4090 GPU.

Component	Time (s/frame)
Physics optimization	1.9
Visual rendering	0.3
Total	2.2

Tab. 6 reports the average processing time per output frame on a single NVIDIA RTX 4090 GPU, using the same output-frame denominator for both stages. Physics-based optimization takes 1.9 seconds per frame and visual rendering takes 0.3 seconds per frame. The sequential stages therefore take approximately 2.2 seconds per frame in total, corresponding to $1/2.2 \approx 0.45$ FPS.

Physics optimization accounts for approximately 86% of the combined processing time, as each attempt evaluates many candidate control sequences through physical rollouts. Rendering is performed only for accepted rollouts, with $R = 3$ visual variants and two camera views per variant. Dataset-level generation cost also depends on trajectory lengths and the number of rejected attempts. In our experiments, generating 1000 successful demonstrations typically requires 1.5–$3.0\times$ more attempts depending on the task and randomization regime.