SIMWEAVER: ZERO-SHOT RGB SIM-TO-REAL FOR DEFORMABLE MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

RGB sim-to-real for deformable manipulation has remained largely unsolved without real-world fine-tuning. We present SIMWEAVER, which trains zero-shot RGB VLA policies on 200 simulated demonstrations per task, reaching above 80% per-task and 91% average real-world success across 5 diverse deformable tasks including handled non-woven bag manipulation, without teleoperation or per-task physical-parameter calibration. SimWeaver combines a reliable measurement-backed simulator (SimWeaver-Sim) with an extensible asset framework supporting single-image generation (SimWeaver-Asset), a deterministic topology-aware trajectory synthesizer (SimWeaver-Syn), and a sim-to-real protocol with ISP-aware photometric augmentation (SimWeaver-Real). On silk grasping, the sim-trained policy reaches 100% under visual distribution shifts where real-data baselines drop to 9–70%, at two orders of magnitude lower per-trajectory cost. We will release SimWeaver and a representative asset subset. Project page: anonymous link.

1 INTRODUCTION

Recent studies (Lin et al., 2025; AgiBot-World-Contributors et al., 2025) show that policy learning for robotic manipulation follows an approximate power-law relationship with respect to the number of training demonstrations and environments. Real-world data offer the highest fidelity for policy learning but are labor-intensive, costly, and difficult to scale across environments. While such collection is tractable for rigid-object manipulation, it becomes prohibitive for deformable objects due to large configuration spaces, non-linear shape variation, and complex interactions with rigid end-effectors. To enable scalable data generation, the community has extensively explored synthetic demonstrations via physics-based simulation. Although effective for rigid manipulation, prior approaches have not achieved success in deformable-object settings without real-world fine-tuning.

Several factors contribute to this limitation. Widely used simulators, such as NVIDIA Isaac Sim, exhibit unreliable contact dynamics for thin structures and fail to capture realistic physical behavior of deformable materials. Consequently, demonstration synthesis pipelines developed for rigid-body manipulation transfer poorly to deformable settings. Moreover, pixel-based end-to-end policy learning for deformable object manipulation has stagnated due to the unresolved sim-to-real gap, prompting a shift in the community toward depth and point-cloud representations over raw visual inputs.

Concurrent SIM1 (Zhou et al., 2026) demonstrates pixel-based sim-to-real transfer for garment folding, reporting 90% zero-shot real-world success on T-shirt folding with 10k generated samples. However, this relies on per-task expert-guided material calibration and teleoperated demonstrations for trajectory generation. These constraints leave unresolved whether pixel-based sim-to-real transfer for deformable manipulation can be made teleoperation-free, scalable beyond folding-centric tasks, and deployable without per-asset manual calibration.

In this work, we present *SimWeaver*, a sim-to-real framework that addresses these three constraints (Fig. 1). SimWeaver integrates a deformable simulator (SimWeaver-Sim) with an extensible asset framework (SimWeaver-Asset), a topology-aware trajectory synthesizer (SimWeaver-Syn), and a sim-to-real deployment protocol (SimWeaver-Real). Unlike point-cloud-based pipelines, SimWeaver adopts a pixel-based formulation that robustly handles visually challenging materials including dark, low-texture, and reflective fabrics.

We summarize our main contributions as follows:

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.

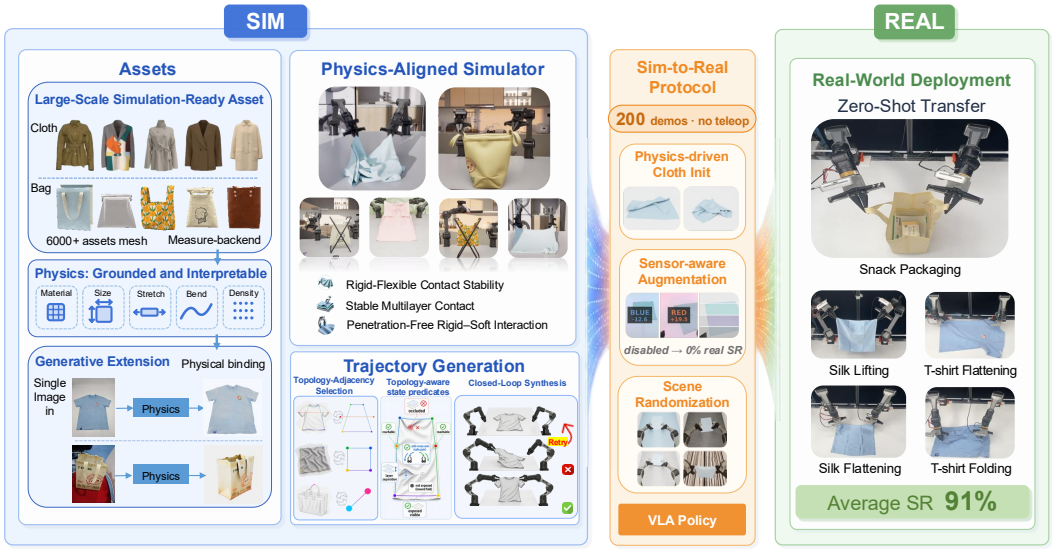


Figure 1: **System architecture.** Assets feed the simulator; SimWeaver-Syn produces 200 deterministic demonstrations per task from a single seed; the demonstrations train a VLA policy that is deployed zero-shot on real hardware; the pipeline also supports closed-loop evaluation in simulation.

- **Zero-shot RGB sim-to-real for deformable manipulation with only 200 synthesized demonstrations per task.** SimWeaver achieves an average of 91% real-world success (>80% per task) across 5 diverse deformable tasks—including handled non-woven bag manipulation—without teleoperation or per-task physical-parameter calibration. On silk grasping, the sim-trained policy maintains 100% under visual distribution shifts where real-robot-trained baselines drop to 9–70%.
- **Reliable deformable simulator with robust collision handling, measurable physical correspondence, and an extensible asset framework.** SimWeaver-Sim introduces robust collision handling, penetration-prevention mechanisms, and trajectory-replay determinism, addressing contact-reliability failures of widely-used solvers (Isaac Sim, VBD-based) on thin fabrics, with physical parameters in direct real-world correspondence. The simulator additionally supports an extensible asset framework (SimWeaver-Asset) for mesh import and single-image asset generation across diverse deformable categories including bags with deformable handles.
- **Topology-aware trajectory synthesis without teleoperation.** SimWeaver-Syn generates high-quality deterministic demonstrations from a single seed without learned generative models, teleoperation, or post-hoc discriminator filtering.

2 Related Work

2.1 Deformable Simulation

General-purpose deformable simulators such as SoftGym (Lin et al., 2021), DiffCloth (Li et al., 2022), PlasticineLab (Huang et al., 2021), and DaXBench (Chen et al., 2023b) cover cloth, rope, fluids, and elastoplastic materials, but are not primarily designed for robot–deformable interaction and often suffer from unreliable grasping contact, instability in contact-rich regimes, or limited support for thin-shell objects with handles.

Recent robot-oriented simulators reduce this gap. GarmentLab (Lu et al., 2024) and DexGarmentLab (Wang et al., 2025b) support large-scale policy learning for garment manipulation with Position-Based Dynamics (Müller et al., 2007) for cloth simulation, but remain tailored to standard fabrics and depend on simulator-specific contact tuning rather than physically measurable parameters. Concurrent VBD-based (Chen et al., 2024) solvers including SIM1 further improve fidelity for

robot–deformable interaction; remaining failure modes (cloth–rigid penetration, grasp instability, replay non-determinism) are characterized in §3.

2.2 Sim-to-Real for Deformable Manipulation

Deformable sim-to-real has split along observation modality. Point-cloud methods (Ha & Song, 2022; Weng et al., 2022; Canberk et al., 2023; Xue et al., 2023; Sundaresan et al., 2022) and GarmentLab/DexGarmentLab achieve effective transfer on standard fabrics by leveraging explicit geometry, but become unreliable when depth is noisy or incomplete—dark, reflective, low-texture, or self-occluded materials. RGB end-to-end has historically lagged: VSF (Hoque et al., 2020) reported pure RGB underperforming RGBD by 44% on fabric folding, and the gap has not closed—DexGarmentLab's RGB diffusion policy reaches only a 58% mean zero-shot real success across four garment tasks, and recent RGB diffusion policies on bimanual deformables continue to struggle. Concurrent work SIM1 shows RGB sim-to-real transfer for garment folding, but relies on teleoperated trajectories and post-hoc filtering (head-to-head in §7.3).

Plastic-bag manipulation is largely real-robot-first: AutoBag relies on visual markers (Chen et al., 2023a), ShakingBot studies dynamic shaking on a specific archetype (Gu et al., 2024), DextAIRity uses a multi-arm airflow setup on one bag (Xu et al., 2022), and SOI learns dynamics from real point clouds due to unreliable bag simulation (Zhou et al., 2025). Simulation benchmarks remain simplified—DeformableRavens covers single-topology bags (Seita et al., 2021), SoftMimicGen omits handle structure (Moghani et al., 2026)—because thin-shell deformation, 2-torus handles, self-contact, and large pick-and-place deformation resist standard tuning. Our pipeline handles bags with explicit handles within the same RGB sim-to-real protocol as garment tasks (§3, §7).

2.3 Trajectory Generation for Manipulation

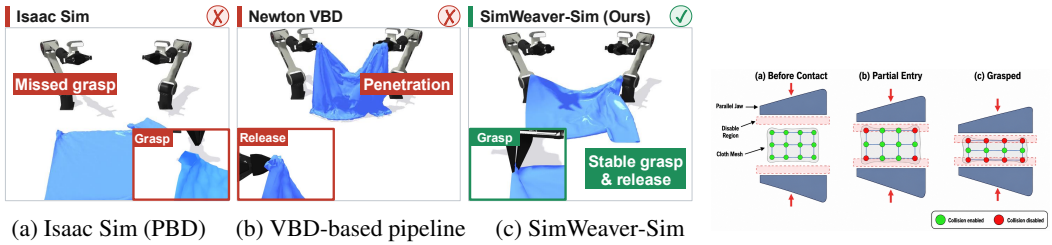
Trajectory generation expands a small set of demonstrations into larger training datasets. Existing methods are most developed for rigid manipulation: MimicGen (Mandlekar et al., 2023), SkillMimicGen (Garrett et al., 2025), and DexMimicGen (Jiang et al., 2025) synthesize demonstrations by transforming source trajectories through object-centric rigid-frame correspondence. This assumption breaks down for deformables, which lack canonical pose frames and undergo large non-rigid (sometimes topological) deformation.

Deformable extensions use non-rigid registration, task-specific correspondence, or learned generation. SoftMimicGen warps trajectories via non-rigid registration but stays at 17% on bimanual towels even at 750 generated demos, partly because warping enforces no IK feasibility or bimanual consistency. Garment-specific methods exploit canonical structure—DexGarmentLab/HALO via sleeve/collar correspondence, FoldNet's KG-DAgger (Chen et al., 2026) via keypoint-driven recovery—but neither transfers to non-garment deformables such as bags. Concurrent SIM1 pairs diffusion-based generation with post-hoc filtering at the cost of teleoperation seeds.

These methods follow a generate-then-filter paradigm. In contrast, our generator is deterministic and topology-aware, producing feasible high-quality trajectories without learned generative models, teleoperation seeds, or post-hoc rejection (§4, §7).

3 SimWeaver-Sim: Reliable Deformable Simulation

Leveraging large-scale low-cost simulation data is promising for robot policy learning, especially for VLA training. However, the sim-to-real gap degrades real-world performance of policies trained in simulation. This challenge is particularly severe in deformable object manipulation, where rigid-flexible interactions among deformable objects, robots, and the environment introduce substantial physical complexity. Most methods therefore combine simulation pretraining with limited real-world adaptation. Since sim-to-real performance fundamentally depends on simulator fidelity, improving simulator realism can substantially enhance zero-shot real-world performance.



(a) Isaac Sim (PBD) (b) VBD-based pipeline (ours) (c) SimWeaver-Sim (d) Active collision region

Figure 2: **Simulator failure modes (a–c) and SimWeaver-Sim's active-collision-region scheme (d).** (a) Isaac Sim with PBD cloth exhibits instability and fails to grasp the garment. (b) Newton VBD produces cloth–rigid penetration during dual-arm manipulation. (c) SimWeaver-Sim achieves stable, penetration-free behavior under identical trajectories. (d) Green: the region where self-intersection untangling remains active. Adjacent exclusion bands suppress only the local untangling correction; rigid–cloth contact remains enabled.

3.1 Base Simulator

We select the simulation backbone using three requirements: **Fabric realism:** stable simulation across soft-to-stiff and thin-to-thick materials; **Multilayer contact stability:** robust contact among fabric layers and multiple deformable objects; **Rigid-flexible contact stability:** reliable robot–cloth interaction during grasping and compression.

We compare NVIDIA Isaac Sim's Position-Based Dynamics (PBD), a VBD-based simulator, and RGBench against these requirements. We adopt RGBench (Hu et al., 2026) as SimWeaver-Sim's base solver and adapt its contact handling for closed-loop robot manipulation (§3.2). Our contribution addresses penetration, grasp-contact jitter, and replay reliability rather than introducing a new solver.

Figure 2 illustrates (a) grasp instability in Isaac Sim and (b) robot–cloth penetration in Newton VBD. In contrast, (c) SimWeaver-Sim maintains stable multilayer contact and grasping under the evaluated trajectories, using the active-collision-region scheme described next. Quantitative reliability and efficiency results are reported in §7.2.

3.2 Optimized Robot-Deformable Interaction

Reliable robot–deformable contact is essential for sim-to-real transfer: penetration and unstable grasps can make simulated trajectories infeasible on real hardware.

Position-controlled jaws can compress multiple cloth layers into a gap that leaves insufficient space for self-intersection repair. In this regime, the base solver's post-detection untangling corrections can conflict with jaw-contact constraints, causing jitter, nonphysical attachment, or cloth–jaw penetration.

To stabilize compressed grasps (Figure 2(d)), we skip only the post-detection untangling of cloth self-intersections involving vertices in fixed exclusion bands next to the inner jaw surfaces, within the local inter-jaw volume. Rigid–cloth contact, friction, elasticity, and the remaining collision handling stay enabled. Let $d_g(\tau)$ be the jaw gap and d_f the exclusion depth. The central active untangling region has width

$$d_a(\tau) = \max\{0, d_g(\tau) - 2d_f\}.$$

For an inter-jaw vertex at minimum inward normal distance d_v from either inner jaw surface, untangling is skipped when $d_v \leq d_f$ and retained otherwise. We use a fixed reference scale $h_{\text{ref}} = 2\,\text{mm}$, $d_f = h_{\text{ref}}/2$, and a closing gap $d_g^{\text{close}} = h_{\text{ref}}$ across all evaluated tasks; this reference scale is distinct from the fabric's physical thickness. Thus $d_a = 0$ at full closure. As the jaws reopen, the active region expands and vertices leaving the exclusion bands resume standard untangling. This rule improves task-level contact reliability without establishing exact inter-layer separation or contact-force accuracy inside the excluded bands.

3.3 Quantitative Reliability Comparison

We evaluate contact reliability and physical realism in thin cloth–rigid interaction, cloth grasping, and garment folding. In a multilayer-grasp ablation with 50 trials per condition, the active-collision-region scheme reduced observed cloth–jaw penetration from 50/50 to 0/50 under identical trajectories.

For each setting, we report quantitative metrics over n trials per condition in Section 7.2. SimWeaver-Sim achieves significantly lower penetration rates and higher trajectory success rates than both NVIDIA Isaac Sim cloth simulation and a VBD-based baseline, while maintaining competitive simulation efficiency. These reliability improvements directly enable the downstream sim-to-real performance reported in Section 7.

4 SimWeaver-Syn: Topology-Aware Trajectory Synthesis

SimWeaver-Syn synthesizes training trajectories without teleoperated demonstrations. Existing learned samplers such as SIM1 still rely on human teleoperation to seed generative models, degrade under shifts in the cloth's initial pose, and report pass rates below 50% on bimanual cloth tasks. Our pipeline removes this dependency through three deterministic components: **topology-adjacency selection** (TAS) defines semantically valid bimanual grasp pairs from the asset's canonical mesh; **topology-aware state predicates** determine which pairs are feasible under the current observation; and **closed-loop synthesis** verifies each grasp attempt and reselects when execution fails. A constrained planner then realizes the selected pairs as smooth, executable trajectories (Appendix F); diversity comes independently from randomized simulator initializations.

Topology-adjacency selection (TAS). Across deformable objects, manipulation semantics are determined not by keypoints alone but by their topological relations: the same pair of corners may unfold or fold a cloth depending on which two are grasped. We represent feasible bimanual configurations as a labeled graph $G = (V, E)$ on the canonical mesh, where nodes V are semantic landmarks (cloth corners, T-shirt neck/hem/shoulder/sleeve points, bag strap tips) and edges $(u, v) \in E$ are task-feasible grasp pairs. G is extracted automatically for symmetric assets and labeled once per asset class otherwise (Appendix E). Given an observation obs_t, we select

$$(l^\star, r^\star) \;=\; \arg \max_{(u,v) \,\in\, E \,\cap\, \mathcal{F}(\text{obs}_t)} S(u,\, v\,;\, \text{obs}_t), \tag{1}$$

where $\mathcal{F}(\text{obs}_t)$ is the geometric feasibility set on the current cloth state and S scores the surviving candidates. The adjacency graph G does most of the work: by restricting the argmax to E, all non-adjacent pairs are excluded by construction and never enter the score. S is then a lightweight, task-specific selector among feasible adjacency edges. Unlike continuous-registration methods that match deformation fields point by point (e.g., SoftMimicGen), our discrete graph formulation applies uniformly across cloth, garment, and bag topologies; only G changes across assets.

Topology-aware state predicates. A topologically valid grasp pair may still be infeasible in the current state because deformable objects exhibit severe self-occlusion, folding, and multi-layer stacking. Selection in Eq. 1 and the verifier in the synthesis loop therefore share a set of closed-form predicates on the current cloth observation. **Occlusion-above** flags a candidate u when a non-geodesic neighbor lies in the cylinder above it,

$$\exists\, v \in V \setminus \mathcal{G}_\delta(u) : \quad \|\mathbf{p}_v^{xy} - \mathbf{p}_u^{xy}\| < r, \;\; z_v - z_u \in [\delta_z,\, h], \tag{2}$$

where $\mathcal{G}_\delta(u)$ is the δ-radius geodesic neighborhood of u on the canonical mesh; masking $\mathcal{G}_\delta(u)$ separates self-curvature from true occluding overlap. **Layer separation** flags u when the canonical-coordinate ℓ_∞ spread of its physical XY-neighborhood exceeds a threshold, remaining reliable on thin fabrics where simple z-difference heuristics fail. Additional predicates enforce reachability, anti-cross-arm safety, and surface exposure on the current 2-D convex hull, augmenting the candidate set when canonical landmarks fold inward. These predicates gate selection, terminate synthesis upon success, and define task success during evaluation; being closed-form functions of the observation, they keep the loop deterministic and label-noise-free.

Closed-loop synthesis. Grasp failure is among the most common failure modes for deformables, particularly under occlusion and self-contact, so trajectory synthesis cannot be a single-shot selection.

Each grasp attempt in SimWeaver-Syn couples topology-aware selection with verification: at every iteration, the system re-observes the cloth, updates the visible vertex set under the occlusion mask, re-evaluates Eq. 1 under the latest feasibility constraints, executes the grasp while anchoring already-verified arms, and retries up to T attempts (full pseudocode in Appendix D).

5 SIMWEAVER-REAL: SIM-TO-REAL PROTOCOL

Beyond rendering realism and standard domain randomization, we identify two deformable-specific sim-to-real gaps and address each at the appropriate layer: the cloth *state distribution* that rigid-body pose randomization cannot cover (§5), and the per-unit ISP noise of consumer RGBD cameras (§5). We further apply conventional DR over robot and scene axes (§5). Demonstration synthesis (§4) and in-simulation policy evaluation share the same environment, so sim-to-sim closed-loop checks are available throughout development; the generalization claims in this paper are evaluated against real-robot deployment in §7.4.

Physics-driven cloth state randomization. Deformable manipulation involves a high-dimensional state space of folds, wrinkles, and self-contact that simple pose randomization cannot capture. We therefore initialize cloth states using task-conditioned physics primitives (e.g., pin-and-release or random-fold-and-settle), allowing the simulator to generate physically plausible configurations through contact, gravity, and bending dynamics.

Sensor-distribution-aware photometric augmentation. Real-world deployment uses three RealSense D435i cameras (one overhead, two wrist-mounted). We characterize their per-unit ISP behavior and identify two sensor-internal failure modes—color bias across cameras and gain-loop channel drift within a single camera—that conventional pipelines often absorb into generic environmental randomization. We instead apply photometric augmentation with ranges fitted to the measured per-camera statistics; disabling it collapses real-world success to 0% on all five tasks, while enabling it restores the rates reported in §7.4. Procedure and ranges are in Appendix B.

Robot and scene domain randomization. Conventional DR covers per-joint robot home-pose noise, table height, table texture, and lighting. Deformable object physical parameters, such as bending stiffness and friction, are matched to direct measurements of the corresponding real fabrics without task-specific tuning, enabled by SimWeaver-Sim's physically meaningful parameterization (§3). All axes are sampled per episode via a scrambled Sobol quasi-random sequence to improve coverage under the 200-episode budget; full ranges in Appendix A.

6 SIMWEAVER-ASSET: EXTENSIBLE ASSET FRAMEWORK

SimWeaver-Asset is an extensible deformable-asset framework for sim-to-real research. It bundles about $2{,}000$ garment assets sourced from Style3D's GarmageSet (Li et al., 2025; Style3D, n.d.), together with additional asset categories underrepresented in prior deformable datasets, most notably bags with deformable handles. We integrate these assets into our simulation and trajectory-generation pipeline through mesh preprocessing and task-topology metadata. All assets share a unified triangle-mesh representation, enabling interchange across solvers and renderers.

A key feature of SimWeaver-Asset is its measurement-backed physical parameterization. Unlike PBD-based simulators whose parameters, such as inter-particle distance, often lack direct real-world correspondence, SimWeaver-Asset represents material properties as measurable physical quantities (mass per unit area, bending stiffness, stretch, friction). The physical-property values used in this paper are drawn from the large-scale fabric measurement library released by RGBench, which catalogs real fabrics characterized under standard textile-testing protocols (e.g., ASTM D1388 cantilever bending (ASTM International, 2018), ASTM D3107 stretch/extensibility (ASTM International, 2019)). Each asset is bound to a fabric class entry from this library, so sim-to-real experiments use physically grounded settings rather than manually tuned simulator proxies.

To further scale asset collection, SimWeaver-Asset integrates a single-image asset-generation pipeline based on EmbodiedGen (Wang et al., 2025a). Unlike standard 3D generation pipelines that mainly recover geometry and appearance, our pipeline additionally assigns deformable physical properties

to generated assets by sampling from the same RGBench-derived property library according to the inferred fabric category, optionally predicted by a VLM-based classifier. As a result, generated assets can be directly instantiated in SimWeaver-Sim with measurement-backed dynamics, avoiding generic defaults or manual tuning. Imported meshes and generated assets share the same physical-parameter interface, so the collection grows without retuning the simulator; release details are in Appendix K.

7 EXPERIMENTS

7.1 SETUP

Our bimanual setup—two Piper 6-DOF arms with parallel-jaw grippers, one overhead and two wrist-mounted RealSense D435i cameras—is described in detail in §5. We train $\pi_{0.5}$ (Physical Intelligence et al., 2025) with full fine-tuning on the 200 demonstrations generated by SimWeaver-Syn per task. A DP3 (Ze et al., 2024) point-cloud baseline trained on the same demonstrations fails on all five real tasks; the failure reproduces across the consumer RGBD sensors we tested, so we attribute it to depth-acquisition limits on matte-black grippers and silk/bag surfaces rather than to point-cloud methods in general (see §5 and Appendix G).

7.2 SIMULATOR RELIABILITY AND EFFICIENCY

We benchmark SimWeaver-Sim against two cloth simulators previously used for deformable data generation: Isaac Sim with PhysX particle cloth (parameters inherited from DexGarmentLab) and the Newton implementation of VBD, each under its officially recommended parameters. All three systems import the same garment asset and execute the same predefined dual-arm grasp-and-lift; per-step time is averaged over 200 physics steps after a 100-step warm-up at a $1/500\,\mathrm{s}$ timestep on the same workstation equipped with an NVIDIA GeForce RTX 5070 Ti GPU. Figure 2 shows representative failure frames.

Table 1 compares simulator reliability on repeated bimanual garment grasping. Isaac Sim fails to establish stable grasps, while Newton VBD frequently suffers from physically invalid contact failures such as penetration and explosion. In contrast, SimWeaver-Sim achieves stable and efficient simulation without these failure modes, resulting in substantially higher usable demonstration yield for large-scale data generation.

Table 1: **Reliability comparison on repeated bimanual garment grasping**. Penetration / explosion are physically invalid contact failures; per-step time is measured with a fixed asset and timestep on the same NVIDIA GeForce RTX 5070 Ti GPU.

Simulator	Task success ↑	Grasp success ↑	Penetration ↓	Explosion ↓	Per-step time ↓
Isaac Sim (PhysX particle)	0.0%	0.0%	0.0%	0.0%	7.80 ms
Newton VBD	0.0%	100.0%	77.5%	22.5%	10.38 ms
Ours	**100.0%**	**100.0%**	**0.0%**	**0.0%**	**4.44 ms**

7.3 TRAJECTORY SYNTHESIS QUALITY AND REPLAY STABILITY

We assess SimWeaver-Syn on three axes: **Pass** (synthesis success rate over n trajectories); **Replay** (a successful trajectory replayed $100\times$ from fresh simulator resets, isolating contact determinism); and two reported failure categories (**Lift fail / Stall fail**) defined in Appendix H. We independently ablate two structural components: **w/o TAS** replaces E in Eq. (1) with $V \times V$ (uniform sampling over visible-and-reachable bimanual pairs); **w/o closed-loop** restricts Algorithm 1 to a single attempt ($T = 0$). For cross-method comparison we run SIM1's official released train–sample–filter pipeline on its T-shirt fold task under identical physics.

Ablation (flatten). Independently removing closed-loop iteration or TAS reduces pass rate from 89.7% to 65.0% or 56.3%, respectively. Most failures are lift failures, indicating that both closed-loop retry and topology-aware pair selection are critical for reliable trajectory synthesis.

Cross-method comparison (fold). Under matched $n = 100$, SimWeaver-Syn substantially outperforms SIM1 in T-shirt folding. SIM1 suffers from both generator-side and simulator-side failures: the

Table 2: **Trajectory synthesis: pass rate, replay stability, and selected failure categories.** *Top*: ablation of SimWeaver-Syn's two structural components on T-shirt flatten ($n = 300$, same simulator and task). *Bottom*: cross-method comparison on T-shirt fold ($n = 100$, matched physics).

Method	Task	n	Pass (%)	Replay 100×	Lift fail (%)	Stall fail (%)
SimWeaver-Syn	T-shirt flatten	300	**89.7**	**100/100**	10.3	0
w/o closed-loop	T-shirt flatten	300	65.0	—	35.0	0
w/o TAS	T-shirt flatten	300	56.3	—	43.7	0
SimWeaver-Syn	T-shirt fold	100	**97.2**	**100/100**	2.8	0
SIM1	T-shirt fold	100	24.0	13/100	43.0	32.0

learned trajectory sampler introduces unstable grasps and large trajectory noise, while the contact solver frequently fails under bimanual cloth interactions, leading to lift and stall failures respectively.

Replay isolates the simulator-side cause. To separate the two causes, we replay one successful trajectory $100\times$ from fresh simulator resets on each system. SimWeaver-Syn achieves **100/100** replay success, while SIM1 reaches only **13/100**. Since the trajectory is fixed across replays, the gap isolates simulator-side contact-resolution determinism, ruling out generator noise.

7.4 Main Sim-to-Real Results Across Five Tasks

The five tasks are: **snack packaging** (insert a snack into a handled non-woven fabric bag, then lift the bag by its deformable handles), **garment folding** (T-shirt fold), **garment unfolding** (T-shirt flatten from a wrinkled start), **silk unfolding** (silk flatten from arbitrary draping), and **silk grasping** (bimanual thin-silk grasp). They span bag–object interaction, structured garment manipulation, and

Table 3: **Main sim-to-real result** ($n = 23$ consecutive real-robot trials per task; 200 sim demos per task, zero-shot bimanual deployment). Per-task and pooled Wilson 95% CIs are in Appendix Tab. 10.

	Snack packaging	Garment folding	Garment unfolding	Silk unfolding	Silk grasping	Average
Real success (%)	86.96	91.30	82.61	95.65	100.00	**91.30**

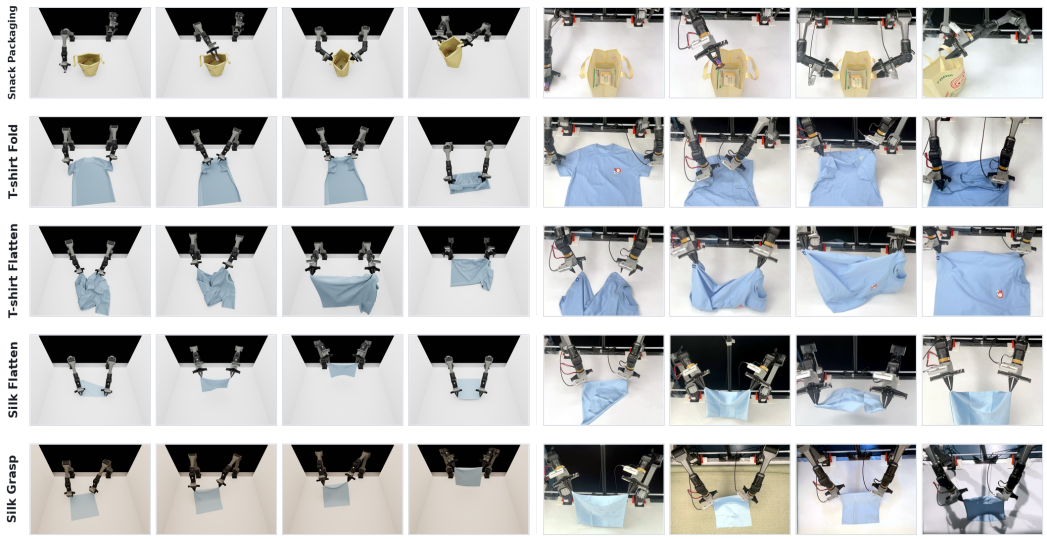


Figure 3: **Sim-to-real qualitative comparison across five deformable-manipulation tasks.** Within each row, the simulated rollout (left) and the zero-shot real-robot rollout (right) share the same checkpoint; one policy is trained per task on 200 sim demos, so checkpoints differ across rows.

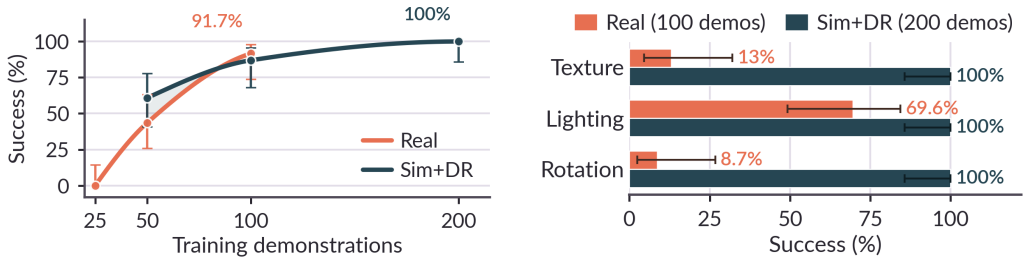


Figure 4: **Silk grasping: real vs. sim. (a)** In-distribution scaling. **(b)** OOD texture/lighting/rotation.

thin-fabric handling under large deformation. Snack packaging extends the suite from garment and silk manipulation to handled bags with deformable grasp points.

Real-world success ranges from 82.61% (garment unfolding) to 100% (silk grasping), with a mean of $\mathbf{91.30}\%$ over $n = 23$ consecutive trials per task (Wilson 95% CIs in Appendix Tab. 10). The two lowest cases—snack packaging (86.96%) and garment unfolding (82.61%)—each require recovery from wrinkled or arbitrarily-draped initial states, and their CIs overlap the higher-scoring tasks. The same 200-demonstration budget transfers stably across all three task families. Additional setup details are provided in Appendix H and Appendix A.

7.5 Sample Efficiency and Generalization on Silk Grasping

Silk grasping is challenging due to weak texture, specular reflection, and sensitivity to grasp errors. We compare real-data policies trained on $25/50/100$ demonstrations with sim-data policies trained on $50/100/200$ demonstrations. Both pipelines share the same photometric augmentation; sim policies additionally receive scene-randomization axes (texture, lighting) unavailable to real-data collection (matched-augmentation discussion in Appendix H). Evaluation uses $n = 23$ real-robot trials per cell under in-distribution scaling and OOD shifts in texture, lighting, and rotation.

Sample efficiency (in-distribution scaling, panel a). The sim-trained policy outperforms real-data training at small data scales and reaches 100% at $n = 200$; we read panel (a) as a scaling *trend* rather than a per-budget significance test (Wilson 95% CIs in Appendix H). The trend suggests that consistent synthesized trajectories are more sample-efficient than noisy teleoperated demonstrations, whose operator-specific variations require larger datasets to average out.

Generalization under distribution shift (panel b). On texture, lighting, and rotation shifts the sim-trained policy (200 demos + DR) scores 100% uniformly versus 13.0%, 69.6%, and 8.7% for real (100 demos)—all gaps significant under single-tail Fisher ($\alpha = 0.05$). The lighting gain is partly a side effect of the photometric augmentation shared by both pipelines; texture and rotation gaps come from simulator-only DR axes (full CIs in Appendix H).

Cost efficiency. Reporting per-*usable* rather than per-raw trajectory matters here: fast solvers that emit unstable rollouts incur post-hoc filtering whose discards inflate the true unit cost of training-ready data. Under matched server-cost assumptions ($8\times$RTX 4090), SimWeaver-Syn reaches **2824** trajectories/day at **$0.03** each—$4\times$ SIM1's throughput at $3\times$ lower unit cost, **two orders of magnitude cheaper** than real-robot collection (Appendix I).

8 Conclusion

We presented SimWeaver, which trains zero-shot RGB VLA policies on 200 sim demos per task to reach an average of 91% real-world success ($>80\%$ per task) across five deformable tasks—including handled non-woven bag manipulation—without teleoperation, by combining a measurement-backed simulator and asset framework (SimWeaver-Sim, SimWeaver-Asset), a deterministic topology-aware synthesizer (SimWeaver-Syn), and an ISP-aware sim-to-real protocol (SimWeaver-Real). On silk grasping, the sim-trained policy matches real-data training in-distribution and exceeds it under distribution shift, at two orders of magnitude lower cost (limitations in Appendix K).

AI USE STATEMENT

We used generative AI tools (large language models) as collaborative coding assistants when writing and debugging parts of our software, for translation between Chinese and English during drafting, to edit the paper for readability and concision, to format and verify bibliography entries against their primary sources, and to assist with LaTeX formatting. We did not use generative AI tools to formulate hypotheses, design the methodology or experiments, generate or process any data, or interpret the results; all research ideas, methods, experiments, and data are the authors' own. We have reviewed all AI-assisted content and take full responsibility for the content of this paper, including all text, claims, and code.

REPRODUCIBILITY STATEMENT

We document every component needed to reproduce our results. The contact-reliable simulator and its active-collision-region scheme are described in §3; the trajectory synthesizer is given as pseudocode in Appendix D, with per-asset topology graphs in Appendix E and the planner/IK backends in Appendix F. Per-task domain-randomization ranges and initialization protocols are listed in Appendix A, the photometric augmentation pipeline in Appendix B, the real-robot hardware and deployment protocol in Appendix C, and policy training hyperparameters in Appendix J. Evaluation protocols, trial counts, and Wilson confidence intervals are reported in §7 and Appendix H, and cost accounting in Appendix I. Fabric physical parameters follow published textile-testing protocols (ASTM D1388/D3107). We will release the trajectory-synthesis, asset-generation, and sim-to-real training code, a representative asset subset, and a usable version of the simulator (Appendix K).

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025. URL https://arxiv.org/abs/2503.06669.

ASTM International. Standard Test Method for Stiffness of Fabrics. Standard ASTM D1388-18, ASTM International, West Conshohocken, PA, USA, 2018. URL https://store.astm.org/d1388-18.html.

ASTM International. Standard Test Methods for Stretch Properties of Fabrics Woven from Stretch Yarns. Standard ASTM D3107-07(2019), ASTM International, West Conshohocken, PA, USA, 2019. URL https://store.astm.org/d3107-07r19.html.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

Alper Canberk, Cheng Chi, Huy Ha, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Cloth Funnels: Canonicalized-alignment for multi-purpose garment manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5872–5879, 2023. doi: 10.1109/ICRA48891.2023.10161546.

Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. Vertex block descent. *ACM Transactions on Graphics (TOG)*, 43(4):116:1–116:16, 2024. doi: 10.1145/3658179.

Lawrence Yunliang Chen, Baiyu Shi, Daniel Seita, Richard Cheng, Thomas Kollar, David Held, and Ken Goldberg. AutoBag: Learning to open plastic bags and insert objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3918–3925, 2023a. doi: 10.1109/ICRA48891.2023.10161402.

Siwei Chen, Yiqing Xu, Cunjun Yu, Linfeng Li, Xiao Ma, Zhongwen Xu, and David Hsu. DaXBench: Benchmarking deformable object manipulation with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://openreview.net/forum?id=1NAzMofMnWl.

Yuxing Chen, Bowen Xiao, and He Wang. FoldNet: Learning generalizable closed-loop policy for garment folding via keypoint-driven asset and demonstration synthesis. *IEEE Robotics and Automation Letters*, 11(3):3820–3827, 2026. doi: 10.1109/LRA.2026.3656770.

Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. SkillMimicGen: Automated demonstration generation for efficient skill learning and deployment. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2750–2790. PMLR, 2025. URL https://proceedings.mlr.press/v270/garrett25a.html.

Ningquan Gu, Zhizhong Zhang, Ruhan He, and Lianqing Yu. ShakingBot: Dynamic manipulation for bagging. *Robotica*, 42(3):775–791, 2024. doi: 10.1017/S0263574723001753.

Huy Ha and Shuran Song. FlingBot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 24–33. PMLR, 2022. URL https://proceedings.mlr.press/v164/ha22a.html.

Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for multi-step, multi-task fabric manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020. doi: 10.15607/RSS.2020.XVI.034.

Wenkang Hu, Xincheng Tang, Yanzhi E, Yitong Li, Zhengjie Shu, Wei Li, Huamin Wang, and Ruigang Yang. Real garment benchmark (RGBench): A comprehensive benchmark for robotic garment manipulation featuring a high-fidelity scalable simulator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18306–18314, 2026. doi: 10.1609/aaai.v40i22.38894.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2104.03311.

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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930, 2025. doi: 10.1109/ICRA55743.2025.11127809.

Siran Li, Ruiyang Liu, Chen Liu, Zhendong Wang, Gaofeng He, Yong-Lu Li, Xiaogang Jin, and Huamin Wang. GarmageNet: A multimodal generative framework for sewing pattern design and generic garment modeling. *ACM Transactions on Graphics*, 44(6), December 2025. doi: 10.1145/3763271. URL https://doi.org/10.1145/3763271.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. DiffCloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 42(1), 2022. doi: 10.1145/3527660.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=pISLZG7ktL.

Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 432–448. PMLR, 2021. URL https://proceedings.mlr.press/v155/lin21a.html.

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. GarmentLab: A unified simulation and benchmark for garment manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://garmentlab.github.io/.

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. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Masoud Moghani, Mahdi Azizian, Animesh Garg, Yuke Zhu, Sean Huver, and Ajay Mandlekar. SoftMimicGen: A data generation system for scalable robot learning in deformable object manipulation. *arXiv preprint arXiv:2603.25725*, 2026. URL https://arxiv.org/abs/2603.25725.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660, 2025. doi: 10.1109/CVPR52734.2025.02575.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. doi: 10.1016/j.jvcir.2007.01.005.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4568–4575, 2021. doi: 10.1109/ICRA48506.2021.9561391.

Style3D. GarmageSet. Hugging Face dataset, n.d. URL https://huggingface.co/datasets/Style3D/GarmageSet. Accessed: 2026-09-26.

Priya Sundaresan, Rika Antonova, and Jeannette Bohg. DiffCloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10828–10835, 2022. doi: 10.1109/IROS47612.2022.9981101.

Xinjie Wang, Liu Liu, Yu Cao, Ruiqi Wu, Wenkang Qin, Dehui Wang, Wei Sui, and Zhizhong Su. EmbodiedGen: Towards a generative 3D world engine for embodied intelligence, 2025a. URL https://arxiv.org/abs/2506.10600.

Yuran Wang, Ruihai Wu, Yue Chen, Jiarui Wang, Jiaqi Liang, Ziyu Zhu, Haoran Geng, Jitendra Malik, Pieter Abbeel, and Hao Dong. DexGarmentLab: Dexterous garment manipulation environment with generalizable policy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. URL https://wayrise.github.io/DexGarmentLab/.

Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. FabricFlowNet: Bimanual cloth manipulation with a flow-based policy. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 192–202. PMLR, 2022. URL https://proceedings.mlr.press/v164/weng22a.html.

Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. DextAIRity: Deformable manipulation can be a breeze. In *Proceedings of Robotics: Science and Systems (RSS)*, 2022. URL https://www.roboticsproceedings.org/rss18/p017.html.

Han Xue, Yutong Li, Wenqiang Xu, Huanyu Li, Dongzhe Zheng, and Cewu Lu. UniFolding: Towards sample-efficient, scalable, and generalizable robotic garment folding. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3321–3341. PMLR, 2023. URL https://proceedings.mlr.press/v229/xue23b.html.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems (RSS)*, 2024. URL https://www.roboticsproceedings.org/rss20/p067.html.

Peng Zhou, Pai Zheng, Jiaming Qi, Chengxi Li, Hoi-Yin Lee, Yipeng Pan, Chenguang Yang, David Navarro-Alarcon, and Jia Pan. Bimanual deformable bag manipulation using a structure-of-interest based neural dynamics model. *IEEE/ASME Transactions on Mechatronics*, 30(5):3254–3265, 2025. doi: 10.1109/TMECH.2024.3485471.

Yunsong Zhou, Hangxu Liu, Xuekun Jiang, Xing Shen, Yuanzhen Zhou, Hui Wang, Baole Fang, Yang Tian, Mulin Yu, Qiaojun Yu, Li Ma, Hengjie Li, Hanqing Wang, Jia Zeng, and Jiangmiao Pang. SIM1: Physics-aligned simulator as zero-shot data scaler in deformable worlds. *arXiv preprint arXiv:2604.08544*, 2026. URL https://arxiv.org/abs/2604.08544.

APPENDIX

A DOMAIN RANDOMIZATION CONFIGURATION

Domain randomization is configured per task. The silk-grasping task receives the broadest DR — including lighting, texture, and a table-height axis — because it serves as the centerpiece of the visual-generalization study (§7.5). The remaining four tasks use only cloth-pose, cloth-yaw, init-primitive, and (where indicated) joint-noise randomization at task-appropriate scales.

Axis	Snack pkg.	G. fold	G. unfold	Silk unfold	Silk grasp
Cloth XY offset	± 2 cm	± 5 cm	–	± 5 cm	± 10 cm
Cloth yaw (Z)	$\pm 5^\circ$	$\pm 15^\circ$	–	$\pm 10^\circ$	$\pm 15^\circ$
Init primitive	free-fall	free-fall	2-pt hang-and-drop	random fold	free-fall
Joint-noise scale	$0.5\times$	off	off	$0.5\times$	$0.5\times$
Table height	–	–	–	–	$-1.5 \rightarrow 0$ cm
Light exposure	–	–	–	–	± 0.7 stops
Color temperature	–	–	–	–	4000–6500 K
Dome ambient	–	–	–	–	0–0.15
Table texture	–	–	–	–	10,000 var.†
Sampling	uniform	uniform	uniform	uniform	Sobol

Table 4: Per-task domain randomization ranges. "–" indicates the axis is disabled for that task. Joint-noise scale is relative to a task-conditional preset and applied symmetrically to both arms. Table-height randomization is asymmetric ($-1.5 \rightarrow 0$ cm) because thin-fabric grasping is sensitive to downward calibration error but not to upward offset. †Texture set: 10,000 training variants from the RoboTwin background-texture library (Mu et al., 2025), with 1,000 held-out variants reserved for the out-of-distribution evaluation in §7.5.

Per-task randomization ranges.

Per-task initialization protocol. The *Init primitive* row of Tab. 4 encodes the task-specific cloth-state generator that produces each episode's starting configuration. Free-fall settle is used wherever the task naturally operates from an arbitrary draped state (snack packaging, garment folding, silk grasping). The two flattening tasks instead use a structured generator that already determines the starting configuration's pose family: the T-shirt (garment unfolding) is initialized by pinning two key points and releasing under gravity, producing a wrinkled but recognizable shape that matches how the task is presented downstream; the silk (silk unfolding) is initialized via a random fold of two diagonal corners. Because these generators already place the cloth in a task-appropriate distribution, explicit cloth-XY-offset and yaw randomization are unnecessary on top and the corresponding cells in Tab. 4 read "–".

Sampling. For the silk-grasping task, where covering the joint randomization space at the 200-episode budget is critical, all axes are drawn per episode from a scrambled Sobol quasi-random sequence. The other four tasks use independent uniform sampling. At small data scales we empirically find Sobol's better joint-axis coverage outweighs the extra implementation cost.

Cloth physical parameters. Cloth physical parameters — per-axis stretch and bend stiffness, density, thickness, and gripper static/dynamic friction — are taken from a pre-built, *industrially measured* fabric library: each fabric class is characterized once under standard textile-testing protocols (RGBench), and we assign every asset the measured values of its matching fabric class. The parameters are therefore measurement-backed rather than hand-tuned, yet deployment requires *no per-asset measurement and no per-task physical-parameter calibration*: binding an asset to its fabric class is a one-time library lookup, and the resulting values are neither randomized nor tuned per task. This lookup is feasible only because the simulator's parameters are in direct correspondence with physically meaningful quantities (§3): each denotes a real, measurable property carried by the library entry, rather than an opaque simulator proxy that must be fitted by trial and error.

Table-height-randomization ablation (silk grasping). We isolate the contribution of the table-height axis on silk grasping in simulation, holding all other axes fixed:

Configuration	Simulation success rate
All randomization *except* table height	90%
All randomization *including* table height	97%

B Image Augmentation Pipeline

Empirical sensor characterization. We instrument all three Intel RealSense D435i units in the deployment rig (one overhead, two wrist-mounted) under fixed manual exposure, gain, and white-balance settings, and capture 125 frames per camera across two distinct robot poses (750 frames total). For each frame we compute per-channel mean intensity on white pixels (all channels $> 200/255$). Two systematic failure modes emerge (Fig. 5). First, *per-unit color bias*: identical sensor settings yield camera-dependent color casts spanning $\Delta(R{-}G) \approx 39$ units on a 0–255 scale; two of three cameras are pose-stable to within $\Delta < 0.2$ units across the two poses, while the third drifts. Second, *ISP gain-loop drift*: the third camera's R-channel mean drifts P2P ≈ 29 units within a single 5-second window ($\sigma \approx 12$), with the drift triggered by scene/angle changes and not removable by pre-deployment calibration. Both effects are sensor-internal and require absorbing on the simulation side rather than calibrating out at capture time.

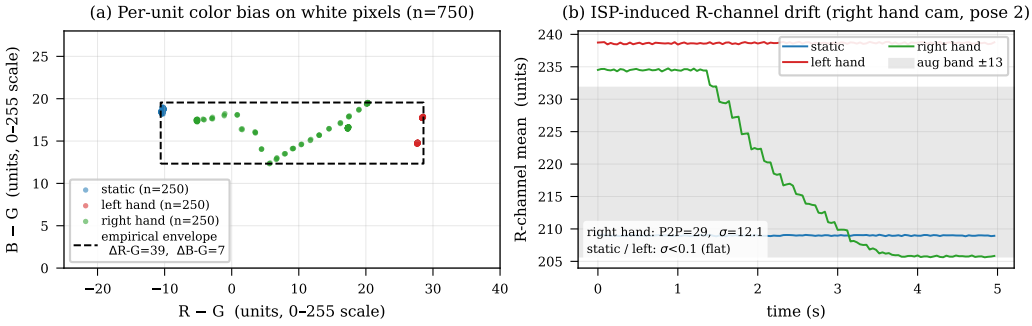


Figure 5: **Empirical D435i ISP characterization grounding the augmentation design.** (a) Per-unit color bias on white pixels across three cameras over two robot poses (250 frames each). The empirical bounding envelope spans $\Delta(R{-}G) = 39$, $\Delta(B{-}G) = 7$ units. (b) Frame-time R-channel drift on the third camera under pose 2: P2P $= 29$, $\sigma = 12.1$, while the other two cameras are flat ($\sigma < 0.1$). Augmentation ranges in our pipeline (hue ± 0.15 and per-channel jitter $\sigma = 0.05 \approx \pm 13$ units) are chosen against these empirical envelopes rather than literature defaults.

A residual sim-real brightness offset of 7–10% (simulator brighter than capture) is also present and is absorbed by the same photometric pipeline.

Training stack. Our policy training is built on the LeRobot stack (Cadene et al., 2024), so that data format, dataloader, training scripts, and augmentation tooling all integrate with the open-source robotics community.

Augmentation parameters. The augmentation pipeline reuses LeRobot's photometric transforms, with the brightness and hue sampling weights elevated and the hue range expanded relative to its defaults to cover the D435i cross-camera color bias.

Augmentation is necessary for transfer. We ablate the pipeline by training otherwise-identical policies with all photometric augmentation disabled. Real-world success rates collapse to 0% uniformly across all five tasks:

Transform	Type	Range	Sampling weight
Brightness	ColorJitter	$[0.75, 1.15]$	3.0
Hue	ColorJitter	$[-0.15, 0.15]$	3.0
Contrast	ColorJitter	$[0.80, 1.20]$	1.0
Saturation	ColorJitter	$[0.50, 1.50]$	1.0
Sharpness	SharpnessJitter	$[0.50, 1.50]$	1.0
Affine	RandomAffine	$\pm 5^\circ$, $\pm 5\%$ trans.	1.0

Table 5: Per frame, a random subset of three transforms is drawn (with weights as sampling probabilities) and applied in random order.

Task	w/ image augmentation	w/o image augmentation
Snack packaging	86.96%	0%
Garment folding	91.30%	0%
Garment unfolding	82.61%	0%
Silk unfolding	95.65%	0%
Silk grasping	100.00%	0%
Average	**91.30%**	**0%**

Table 6: Per-task real-world success rates with and without the photometric augmentation pipeline ($n = 23$ trials per cell). The "with augmentation" column matches the main-result row of Tab. 3; the "without" column is the ablation, in which all five tasks collapse to 0% uniformly. Image augmentation is a prerequisite for sim-to-real transfer, not a marginal improvement.

C Real-Robot Deployment

Render-pipeline latency. RGB observations are produced by one Omniverse Kit frame update per simulation tick (`kit_frame_updates=1`), so each rendered image lags the underlying simulator state by one to two control steps — approximately 40–80 ms at the 25 Hz control rate. This deliberately matches the natural capture-to-output pipeline latency of the D435i RGB stream (30–50 ms in our measurements), avoiding the unrealistically instantaneous-render regime that a default zero-lag rendering loop would impose on the training data. We do not isolate the contribution of this setting with a dedicated ablation; we report it for reproducibility.

D SimWeaver-Syn Pseudocode

Algorithm 1 SimWeaver-Syn: closed-loop topology-aware grasp synthesis.

Require: topology graph G, $\textsc{Observe}(\cdot)$, attempt budget T
for $t = 0, \ldots, T$ **do**
  obs $\leftarrow \textsc{Observe}()$; $(l^\star, r^\star) \leftarrow \textsc{TopoSelect}(G, \text{obs})$ ▷ Eq. (1)
  Execute grasp at $(l^\star, r^\star)$, anchoring already-verified arms at obs.q
  $(l_{\text{ok}}, r_{\text{ok}}) \leftarrow \textsc{Verify}(\textsc{Observe}())$
  if $l_{\text{ok}} \wedge r_{\text{ok}}$ **then return** $\textsc{Execute}(\tau_{\text{post}})$
  end if
end for
return ABORT

E Topology-Graph Instantiations Across Assets

The same selection–verification loop (Algorithm 1) applies across all three asset categories. Only the canonical mesh and the graph $G = (V, E)$ change between assets; predicates, the closed-loop iteration, and the verifier are shared.

Asset class	Landmark set V	Adjacency E
Cloth (rectangular)	4 corners	4 boundary edges; diagonals excluded
Garment (T-shirt)	7 landmarks (neck, $\text{hem}_{l/r}$, $\text{shoulder}_{l/r}$, $\text{sleeve}_{l/r}$)	task-feasible subset (e.g. hem$\leftrightarrow$sleeve)
Bag (with strap)	2 strap-tips (one per handle)	$\text{strap-tip}_l \leftrightarrow \text{strap-tip}_r$

Table 7: Topology graph $G = (V, E)$ instantiated per asset class.

Graph construction across asset classes. The construction of $G = (V, E)$ takes one of two paths depending on the asset's symmetry.

Rectangular cloth. For symmetric assets, G is extracted automatically from the canonical mesh. We identify the boundary edges of the mesh (edges incident to exactly one triangle), traverse them into a closed boundary loop, and select four farthest-from-centroid vertices as corners; midpoints of the four boundary segments between corners serve as edge-center landmarks. The adjacency edge set E contains the four boundary-aligned corner pairs and excludes the two diagonal pairs, since the cloth folds rather than unfolds along the diagonal axis under fling. No human input is required.

Garments and bags. For semantically structured assets, the landmark vertex indices are labeled once per asset class on the canonical mesh during asset preprocessing. T-shirt classes carry $\{\text{neck}, \text{hem}_{l/r}, \text{shoulder}_{l/r}, \text{sleeve}_{l/r}\}$ as the typical V; bag-strap classes carry two strap-tip landmarks (one per handle). The edge set E encodes the task-feasible bimanual pairs for that asset class (e.g., hem $\leftrightarrow$ sleeve for T-shirt sleeve folding; $\text{strap-tip}_l \leftrightarrow \text{strap-tip}_r$ for bag bimanual lift). Once labeled, both V and E are reused across all instances of the asset class and across all rendered episodes, so per-trajectory annotation is never required.

Across the entire training corpus, the human input on G amounts to a one-time labeling of $|V|$ vertex indices per asset class, typically a few minutes per new garment or bag class, and zero for symmetric assets where G is auto-extracted.

Geometric feasibility set $\mathcal{F}(\mathbf{obs}_t)$. The feasibility set in Eq. 1 is the intersection of six closed-form predicates, evaluated on the current observation $\text{obs}_t = \{p_v\}_{v \in V}$.

Reachability, anti-cross-arm, and safety. A candidate vertex must lie within the workspace of at least one arm; a candidate pair must remain on the same side of the inter-base axis so the arms do not cross, and must maintain a physical safety margin:

$$\mathcal{F}_{\text{reach}}(u) \;:\; \min\big(\|p_u - b_l\|,\ \|p_u - b_r\|\big) \;\le\; R_{\text{ws}}, \tag{3}$$

$$\mathcal{F}_{\text{cross}}(u, v) \;:\; \langle p_u - p_v,\ b_l - b_r \rangle \;\ge\; 0, \tag{4}$$

$$\mathcal{F}_{\text{safe}}(u, v) \;:\; \|p_u - p_v\| \;\ge\; d_{\min}. \tag{5}$$

Surface exposure. A candidate must lie near the cloth's current 2-D outline, allowing observation-driven hull vertices to augment V when canonical landmarks fold inward and become unreachable:

$$\mathcal{F}_{\text{surf}}(u) \;:\; \min_{q \in \mathcal{H}_{xy}(\text{obs}_t)} \|p_u^{xy} - q\| \;<\; d_{\text{hull}}, \tag{6}$$

where $\mathcal{H}_{xy}(\text{obs}_t)$ is the set of vertices on the 2-D convex hull of the observed cloth.

Occlusion-above. Restating Eq. 2 as a feasibility predicate, u is feasible only when no non-geodesic neighbor occupies the cylinder above it:

$$\mathcal{F}_{\text{occ}}(u) \;:\; \neg\, \exists\, v \in V \setminus \mathcal{G}_\delta(u) :\ \|p_v^{xy} - p_u^{xy}\| < r,\ \ z_v - z_u \in [\delta_z, h]. \tag{7}$$

Layer separation. The canonical-coordinate ℓ_∞ spread of u's physical XY-neighborhood $\mathcal{N}(u) = \{v : \|p_v^{xy} - p_u^{xy}\| < r\}$ must remain below a threshold:

$$\mathcal{F}_{\text{layer}}(u) \;:\; \max_{i \in \{x,y\}} \Big(\max_{v \in \mathcal{N}(u)} c_i(v) \;-\; \min_{v \in \mathcal{N}(u)} c_i(v) \Big) \;\le\; \tau_{\text{spread}}, \tag{8}$$

where $c(\cdot)$ denotes canonical-mesh position. Vertices physically close in 3-D but topologically far on the canonical mesh indicate multi-layer overlap.

Composition. The full feasibility set is the intersection of these predicates, evaluated at both u and v where applicable:

$$\mathcal{F}(\text{obs}_t) \;=\; \Big\{(u,v) \in V \times V : \mathcal{F}_{\text{cross}}(u,v) \;\wedge\; \mathcal{F}_{\text{safe}}(u,v) \;\wedge \bigwedge_{x \in \{u,v\}} \big(\mathcal{F}_{\text{reach}}(x) \;\wedge\; \mathcal{F}_{\text{surf}}(x) \;\wedge\; \mathcal{F}_{\text{occ}}(x) \;\wedge\; \mathcal{F}_{\text{layer}}(x)\big)\Big\}. \tag{9}$$

F Multi-Backend Planner and Inverse Kinematics

The trajectory synthesizer is decoupled from any single planning or kinematics implementation through a unified `PlannerManager` interface and a `BaseKinematics` abstraction. Each backend is swappable at configuration time, and the synthesis loop never references a backend directly. This decoupling supports two practical needs: hardware-portability, since CPU-only deployments can fall back to analytical IK while GPU-equipped servers leverage batched IK, and faithful cross-platform reproduction of the synthesis pipeline without re-implementing higher-level logic.

Layer	Backend	Implementation	Notes
Motion planning	TOPP-RA	CPU, deterministic	Default; time-optimal parameterization
	cuRobo	GPU, batched	High-throughput data generation
	Min-jerk	CPU, analytical	Reference baseline
	Adaptive	Hybrid	Per-segment backend selection
Inverse kinematics	Pinocchio	Analytical Jacobian, CPU	Default
	cuRobo	GPU, batched	Pairs with cuRobo planner
	RTB	Robotics Toolbox	Cross-validation
	TRAC-IK	Constrained quadratic	Singular-configuration fallback

Table 8: **Planner and IK backends sharing a unified interface.** The synthesizer is invariant to backend choice; selection is configuration-driven. The pipeline used throughout this paper combines TOPP-RA + Pinocchio as the default; cuRobo is enabled when batch IK throughput becomes the bottleneck.

Multi-stage skill composition. Skills with qualitatively different motion phases such as bimanual fling, drag, or contact-critical grasp cannot be expressed under a single fixed-parameter motion plan. The synthesizer therefore exposes per-stage specifications: each stage of a skill carries its own velocity scale, its own per-axis pose-constraint vector (in either world or end-effector frame), and its own goal type (Cartesian pose or joint configuration). A configurable via-point is inserted before each grasp to relax IK reachability without altering the camera viewpoint, and IK solutions propagate as warm-starts across consecutive stages to minimize wrist rotation.

The bimanual fling is the canonical example. It chains a slow lift to the fling start pose, a high-velocity forward swing that injects momentum into the cloth, a mid-low transition that absorbs dynamic forces, a low-z drag with tightly locked gripper orientation that keeps the fabric flat against the table, and a back-amplitude return that resets the configuration. Each stage carries its own velocity scale and constraint vector, specified once at task-config time and consumed by the planner without solver-side intervention; no fixed-parameter pipeline could produce this trajectory uniformly.

Behavioral comparison with human teleoperation. As a post-hoc behavioral check, we compare SimWeaver-Syn-generated trajectories with 100 human-teleoperated episodes of the same task on the same hardware. The comparison is descriptive only: it asks whether synthesized trajectories sit within the same family of motion profiles as humans, not whether they are derived from human data. SimWeaver-Syn does not consume any teleoperation in either training or planning.

We fit five candidate velocity-profile models to the normalized velocity curve of each human episode (Table 9). An asymmetric bell explains human motion best, with peak velocity at $\tau \approx 0.35$ rather than the symmetric $\tau = 0.5$ that min-jerk assumes. The synthesizer's TOPP-RA backend emits symmetric trapezoidal profiles; the median-MSE gap from symmetric trapezoidal to the best-fit asymmetric bell (0.081 vs. 0.007) is roughly an order of magnitude smaller than gaps induced by initial-pose

distribution mismatch and gripper-close timing. The synthesizer's profile family is therefore in-distribution with human motion behavior, with the residual shape difference dominated by other sim–real factors.

Profile model	MSE (median)	Best-fit rate
Asymmetric bell	**0.007**	**97%**
Asymmetric trapezoidal	0.056	1.0%
Symmetric trapezoidal	0.081	0.0%
Gaussian	0.088	1.5%
Min-jerk	0.096	0.5%

Table 9: Profile fit to 100 human-teleoperated episodes; SimWeaver-Syn does not consume teleoperation in any pipeline stage.

G Failure Cases

Point-cloud baseline (DP3) failure mode. We trained 3D Diffusion Policy (DP3) on the same SimWeaver-Syn-generated demonstrations and evaluated on the same real hardware. **DP3 fails across all five tasks.** The dominant cause is fundamental coverage degradation of the point-cloud modality on our scene. Fig. 6 shows a reconstruction captured by our highest-fidelity scanning setup — an upper bound on what any depth-based pipeline can extract on this scene — and even at this quality, large portions of the table surface and the robot arm are missing, the gripper collapses to a handful of sparse points, and the silk fabric returns almost no signal. The real-time D435i depth stream that DP3 actually consumes at deployment is strictly noisier and sparser than this offline scan, so the policy never sees a reliable geometric signature for either the manipulator or the cloth and loses track of both before reaching the grasp.



Figure 6: **Point-cloud coverage failure on our real-world scene (best-case scan).** A high-fidelity offline reconstruction — the upper bound on what any depth-based pipeline can extract on this scene — already exhibits large holes on both the table surface and the robot arm, collapses the gripper to a few isolated points, and recovers virtually no points on the silk fabric. The real-time D435i depth stream consumed by DP3 at deployment is strictly worse, which directly drives the universal failure of the point-cloud baseline across all five tasks.

H EXTENDED EXPERIMENT READOUT

Failure-mode definitions (Tab. 2). **Lift fail** counts trajectories where the gripper closes but does not lift the cloth past the height threshold—the cloth was not actually grasped. **Stall fail** counts trajectories where the cloth is lifted but its dynamic motion stalls during the subsequent fling/fold phase: contact-resolution interference (any of contact-force balancing, self-collision, or cloth–rigid collision) blocks the cloth from following the planned trajectory, leaving it in a wrong terminal configuration. This is a simulator-side breakdown rather than a planning failure.

Simulator metrics and failure interpretation. Section 7.2 reports five metrics. **Task success** measures end-to-end completion. **Grasp success** measures whether the garment is successfully grasped and lifted before the release phase. **Penetration** measures cloth–rigid interpenetration or invalid post-contact attachment that makes the rollout physically implausible. **Explosion** measures catastrophic simulator instability with visually implausible cloth motion. For **per-step time**, we import the same asset into all simulators, fix the physics timestep to $1/500$ s, discard the first 100 warm-up steps, and average the next 200 pure physics steps over repeated runs. All simulator timing runs use the same workstation equipped with an NVIDIA GeForce RTX 5070 Ti GPU.

The detailed readout is consistent with the summary in the main text. Our simulator avoids the two failure modes that are most damaging for downstream data generation: visible cloth–rigid penetration and catastrophic instability. By contrast, the VBD-based baseline can nominally lift the garment but frequently exhibits penetration and explosive post-contact behavior, so apparently correct planned trajectories do not remain executable after replay: 77.5% of trials exhibit penetration, 22.5% end in explosion, and end-to-end task success drops to 0.0%, while its per-step time rises to 10.38 ms. Isaac Sim cloth fails differently. It typically cannot establish or maintain a stable grasp at all, which results in 0.0% grasp success, 0.0% task success, and a slower per-step time of 7.80 ms. At data-collection scale, both patterns produce large numbers of unusable demonstrations.

Replay protocol and trajectory-synthesis metrics. At the trajectory level, we start from a single successful garment-folding trajectory collected in our simulator, then reset the simulator and replay the generated trajectory under the same task definition to test whether the grasp and subsequent execution remain valid. This replay-based protocol directly validates the determinism claims made in Section 4: the question is not whether a trajectory can succeed once, but whether it remains executable across repeated simulator resets.

Table 2 is organized along three axes. **Pass** is the success rate over n independently synthesized trajectories. **Replay** is the success rate of a single successful trajectory replayed $100\times$ in our simulator, isolating simulator-side determinism under contact. **Lift fail / Stall fail** report two mutually exclusive failure modes (defined above)—gripper closes but cloth is not lifted past the height threshold; cloth is lifted but its dynamic motion stalls during the fling/fold phase leaving the wrong terminal configuration—but do not exhaust all failures. We independently ablate the two structural components introduced in Section 4: removing the topology graph (**w/o TAS**) replaces E in Eq. (1) with $V \times V$, sampling bimanual pairs uniformly from the visible-and-reachable subset; removing the closed-loop iteration (**w/o closed-loop**) restricts Algorithm 1 to a single attempt ($T = 0$). We further compare against SIM1's *official released implementation and trajectory sampler* under its default configuration (not a reimplementation), evaluated on its original T-shirt fold task. The $100\times$ replay is conducted in SIM1's own released simulator, so the $13/100$ result reflects that solver's contact determinism under a fixed successful trajectory, independent of our pipeline.

Detailed synthesis readout. On T-shirt flatten, independently removing closed-loop iteration or the topology graph drops SimWeaver-Syn's pass rate from 89.7% to 65.0% or 56.3%, respectively. These correspond to drops of 24.7 and 33.4 pp relative to the full method; each ablation retains the other component. The latter ablation samples bimanual pairs uniformly from the visible-and-reachable subset, so the $1/3$ probability of selecting a diagonal pair—which fails under fling because the cloth folds along the diagonal axis rather than unfolding—accounts for part of the remaining gap. On T-shirt fold under matched conditions with SIM1, SimWeaver-Syn reaches 97.2% versus 24.0%, a 73-point same-task gap.

A single successful trajectory replayed $100\times$ yields $100/100$ for SimWeaver-Syn and only $13/100$ for SIM1. Because replay is independent of both the synthesis method and the task, this 87-point gap

reflects contact-rich simulator determinism alone: trajectories produced by a learned sampler sit at the boundary of stable contact regimes, where small numerical perturbations cascade into task failure, whereas deterministic synthesis stays in the regime's interior.

SimWeaver-Syn's failures concentrate exclusively in the Lift-fail mode—10.3% on flatten and 2.8% on fold—the same mode that dominates SIM1's failures at 43%. The same-mode reduction is $4\times$ on flatten and $15\times$ on fold, which isolates the value of the closed-loop mechanism in the slip-prone grasp regime. SimWeaver-Syn also produces zero Stall-fail on either task versus 32% for SIM1, which is consistent with TAS preventing structurally invalid grasps from cascading into wrong terminal configurations.

Main results: per-task Wilson 95% CIs (Tab. 3). With $n = 23$ trials per task and $N = 115$ pooled trials, the per-task and pooled Wilson 95% confidence intervals corresponding to the main sim-to-real result are:

Task	Success rate (%)	Wilson 95% CI
Snack packaging	86.96	[67.9, 95.5]
Garment folding	91.30	[73.2, 97.6]
Garment unfolding	82.61	[62.9, 93.0]
Silk unfolding	95.65	[79.0, 99.2]
Silk grasping	100.00	[85.7, 100.0]
Average (N=115)	**91.30**	[**84.7**, **95.2**]

Table 10: Per-task and pooled Wilson 95% confidence intervals for the main sim-to-real result (Tab. 3); $n = 23$ consecutive real-robot trials per task.

Silk grasping: matched-augmentation comparison and CIs (§7.5). The OOD comparison isolates the data-source axis (sim vs. real) at matched 2D image augmentation. Both pipelines apply identical photometric augmentation (color jitter, brightness/contrast, photometric ranges fitted to per-camera ISP statistics, §5); the only deliberate asymmetry is that the sim policies additionally receive appearance and lighting randomization that requires programmatic scene control and is therefore unavailable to real-data collection. Simulator-only DR is documented as an additional axis on the sim side rather than absorbed into the matched augmentation.

With $n = 23$ trials per cell the Wilson 95% confidence intervals are wide ($\pm\sim$19 pp at the mid-range 50% probability), so the in-distribution scaling result (panel a) is read as a trend rather than a per-budget significance test. For OOD (panel b), the sim-trained policy (200 demos + DR) scores 100% uniformly with Wilson 95% CI $[85.7, 100.0]$ versus 13.0% ($[4.5, 32.1]$), 69.6% ($[49.1, 84.4]$), and 8.7% ($[2.4, 26.8]$) for real (100 demos) on texture, lighting, and rotation respectively; all gaps remain significant under a single-tail Fisher test ($\alpha = 0.05$).

I Cost Efficiency

Methodology (Table 11). "Single-trajectory time" denotes the wall-clock time to obtain one usable trajectory before dataset serialization; for our pipeline this includes planning, physics execution, and Isaac rendering. "Throughput" denotes usable trajectories collected per day under the reported hardware setup. "Unit cost" denotes amortized cost per usable trajectory. "Relative cost" is normalized to the real-robot collection cost. Our throughput and unit-cost entries are computed from a measured 244.8 s per usable trajectory under the same $8\times$RTX 4090 server-cost assumption used in SIM1. SIM1 and real-robot numbers follow the cost-efficiency analysis reported in SIM1; the real-robot single-trajectory time is derived from 104 trajectories collected over an 8-hour workday.

Beyond transfer performance, we also evaluate whether the same pipeline is efficient enough to serve as a practical data-generation engine. We compare our method against SIM1 and direct real-robot collection using four metrics: single-trajectory time, throughput, unit cost, and relative cost. All quantities are reported per usable trajectory, so the comparison reflects the cost of data that can actually be retained for policy training rather than the cost of raw rollouts with mixed quality.

Table 11: **Per-trajectory collection efficiency.** All quantities are per *usable* trajectory under matched server-cost assumptions ($8\times$RTX 4090).

Method	Single-Trajectory Time (min/traj.) ↓	Throughput (usable traj./day) ↑	Unit Cost ($/usable traj.) ↓	Relative Cost ↓
Real Robot	4.6	104	2.71	1.0×
SIM1	16.2	710	0.10	0.037×
Ours	**4.1**	**2824**	**0.03**	**0.011×**

Table 11 shows that the proposed pipeline is not only effective, but also substantially more scalable than prior alternatives. Under the same server-cost assumption used in SIM1, our measured single-trajectory time is 4.1 minutes per usable trajectory, compared with 16.2 minutes for SIM1 and 4.6 minutes for direct real-robot collection. More importantly, because our trajectories can be generated in parallel with high yield, the system produces 2824 usable trajectories per day, compared with 710 for SIM1 and 104 for direct real-world collection.

The cost advantage is equally clear. Our unit cost is $\$0.03$ per usable trajectory, compared with $\$0.10$ for SIM1 and $\$2.71$ for direct real-robot collection. In normalized form, this corresponds to a relative cost of $0.011\times$ the real-robot baseline, versus $0.037\times$ for SIM1. Under the same accounting rule, our method therefore delivers roughly four times the usable-trajectory throughput of SIM1 while reducing the cost per usable trajectory by more than a factor of three, and remains nearly two orders of magnitude cheaper than collecting the data directly on hardware.

The key point is that the comparison is not about raw simulator speed alone. A simulator can run quickly while still producing unstable or unusable data that must later be discarded. The relevant quantity is therefore the cost of producing trajectories that survive quality control and can enter the final training set. Under that definition, our pipeline combines low single-trajectory latency, high parallel throughput, and high usable-data yield, making it well suited for scalable deformable-data generation in practice.

J Training Details

Policy backbone. We use a $\pi_{0.5}$-class vision–language–action (VLA) policy. The model consumes RGB observations from one fixed-scene camera and two wrist-mounted cameras together with the bimanual joint state, and predicts a chunk of 50 joint-action steps. We perform full-parameter fine-tuning from the publicly released $\pi_{0.5}$ base checkpoint; the visual encoder is not frozen.

Training recipe. Each policy is trained for $30{,}000$ optimizer steps on 200 SimWeaver-Syn-generated demonstrations per task. The effective batch size is 32 (8 per GPU $\times$ 4 GPUs). Optimization uses AdamW with cosine learning-rate decay; the peak learning rate is 2.5×10^{-5} after a $1{,}000$-step linear warmup. We use mixed-precision bfloat16 with gradient checkpointing throughout, and quantile-based normalization on both state and action streams. Hardware is $4\times$NVIDIA RTX 4090 (48 GB each).

Data format. Demonstrations are stored in the LeRobot v3 schema (Parquet for state and action streams, MP4 for visual observations). Physics simulation runs at 500 Hz; observations and actions are sub-sampled to the policy's 25 Hz control cadence, and the recorded action chunk length matches the policy's prediction horizon.

Visual augmentation. The photometric augmentation parameters and the ablation showing that augmentation is required for sim-to-real transfer are documented in Appendix B.

K Limitations and Future Work

Strict comparison. Our most rigorous head-to-head comparison with real-robot training is on silk grasping (§7.5); broader cross-task comparisons require additional real-robot data collection and are left to future work.

Release. We plan to make our pipelines (trajectory synthesis, asset generation, and the sim-to-real training stack) and a representative subset of deformable assets available to the community. A usable version of the SimWeaver-Sim simulator will likewise be made available, while the full industrial-grade collection is available under licensed access, reflecting the cost of acquiring high-quality industrial deformable measurements.

Benchmark. A comprehensive benchmark with standardized task taxonomies, evaluation protocols, and policy leaderboards—built on the simulation infrastructure presented here—is in preparation as a separate contribution.