arXiv:2611.01190v1 [cs.RO] 31 Oct 2026

ClothDojo: Learned Data Generation and Benchmarking for Bimanual Garment Flattening and Folding

Zilong Huang[1,2], Zipeng Ye[2,†], Caicheng Wang[2], Yuchen Xie[2], Jiaxiang Luo[1,*]
[1]South China University of Technology, [2]Meituan
https://clothdojo.github.io/

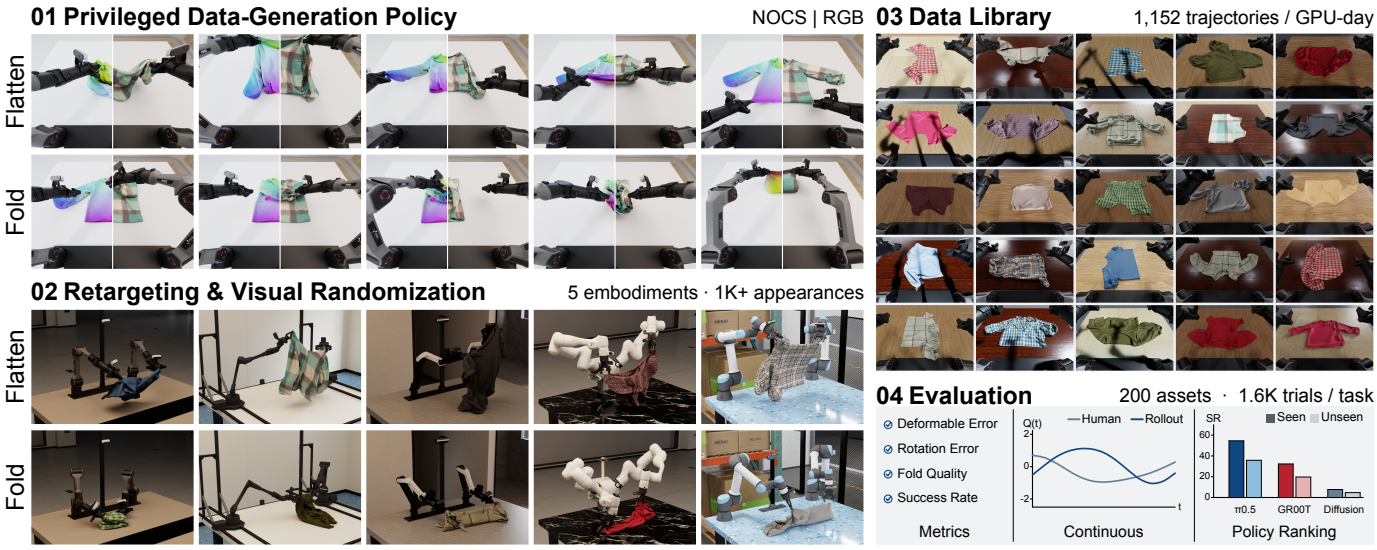


Fig. 1. **ClothDojo connects data generation, data expansion, and closed-loop evaluation.** (01) A data-generation policy with privileged visual observations generates flattening and folding trajectories, shown with normalized object coordinate space (NOCS) and standard RGB appearances side by side. (02) Retargeting and visual randomization expand embodiment and appearance coverage. (03) On a single NVIDIA GeForce RTX 5090 D, the pipeline executes approximately 1,152 trajectories per GPU-day, excluding RGB re-rendering. (04) Evaluation summarizes discrete success criteria, continuous task metrics, and policy rankings.

***Abstract*—Garment flattening and folding require robots to identify garment structure during manipulation and adapt their actions as the cloth deforms. We present ClothDojo, a simulation platform that generates adaptive manipulation trajectories at scale across diverse garments and initial states and evaluates policies in closed loop. ClothDojo encodes each garment's canonical mesh coordinates as an NOCS texture, providing canonical surface-position cues through the visual interface of a pretrained vision-language-action model. We fine-tune the model on human demonstrations and corrections to obtain a data-generation policy. The policy autonomously generates simulated manipulation trajectories, and we retain rollouts that satisfy task-specific success criteria. Each retained trajectory is re-rendered under a randomly sampled appearance to provide RGB observations for downstream policies that do not receive NOCS. Across both tasks, the policy produces nearly 8,000 successful trajectories over hundreds of garments. RGB policies trained on these generated trajectories outperform those trained directly on the original human demonstrations and corrections across all evaluated garment splits and appearance settings. Evaluations across multiple policy models show that overall performance and appearance robustness vary across models and tasks, while generalization to unseen garments remains a common challenge.**

†Project lead. *Corresponding author.

I. Introduction

Garment flattening and folding require visual understanding and bimanual control throughout long sequences of interaction. Garments differ in structure and appearance. Wrinkles and self-occlusion hide relevant parts, and each contact changes the cloth configuration and the conditions for subsequent actions. Policies must continually adjust to the outcomes of their actions. Training across garments therefore requires continuous trajectories covering many initial states. Collecting these trajectories on real robots is time-consuming and requires repeatedly setting up complex configurations that are difficult to reproduce. Differences in robots and scenes further limit the reuse of data and experiments, making training and evaluation at scale costly.

Simulation allows initial states to be reused and rollouts to run in batches, but successful trajectories must still be designed or collected. Task-specific primitives, keypoint planners, and prescribed scripts can execute fixed procedures at scale, but cannot readily capture the state-dependent action transitions in long-horizon garment manipulation. Fig. 2 illustrates such behaviors: flattening involves midair regrasping, dynamic unfolding, and local adjustment, while folding involves hand

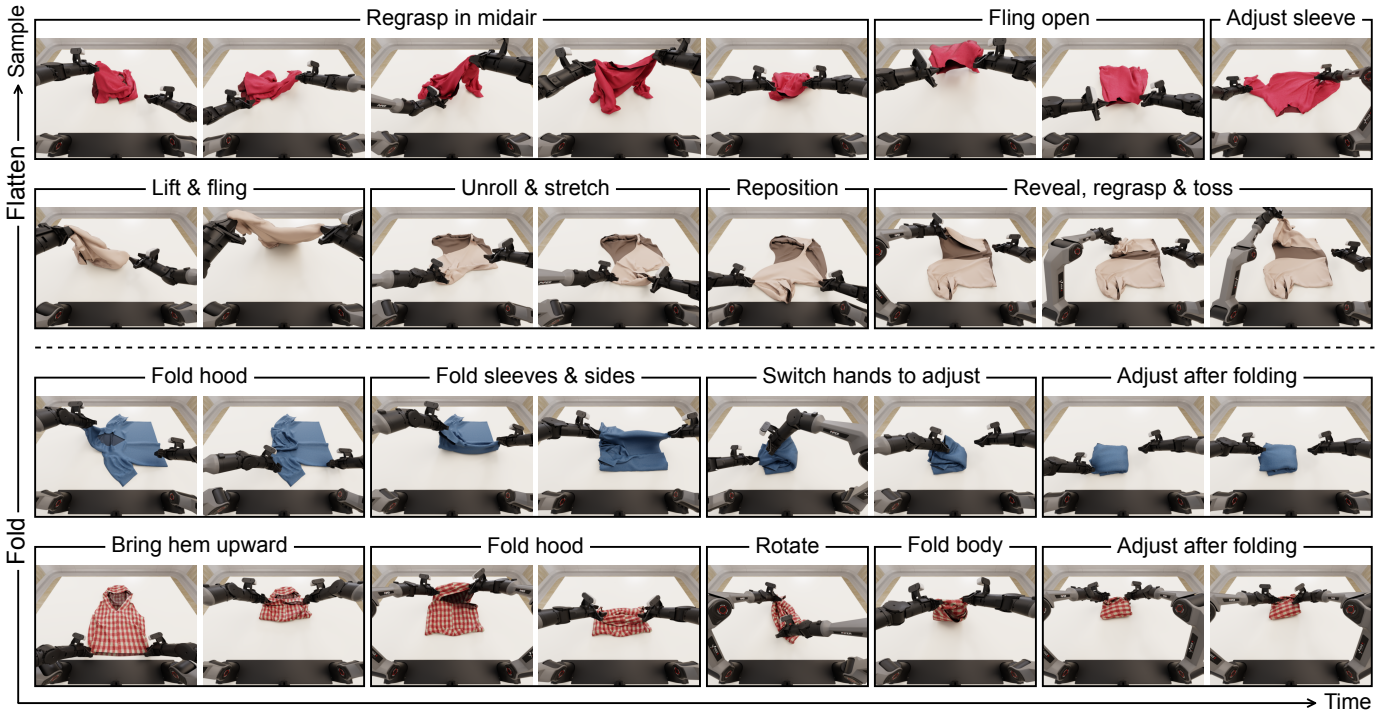


Fig. 2. **Autonomous policies generate long-horizon behaviors with varied action transitions and adjustments during execution.** Flattening trajectories combine midair regrasping, dynamic unfolding, and local adjustments; folding includes hand switching, garment rotation, and post-fold adjustments. Each row shows keyframes from an independent rollout, ordered from left to right with nonuniform time intervals. Box labels describe observed behaviors; the images are re-rendered from recorded simulation states.

switching, rotation, and post-fold refinement. Their timing, arm assignment, and subsequent adjustments depend on the cloth state produced by preceding actions and are difficult to prescribe as a uniform procedure. Human teleoperation naturally produces such trajectories, but its cost scales with the number collected. We therefore study how to extend adaptive garment manipulation to more garments and initial states.

Real-robot laundry systems show that demonstrations, human corrections, and autonomous experience can form an effective data collection and learning cycle [1]–[3]. Inspired by these systems, ClothDojo trains a closed-loop data-generation policy to collect trajectories from new states. We first train the policy on human demonstrations with supervised fine-tuning (SFT), then perform batched rollouts. Task metrics automatically retain successful trajectories. For failed executions, a human selects states and provides human-in-the-loop (HIL) corrections to update the policy. Simulation also allows normalized object coordinate space (NOCS) to be encoded as the garment texture. Through the standard image interface of a vision-language-action model (VLA), these coordinates provide canonical surface-position cues for learning cloth interaction. Successful trajectories are then re-rendered with randomized appearances to train downstream RGB policies without NOCS. Human effort is thus concentrated on demonstrations and necessary corrections, while the data-generation policy scales trajectory collection through autonomous rollouts.

As shown in Fig. 1, ClothDojo connects learned trajectory generation, data expansion, and closed-loop evaluation through dynamic interaction between bimanual robots and cloth. Our contributions are:

1) **A simulation and evaluation platform.** A simulation platform for continuous bimanual garment manipulation, with shared interfaces for human teleoperation, policy rollouts, and closed-loop evaluation.
2) **Learned data generation.** A data-generation pipeline that learns from human demonstrations and corrections, uses NOCS through a VLA's image interface, and autonomously collects trajectories without task-specific primitives.
3) **Trajectory data and training value.** Nearly 8,000 successful trajectories across hundreds of garments. Under the same human supervision budget, RGB policies trained on generated data outperform those trained directly on human data across all evaluated garment and appearance conditions.

II. Related Work

A. Garment manipulation policies

Many methods define a set of task-specific primitives for garment flattening and folding. A policy selects a primitive or predicts parameters such as grasp locations from visual observations, while a predefined controller executes the continuous motion [4]–[8]. The policy can observe the cloth again between primitives, and some primitives also use feedback

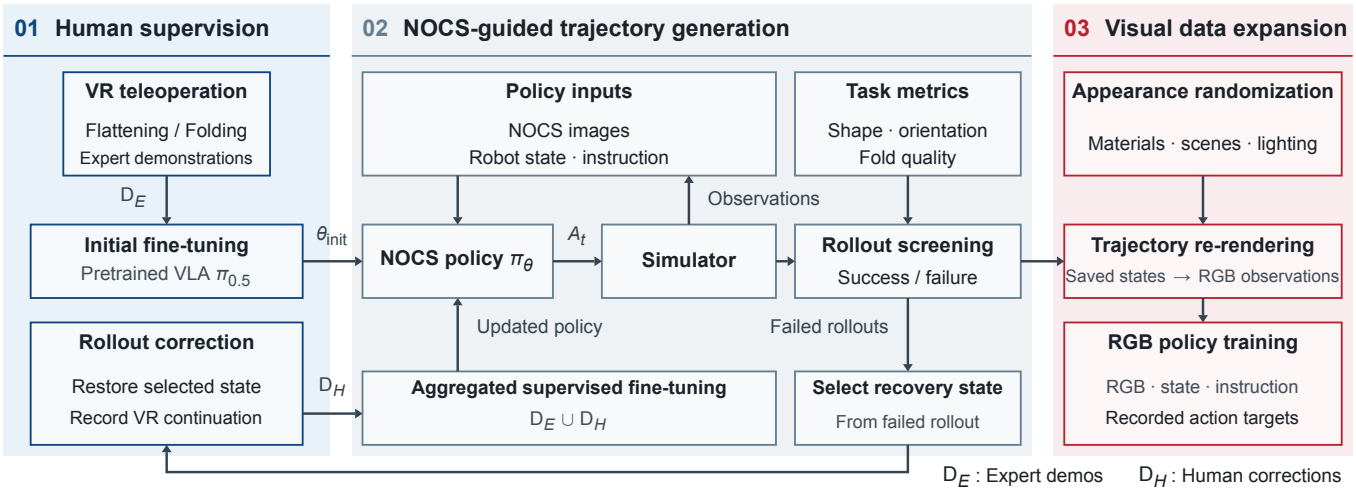


Fig. 3. **Overview of the NOCS-guided data-generation pipeline.** Expert demonstrations $\mathcal{D}_E$ initialize a data-generation policy. For selected failures encountered during autonomous execution, an operator restores a chosen state and provides a correction $\mathcal{D}_H$. The two sources of human data update the policy, while successful rollouts are re-rendered under randomized appearances to train an RGB policy.

during operations such as stretching. These action priors reduce the learning space and have been used for flattening multiple garment types and for structured folding.

Another approach learns continuous actions from data. Recent VLAs directly predict action chunks and perform long sequences of picking up, flattening, folding, and stacking garments on real robots [1]–[3]. These systems establish initial behavior with human demonstrations, expose failures through autonomous rollouts, and use human corrections to cover states encountered by the policy. This learning cycle also lets the policy learn action ordering, bimanual coordination, and adjustments during execution.

B. Simulation environments and data generation

Existing garment simulation platforms provide diverse garment assets and dynamic robot–cloth interaction for tasks including flattening and folding [9]–[12]. Their publicly described trajectories primarily come from task primitives, demonstration transfer, or limited teleoperation.

Automatic trajectory expansion largely follows two approaches. Programmatic methods construct trajectories using keypoints, task stages, or folding patterns [13], [14]. Demonstration transfer adapts segmented behaviors to new assets and initial states, then executes them to obtain new physical trajectories [15], [16]. Existing large-scale results mainly concern folding tasks with well-defined steps. For severely crumpled garments, however, the course of flattening changes with the cloth state, making fixed procedures and segmented demonstrations difficult to reuse directly.

ClothDojo brings the demonstration–rollout–correction cycle used in real-robot garment manipulation to simulation trajectory generation. The data-generation policy learns from human interaction and then runs autonomously on diverse garments and complex initial states; Fig. 2 shows examples of the resulting behavior.

III. Method

ClothDojo separates physical trajectory generation from RGB training-data construction (Fig. 3). We first train NOCS-conditioned data-generation policies from human demonstrations and corrections, then deploy them on a broader set of garments and initial states. Task metrics automatically retain successful rollouts. We sample one appearance for each retained trajectory and replay its states to render RGB observations for downstream policy training.

A. Interaction environment and tasks

ClothDojo combines MuJoCo robot dynamics [17] with Newton's SolverStyle3D cloth solver [18] to support continuous bimanual control. Virtual reality (VR) teleoperation and policy execution use the same physics, so human demonstrations, corrections, and autonomous rollouts share the same robot–cloth interaction.

We convert more than 10,000 garments from the large-scale parametric GarmentCodeData dataset [19] into simulation assets and select 500 structurally diverse garments as a core pool for trajectory generation and evaluation. Human demonstrations and corrections are not restricted to this pool. As shown in Fig. 4, this set spans large variations in garment length, body width, sleeve length, and sleeve width, and includes sleeveless, hooded, low-cut, open-back, asymmetric, and unconventional designs.

Flattening starts from a crumpled state produced by randomly hanging, releasing, and settling a garment. The goal is a naturally flat, face-up configuration. All four cardinal orientations on the table are accepted, and its position within the tabletop is unconstrained. **Folding** starts from a preconstructed state that satisfies the flattening criterion. Depending on data availability, these states come from human-demonstration endpoints, successful rollouts from the data-generation policy, or standard flattened configurations. The goal is a compact, flat



Fig. 4. **The core garment pool spans diverse geometries.** The figure shows most of the 500 assets, covering different garment lengths, widths, sleeve types, necklines, and asymmetric cuts. All garments are naturally settled and shown at their relative physical sizes.

rectangular bundle with little protruding material. We prescribe no aspect ratio, crease pattern, or action order. Initial states for both tasks are saved in advance so that policies can run from identical states; the quantitative success criteria are given in Sec. IV-A.

B. NOCS-conditioned data-generation policies

We encode normalized object coordinate space (NOCS) [20] as the garment texture to provide a consistent surface-position reference as the cloth deforms. Let v_i denote a vertex of the canonical mesh, with component-wise bounds $v_{\min}$ and $v_{\max}$. Its normalized coordinate is

$$u_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}}, \tag{1}$$

whose three components form the RGB channels. Each color is assigned in the canonical configuration and remains attached to the corresponding vertex as the garment deforms. Similar relative positions across garments therefore receive similar colors. This exposes canonical surface-position cues through the model's existing image interface. The policy still observes only visible surfaces, and matching colors do not imply exact semantic correspondence across garments.

We use the pretrained $\pi_{0.5}$ model [21] as the base and separately fine-tune data-generation policies for flattening and folding. Each policy receives NOCS images through the model's original visual interface, together with the robot state and task instruction, and predicts joint targets and gripper commands for both arms. During execution, the policy applies only a prefix of each predicted action sequence and then replans from the latest observation, producing a closed-loop trajectory.

C. Learning from demonstrations and corrections

We collect separate flattening and folding demonstrations through VR teleoperation. To collect a folding demonstration, we restore the final state of its corresponding flattening demonstration and continue from that state. The demonstrations for each task are recorded separately and used to fine-tune the respective data-generation policies with the base model's action-prediction objective. Sec. IV-B reports the garment coverage and number of demonstrations.

To cover states visited by the data-generation policies, we collect one round of human corrections in a DAgger-like manner [22]. For each task, we first generate a candidate pool of 800 autonomous rollouts. A human collector selects executions with action loops or substantial deterioration of the cloth state, chooses an intervention point, and restores the corresponding simulation state. The collector then uses VR to complete the task, producing a correction trajectory $\mathcal{D}_H$. We combine these corrections with the original demonstrations $\mathcal{D}_E$ and update the policy using the same supervised objective. All supervised actions are provided by humans; autonomous rollouts only supply the starting states for correction.

To add training examples with exchanged arm roles, we apply bilateral morphological symmetry augmentation [23] to every human trajectory. At each frame, we reflect the garment state and transform the robot states and actions while exchanging the two arms and grippers. We then re-render NOCS observations by replaying the transformed states. Each human trajectory yields one mirrored sample, which is not counted as an additional human demonstration.

D. RGB data from successful trajectories

After training, we freeze the data-generation policies and generate trajectories autonomously. For flattening, we select 400 garments and generate 20 random initial states per garment. We execute the policy on these states, truncate a trajectory as soon as it satisfies the success criterion in Sec. IV-A, and retain at most ten successful trajectories per garment. For folding, we sample successful terminal states from the generated flattening trajectories and apply the four cardinal rotations to construct a pool of 20 initial states per garment. We then use the same success filtering and truncation procedure.

On a single NVIDIA GeForce RTX 5090 D, our implementation executes approximately 1,152 trajectories per GPU-day, excluding RGB re-rendering.

For each successful trajectory, we sample from 34 base garment materials with 39 color variants, 17 table materials, 12 floor materials, and eight background scenes. Light position, intensity, and color temperature are drawn from continuous ranges. We sample one appearance per physical trajectory and keep it fixed throughout the trajectory. We replay the saved states to render one RGB version while preserving the robot and garment motion, robot states, and action targets. The resulting data train downstream policies that receive RGB observations, robot states, and task instructions without NOCS. The appearance subsets used for training and evaluation are specified in the experimental setup.

IV. Experiments

A. Metrics and success criteria

Flattening. We align the current garment with a naturally laid-flat reference of the same garment using translation-invariant planar Procrustes alignment and normalize distances

by the diagonal of the reference XY bounding box. The mean three-dimensional distance between corresponding vertices after alignment defines the deformation error d_{def}. The mean reference-vertex displacement induced by the difference between the fitted rotation and the nearest valid cardinal orientation defines the orientation error d_{rot}; all four cardinal orientations are accepted. Using 890 expert terminal states from 182 garments, weighted equally by garment, we set median-and-MAD upper bounds of 0.088 and 0.043. A rollout succeeds when neither error exceeds its respective threshold for 1 s.

Folding. Because the task admits multiple valid outcomes, we do not match the result to a single template. Instead, the fold-quality score Q combines area compression, overall flatness, top-surface rectangularity, three-dimensional material overlap, and proximity to the table. We fit a fixed linear score using expert terminal states and successful HIL states as positive examples and unfinished original rollouts as negative examples, splitting both training and validation data by garment. The score reaches 85.4% balanced accuracy under garment-split cross-validation. We calibrate a threshold of 0.142 from the cross-validation predictions; a rollout succeeds when $Q \geq 0.142$ for 3 s. Score fitting and threshold calibration exclude Unseen garments in the RGB policy evaluation.

Task-specific evaluations use 50 Hz robot control, with time limits of 60 s for flattening and 40 s for folding. Throughout this section, arrows indicate the preferred direction of a metric; for comparisons among like policies, bold marks the best result for each task and evaluation condition.

B. Can the data-generation policy scale trajectory collection?

We test whether NOCS, mirror augmentation, and HIL progressively improve the policy's ability to generate successful trajectories. Each policy first learns from expert demonstrations; HIL corrections then cover states encountered during autonomous execution. Table I summarizes the two sources of human data and their garment coverage. Folding demonstrations start from the terminal states of their corresponding flattening demonstrations. Because these states are relatively controlled, we collect fewer demonstrations per garment. All trajectories are cleaned before training; the table also reports their sizes after the mirror augmentation in Sec. III-C.

TABLE I
HUMAN TRAINING DATA FOR THE DATA-GENERATION POLICIES.

Task	Expert raw → aug. / garments	HIL corrections raw → aug. / garments
Flatten	890 → 1,780 / 182	201 → 402 / 88
Fold	537 → 1,074 / 156	213 → 426 / 125

Starting from White, which uses expert demonstrations with a uniform white garment appearance, we successively add NOCS, mirror augmentation, and HIL corrections to form four cumulative configurations. Except for these cumulative changes, all configurations use the same training hyperparameters, train for three epochs on their respective datasets, and evaluate the final checkpoint. All data-generation policies are initialized from the pretrained $\pi_{0.5}$ model and trained separately for flattening and folding. For each task, we evaluate 200 garments from four saved initial states, for 800 closed-loop rollouts in total. All configurations share the same initial states and execution settings. Seen garments occur in $\mathcal{D}_E$ (or $\mathcal{D}_H$ for folding), whereas Unseen garments occur in neither $\mathcal{D}_E$ nor $\mathcal{D}_H$. The flattening evaluation has 100 Seen and 100 Unseen garments; folding has 130 and 70, respectively. We report overall and split success rates, together with the successful-trajectory quality metrics defined in Sec. IV-A. The Expert rows use terminal states that pass the success criteria as quality references and are excluded from the success-rate comparison. In Table II, Def. and Rot. are garment-equal means at first confirmed success.

TABLE II
DATA-GENERATION-POLICY PERFORMANCE ON FLATTENING.

Method	SR (%) ↑	Seen	Unseen	Def. ↓	Rot. ↓
White	24.8	26.3	23.3	0.081	0.024
+ NOCS	62.9	66.3	59.5	0.080	0.024
+ Mirror	69.6	76.0	63.3	0.080	0.025
+ HIL	**76.5**	**82.3**	**70.8**	0.080	0.024
Expert	–	–	–	0.068	0.019

TABLE III
DATA-GENERATION-POLICY PERFORMANCE ON FOLDING.

Method	SR (%) ↑	Seen	Unseen	Q ↑
White	55.1	61.0	44.3	0.809
+ NOCS	57.6	61.2	51.1	0.779
+ Mirror	60.5	64.2	53.6	0.846
+ HIL	**70.0**	**73.7**	**63.2**	0.825
Expert	–	–	–	1.037

NOCS provides the largest gain for flattening, increasing overall success by 38.1 percentage points. Its overall effect on folding is smaller: performance is nearly unchanged on Seen garments but improves by 6.8 points on Unseen garments. This contrast suggests that canonical position cues matter most when severe deformation makes garment parts difficult to identify, while also supporting transfer of learned manipulation across garment structures.

Mirror augmentation further improves both tasks without adding human trajectories, by 6.7 points for flattening and 2.9 points for folding. The larger flattening gain is consistent with its arbitrary initial orientations and flexible division of labor between the arms, indicating that known bilateral symmetry can expand the execution patterns covered by limited human data.

HIL further improves both tasks and provides the largest single-step gain for folding. Unlike NOCS and mirror augmentation, HIL adds corrections from failure states actually visited by the policy. With HIL, success on Unseen garments reaches 70.8% for flattening and 63.2% for folding, enabling

the policy to expand successful trajectories beyond the human data.

Higher success rates do not coincide with uniformly better success-state metrics. Def. and Rot. remain stable across flattening configurations, whereas Q fluctuates because each value is measured on a different successful subset. Human demonstrations have finer terminal states on average. Fig. 5 compares their visual quality with successful policy rollouts at selected quality percentiles.

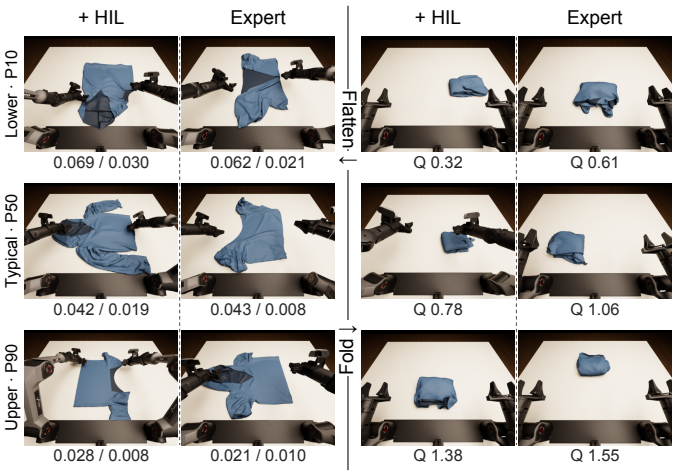


Fig. 5. **The data-generation policy produces successful terminal states with visual quality close to human demonstrations.** Rows show selected released states at P10, P50, and P90 of each source's eligible subset, ranked jointly by deformation and orientation errors for flattening, and by Q for folding. The two values below each flattening image are d_{def} and d_{rot}, respectively (lower is better); folding images report Q (higher is better). Flattening images use later released states, whereas Table II measures first confirmed success.

C. What is the value of automatically generated data?

We test whether autonomously generated trajectories improve RGB policy learning without additional human supervision. Starting from the same pretrained $\pi_{0.5}$ weights, we train separate RGB policies for flattening and folding on three data sources. **Human** contains the original demonstrations and corrections, whereas **Generated** contains autonomous trajectories retained by the success criteria. Their garment coverage partially overlaps, and **Combined** pools both datasets. Human and Generated use the same RGB re-rendering procedure. All three conditions draw on the same original human data, and autonomous generation adds no human action labels. Each policy uses the same training hyperparameters, trains for three epochs on its respective dataset, and is evaluated at the final checkpoint. Table IV summarizes the number of trajectories and garments in each training set.

We evaluate garment and appearance generalization on a shared test set. The Seen split contains 100 garments represented in all three training sets, while the Unseen split contains another 100 garments absent from all three. Of these Unseen garments, 73 come from the core 500-garment pool and 27 from the larger converted asset pool. For each garment, we evaluate four fixed held-out initial states under both Seen and Unseen appearances. Seen appearances are sampled from the training randomization distributions; Unseen appearances use garment textures, support surfaces, backgrounds, and lighting conditions withheld from training. All policies share the garments, initial states, and appearance settings. Each of the four garment–appearance combinations therefore contains 400 closed-loop rollouts; Avg. is their unweighted mean.

TABLE IV
RGB-POLICY TRAINING DATA (TRAJECTORIES / GARMENTS).

Task	Human	Generated	Combined
Flatten	1,091 / 217	3,992 / 400	5,083 / 470
Fold	750 / 242	3,856 / 399	4,606 / 458

TABLE V
RGB-POLICY SUCCESS RATES (%) BY TRAINING SET.

Task	Training set	Seen garments		Unseen garments		Avg.
		S app.	U app.	S app.	U app.	
Flatten	Human	33.3	30.5	22.3	19.3	26.3
	Generated	**41.3**	**33.3**	**23.5**	**23.8**	**30.4**
	Combined	32.8	28.0	20.3	19.5	25.1
Fold	Human	68.0	61.8	40.0	37.8	51.9
	Generated	**72.3**	69.0	**47.8**	**47.0**	**59.0**
	Combined	71.0	**69.8**	43.0	46.0	57.4

S app. and U app. denote Seen and Unseen appearances, respectively.

Although Generated trajectories end in less refined states on average than Human trajectories, the RGB policies trained on Generated achieve higher success rates in every evaluation condition in Table V. Autonomous generation therefore broadens the training experience available to downstream policies.

Combined contains the most trajectories and garments, yet provides no further improvement. The benefit of Generated therefore cannot be explained by data scale and coverage alone: the distribution induced by autonomous rollouts and success filtering may also matter, and simply pooling human and generated trajectories does not automatically exploit their complementarity. How to combine the two sources effectively remains an open question.

Across the two generalization axes, Unseen garments cause a larger performance drop than appearance shifts, consistently across all three training sets and both tasks. Generalizing to new garment structures and deformation patterns therefore remains the primary bottleneck for current RGB policies.

D. Comparison of policy models

We train four policy models on the Generated data from Sec. IV-C and compare them under the same closed-loop protocol across tasks, garments, and appearances. Specifically, we fine-tune $\pi_{0.5}$ [21], GR00T N1.7 [24], and an internal policy implemented with the public StarVLA framework [25], and train Diffusion Policy [26] from scratch.

Table VI shows that $\pi_{0.5}$ achieves the highest success rate across both tasks and all evaluation conditions, whereas Diffusion Policy achieves the lowest; StarVLA and GR00T lie

TABLE VI
RGB-POLICY SUCCESS RATES (%) BY METHOD.

Task	Method	Seen garments		Unseen garments		Avg.
		S app.	U app.	S app.	U app.	
Flatten	$\pi_{0.5}$	**41.3**	**33.3**	**23.5**	**23.8**	**30.4**
	StarVLA	25.0	14.3	12.8	8.3	15.1
	GR00T	16.0	17.3	10.8	12.3	14.1
	Diffusion	3.5	4.5	1.3	3.0	3.1
Fold	$\pi_{0.5}$	**72.3**	**69.0**	**47.8**	**47.0**	**59.0**
	StarVLA	67.0	67.5	38.3	40.3	53.3
	GR00T	47.0	47.0	26.5	27.0	36.9
	Diffusion	16.8	5.0	10.8	2.0	8.6

S app. and U app. denote Seen and Unseen appearances, respectively.

between them. These consistent gaps under a common protocol show that ClothDojo can differentiate policy models by their closed-loop performance.

Consistent with Sec. IV-C, all four models degrade on Unseen garments. Appearance shifts, however, have model- and task-dependent effects: they affect StarVLA mainly in flattening and Diffusion Policy mainly in folding, while $\pi_{0.5}$ and GR00T remain comparatively stable. Garment generalization and appearance robustness therefore capture two complementary dimensions of policy performance.

All four policies achieve lower success rates on flattening than on folding in our evaluation. The two tasks differ in both initial-state complexity and behavioral structure. Flattening starts from randomly crumpled states, requiring the policy to recognize garment structure under self-occlusion and adapt its regrasping, unfolding, and local adjustments as the garment evolves. Folding starts from a valid flattened state, where garment structure is clearer and the manipulation sequence is more regular. Flattening therefore requires more state-dependent long-horizon decisions.

E. Task composition and cross-embodiment reuse

As a supplementary experiment, we use the full configuration in Sec. IV-B, including NOCS, mirror augmentation, and HIL, to study task composition and cross-embodiment reuse. For task composition, we sample flattening and folding data equally and jointly train the policy to predict the semantic subtask and its corresponding action under the overall instruction "flatten and fold the garment." At inference time, the policy selects the current subtask from its observation. For cross-embodiment reuse, we retarget the same training trajectories to UR5e using inverse kinematics and dynamics replay, without collecting additional human trajectories.

Flatten $\rightarrow$ Fold starts from a crumpled state and runs continuously for 120 s; Fold starts from a fixed state that satisfies the flattening criterion and runs for 40 s. Both evaluations use the task metrics and initial-state setup from Sec. IV-B. For Flatten $\rightarrow$ Fold, the two SRs use the same set of trials and separately measure the fraction that reaches each criterion. Table VII reports the results.

TABLE VII
TASK COMPOSITION AND CROSS-EMBODIMENT REUSE.

Robot	Flatten $\rightarrow$ Fold		Fold
	Flatten SR	Fold SR	Fold SR
Piper	49.4	20.0	17.8
UR5e	68.9	7.1	27.8

The joint policy selects semantic subtasks from its observations and reaches both success criteria in some executions, but it does not yet complete the full process reliably; its performance in the standalone Fold evaluation also trails that of the task-specific policy. Balanced joint training and semantic supervision thus provide an initial mechanism for task composition, while task interference and the lack of transition experience may still limit continuous execution.

The UR5e policy, trained only on retargeted trajectories, can likewise perform both flattening and folding. ClothDojo's data-generation process is therefore not tied to Piper, although reuse requires the target robot's workspace and kinematics to be compatible with the original trajectories.



Fig. 6. **Potential task extensions of ClothDojo.** Left: humanoid dressing. Right: bimanual garment hanging. Both settings introduce more complex cloth–rigid-body contacts than tabletop flattening and folding.

V. CONCLUSION

Contributions. ClothDojo unifies learned trajectory generation and closed-loop evaluation for continuous bimanual garment manipulation. Its infrastructure supports VR demonstrations, HIL corrections, batched policy execution, success filtering, and state-based RGB re-rendering for both flattening and folding. Using NOCS-conditioned policies, we generate nearly 8,000 successful trajectories across hundreds of garments; under our evaluation protocol, RGB policies trained on these trajectories outperform those trained directly on the original human trajectories. The trained checkpoints and pipeline can continue generating trajectories for new garments and initial states, making ClothDojo an extensible source of adaptive garment-manipulation data rather than a fixed dataset.

Limitations. We validate the complete ClothDojo pipeline only in simulation and do not evaluate the learned policies on real robots. Evaluation at a comparable scale would require repeated physical setup and execution; reliable transfer also requires real-to-sim calibration of cloth material and contact parameters, robot dynamics, and low-level control—a substantial research problem in its own right. Finally, the

training trajectories do not cover the transition from flattening to folding, and the joint policy does not yet execute the two stages reliably in sequence.

Future work. The current tasks primarily involve cloth self-contact and relatively simple cloth–rigid-body contact. We will leverage the solver's bidirectional force coupling and watertight contact handling to extend ClothDojo to more contact-rich tasks, including humanoid dressing and garment hanging (Fig. 6).

REFERENCES

[1] Physical Intelligence *et al.*, "$\pi^*_{0.6}$: A VLA that learns from experience," *arXiv preprint arXiv:2511.14759*, 2025. [Online]. Available: https://arxiv.org/abs/2511.14759

[2] C. Yu, C. Sima, G. Jiang, H. Zhang, H. Mai, H. Li, H. Wang, J. Chen, K. Wu, L. Chen, L. Zhao, M. Shi, P. Luo, Q. Bu, S. Peng, T. Li, and Y. Yuan, "χ_0: Resource-aware robust manipulation via taming distributional inconsistencies," *arXiv preprint arXiv:2602.09021*, 2026. [Online]. Available: https://arxiv.org/abs/2602.09021

[3] T. Su, J. Zhu, T. Wang, Y. He, Z. Huang, J. Zhang, C. Ma, H. Wang, T. Zhang, M. Yin, W. Ding, and Y. Xu, "DeMaVLA: A vision-language-action foundation model for generalizable deformable manipulation," *arXiv preprint arXiv:2605.31286*, 2026. [Online]. Available: https://arxiv.org/abs/2605.31286

[4] H. Ha and S. Song, "FlingBot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2022, pp. 24–33. [Online]. Available: https://proceedings.mlr.press/v164/ha22a.html

[5] J. Luo, Z. Huang, H. Cheng, and Z. Hong, "ClothMate: Leveraging grasp-fling consistency for generalizable and data-efficient garment flattening," *IEEE Robotics and Automation Letters*, vol. 11, no. 1, pp. 298–305, 2026.

[6] A. Canberk, C. Chi, H. Ha, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Cloth Funnels: Canonicalized-alignment for multi-purpose garment manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5872–5879. [Online]. Available: https://clothfunnels.cs.columbia.edu/

[7] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, "SpeedFolding: Learning efficient bimanual folding of garments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1–8. [Online]. Available: https://arxiv.org/abs/2208.10552

[8] H. Xue, Y. Li, W. Xu, H. Li, D. Zheng, and C. Lu, "UniFolding: Towards sample-efficient, scalable, and generalizable robotic garment folding," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 2023, pp. 3321–3341. [Online]. Available: https://proceedings.mlr.press/v229/xue23b.html

[9] H. Lu, R. Wu, Y. Li, S. Li, Z. Zhu, C. Ning, Y. Shen, L. Luo, Y. Chen, and H. Dong, "GarmentLab: A unified simulation and benchmark for garment manipulation," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 11 866–11 903. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/15f80ec0fed53885d2ca6272edb96ede-Paper-Conference.pdf

[10] Y. Wang, R. Wu, Y. Chen, J. Wang, J. Liang, Z. Zhu, H. Geng, J. Malik, P. Abbeel, and H. Dong, "DexGarmentLab: Dexterous garment manipulation environment with generalizable policy," in *Advances in Neural Information Processing Systems*, D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, Eds., vol. 38, Main Conference. Curran Associates, Inc., 2025, pp. 133 757–133 796. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2025/file/c20ac0df6c213db6d3a930fe9c7296c8-Paper-Conference.pdf

[11] Z. Li, Y. Yang, S. Xie, K. Xu, T. Chen, Y. Wang, Z. Shen, Y. Shen, Y. Chen, W. Li, Y. Zheng, C. Zhang, S. Lin, F. Teng, H. Yang, M. Chen, S. Xie, and R. Wu, "LeHome: A simulation environment for deformable object manipulation in household scenarios," *arXiv preprint arXiv:2604.22363*, 2026. [Online]. Available: https://arxiv.org/abs/2604.22363

[12] W. Hu, H. Wang, Y. Li, L. Liu, M. Zhao, L. Jiang, X. Tang, J. Wei, Z. Shu, Z. Wang, Z. Su, H. Wang, and R. Yang, "SimWeaver: Zero-shot RGB sim-to-real for deformable manipulation," *arXiv preprint arXiv:2606.15338*, 2026. [Online]. Available: https://arxiv.org/abs/2606.15338

[13] Y. Chen, B. Xiao, and H. Wang, "FoldNet: Learning generalizable closed-loop policy for garment folding via keypoint-driven asset and demonstration synthesis," *IEEE Robotics and Automation Letters*, vol. 11, no. 3, pp. 3820–3827, 2026.

[14] Y. Wang, C. Qian, and E. Johns, "Instant-Fold: In-context imitation learning for deformable object manipulation," *arXiv preprint arXiv:2606.04269*, 2026. [Online]. Available: https://arxiv.org/abs/2606.04269

[15] M. Moghani, M. Azizian, A. Garg, Y. Zhu, S. Huver, and A. Mandlekar, "SoftMimicGen: A data generation system for scalable robot learning in deformable object manipulation," *arXiv preprint arXiv:2603.25725*, 2026. [Online]. Available: https://arxiv.org/abs/2603.25725

[16] Y. Zhou, H. Liu, X. Jiang, X. Shen, Y. Zhou, H. Wang, B. Fang, Y. Tian, M. Yu, Q. Yu, L. Ma, H. Li, H. Wang, J. Zeng, and J. Pang, "SIM1: Physics-aligned simulator as zero-shot data scaler in deformable worlds," *arXiv preprint arXiv:2604.08544*, 2026. [Online]. Available: https://arxiv.org/abs/2604.08544

[17] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[18] Newton Developers, "Newton: A GPU-accelerated physics simulation engine," 2026, version 1.5.0. [Online]. Available: https://github.com/newton-physics/newton

[19] M. Korosteleva, T. L. Kesdogan, F. Kemper, S. Wenninger, J. Koller, Y. Zhang, M. Botsch, and O. Sorkine-Hornung, "GarmentCodeData: A dataset of 3D made-to-measure garments with sewing patterns," in *Computer Vision – ECCV 2024*, ser. Lecture Notes in Computer Science, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., vol. 15118. Cham: Springer Nature Switzerland, 2025, pp. 110–127.

[20] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[21] Physical Intelligence *et al.*, "$\pi_{0.5}$: A vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025. [Online]. Available: https://arxiv.org/abs/2504.16054

[22] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html

[23] Y. Li, X. Ma, J. Xu, Y. Cui, Z. Cui, Z. Han, L. Huang, T. Kong, Y. Liu, H. Niu, W. Peng, J. Qiao, Z. Ren, H. Shi, Z. Su, J. Tian, Y. Xiao, S. Zhang, L. Zheng, H. Li, and Y. Wu, "GR-RL: Going dexterous and precise for long-horizon robotic manipulation," *arXiv preprint arXiv:2512.01801*, 2025. [Online]. Available: https://arxiv.org/abs/2512.01801

[24] NVIDIA, "NVIDIA Isaac GR00T N1.7," 2026, software release. [Online]. Available: https://github.com/NVIDIA/Isaac-GR00T

[25] StarVLA Community, "StarVLA: A lego-like codebase for vision-language-action model developing," *arXiv preprint arXiv:2604.05014*, 2026. [Online]. Available: https://arxiv.org/abs/2604.05014

[26] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. C. M. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023.