Enabling Robust Cloth Manipulation via Inference-Time Simulator-in-the-Loop Refinement

Anonymous Author(s)
Affiliation
Address
email

Abstract: *Simulator-in-the-loop* optimization offers a promising inference-time mechanism for robot manipulation. It uses a physical simulator as a backend rollout engine to evaluate candidate trajectories in parallel and refine nominal actions online, a paradigm proven effective in rigid-body manipulation where state and contact are relatively tractable. We bring this paradigm to real-world cloth manipulation from a single RGB input through three pillars. *(i)* We design a scalable synthetic-data generation and inference-time rollout pipeline built on FLASH, a deformable-object simulator that provides a practical balance among physical fidelity, numerical stability, and rollout efficiency. *(ii)* We develop a real-to-sim module, trained purely on synthetic data, that maps a single RGB observation to simulation-compatible cloth state by fusing pretrained visual features with learnable canonical tokens. *(iii)* We perform online planning by coupling a sparse-mesh rollout backend with prior-guided MPPI, anchored at an offline-distilled policy trajectory, preserving manipulation-relevant deformation and contact while enabling sufficient parallel rollout batches. Real-robot experiments show higher success rates and stronger robustness than baseline methods. Project page: https://anonymous-clothmani.github.io/

Keywords: Deformable Object Manipulation, Simulator-in-the-Loop Optimization, Cloth State Estimation

1 Introduction

Deformable-object manipulation presents fundamental challenges: infinite-dimensional configuration spaces, continuously evolving local geometry, severe self-occlusion, and pervasive frictional contact. These factors substantially complicate both state estimation and action reasoning, making reliable cloth manipulation considerably more challenging [1, 2, 3, 4].

Imitation learning has become a dominant paradigm for deformable-object manipulation, bypassing explicit dynamics modeling by fitting policies to the distribution of expert demonstrations and robot interaction data [5, 6, 7, 8]. Recent robot foundation models [9, 10, 11, 12, 13] further scale this idea and show promising task-level generalization [14, 15]. However, at the deployment stage, these policies have limited ability to correct their own execution errors. Small deviations in perception, grasp selection, timing, or action prediction can accumulate through contact-rich cloth dynamics and drive the system away from the intended task state. Reliable cloth manipulation, therefore, requires an inference-time mechanism that refines policy proposals and recovers from accumulated errors.

High-fidelity deformable-object simulators offer a promising foundation for physically grounded inference-time reasoning in cloth manipulation. Recent work has explored this direction by using simulators to generate data for diffusion-based neural dynamics models, paired with full-state reconstruction from partially observed point clouds [16]. We argue instead that the physical simulator itself is the most natural forward dynamics model [17]: direct simulator rollout avoids task- or

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

asset-specific dynamics training, removes the additional approximation error of learned predictors, and can serve as a general physical reasoning engine across deformable assets.

This raises the central question of this work: *Can we directly use deformable-object simulators as inference-time forward models for real-world cloth manipulation?* Answering this question requires solving the full real-to-sim-to-real loop. The simulator must be physically faithful and fast enough for online rollout. The real cloth state must be recovered accurately enough to synchronize a simulated twin. The rollout representation must balance state granularity, dynamics fidelity, and parallel candidate diversity under a limited control budget. If any part fails, the optimized action becomes misaligned with the physical scene or too expensive to compute online.

Contributions. We structure the investigation around three coupled system-level questions: *(i)* how simulator fidelity, numerical stability, and rollout efficiency affect inference-time refinement for cloth manipulation; *(ii)* how to recover simulation-compatible cloth states from practical real-world observations; and *(iii)* how to refine nominal policy trajectories into robust closed-loop actions through inference-time physics rollouts under limited online budgets.

Based on this joint study, we propose a unified *simulator-in-the-loop* framework for real-world cloth manipulation shown in Fig. 1. The framework contains two coupled flows. Offline, we use FLASH [18] to generate scalable synthetic cloth-deformation data and train an RGB-native real-to-sim module with pretrained visual features, canonical query tokens, and random grasping-point masking. Online, a single RGB observation is lifted into a sparse simulation-compatible cloth state, which synchronizes a sparse cloth-mesh rollout backend. A prior-guided MPPI controller then refines the nominal policy trajectory through parallel physics rollouts and applies the optimized action in a receding-horizon loop on hardware. The proposed pipeline provides a general mechanism for refining prior policies into robust closed-loop execution through online physics inference under limited rollout budget. To our knowledge, this is the first framework that directly uses a deformable-object simulator as an inference-time forward model for cloth manipulation under a real-time robot control budget. Real-robot experiments show that replacing or weakening any single component degrades closed-loop performance, and that the full system achieves higher success rates and stronger robustness, with further evaluations demonstrating generalization potential across deformable assets of different shapes and local deformation patterns.

2 Related Works

We use *simulator-in-the-loop control* to refer to methods that refine policy execution with an inference-time forward model. Given an estimated state and candidate actions, the model predicts short-horizon physical consequences for online optimization, and the selected action is applied in a receding-horizon manner. We review prior work along three axes central to this paradigm.

State Estimation. State estimation for rigid objects is commonly formulated as 6D pose tracking from RGB or RGB-D observations [19, 20, 21]. Cloth manipulation requires a different representation: the object state is high-dimensional, continuously deforming, and often self-occluded. Prior work reconstructs garment or cloth geometry from partial observations, including canonical garment completion from point clouds [22], mesh reconstruction with occlusion reasoning [23], diffusion-based mesh prediction from multi-view partial point clouds [16], and RGB-based refinement through 3D Gaussian splatting [24]. Other methods avoid explicit full-state reconstruction by feeding point clouds [25] or 3D keypoints [26] directly to the policy. Depth and point-cloud inputs are common, but thin fabric remains difficult for depth sensing, and geometry-only observations discard texture cues. Recent benchmarks also show persistent sim-to-real gaps for cloth perception [4, 27].

Forward Model. A simulator-in-the-loop controller requires a forward model that predicts how cloth responds to candidate actions with sufficient physical fidelity and rollout speed. One direction is to use physical simulators directly. GPU-parallel engines such as Isaac Sim [28], SAPIEN [29], Genesis [30], and FLASH [18] provide task-agnostic dynamics and support large-scale rollout, synthetic data generation, and contact-rich interaction. Another direction is to learn dynamics from



Figure 1: **Overview of the proposed framework.** Offline, we use FLASH to generate synthetic data for real-to-sim training. Online, RGB observations initialize physics rollouts that refine the prior policy through MPPI for closed-loop hardware execution.

simulated or real interaction data. For deformable objects, graph-based models such as MeshGraphNets [31], DPI-Net [32], VCD-Cloth [33], AdaptiGraph [34], and GraphGarment [35] model mesh, particle, or graph-structured dynamics, while UniClothDiff [16] learns diffusion-based cloth dynamics from synthetic rollouts. These learned models reduce per-rollout cost but remain tied to their training distribution and source simulator.

Rollout-based Action Optimization. Sampling-based optimization is well suited to contact-rich manipulation because it does not require differentiating through non-smooth dynamics [36, 37, 38]. CEM [39] and MPPI [40, 41] optimize action sequences by evaluating sampled rollouts, and recent variants improve sampling through learned values, covariance adaptation, biased proposals, or diffusion-style annealing [42, 43, 44, 45, 46, 47]. However, most applications focus on rigid-body systems with relatively compact state spaces. In cloth manipulation, high-dimensional deformation and nonconvex geometric objectives make unguided sampling inefficient, often limiting prior work to open-loop primitives or heavily randomized policies [48, 49, 50].

3 Method

In this section, we detail the implementation of the proposed framework illustrated in Fig. 1. We begin with the closed-loop execution formulation and notation, then describe the simulator backbone, real-to-sim reconstruction module, and prior-guided MPPI controller.

3.1 System Overview

At each control step t, the robot receives a single RGB observation I_t of the scene. The real-to-sim module reconstructs a simulation-compatible cloth state

$$\hat{s}_t = f_\theta(I_t), \tag{1}$$

which is used to synchronize the simulator. Given a nominal action sequence $\bar{U}_t$ from an offline policy, the controller samples candidate sequences, evaluates them through parallel rollouts in FLASH, and executes the first action of the refined sequence:

$$a_t^\star = \mathrm{MPPI}\left(\hat{s}_t, \bar{U}_t, \mathcal{F}, c\right), \tag{2}$$

where $\mathcal{F}$ is the simulator rollout operator and c the task cost. The executed action yields a new observation I_{t+1}, and the process repeats as a receding-horizon real-to-sim-to-real loop.

3.2 Simulator Backbone

In this loop, the simulator plays a dual role: it provides synthetic supervision and defines the simulation-compatible state space for real-to-sim training, and it serves as the inference-time rollout engine for MPPI. Its physical fidelity, numerical stability, and rollout efficiency therefore directly determine the performance of the entire real-to-sim-to-real pipeline.

We compare several deformable simulators in Section 4.1 and build the framework on FLASH [18]. FLASH represents cloth as a triangular mesh, where vertices are the simulated degrees of freedom and mesh connectivity encodes local deformation and surface geometry. For contact-rich cloth dynamics, FLASH resolves frictional contact through non-smooth Newton iterations on Signorini–Coulomb conditions, preserving stick–slip behavior under large deformation. Its local-global integrator and contact formulation reduce the dominant computation to GPU-friendly sparse matrix operations, enabling stable repeated rollouts for online control [51].

Let $x_t = (q_t, \dot{q}_t)$ denote the simulator state, where q_t and $\dot{q}_t$ are the mesh vertex positions and velocities. Given a robot action a_t applied through the end-effector or selected grasping points, FLASH advances the dynamics as

$$x_{t+1} = \mathcal{F}(x_t, a_t), \tag{3}$$

where $\mathcal{F}$ denotes one simulation step under manipulation, gravity, contact, and friction. During online control, this update is rolled out in parallel over candidate action sequences and passed to MPPI for action refinement.

3.3 RGB-native Real-to-Sim Reconstruction

We develop the state estimator f_θ in Equation (1) to recover a simulator-compatible cloth state $\hat{s}_t$ from a single RGB image. The estimator predicts the deformed positions of N material vertices on the garment asset at each time step t:

$$\hat{s}_t = \{\hat{\mathbf{p}}_{t,i}\}_{i=1}^{N}, \qquad \hat{\mathbf{p}}_{t,i} \in \mathbb{R}^3, \tag{4}$$

where each vertex corresponds to a fixed material point on the rest-shape mesh constructed offline.

Simulator asset construction. For each garment, we reconstruct a fixed-topology rest-shape mesh offline from ZED 2i RGB-D frames. Given a target mesh granularity, we segment the garment with SAM 2 [52], back-project the masked depth region into a point cloud, triangulate it via ball pivoting [53], and regularize the surface through isotropic remeshing with boundary-locked smoothing [54, 55]. This yields a triangular mesh whose material vertices are shared by real-to-sim reconstruction and rollout.

Synthetic supervision. Given the reconstructed asset, we train f_θ purely on synthetic data generated in FLASH. We simulate manipulation episodes with randomized cloth poses, configurations, and deformation patterns. Every frame is rendered from camera poses matched to the calibrated real setup and paired with the ground-truth vertex positions, giving M image–state pairs:

$$\mathcal{D}_{\text{syn}} = \left\{ \left(I^{(m)}, \left\{ \mathbf{p}_i^{\star(m)} \right\}_{i=1}^{N} \right) \right\}_{m=1}^{M}. \tag{5}$$

Network architecture. Given the RGB observation I_t, a frozen DINOv2 encoder [56] extracts a set of dense patch features, which a learnable linear layer P projects to the model dimension d:

$$V_t = P(\Phi_{\text{DINO}}(I_t)), \qquad V_t \in \mathbb{R}^{M_v \times d}, \tag{6}$$

where Φ_{DINO} denotes the frozen encoder and M_v is the number of patch tokens. In parallel, we maintain N learnable canonical tokens $C = \{c_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d}$, one per mesh vertex, optimized during training to encode the persistent identity and canonical location of each material point.

The N learnable canonical tokens and M_v visual tokens are concatenated and processed jointly by self-attention layers. Each canonical token aggregates visual evidence from the image while

exchanging structural context with others. We take the output slice corresponding to the N canonical tokens:

$$Z_t = \mathrm{SelfAttn}\left([\, C \,;\, V_t \,]\right)_{1:N}, \qquad Z_t \in \mathbb{R}^{N\times d}, \tag{7}$$

where $[\,\cdot\,;\,\cdot\,]$ denotes token concatenation and $(\cdot)_{1:N}$ extracts the canonical-token outputs. A shared decoder D_θ maps each token feature to its deformed 3D position:

$$\hat{\mathbf{p}}_{t,i} = D_\theta(Z_{t,i}), \qquad i = 1, \ldots, N. \tag{8}$$

Training objective. We supervise f_θ against the simulator ground truth with a smooth-L_1 loss:

$$\mathcal{L}_{\mathrm{r2s}} = \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{smooth-}L_1}\left(\hat{\mathbf{p}}_i, \mathbf{p}_i^\star\right). \tag{9}$$

We optimize only the projection layer P, the learnable canonical tokens C, the attention layers, and the decoder D_θ, while keeping DINOv2 frozen.

Robust augmentation. To improve robustness to sim-to-real visual shift, we inject perturbations at two levels during training. At the input level, we apply image-space randomization to the rendered RGB. At the latent level, we add noise and masking to the visual tokens, including random grasping-point masking around sampled contact regions, so the model learns to infer the cloth state from incomplete and corrupted visual evidence. *Details of dataset configuration, network architecture, and augmentation parameters are provided in Appendix A.1.*

3.4 Prior-Guided MPPI Refinement

We assume access to a base policy that maps the current cloth state to robot actions. At deployment, the base policy produces a nominal action sequence $\bar{U}_t$ over horizon H. Rather than being executed directly, this sequence serves as a task-level prior and is refined online through simulator rollouts.

Prior-guided online refinement. Given the synchronized cloth state $\hat{s}_t$, we sample K candidate action sequences around the nominal proposal:

$$U_t^{(k)} = \bar{U}_t + \epsilon^{(k)}, \qquad \epsilon^{(k)} \sim \mathcal{N}(0, \Sigma), \qquad k = 1, \ldots, K, \tag{10}$$

where Σ controls the local exploration scale. Each candidate is evaluated by rolling it out in the simulator backend:

$$\tau_t^{(k)} = \mathcal{F}_{0:H}\left(\hat{s}_t, U_t^{(k)}\right), \tag{11}$$

and assigned a task cost $S(\tau_t^{(k)})$ based on the folding objective and regularization terms such as workspace limits, table penetration, and action smoothness.

MPPI converts the rollout costs into normalized weights:

$$w_k = \frac{\exp(-S(\tau_t^{(k)})/\lambda)}{\sum_{j=1}^{K}\exp(-S(\tau_t^{(j)})/\lambda)}, \tag{12}$$

where λ is the temperature. The refined action sequence is computed as

$$U_t^\star = \sum_{k=1}^{K} w_k U_t^{(k)}. \tag{13}$$

Receding-horizon execution. The optimized sequence is applied in a receding-horizon manner: the robot executes $a_t^\star = U_t^\star[0]$, receives a new RGB observation, and repeats the real-to-sim-to-real loop. In this way, the base policy is continuously corrected by online physics inference, allowing the system to recover from local execution errors and cloth-state deviations.

4 Experiments

We organize the evaluation around the design questions in Section 1, validating the necessity of each component and demonstrating the effectiveness of the developed simulator-in-the-loop framework on real-world cloth manipulation tasks.

4.1 Simulator Backbone

Baselines. We compare representative deformable-object simulators, including GENESIS [30], NEWTON [57], ISAAC SIM [28], and FLASH [18], using the same cloth manipulation task as a controlled probe. The cloth asset, task cost, rollout horizon, MPPI controller, and compute environment are identical.

Table 1: **Simulator backbone evaluation** under the same cloth manipulation task and MPPI controller.

Simulator	K	Time ↓		KP err. ↓	MSE ↓	SR ↑
		Step	MPPI	(cm)	(cm^2)	
NEWTON	64	41.0	0.85	8.40 ± 8.72	56.8 ± 73.1	40%
	128	55.1	1.14	10.28 ± 11.08	68.9 ± 93.7	40%
	256	104.3	2.15	8.85 ± 10.71	58.1 ± 89.6	60%
	512	232.8	4.80	13.02 ± 13.06	92.3 ± 105.3	35%
ISAAC SIM	64	**11.7**	**0.26**	9.78 ± 10.74	60.5 ± 104.4	55%
	128	**22.2**	**0.50**	10.88 ± 11.13	67.4 ± 108.0	45%
	256	**41.1**	**0.92**	7.09 ± 5.60	32.4 ± 47.6	55%
	512	**80.7**	**1.79**	5.84 ± 4.73	24.6 ± 32.8	60%
FLASH	64	19.4	0.40	$\mathbf{1.61 \pm 1.18}$	$\mathbf{3.7 \pm 4.3}$	**100%**
	128	36.0	0.75	$\mathbf{1.54 \pm 0.88}$	$\mathbf{3.1 \pm 1.9}$	**100%**
	256	68.9	1.42	$\mathbf{1.95 \pm 1.82}$	$\mathbf{5.5 \pm 11.0}$	**95%**
	512	137.3	2.84	$\mathbf{2.08 \pm 1.93}$	$\mathbf{3.9 \pm 3.2}$	**95%**

Metrics. We evaluate each simulator in the same MPPI loop under different numbers of parallel environments. For rollout efficiency, we report the wall-clock time per deformable dynamics step (ms) and per MPPI rollout batch (s). For control quality, we report the task-keypoint alignment error (KP err), which is the planner's geometric objective, and define success by whether this error falls below a task-specific threshold. To assess physically implausible final states beyond keypoint alignment, we also report mean per-vertex MSE, Chamfer Distance (CD), and Earth Mover's Distance (EMD) against the target cloth state.

Results. Quantitative results are summarized in Table 1 and Fig. B.1. Across all parallel-environment counts, FLASH achieves the best overall trade-off among control accuracy, computation time, and final-state fidelity. Isaac Sim is marginally faster per step (1.5–1.9×), but FLASH attains 95–100% success across all K while Isaac Sim and Newton stay below 60%, with per-vertex MSE and EMD roughly an order of magnitude lower. Newton further shows occasional numerical instabilities, reflected in its large standard deviations, and Genesis is omitted as it fails to produce a stable fold across all K and seeds. *Implementation details are provided in Appendix B.1.*

4.2 State Estimation

Baselines. We evaluate on two cloth assets, a towel and a long-sleeve shirt. Baselines include a point-cloud DPM state estimator from UniClothDiff [16] with 50 denoising steps, and a depth-based variant using DeFM [58] in place of our RGB encoder.

Metrics. We report mean per-vertex MSE, CD, and EMD against the simulator ground truth, following [16]. Single-frame inference latency is also recorded for inference efficiency.

Results. Table 2 summarizes the quantitative comparison. Our RGB-native module achieves the lowest reconstruction error on both assets across all three metrics while maintaining the fastest inference, showing stronger material-point correspondence and better preservation of the overall cloth shape. The per-vertex MSE is 24.23 mm^2 on the long-sleeve shirt and 2.33 mm^2 on the towel (about 4.9 vs. 1.5 mm in RMS), while the CD and EMD remain within a few millimeters

Table 2: **State-estimation comparison.** Per-vertex MSE, CD, EMD, and single-frame inference latency on the validation split.

Asset	Real-to-Sim	MSE (mm^2) ↓	CD (mm) ↓	EMD (mm) ↓	Lat. (ms) ↓
Towel	**Ours**	$\mathbf{2.33 \pm 1.21}$	$\mathbf{1.94 \pm 0.41}$	$\mathbf{1.98 \pm 0.45}$	$\mathbf{7.4 \pm 0.6}$
	DPM	7.17 ± 2.22	3.89 ± 0.67	4.01 ± 0.72	318.5 ± 3.7
	DeFM	43.36 ± 34.66	5.51 ± 0.99	6.53 ± 1.52	12.6 ± 0.5
Long-sleeve Shirt	**Ours**	$\mathbf{24.23 \pm 11.49}$	$\mathbf{3.65 \pm 0.78}$	$\mathbf{4.67 \pm 1.24}$	$\mathbf{12.4 \pm 0.4}$
	DPM	45.74 ± 14.19	5.93 ± 0.27	8.21 ± 0.65	512.9 ± 6.0
	DeFM	43.52 ± 22.02	4.05 ± 0.38	5.93 ± 0.91	18.0 ± 0.6

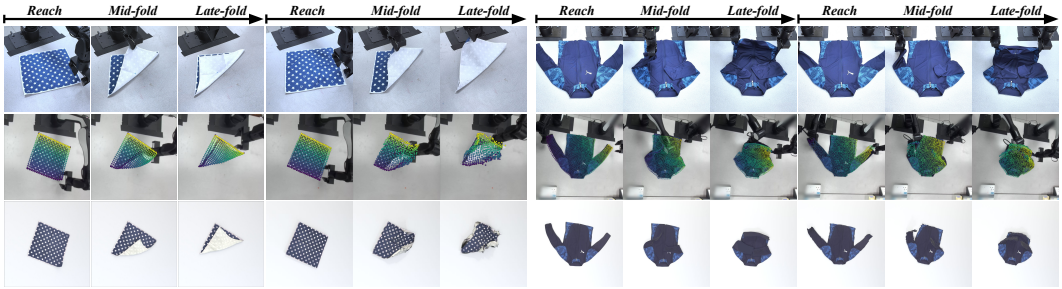


Figure 2: **Real-world real-to-sim reconstruction.** Panel (a) towel and (b) long-sleeve top, each comparing our RGB-native estimation with DPM at different folding stages. From top to bottom, the real observation, predicted vertices overlaid on the image, and the reconstructed cloth.

and the lowest among all methods. Fig. 2 further visualizes real-world deployment examples. Poor vertex accuracy or weak shape preservation can produce abnormal real-to-sim assets, whereas our method maintains both material-point consistency and global cloth shape, enabling faithful simulator initialization for rollout. *Baseline details are provided in Appendix B.2.*

4.3 Real-World Evaluation

Tasks. We construct end-to-end pipelines with different module choices and evaluate them on two real-world tasks: *single-arm diagonal folding* and *dual-arm symmetric folding*. For each trial, we randomize the initial towel pose and run the method on the same robot platform.

Baselines. We evaluate variants by replacing the state estimator, simulator backbone, or controller in the full pipeline. Specifically, we compare the RGB-native module with DPM [16], the FLASH backend with Isaac Sim [28], and prior-guided MPPI refinement with direct base-policy execution. A trial is successful if every target corner pair is aligned within 2 cm in the final cloth state.

Results. The full pipeline achieves the highest success rate on both tasks, demonstrating a substantial closed-loop improvement over direct base-policy execution. The drop in performance from replacing any single component further shows that state synchronization, simulator fidelity, and online refinement are tightly coupled, and that weakening any part degrades the entire real-to-sim-to-real loop. *All variants use the same robot platform, camera setup, action primitive, task cost, and stopping criterion. Detailed configurations and additional trial visualizations are provided in Appendix B.3.*

Table 3: **Real-world pipeline-variant evaluation.** Each row replaces one component of the full pipeline (first row).

Task	Real-to-Sim	Backend	Controller	Real SR (↑)
Single-arm diagonal	**Ours (RGB)**	**FLASH**	**MPPI**	**9/10**
	DPM	FLASH	MPPI	6/10
	Ours (RGB)	Isaac Sim	MPPI	5/10
	Ours (RGB)	FLASH	Base-policy	3/10
Dual-arm symmetric	**Ours (RGB)**	**FLASH**	**MPPI**	**8/10**
	DPM	FLASH	MPPI	3/10
	Ours (RGB)	Isaac Sim	MPPI	4/10
	Ours (RGB)	FLASH	Base-policy	1/10

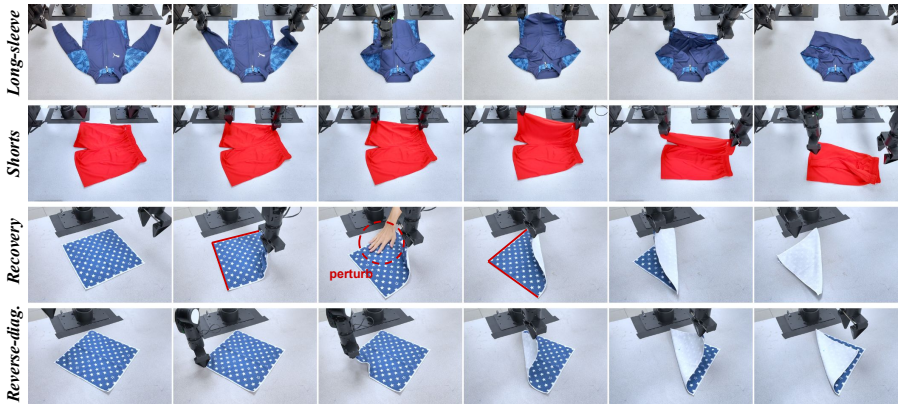


Figure 3: **Representative rollouts for generalization and robustness evaluation.** Time progresses from left to right. The four rows visualize long-sleeve shirt folding, shorts folding, recovery from mid-fold human disturbance, and reverse-diagonal towel folding.

4.4 Generalization and Robustness

Tasks. We further evaluate the framework on four additional real-world settings: long-sleeve shirt folding, shorts folding, reverse-diagonal towel folding, and mid-fold disturbance recovery. These settings test generalization across garment assets and folding variants, as well as robustness to external perturbations. All tasks follow the same evaluation protocol as the towel experiments, with task-specific geometric success criteria. Full task configurations are provided in Appendix B.4.

Table 4: **Generalization and robustness evaluation.**

Task	Real SR
Long-sleeve	8/10
Shorts	7/10
Towel, reverse diag.	8/10
Towel, mid-fold disturb.	8/10

Results. Table 4 reports the success rate over ten trials for each task, and Fig. 3 shows one representative execution sequence per setting. The framework remains reliable across new garment assets, folding variants, and mid-fold disturbances without additional dynamics retraining, demonstrating robustness to task and asset variations as well as recovery from execution perturbations.

5 Conclusion

We presented a simulator-in-the-loop framework that brings inference-time physics refinement to real-world cloth manipulation from a single RGB observation, coupling the FLASH deformable simulator, an RGB-native real-to-sim module, and a prior-guided MPPI controller into one closed-loop real-to-sim-to-real pipeline. Real-robot experiments show that the framework attains higher success rates and stronger robustness than baselines, and remains robust across broad tasks.

6 Limitations and Future Work

Two open problems remain. First, although our system strikes a good balance between rollout efficiency and physical fidelity, solving deformable dynamics is far less parallelizable than the rigid-body case, so rollouts easily saturate the single-GPU budget, and the achievable batch size, which upper-bounds the closed-loop control rate, remains tied to the simulation backend. Future work will scale inference across multiple GPUs via distributed rollout and further backend optimization. Second, generalization to novel garments is still limited by two per-asset stages: the offline cloth-mesh reconstruction and the real-to-sim module, which is currently trained per garment. We see a promising direction in image-conditioned mesh foundation models that produce a simulator-compatible asset from a single observation, paired with a category-level real-to-sim module that removes the per-garment training step.

References

[1] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, et al. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022.

[2] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 7:82, 2020.

[3] X. Lin, Y. Wang, J. Olkin, and D. 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*, pages 432–448, 16–18 Nov 2021.

[4] W. Hu, X. Tang, Y. Li, Z. Shu, W. Li, H. Wang, R. Yang, et al. Real garment benchmark (rgbench): A comprehensive benchmark for robotic garment manipulation featuring a high-fidelity scalable simulator. *arXiv preprint arXiv:2511.06434*, 2025.

[5] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. C. Burchfiel, and S. Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[6] Z. Fu, T. Z. Zhao, and C. Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[7] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[8] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[9] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183, 06–09 Nov 2023.

[10] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713, 06–09 Nov 2025.

[11] S. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. P. Kaelbling, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations (ICLR)*, 2024.

[12] L. Maes, Q. L. Lidec, D. Scieur, Y. LeCun, and R. Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

[13] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[14] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[15] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, et al. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024.

[16] T. Tian, H. Li, B. Ai, X. Yuan, Z. Huang, and H. Su. Diffusion dynamics models with generative state estimation for cloth manipulation. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 1703–1725, 27–30 Sep 2025.

[17] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa. Predictive sampling: Real-time behaviour synthesis with MuJoCo. *arXiv preprint arXiv:2212.00541*, 2022.

[18] S. Luo, B. Zhou, C. Zhang, X. Liu, Z. Huang, G. Yang, Z. Han, X. Hu, E. Yang, R. Yu, et al. Flash: Fast learning via gpu-accelerated simulation for high-fidelity deformable manipulation in minutes. *arXiv preprint arXiv:2604.17513*, 2026.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[20] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 606–617, 2023.

[21] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17868–17879, 2024.

[22] C. Chi and S. Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3304–3313, 2021.

[23] Z. Huang, X. Lin, and D. Held. Mesh-based Dynamics with Occlusion Reasoning for Cloth Manipulation. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[24] A. Longhini, M. Büsching, B. P. Duisterhof, J. Lundell, J. Ichnowski, M. Björkman, and D. Kragic. Cloth-splatting: 3d cloth state estimation from rgb supervision. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2845–2865, 06–09 Nov 2025.

[25] H. Chen, J. Li, R. Wu, Y. Liu, Y. Hou, Z. Xu, J. Guo, C. Gao, Z. Wei, S. Xu, J. Huang, and L. Shao. Metafold: Language-guided multi-category garment folding framework via trajectory generation and foundation model. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4339–4346, 2025.

[26] Y. Deng and D. Hsu. General-purpose clothes manipulation with semantic keypoints. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13181–13187, 2025.

[27] Y. Ru, L. Zhuang, Z. He, F. P. Audonnet, and G. Aragon-Caramasa. Can real-to-sim approaches capture dynamic fabric behavior for robotic fabric manipulation? *arXiv preprint arXiv:2503.16310*, 2025.

[28] NVIDIA. Isaac Sim. https://github.com/isaac-sim/IsaacSim, 2025.

[29] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. Sapien: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11094–11104, 2020.

[30] G. Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[31] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[32] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019.

[33] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 256–266, 08–11 Nov 2022.

[34] K. Zhang, B. Li, K. Hauser, and Y. Li. AdaptiGraph: Material-Adaptive Graph-Based Neural Dynamics for Robotic Manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[35] W. Chen, K. Li, D. Lee, X. Chen, R. Zong, and P. Kormushev. Graphgarment: Learning garment dynamics for bimanual cloth manipulation tasks. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7615–7621, 2025.

[36] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[37] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(104):3137–3181, 2010.

[38] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake. Do differentiable simulators give better policy gradients? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20668–20696, 17–23 Jul 2022.

[39] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science & Business Media, 2004.

[40] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.

[41] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[42] N. A. Hansen, H. Su, and X. Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406, 17–23 Jul 2022.

[43] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[44] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras. Trajectory distribution control for model predictive path integral control using covariance steering. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1478–1484, 2022.

[45] Z. Yi, C. Pan, G. He, G. Qu, and G. Shi. CoVO-MPC: Theoretical analysis of sampling-based MPC and optimal covariance design. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pages 1122–1135, 15–17 Jul 2024.

[46] E. Trevisan and J. Alonso-Mora. Biased-mppi: Informing sampling-based model predictive control by fusing ancillary controllers. *IEEE Robotics and Automation Letters*, 9(6):5871–5878, 2024.

[47] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4974–4981, 2025.

[48] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *5th Annual Conference on Robot Learning*, 2021.

[49] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 734–743, 29–31 Oct 2018.

[50] R. Jangir, G. Alenyà, and C. Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636, 2020.

[51] Z. Zeng, S. Luo, F. Shi, and Z. Zhang. Fast but accurate: A real-time hyperelastic simulator with robust frictional contact. *ACM Trans. Graph.*, 44(4), July 2025.

[52] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollar, and C. Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

[53] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4): 349–359, 1999.

[54] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, page 185–192, New York, NY, USA, 2004.

[55] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 351–358, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917014.

[56] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. HAZIZA, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[57] The Newton Contributors. Newton: GPU-accelerated physics simulation for robotics and simulation research. https://github.com/newton-physics/newton, Apr. 2025.

[58] M. Patel, J. Frey, M. Mittal, F. Yang, A. Hansson, A. Bar, C. Cadena, and M. Hutter. DeFM: Learning foundation representations from depth for robotics. *arXiv preprint arXiv:2601.18923*, 2026.

Appendix

A Implementation Details

This section provides the full implementation details of the two modules introduced in the main paper. We describe the RGB-native real-to-sim reconstruction module in §A.1 and the prior-guided MPPI controller, including its cost terms and hyperparameters, in §A.2.

A.1 RGB-native Real-to-Sim Reconstruction

The real-to-sim module maps a single RGB frame to the deformed 3D positions of a fixed set of N cloth mesh vertices, the same vertex set the planner rolls out. The model has four trainable components on top of a frozen visual backbone: a linear projection P, a bank of N learnable canonical tokens, a self-attention fusion backbone, and a per-vertex MLP decoder.

Input. The input is a single RGB frame center-cropped and resized to 392×392 and normalized with the standard ImageNet statistics. The same crop and normalization are applied identically at training and test time.

Visual Encoder. The encoder is a frozen pretrained DINOv2-small [56] ViT-S/14, with patch size 14. The 392×392 input yields $M_v = (392/14)^2 = 784$ dense patch tokens at the backbone hidden dimension $d_{\text{vit}} = 384$. A learnable linear projection $P : \mathbb{R}^{384} \to \mathbb{R}^d$ maps each patch token to the model dimension $d = 256$.

Canonical Tokens. We maintain N learnable canonical tokens $C = \{c_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d}$, one per mesh vertex ($N = 464$ for the towel, $N = 1156$ for the pants, and $N = 1379$ for the long-sleeve top). Each c_i is initialized from a deterministic Gaussian random Fourier-feature map of its rest-shape coordinate $\mathbf{x}_i^{\text{rest}} \in \mathbb{R}^3$:

$$c_i^{(0)} = \left[\sin(2\pi\, \mathbf{x}_i^{\text{rest}} B);\, \cos(2\pi\, \mathbf{x}_i^{\text{rest}} B) \right] \in \mathbb{R}^d, \qquad B \in \mathbb{R}^{3 \times d/2},\ B_{jk} \sim \mathcal{N}(0, \sigma_B^2), \tag{14}$$

with $\sigma_B = 4.0$ and a fixed random seed. The matrix B is sampled once and frozen, while C is optimized as a free parameter table during training.

Fusion Backbone. The N canonical tokens are concatenated with the M_v projected visual tokens into a sequence of length $N + M_v$ and processed by a stack of $L = 4$ standard pre-norm self-attention blocks (multi-head self-attention + feed-forward) operating at $d = 256$ with $h = 8$ heads and dropout 0.1. The output slice corresponding to the first N positions is taken as the fused canonical features $Z \in \mathbb{R}^{N \times d}$. Structural context and visual evidence are exchanged jointly by self-attention.

Decoder. A shared per-vertex MLP decoder maps each Z_i to its predicted 3D position. It applies a LayerNorm followed by two hidden Linear→GELU blocks of width 512 and a final Linear projection to $\mathbb{R}^3$:

$$\hat{\mathbf{p}}_i = W_{\text{out}}\, \text{GELU}(W_2\, \text{GELU}(W_1\, \text{LN}(Z_i))), \tag{15}$$

where $W_1 \in \mathbb{R}^{256 \times 512}$, $W_2 \in \mathbb{R}^{512 \times 512}$, and $W_{\text{out}} \in \mathbb{R}^{512 \times 3}$.

Loss. We supervise the model against the simulator ground-truth vertex positions with a smooth-L_1 loss (Huber threshold $\beta = 1.0$), reduced as the mean over the N vertices and the batch.

Training Data. Paired RGB–mesh examples are generated in FLASH by replaying scripted goal-guided demonstration trajectories and rendering each frame from the camera pose that matches our real-world setup. For every frame we record the ground-truth deformed 3D positions of all N cloth-mesh vertices. The dataset is partitioned $90\%/10\%$ between training and validation at the episode level so that no frame leaks across the split.

Augmentation. Two-level perturbations are applied at training time to bridge the sim-to-real visual gap: photometric, geometric, and lighting randomization on the RGB input, plus token-level masking that propagates a grasp-point pixel occlusion into the overlapping DINOv2 patch tokens. The full parameter ranges are tabulated in Tab. A.1.

Table A.1: **Training-time augmentation for the RGB module.** Every randomized operation applied during training, with its application probability and parameter range. Color jitter and Gaussian blur run on every frame with parameters sampled from the listed ranges. The grasp-point token mask is propagated only when the grasp-point pixel occlusion is applied, so its probability is conditional on that event.

Operation	Level	Prob.	Range / strength
Color jitter	Image	1.0	brightness, contrast, saturation ± 0.4, hue ± 0.1
Random grayscale	Image	0.2	convert RGB to single channel
Gaussian blur	Image	1.0	kernel 5, $\sigma \in [0.1, 2.0]$
Directional lighting	Image	0.6	gradient depth $\alpha \in [0.15, 0.40]$
Random erasing	Image	0.3	erased area fraction $\in [0.02, 0.15]$
Grasp-point occlusion	Image	0.7	half-side $\in [25, 55]$ px, aspect $\in [0.7, 1.5]$, corner jitter ± 12 px, center jitter ± 10 px, fill brightness $\in [0, 0.05]$
Grasp-point token mask	Token	0.8	DINOv2 patches with $> 30\%$ area overlap flagged invalid for fusion

Optimizer and Schedule. The trainable parameters are $\{P, C,$ self-attn, decoder$\}$; the DINOv2 backbone is frozen. We use AdamW with learning rate 3×10^{-4}, weight decay 10^{-4}, and gradient-norm clipping at 1.0. The learning rate follows a cosine schedule with a 3-epoch linear warmup and a minimum-LR ratio of 10^{-2}. Training runs for 50 epochs with a batch size of 32 on a single NVIDIA RTX 4090.

Deployment. At test time, each RGB frame is center-cropped to 392×392 and passed through the network to obtain the predicted 3D positions of all N vertices. The predictions are transformed from camera frame to world frame and broadcast into all K parallel simulation environments. The end-effector pose and gripper state are read from the robot proprioception stack and synchronized in the same step.

A.2 Prior-Guided MPPI Control

Prior-Informed Sampling. At each planning step, the cloth state is synchronized from the real-to-sim module (§A.1) into all K parallel simulation environments. Our default MPPI configuration uses $K = 256$ parallel sampled trajectories, a planning horizon of $H = 5$ steps, a per-dimension noise standard deviation of $\sigma = 5 \times 10^{-3}$ m, and a softmax temperature of $\lambda = 0.08$. The proposal noise is temporally independent Gaussian, $\boldsymbol{\epsilon}_t^{(k)} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, and the K rollouts are dispatched to K parallel simulation environments that remain resident on the GPU across planning steps. The proposal mean is anchored at the distilled-prior nominal $\bar{U}_t$ as described in the main paper, and refined locally by the importance-weighted update.

Cost Function Details. The stage cost combines a task-driving distance term with a small set of safety and smoothness penalties whose mix changes across the manipulation phases. Let $\mathbf{x}_{\text{ee}} \in \mathbb{R}^3$ denote the end-effector position, z_{ee} its vertical coordinate, and z_{table} the table surface height. The distance term is phase-dependent. In the *reach* phase it is the full 3D distance from the end-effector to the pick corner $\mathbf{p}_{\text{pick}}$. In the *fold* phase it is the in-plane distance from the grasped cloth corner $\mathbf{p}_{\text{grasp}}$ to the fold target $\mathbf{p}_{\text{target}}$. The remaining components penalize table penetration, workspace-boundary violation, and action magnitude. The terminal cost $\phi(x_H)$ reuses the phase-dependent distance evaluated at the final rolled-out state. Expressions, weights, and activations are listed in Table A.2.

At deployment we use a table safety margin $\delta_{\text{tbl}} = 5$ mm, and a workspace soft-boundary margin $\delta_{\text{bnd}} = 2$ cm.

Table A.2: **MPPI stage cost components.** $\mathbf{x}_{\text{ee}}^{xy}$ denotes the in-plane end-effector position. $\mathcal{W}_{\text{safe}}^{\text{inner}}$ denotes the interior of the safe workspace after shrinking each axis by a soft margin δ_{bnd}. z_{fold}^{*} is the nominal lift height during folding.

Term	Expression	Weight	Activation
c_{dist} (reach)	$\|\mathbf{x}_{\text{ee}} - \mathbf{p}_{\text{pick}}\|_2$	1.0	*reach* phase
c_{dist} (fold)	$\|\mathbf{p}_{\text{grasp}}^{xy} - \mathbf{p}_{\text{target}}^{xy}\|_2$	1.0	*fold* phase
c_{table}	$\max(0,\, z_{\text{table}} + \delta_{\text{tbl}} - z_{\text{ee}})^2$	50.0	all phases
c_{limit}	axis-wise squared hinge on $\mathcal{W}_{\text{safe}}^{\text{inner}}$	20.0	all phases
c_{vel}	$\|\mathbf{a}_t^c\|_2^2$	10.0	all phases
c_{height}	$(z_{\text{ee}} - z_{\text{fold}}^{*})^2$	2.0	*fold* phase

B Experiment Details

This section documents the detailed configurations behind the experiments referenced in the main paper. We describe the simulator backbone comparison in §B.1, the state-estimation dataset and baseline configurations in §B.2, the real-world pipeline-variant evaluation in §B.3, and the generalization and robustness study in §B.4.

B.1 Simulator Backbone Comparison

We compare GENESIS [30], NEWTON [57], ISAAC SIM [28], and FLASH [18] on the same single-arm diagonal-fold task under a shared vanilla MPPI controller, varying only $K \in \{64, 128, 256, 512\}$. Main results are in Table 1 and Fig. B.1.

Task and controller. The cloth is a 30×30 cm towel mesh of $N_{\text{towel}} = 464$ vertices. MPPI plans an H=6-step horizon at Δt_{act}=40 ms with per-step EE translation clamped to 20 mm; each simulator runs at Δt_{sim}=10 ms (i.e., 4 simulation steps per action). Trials early-stop when KP err. < 5 cm, then settle 150 further steps to acquire the final folding behavior.

Seeds and compute. Each K uses 20 fixed seeds over a ± 8 cm / $\pm 30^\circ$ init randomization. The same per-seed initial pose is replayed on every simulator to ensure a fair comparison. All trials run on a single RTX 4090.

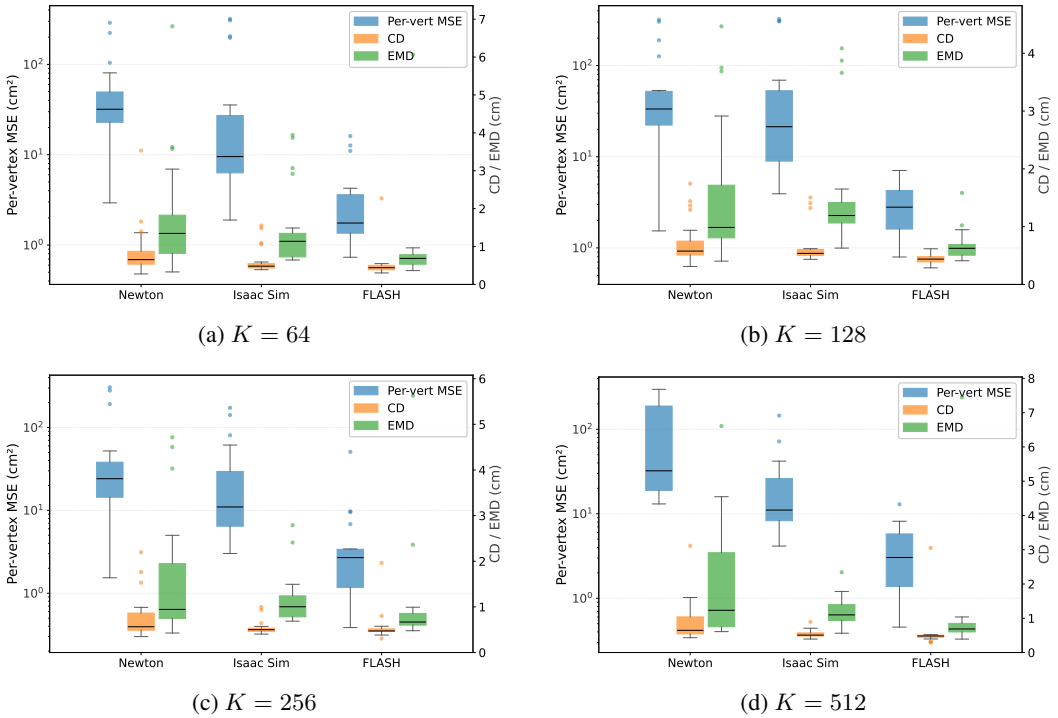
Metrics. *Step* and *MPPI* are wall-clock per simulation step and per MPPI iteration, respectively. *KP err.* is the settled horizontal distance between pick- and target-corner. *MSE* is the mean squared horizontal distance between each moved-half vertex (231 on the 464-vertex mesh) and its mirror partner in the stationary half (cm^2), which measures cloth overlap after the fold, independent of the rest target. *CD* and *EMD* (Fig. B.1) compare the settled cloth to the ideal perfect-fold triangle. *SR* is the seed fraction with KP err. < 5 cm.

Simulator configurations. Table B.3 lists the cross-simulator settings. Cloth material parameters are omitted from the table because they are not commensurable across solver types. Each cloth was properly tuned with solver-specific values given per simulator below.

Table B.3: **Simulator backbone configurations** under the shared cloth asset, MPPI controller, action clamp (20 mm/step), and cost. Per-solver iteration counts use different update rules and are not directly comparable as a compute proxy.

	GENESIS	NEWTON	ISAAC SIM	FLASH
Dynamic solver	PBD	VBD	XPBD	FEM
Solver iters	4	1	16	8
Time step Δt	10 ms	10 ms	10 ms	10 ms
Friction μ	1.0	1.0	1.0	1.0

GENESIS. GENESIS uses position-based dynamics (PBD) with $4/1/1$ iterations on stretch/bending/shear. It re-tessellates the input into ~ 1400 internal particles while keeping 464



(a) $K = 64$ (b) $K = 128$

(c) $K = 256$ (d) $K = 512$

Figure B.1: **Settled-state fold-quality distributions across simulators at four parallel environment counts.** Each panel shows the 20-seed distribution of per-vertex MSE (cm^2, log left axis), CD (cm), and EMD (cm) for Newton, Isaac Sim, and FLASH under identical seeded initial cloth poses.

controllable vertices, with stretch compliance $\alpha_{\text{stretch}} = 10^{-4}$, bending compliance $\alpha_{\text{bend}} = 10^{-2}$, and density 1.0.

NEWTON. NEWTON uses vertex block descent (VBD) on Warp with 1 iteration per step. The material is parameterized by triangle stretch coefficient 200, edge bending coefficient 5×10^{-3}, and density 10.

ISAAC SIM. ISAAC SIM uses the particle-cloth model, wrapping PhysX's XPBD solver with 16 iterations per step. The material is parameterized by spring stretch stiffness 10^4 and bending stiffness 2×10^{-2}, and per-particle mass 10^{-3}.

FLASH. FLASH uses FEM with isometric bending and a Signorini–Coulomb non-smooth Newton contact solver, with 8 local–global iterations and 10 constraint iterations. The material is parameterized by stretch stiffness 4×10^6, bending stiffness 2×10^{-2}, and total mass 1.0.

GENESIS fails the fold task. GENESIS does not produce a usable fold across any K or seed since the plane friction does not take an effect under drag for the held corner to settle on the target. Tuning substeps, compliance, friction, and pin strategy closed only part of the gap, so we omit GENESIS from Table 1.

Results analysis. (i) *Compute.* NEWTON is 2.5–3.5× slower per step than ISAAC SIM. ISAAC SIM and FLASH both scale near-linearly in K, with ISAAC SIM a constant $\sim 1.7\times$ faster per step. (ii) *Control accuracy.* NEWTON and ISAAC SIM stay at 6–13 cm KP err. with SR $\leq 60\%$ at every K, while FLASH attains 1.5–2.1 cm and 95–100% SR. The gap is not closed by more rollouts. (iii) *Physical fidelity.* NEWTON and ISAAC SIM sit at median MSE 30–300 cm^2 and EMD 5–10 cm at every K, while FLASH is one to two orders tighter (median MSE 3–4 cm^2, EMD 1.5–2.5 cm). Furthermore, FLASH is the only candidate that provides a usable physics oracle for online MPPI cloth folding across our K range.

B.2 State Estimation

Cloth assets. We evaluate on two cloth assets, a towel and a long-sleeve top. The towel uses a sparse triangle mesh of $N_{\text{towel}} = 464$ vertices. The long-sleeve top uses $N_{\text{sleeve}} = 1379$ vertex sparse mesh. Both meshes and their vertex orderings are computed once and persisted; the same per-asset vertex set is used by the FLASH rollout backend and as the ground-truth labels for state estimation.

Data collection. Episodes are generated in FLASH with scripted goal-guided demonstration trajectories that drive the cloth toward target fold configurations or settle it at rest. Per asset, we collect two subsets. The *fold* subset samples random goal keypoints on a workspace grid so each episode covers a different fold target. The *rest-state* subset drops the cloth under gravity from a randomized initial pose and stores one frame per episode at rest.

At episode reset, the cloth is randomized by drawing the centroid offset $\Delta x \sim \mathcal{U}(0.16, 0.40)$ m and $\Delta z \sim \mathcal{U}(-0.40, -0.20)$ m, and the yaw angle $\sim \mathcal{U}(-0.50, 0.50)$ rad.

Each frame logs the RGB image, the depth map, and the ground-truth mesh-vertex positions in the world frame, written as a `.pt` file per episode. The point cloud is obtained by back-projecting valid depth pixels through the camera intrinsics. Representative samples are shown in Fig B.2.

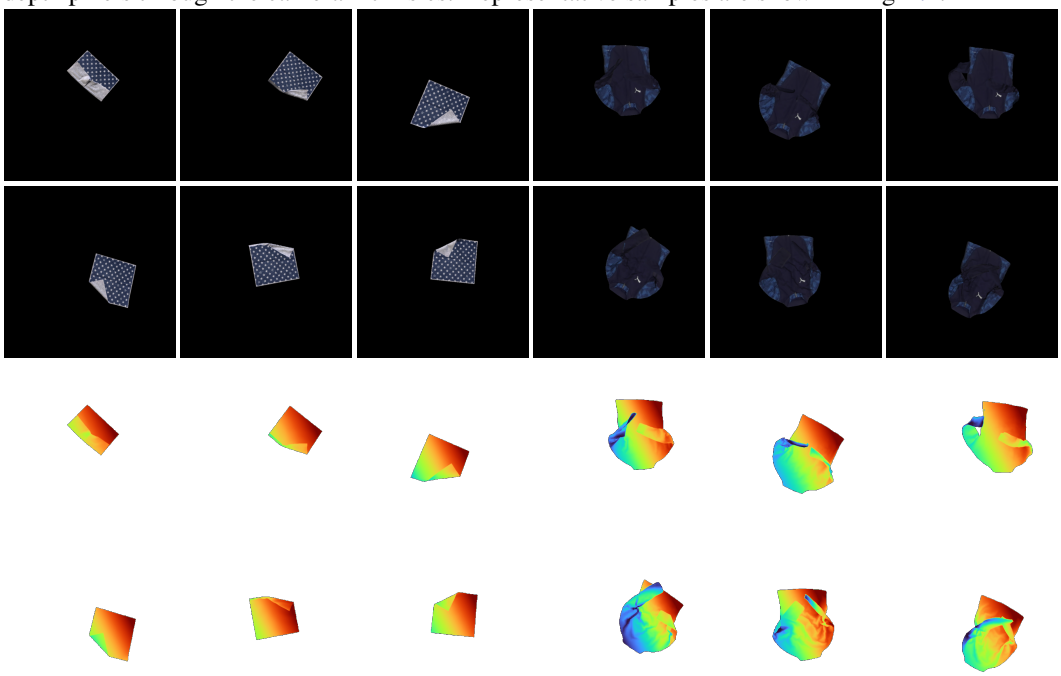


Figure B.2: Representative synthetic cloth deformation samples from the state-estimation dataset, covering the towel and the long-sleeve top across the fold and rest-state subsets. Top, RGB views. Bottom, depth maps.

Baseline configurations. All three methods predict the same N indexed material vertices per asset and are trained on the dataset described above. They differ in input modality and encoder. Table B.4 summarizes the architecture and training, with per-method notes below.

B.3 Real-World Evaluation

We deploy on the AIRBOT Play robot arm in single-arm and dual-arm configurations. Visual observations come from a single ZED 2i stereo camera mounted overhead. Our pipeline uses only the left RGB stream, and the stereo depth is used to provide the point-cloud input required by one baseline. All policy distillation and online planning run on a single NVIDIA RTX 4090.

Table B.4: Architecture and training configuration of the three state-estimation methods.

	Ours (RGB)	DeFM (depth)	UniClothDiff DPM (point cloud)
Input	RGB 392^2	depth 392^2	10,000 points
Encoder	DINOv2 ViT-S/14, frozen	DeFM ViT-S/14, frozen	KNN-grouper + patch MLP
Output head	N canon. tokens, 4L self-attn (d=256, h=8), MLP	same as Ours	DDPM denoiser, template-cond.
Loss	smooth ℓ_1	smooth ℓ_1	DDPM noise pred.
Optimizer (AdamW)	lr 3e$-$4, wd 10^{-4}	lr 3e$-$4, wd 10^{-4}	lr 10^{-3}, wd 10^{-2}
Batch size	32	32	4 per GPU
Schedule	50 ep., cosine, 3-ep. warmup, EMA 0.999	same as Ours	900K steps, const., 500-step warmup
Gradient clip	1.0	1.0	1.0

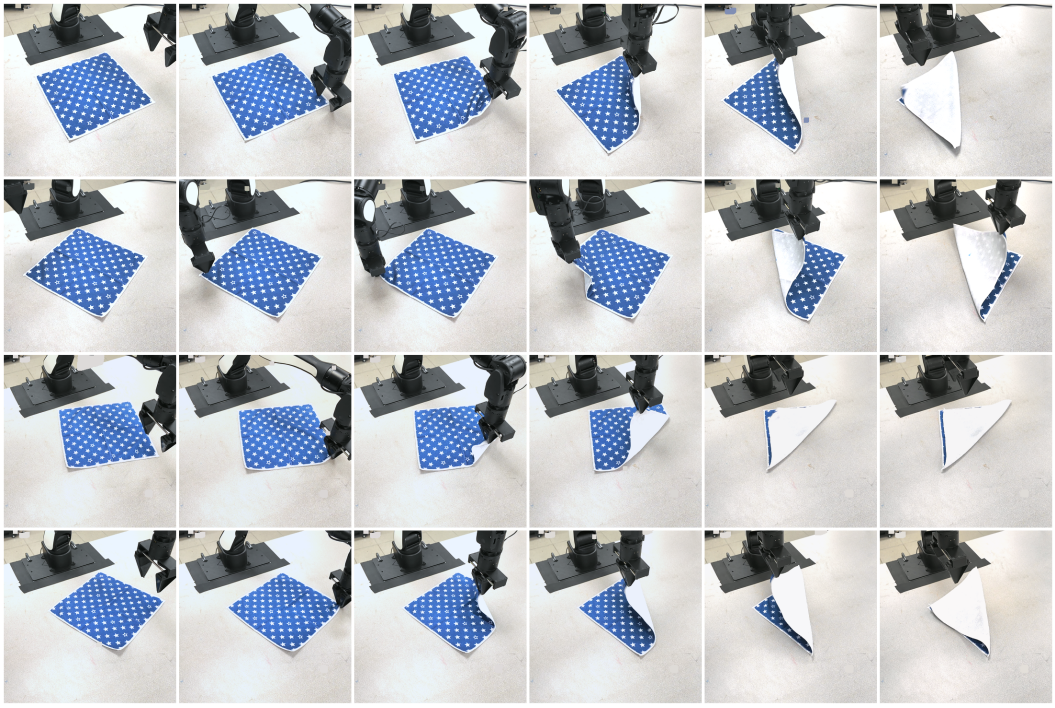


Figure B.3: Full-rollout visualization of successful single-arm diagonal folding trials. Time progresses from left to right.

Robot Platform. We use the AIRBOT Play 6-DoF robot arm in both single-arm and dual-arm configurations. Each arm is commanded through end-effector delta-position. The gripper is a parallel gripper with a binary open and close command. In the dual-arm configuration, the two arms are mounted symmetrically on a common tabletop base. End-effector position commands for both arms are synchronized through a single control loop.

Camera Setup. Visual observations come from a single ZED 2i stereo camera mounted overhead to provide a good view of the manipulation region. Only the left RGB stream is consumed by the real-to-sim module. The camera runs at resolution 1280×720.

Visualizations. Additional full-rollout visualizations of successful real-world trials are provided in Fig. B.3–Fig.B.4.

B.4 Generalization and Robustness

This section provides detailed configurations for the four generalization and robustness settings, together with additional trial visualizations. Each trial randomizes the initial garment pose by drawing

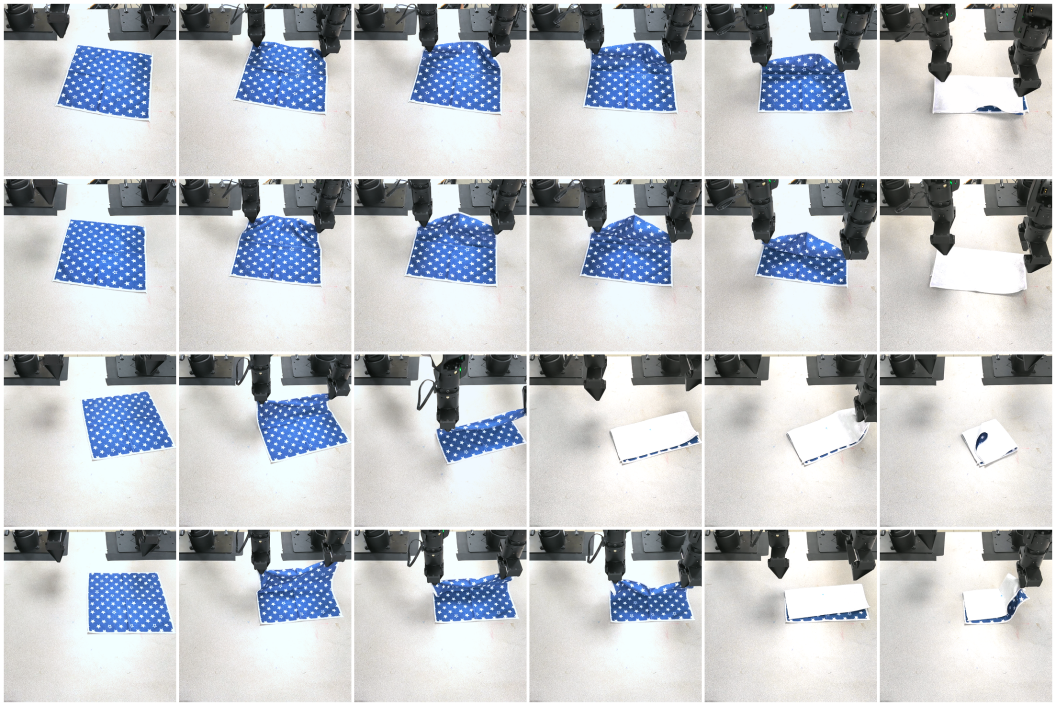


Figure B.4: Full-rollout visualization of successful dual-arm symmetric folding trials. Time progresses from left to right.

a translation uniformly from $[-5, 5]$ cm along the two horizontal axes and a planar rotation from $[-0.5, 0.5]$ rad.

Long-sleeve shirt folding. The task folds both sleeves toward the torso center and then folds the torso along the horizontal midline. Full rollouts are visualized in Fig. B.5.

Shorts folding. The task folds the two legs toward the waistband region and then folds the shorts into a compact final shape. Full rollouts are visualized in Fig. B.6.

Reverse-diagonal towel folding. The task folds the same square towel along the opposite diagonal from the one used in the main single-arm experiment, without retraining. Full rollouts are visualized in Fig. B.7.

Mid-fold disturbance recovery. The task tests whether the system can recover when the towel is manually displaced during the folding process and continue folding from the new state. Full rollouts are visualized in Fig. B.8.

C Real-to-Sim Augmentation Ablation

This section ablates the effect of training-time data augmentation for the RGB-native real-to-sim module. We train the same model with and without the input- and latent-level randomization, and evaluate reconstruction accuracy and real-world closed-loop success. Table C.5 shows that training-time augmentation has little effect on reconstruction accuracy on the synthetic test set, but becomes important during real-world deployment. Without augmentation, the estimator is less robust to real-image noise, visual-domain shift, and robot-arm or gripper occlusions, leading to poorer state synchronization and lower closed-loop success.

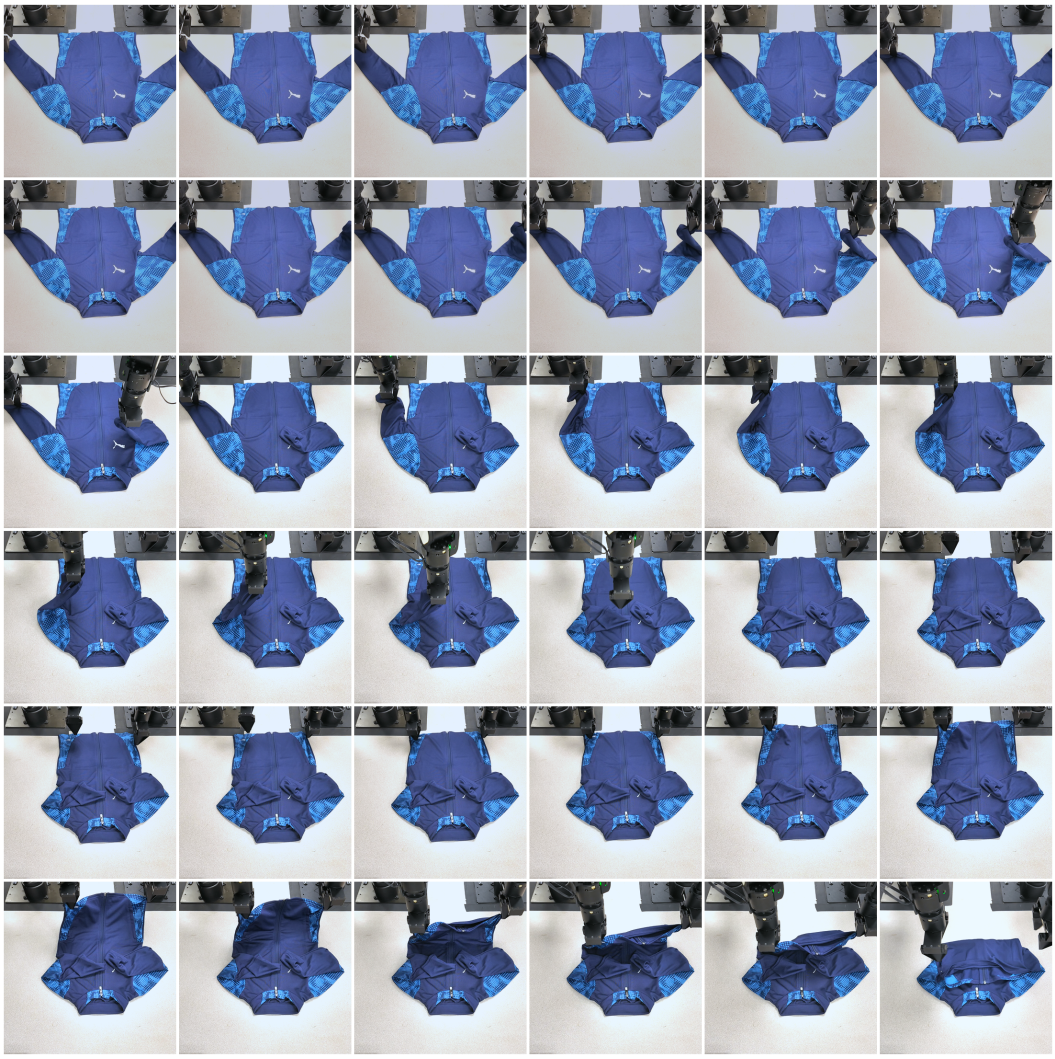


Figure B.5: Full rollout visualization for long-sleeve shirt folding.

Table C.5: **Randomization ablation.** Validation reconstruction metrics and real-world success rate over 10 trials.

Task	R2S Aug	MSE (mm^2)	CD (mm)	EMD (mm)	Real SR (↑)
Single-arm diagonal	on (Ours)	2.33±1.21	1.94±0.41	1.98±0.45	**9/10**
	off	1.85±1.93	1.72±0.47	1.74±0.51	6/10
Dual-arm symmetric	on (Ours)	3.59±2.08	2.42±0.57	2.51±0.68	**8/10**
	off	3.64±2.05	2.37±0.58	2.42±0.60	5/10



Figure B.6: Full rollout visualization for shorts folding.



Figure B.7: Full rollout visualization for reverse-diagonal towel folding.



Figure B.8: Full rollout visualization for mid-fold disturbance recovery.