

Appendices

A Additional Results

A.1 Dynamics Prediction

Quantitative Results We provide additional quantitative results of forward dynamics prediction for both ground truth state and perception noisy input in Table 4 and Table 5. Our dynamics model consistently outperforms all baselines across both evaluation scenarios. When using ground-truth states from the simulator as input, DDM achieves approximately 10× lower error than the best-performing baseline, highlighting the superiority of diffusion models in capturing long-horizon dynamics. When using estimated states from DPM, which introduces additional noise, DDM still achieves 2× lower error than all baselines, demonstrating that the diffusion-based training paradigm significantly enhances noise tolerance through its expressive data distribution modeling capacity.

Type	Method	↓ MSE (10^{-3})	↓ CD (10^{-2})	↓ EMD (10^{-2})
Cloth	GNN	0.75 ± 0.23	3.89 ± 0.80	6.13 ± 2.96
	Transformer	0.61 ± 0.23	1.85 ± 0.33	5.47 ± 1.50
	DDM	0.05 ± 0.04	0.63 ± 0.28	3.47 ± 0.60
T-shirt	GNN	6.36 ± 1.45	8.57 ± 1.06	7.89 ± 1.79
	Transformer	3.22 ± 0.30	2.80 ± 0.23	7.57 ± 0.52
	DDM	0.35 ± 0.13	0.73 ± 0.07	2.84 ± 0.47

Table 4: **Quantitative results on dynamics prediction with ground truth input.** Errors represent a 95% confidence interval.

Type	Method	↓ MSE (10^{-3})	↓ CD (10^{-2})	↓ EMD (10^{-2})
T-shirt	GNN	6.36 ± 1.30	8.88 ± 1.12	8.29 ± 1.94
	Transformer	4.18 ± 0.73	4.26 ± 0.51	7.93 ± 0.70
	DDM	0.55 ± 0.27	1.49 ± 0.13	3.22 ± 0.47
Cloth	GNN	2.17 ± 1.44	5.02 ± 0.90	7.31 ± 4.65
	Transformer	1.30 ± 0.65	2.27 ± 0.46	7.06 ± 2.08
	DDM	0.66 ± 0.45	2.12 ± 0.54	5.51 ± 1.03

Table 5: **Quantitative results on dynamics prediction with perception input.** Errors represent a 95% confidence interval.

Qualitative Results We present additional qualitative results for dynamics prediction in Figure 7 and Figure 8. Each row represents a predicted dynamics sequence. The results demonstrate the physical plausibility of the generated outputs.

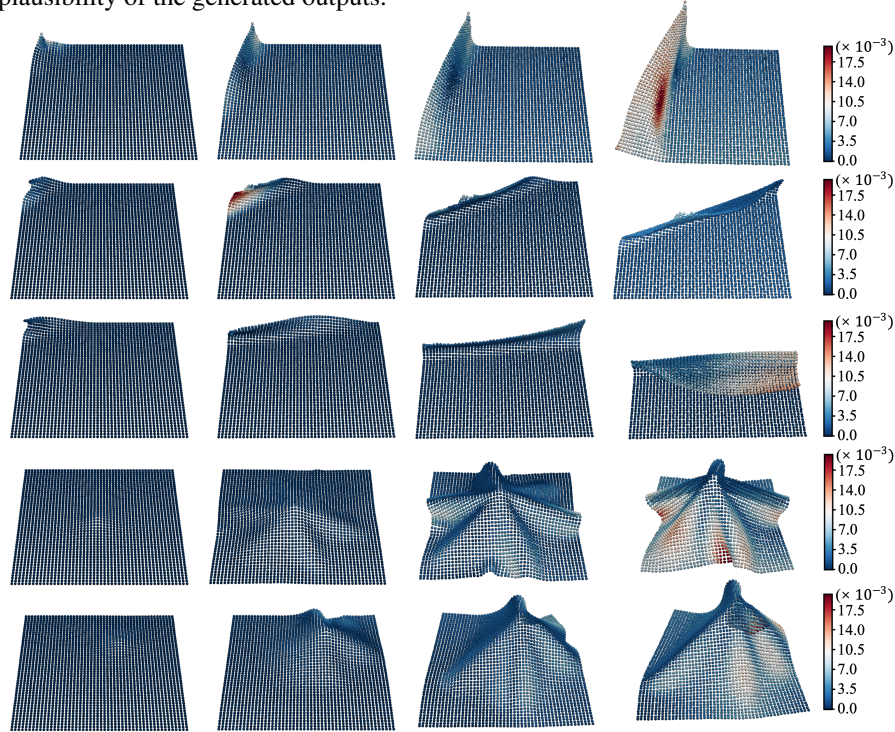


Figure 7: **Qualitative results on cloth dynamics prediction using DDM.**

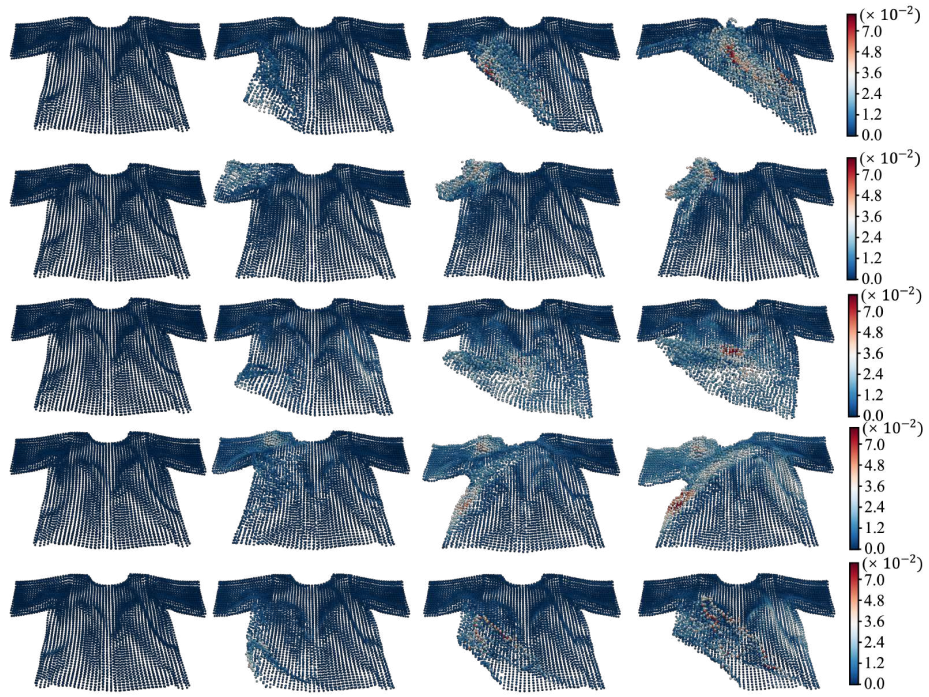


Figure 8: Qualitative results on t-shirt dynamics prediction using DDM.

A.2 Real-world Planning

Additional Quantitative Results We present quantitative results using the EMD metric, which measures the distance from the initial state to the target state in real-world planning scenarios in Figure 9.

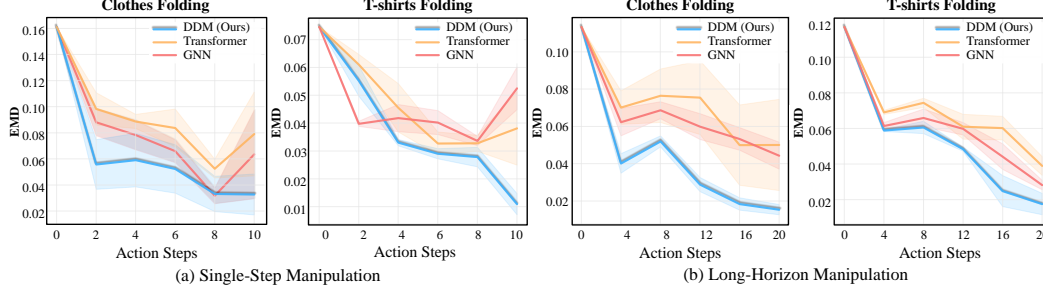


Figure 9: **Quantitative evaluation of planning performance.** Earth Mover’s Distance (EMD) convergence during the planning stage, measured over 10 repeated trials with identical initial and target configurations. Errors represent 95% confidence intervals. Our method outperforms baselines by achieving lower EMD values and faster convergence to goal states.

In single-step manipulation scenarios, our dynamics model exhibits superior performance across both object types. For cloth folding, our method consistently achieves lower EMD values with reduced confidence intervals, indicating enhanced prediction reliability compared to baseline approaches. This performance advantage is particularly evident in T-shirt folding, where topological complexity presents heightened challenges. While baseline methods, especially GNN, exhibit increased variance and elevated EMD values, our approach demonstrates consistent performance improvements throughout the planning horizon, suggesting enhanced handling of complex geometric relationships.

The multi-step scenarios, extending to 20 steps, further highlight our method’s efficacy in long-horizon predictions. Our approach maintains significantly reduced EMD values with a consistent downward trajectory for both cloth and T-shirt manipulation tasks. The performance gap between our method and baselines becomes increasingly pronounced over extended horizons, particularly in T-shirt manipulation, where dual-layer structures introduce additional complexity. This sustained performance advantage in multi-step scenarios underscores our model’s robust capability in mitigating error accumulation while maintaining prediction accuracy across extended planning sequences.

Additional Qualitative Results Accordingly, we also provide additional qualitative results for single step and multi step scenarios on clothes and T-shirts in Figure 10. The results validate our system’s capability to accurately manipulate diverse fabric items from arbitrary initial configurations to challenging target folding states.

Additional Intra-class Generalization Results We provide additional qualitative results for intra-class generalization on square cloth object in Figure 11 with sizes ranging from 20 cm to 40 cm. Our model successfully executes precise folding trajectories across these variations, consistently achieving target configurations and confirming robust intra-class generalization.

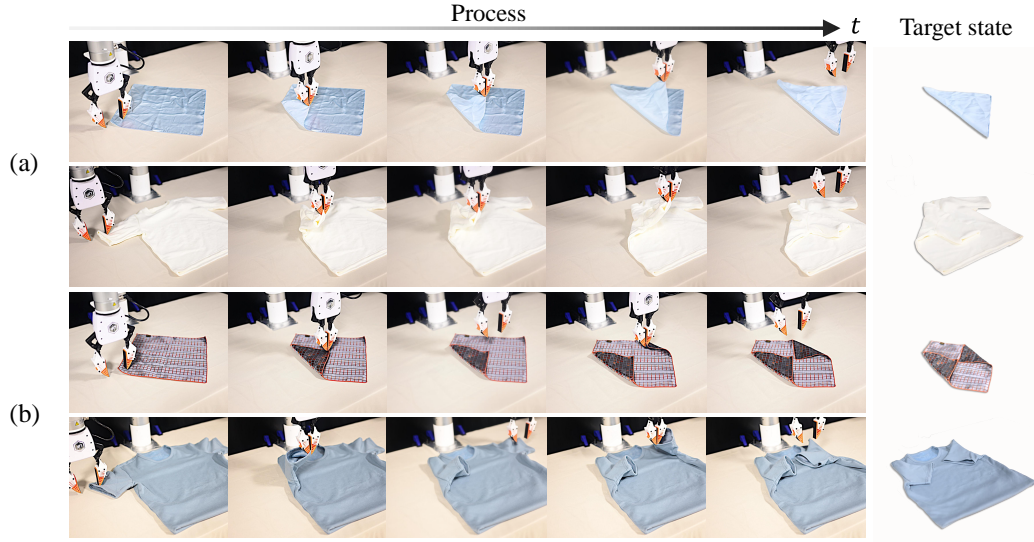


Figure 10: **Qualitative results of real-world system deployment.** The target state is represented in the last column of each row. Each experimental sequence illustrates the progressive deformation states during folding tasks. The first two rows correspond to single-step scenarios, while the last two represent multi-step scenarios.

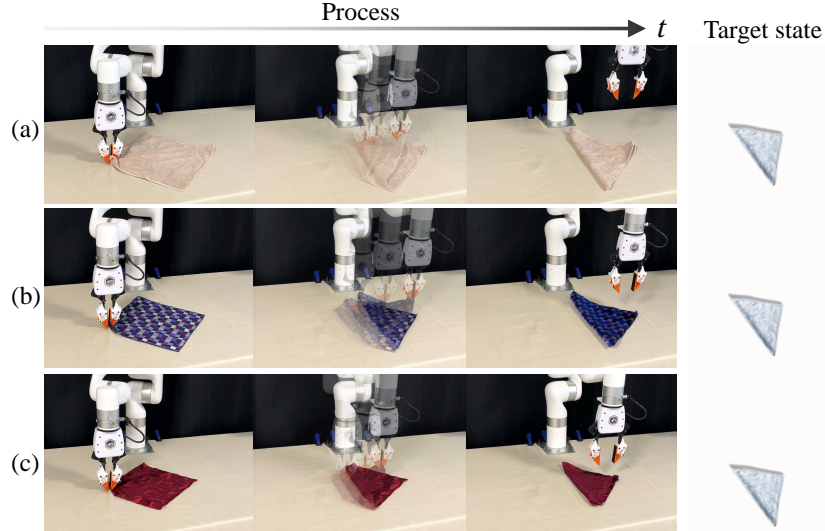


Figure 11: **Intra-class generalization evaluation.** We demonstrate that our method can generalize across garments with varying physical attributes (size, material, and color). The garment size progressively decreases from (a) to (c).

459 A.3 Simulation Planning Results

460 We present more qualitative results in Figure 13 in the simulation environment on planning. We
 461 design four simple tasks in simulation for system validation as visualized in Figure 12.

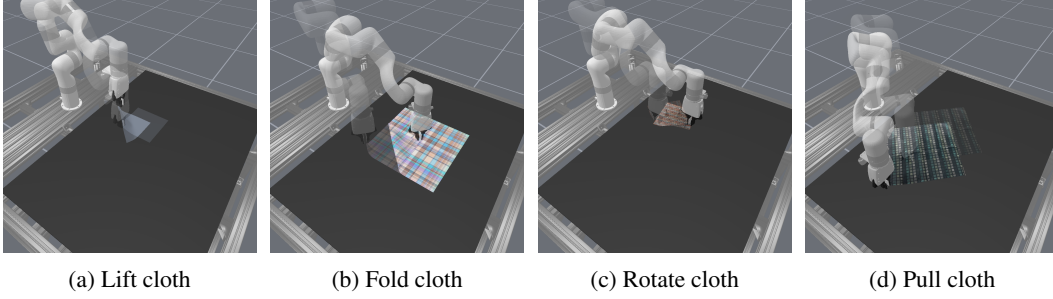


Figure 12: **Simulated cloth manipulation environments.** Visualization of diverse manipulation scenarios in simulation: (a)-(d) demonstrate different cloth-robot interactions with varied object configurations and manipulation tasks.

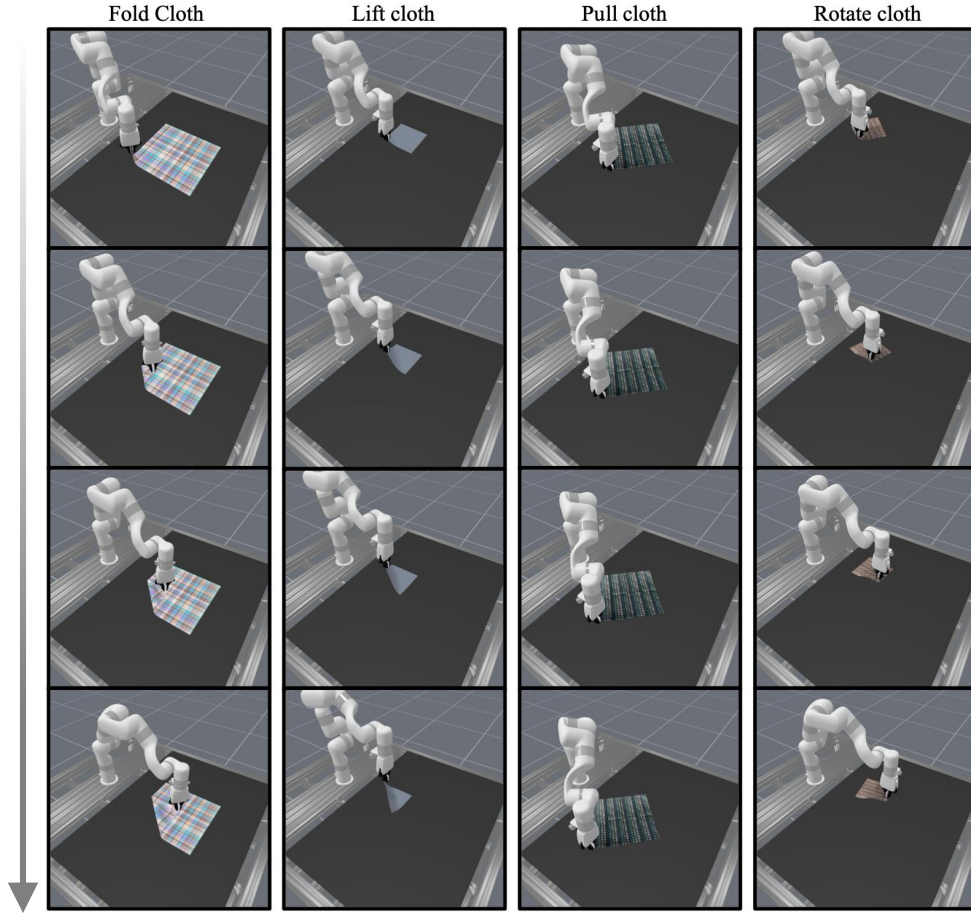


Figure 13: **Model predictive control evaluation in simulation.** Demonstration of our diffusion-based dynamics model integrated with MPC across diverse manipulation tasks using xArm7, validated on various cloth types.

462

463

464
465
466
467
468
469

470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488

489
490
491
492
493
494
495
496
497
498
499
500
501

502

503

504
505
506
507
508
509

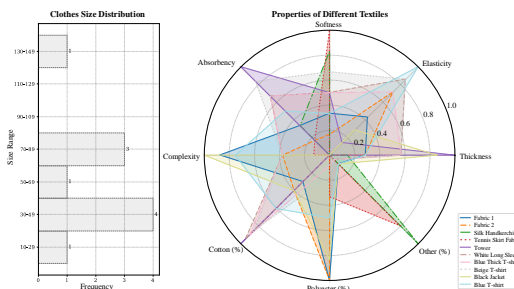


Figure 14: **Coverage of different clothes in our experiments.**

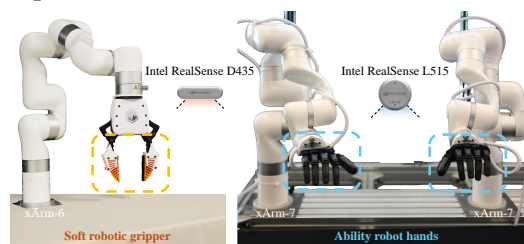


Figure 15: **Hardware overview.** Our real-world platform includes a UFactory xArm-6 and a bi-manual dexterous system consisting of two UFactory xArm-7 robots with Ability hands. Each robot is equipped with one RGB-D camera.

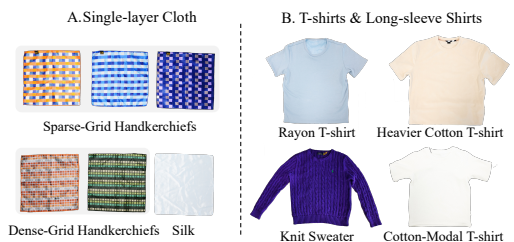


Figure 16: **Cloth overview.** We evaluate our method on different single-layer clothes as well as dual-layer T-shirts and long-sleeve shirts with varying colors and materials.

500

501

502

C Implementation Details

C.1 Data Collection

We collect training data for learning state estimation and dynamics prediction in a simulation environment built on SAPIEN [39]. The rigid bodies, such as the robot arm, are simulated using the built-in PhysX-based simulator, while the cloth is simulated with the projective dynamics (PD) solver [40]. The two systems are coupled at the time step level by alternating updates: the PD system treats the positions and velocities of PhysX-managed objects as boundary conditions, and PhysX does the same for the PD-managed cloth. In the PD system, the cloth is modeled as a hyper-elastic thin shell. We follow Ly et al. [41] to simulate collision and friction in the PD system. We provide detailed physical parameters for cloth simulation in Table 6.

To collect state estimation data, we set up a comprehensive multi-view system that incorporates up to four calibrated stereo-depth sensors, strategically placed at randomized viewing angles within predefined ranges. Cloth is initialized given a randomly "pick-and-place" action. This configuration enables the generation of paired datasets consisting of fused point clouds alongside their corresponding ground-truth mesh states across multiple viewpoints. The system leverages SAPIEN's advanced stereo depth simulation capabilities[42], which significantly reduces the sim-to-real gap by faithfully reproducing point cloud characteristics observed in real-world scenarios. This high-fidelity simulation approach ensures robust and reliable state estimation performance when transferred to physical environments. In our point cloud fusion process, we augment camera extrinsic parameters to simulate real-world calibration errors. Specifically, we introduce rotational variations ranging from -1.5° to 1.5° and translational variations from -0.5 to 0.5 cm. To better mimic real-world conditions, we also simulate depth sensor noise and occlusion effects by applying random point dropout with ratios between 0.1 and 0.2, and introducing noise to the fused point cloud.

Physical Parameter	Value
collision margin	1e-3
collision weight	5e3
collision sphere radius	8e-3
damping	1e-2
thickness	1e-3
density	1e3
stretch stiffness	1e3
bend stiffness	1e-3
friction	0.5
gravity	-9.81

Table 6: Simulation physical parameters.

To collect dynamic data, we employ diverse action sampling strategies to generate a comprehensive dataset of 500K examples. Our sampling approach encompasses two key methodologies designed to capture realistic cloth manipulation scenarios. The first method involves applying directionally-randomized displacements to selected mesh vertices, with particular emphasis on folding-oriented actions where the cloth is manipulated to create various folding patterns. We also simulate picking and relocation actions by applying upward and translational movements to randomly selected vertices. The second methodology focuses on pair-wise vertex manipulation, where vertex pairs are selected based on their spatial distances to simulate actions such as folding one point of the cloth onto another. Each incremental action is precisely controlled, with magnitudes ranging from 0.02 to 0.05 units. To evaluate the model's performance across different time horizons and assess the impact of auto-regressive inference error accumulation, we generate action sequences varying in length from 15 to 35 steps. All resultant mesh deformations throughout these sequences are meticulously recorded to capture the complete dynamics of the cloth's behavior.

C.2 Model Details

Point Cloud Encoder We employ a patch-based architecture for point cloud encoding that processes the input through local grouping and feature extraction. The encoder first groups points using a KNN-based strategy, then processes each local patch through a specialized patch encoder, and finally incorporates positional information through learnable embeddings. This design enables effective capture of both local geometric structures and global spatial relationships.

Model Architecture We design a transformer-based architecture for state estimation, which consists of a point cloud encoder, a positional embedding module, and a series of transformer blocks.

Hyperparameter	Value
Output dimension	1024
Number of groups	256
Group size	64
Group radius	0.15
Position embedding dimension	128
Patch encoder hidden dims	[128, 512]

Table 7: Point cloud encoder hyperparameters.

The model takes both point cloud observations and mesh states as input. The point cloud is first processed through a patch-based encoder, while the mesh states are embedded using a patchified positional encoding scheme. These features are then processed through transformer blocks with cross-attention mechanisms to predict the mesh state.

Hyperparameter	Value
Number of attention heads	16
Attention head dimension	88
Number of transformer layers	4
Inner dimension	1408
Dropout	0.0
Cross attention dimension	1024
Point cloud embedding dimension	1024
Number of input frames	2
Number of output frames	1
Activation function	GELU
Output MLP dimensions	[512, 256]
Normalization type	AdaLayerNorm
Normalization epsilon	1e-5

Table 8: Model hyperparameters.

Action Embedding We employ a Fourier feature-based action encoding scheme to effectively represent mesh manipulation actions in a high-dimensional space. The action encoder consists of two main components: (1) a Fourier feature mapping that projects 3D action vectors into a higher-dimensional space using sinusoidal functions, and (2) a multi-layer perceptron that further transforms these features into the desired embedding dimension.

The Fourier feature mapping applies frequency-based encoding separately to each spatial dimension (A_x, A_y, A_z) of the action vectors using both sine and cosine functions, resulting in an intermediate representation of dimension $2 \times 3 \times F$, where F is the number of Fourier frequencies. Given the input action $a \in \mathbb{R}^{B \times N \times 3}$, where N is the number of actions and 3 represents the dimension of (x, y, z) coordinates, we compute the embedding $e \in \mathbb{R}^{B \times N \times D_3}$ as:

$$\mathbf{e}_{b,n,d} = \begin{bmatrix} \sin(2\pi f_d a_{b,n,x}) & \cos(2\pi f_d a_{b,n,x}) \\ \sin(2\pi f_d a_{b,n,y}) & \cos(2\pi f_d a_{b,n,y}) \\ \sin(2\pi f_d a_{b,n,z}) & \cos(2\pi f_d a_{b,n,z}) \end{bmatrix} \quad (2)$$

where D_3 is the action embedding dimension, $d \in \{0, \dots, D_3/6 - 1\}$, and $f_d = 100^{d/(D_3/6)}$ are the Fourier feature frequencies. The resulting embedding e provides a rich, high-dimensional representation of the action space. This representation is then processed through an MLP to produce the final action embeddings, which is later injected as the condition into our model through cross-attention layers.

Hyperparameter	Value
Fourier frequencies	8
Fourier feature dimension	48
MLP hidden dimensions	[512, 512]
Output dimension	output_dim
Activation function	SiLU
Position normalization	Center & Scale

Table 9: Action encoder hyperparameters.



Figure 17: **Example training data.**

583 C.3 Training Details

584 We train our model using distributed data parallel training on 4 H100 GPUs. The model is trained
 585 with a batch size of 128 per GPU and gradient accumulation steps of 4, resulting in an effective
 586 batch size of 2048. We use the AdamW optimizer with a learning rate of 1e-5 and cosine learning
 587 rate scheduler with 1000 warmup steps. For numerical stability and training efficiency, we employ
 588 mixed-precision training with bfloat16 and enable TF32 on supported hardware.

589 C.4 Planning Details

590 For planning, we employ a hybrid approach combin-
 591 ing Model Predictive Control (MPC) and Cross Entropy
 592 Method (CEM). Our planner optimizes action sequences
 593 by iteratively sampling actions, evaluating their outcomes
 594 using the learned dynamics model, and updating the sam-
 595 pling distribution based on the costs. To enhance plan-
 596 ning efficiency, we introduce two key strategies: (1) an in-
 597 formed action sampling mechanism and (2) a grasp point
 598 selection method. For action sampling, we initialize the
 599 sampling distribution using a prior direction informed by
 600 the target state. Specifically, we identify the K vertices
 601 with the highest mean squared error (MSE) between the
 602 current and target states, and compute a weighted average
 603 direction based on their distances to the grasp point:

Hyperparameter	Value
Number of GPUs	4
Batch size per GPU	128
Gradient accumulation steps	4
Effective batch size	1024
Learning rate	1e-5
Learning rate scheduler	Cosine
Warmup steps	1000
Mixed precision	bfloat16
Number of workers	16

Table 10: Training hyperparameters.

$$d_{main} = \sum_{i=1}^K w_i (s_t^i - s_c^i), \quad w_i = \frac{1}{\|p_g - p_i\| + \epsilon} \quad (3)$$

where s_t^i and s_c^i are target and current states of vertex i , p_g is the grasp point position, and p_i is the position of vertex i . This informed direction guides the initial sampling distribution for more efficient exploration.

For grasp point selection, we employ a temperature-controlled softmax strategy based on vertex displacements. Given the current state S_c and target state S_t , we compute a probability distribution over all vertices:

$$p(i) = \frac{\exp(\|s_t^i - s_c^i\|_2/\tau)}{\sum_j \exp(\|s_t^j - s_c^j\|_2/\tau)} \quad (4)$$

where s_t^i and s_c^i represent the position of vertex i in target and current states, respectively, and τ is a temperature parameter that controls the concentration of the probability distribution. A lower temperature leads to more deterministic selection focusing on maximum displacement vertices, while a higher temperature enables more exploratory behavior. The grasp point is then sampled from this distribution:

$$g \sim p(i) \quad (5)$$

This probabilistic selection mechanism provides several advantages over deterministic maximum displacement selection: (1) it allows for exploration of different grasp points, (2) it can adapt to different manipulation scenarios by adjusting the temperature parameter, and (3) it provides a smoother transition between different grasp point candidates. The planning algorithm is outlined in Algorithm 1. Hyperparameters for model-based planning are listed in Table 11.

C.5 Baseline Implementation

We introduce details of the baseline implementation.

GNNs We adopt the implementation from [6]. We construct a comprehensive graph representation for modeling cloth dynamics, incorporating object particles, end-effector interactions, and material properties. The graph structure consists of four main components: (1) state and action representations, (2) particle attributes and instance information, (3) relation matrices for particle interactions, and (4) material-specific physics parameters. The state representation captures both spatial positions and temporal dynamics through a history buffer of n_{his} frames and future predictions of n_{future} frames. Each state vector contains the 3D positions (x, y, z) of both cloth particles and the end-effector. We maintain a fixed-size particle set through Farthest Point Sampling (FPS) with an adaptive radius range of $[0.05, 0.1]$. We show detailed parameters for graph construction in Table 12.

Parameter	Value
Number of iterations	5
Samples per iteration	16
Sequence length	5
Action dimension	3
Initial std deviation	0.1
Temperature	1.0

Table 11: Planning hyperparameters.

Hyperparameter	Value
Maximum particles (N_{obj})	100
Maximum relations (N_R)	1000
History frames (n_{his})	3
Future frames (n_{future})	5
State dimension	3
Attribute dimension	2
FPS radius range	$[0.05, 0.1]$
Adjacency radius range	$[0.74, 0.76]$
Topk neighbors	5

Table 12: GNN model hyperparameters.

C.6 Manipulation Pipeline Details

Our system integrates OWLV2 [43] and Segment Anything [44] to detect and segment desktop objects from RGB-D input. A single-view partial point cloud of the target object serves as input, which is processed via DPM to infer the state of the cloth. To address the dimensional and positional

645 discrepancies between predicted and observed point clouds, we implement a two-stage alignment
 646 process. First, we compute the spatial dimensions of the observed point cloud and apply appropriate
 647 scaling transformations to the predicted point cloud. Subsequently, we employ the Iterative Closest
 648 Point (ICP) algorithm for fine-grained alignment, ensuring that MPC-generated grasping positions
 649 and motion trajectories can be accurately mapped to the physical object. For manipulation, we
 650 model both soft robotic grippers and dexterous hands by representing their end effectors as particles
 651 that attach to mesh vertices during motion. To evaluate our system, we first collect realistic and
 652 challenging target states through teleoperation. We then conduct 10 experimental trials for the same
 653 target state, executing a delta action sequence through the MPC with the dynamics model. These
 654 actions are transformed into absolute positions in the base frame of the robotic arm, with smooth
 655 Cartesian trajectories generated using joint online trajectory planning.

Algorithm 1 MPC Planning Algorithm

Require: Initial state s_i , target state s_t , dynamics model f_θ , number of iterations N

Require: Number of samples K , sequence length L , action bounds $[a_{min}, a_{max}]$

```

1: Initialize  $\mu \leftarrow \mathbf{0}$ ,  $\sigma \leftarrow 0.1$ 
2:  $a_{best} \leftarrow \text{None}$ ,  $c_{best} \leftarrow \infty$ 
3: for  $i = 1$  to  $N$  do
4:    $A_{mppi} \leftarrow \text{SampleGaussian}(K/2, L, \mu, \sigma, [a_{min}, a_{max}])$ 
5:    $A_{uniform} \leftarrow \text{SampleUniform}(K/2, L, [a_{min}, a_{max}])$ 
6:    $A \leftarrow \text{Concatenate}(A_{mppi}, A_{uniform})$ 
7:    $S_{pred} \leftarrow f_\theta(S, A)$  ▷ Predict trajectories
8:    $C \leftarrow \text{ComputeCost}(S_{pred}, A, T)$  ▷ Evaluate costs
9:   if  $\min(C) < c_{best}$  then
10:     $c_{best} \leftarrow \min(C)$ 
11:     $a_{best} \leftarrow A[\arg \min(C)]$ 
12:   end if
13:    $\mu, \sigma \leftarrow \text{UpdateDistribution}(A, C, \tau)$  ▷ Update using weighted averaging
14:    $\sigma \leftarrow \sigma \cdot (1 - i/N)$  ▷ Anneal exploration
15: end for
16: return  $a_{best}$ 

```
