Mesh-based Dynamics with Occlusion Reasoning for Cloth Manipulation

Zixuan Huang, Xingyu Lin, David Held
Carnegie Mellon University, Pittsburgh PA 15213, USA
{zixuanhu, xlin3,

***Abstract*—Self-occlusion is challenging for cloth manipulation, as it makes it difficult to estimate the state of the cloth. Ideally, a robot trying to unfold crumpled or folded cloth should be able to reason about the cloth's occluded regions. We leverage recent advances in pose estimation for cloth to build a system that uses explicit occlusion reasoning to unfold a crumpled cloth. Specifically, we first learn a model to reconstruct the mesh of the cloth. However, the model will likely have errors due to the complexities of the cloth configurations and due to ambiguities from occlusions. Our main insight is that we can further refine the predicted reconstruction by performing test-time finetuning with self-supervised losses. The obtained reconstructed mesh allows us to use a mesh-based dynamics model for planning while reasoning about occlusions. We evaluate our system both on cloth flattening as well as on cloth canonicalization, in which the objective is to manipulate the cloth into a canonical pose. Our experiments show that our method significantly outperforms prior methods that do not explicitly account for occlusions or perform test-time optimization. Videos and visualizations can be found on our project website.**

I. Introduction

Manipulation of clothing has wide applications but remains a challenge in robotics. Cloth has nearly infinite degrees of freedom (DoF), making state estimation difficult and resulting in complex dynamics. Furthermore, cloth is often subject to severe self-occlusions, especially when it is crumpled.

Due to self-occlusions, prior methods typically rely on visible features for manipulation, such as wrinkles [1] or corners [2]. Data-driven methods have been proposed that can learn a dynamics model in the pixel space [3] or in a latent space [4]. Recently, Lin, Wang, *et al.* [5] proposed to learn a mesh dynamics model on the observed partial point cloud of a crumpled cloth (VCD). However, VCD only considers a graph over the visible points and does not reason explicitly about the occluded regions of the cloth. This simplification sometimes prevents the planner from finding optimal cloth unfolding actions.

Some prior work [6, 7, 8, 9, 10, 11, 12] attempt to reconstruct the full cloth structure. To simplify the task, these approaches typically first grasp and lift the cloth in order to limit the diversity of possible cloth configurations. However, this approach prevents continual state estimation throughout a manipulation task, since the pose can only be estimated while being held in the air by the gripper at a single point. In contrast, we aim to estimate the cloth pose from a more diverse set of poses, such as while it is crumpled on a table in arbitrary configurations.

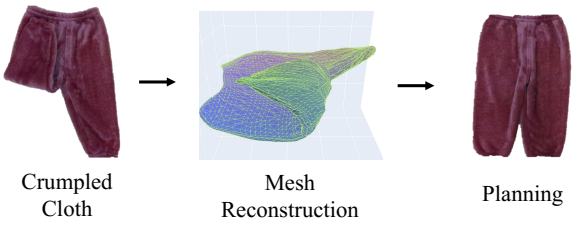


Fig. 1: Our method, MEDOR (MEsh-based Dynamics with Occlusion Reasoning), explicitly reasons about occlusions by reconstructing the full mesh of a cloth; we then use a learned mesh-based dynamics model to plan the robot actions.

In this paper, we propose MEDOR (MEsh-based Dynamics with Occlusion Reasoning) which is built on top of previous methods for mesh-based dynamics learning [5, 13, 14]. MEDOR explicitly reasons about occlusions by reconstructing the full cloth mesh from a single depth image. While previous work such as GarmentNets [12] has demonstrated promising results for mesh reconstruction, we show that, by itself, it is not robust enough for a robot cloth manipulation system. Due to the inherent ambiguity induced by self-occlusions from the crumpled cloth, the reconstructed mesh will likely have many errors, which will create difficulties for downstream planning. Our insight is that we can improve the mesh reconstruction using test-time optimization with self-supervised losses, which can be easily computed without access to the ground-truth mesh and can be directly optimized on real data.

We evaluate our method on two tasks: cloth smoothing and cloth canonicalization, which requires aligning the cloth with a canonical flattened pose. We show that our method of mesh reconstruction, with test-time finetuning, enables a robot to smooth or canonicalize the cloth from crumpled configurations more accurately than previous methods. Our contributions include:

1) a novel perception model that can better estimate the full cloth structure from crumpled configura-

tions, including a self-supervised test-time optimization procedure.
2) a cloth manipulation system that plans over the reconstructed cloth mesh and can perform both cloth flattening and cloth canonicalization.

II. Related works

A. Perception for Cloth Manipulation

There has been a long history of work on cloth perception for manipulation. We refer to Jimenez *et al.* [15] for a comprehensive overview.

Earlier works usually estimate specific visual features of the cloth for manipulation. These include detecting edges and corners for re-grasping [2] or grasping and un-folding [16, 17, 18], or detecting wrinkles for smoothing [1]. If the cloth is loosely extended, simplified models such as the parameterized shape model [19] or the polygonal model [20] can also be used for folding. Other works also detect category-specific features such as collars and hemlines [21]. These features have been used with hand-designed controllers and strategies; in this work, we aim to learn mesh reconstruction and a cloth dynamics model that we can use to plan a cloth manipulation action sequence.

There are also prior works on more generic pose estimation for clothes. One line of works tries to estimate the pose of on-body clothing from video by leveraging a human body shape prior [22, 23, 24, 25, 26, 27, 28]. Another line of works assumes the initial configuration of the cloth is known and uses tracking to estimate the full configuration of the cloth under occlusion [29, 30, 31]. In contrast, we assume that the cloth may be initially crumpled on a table in an unknown initial configuration.

Some other works [6, 7, 8, 9, 10, 11, 12] simplify the problem by lifting up the cloth using the robot gripper for the purpose of easier pose estimation; lifting up the cloth significantly reduces the set of possible poses and simplifies the reconstruction task. After lifting the cloth, Kita *et al.* [6] deform a set of predefined representative shapes to fit the observed data. Other works create a dataset of clothes grasped at different locations and retrieve the observed pose at test-time by classification [7, 8, 10, 11] or using nearest neighbor [9]. In contrast, our method can estimate the cloth directly from a crumpled state on the table, which enables our method to continually re-estimate the cloth state throughout a manipulation sequence.

Recently, GarmentNets [12] performed categorical cloth 3D reconstruction by mapping the cloth point cloud into a normalized canonical space (NOCS) defined for each cloth category [32]. However, like the above methods, GarmentNets also requires grasping and raising the cloth into the air and obtaining four different camera views to reduce the amount of occlusion. In contrast, our perception module only requires a single view of the cloth crumpled on the table. Further, we find that the reconstructions produced by GarmentNets are not sufficient for accurate cloth manipulation. Also, unlike GarmentNets that only considers the perception task, we demonstrate a full cloth manipulation system and show the effectiveness of our perception method for manipulation.

B. Data-driven Methods for Cloth manipulation

Prior works in data-driven cloth manipulation can be categorized as model-free or model-based. Model-based methods train a policy that outputs actions for cloth manipulation. The policies are trained by either reinforcement learning [33, 34], imitation learning [35], or by learning a value function [36]. Alternatively, the policies can be learned as a one-step inverse dynamics model [37, 38].

The second approach is to learn a dynamics model and then plan over the model to find the robot actions. Hoque *et al.* directly learn a video prediction dynamics model [3] and Wilson *et al.* learn a latent dynamics model with contrastive losses [4]. These models do not explicitly reason about the cloth structure, making generalization difficult. Recent work (VCD) trains a mesh dynamics model [5] over the visible points. However, without reasoning about the occluded regions, the planner will often fail to find the optimal smoothing actions. In contrast, our approach plans over a full reconstructed mesh dynamics model.

C. Test-time Optimization

Test-time optimization has been widely used for view-synthesis [39, 40], 3D particle reconstruction [41] and 3D scene flow [42]. For example, Chen et al. [41] trained a network to predict an object point cloud that is consistent with a set of object masks in different camera views. However, these approaches have not been applied to cloth reconstruction or robot manipulation tasks.

III. Background

A. Problem Formulation

We consider the task of manipulating clothes in a planar workspace with a single robot arm. A cloth at time t is represented by a mesh $M^t = (V^t, E^t)$ with vertices $V^t = \{v_i\}_{i=1...N}$ and mesh edges E^t. Each vertex consists of a position x_i and velocity $\dot{x}_i$ that will change with the cloth configuration. The ground-truth configuration of the cloth M^t is unknown, and the robot only observes the RGB-D image I^t, which includes severe self-occlusions for crumpled garments (though our method only uses the color to segment the cloth from the background). Given the camera intrinsics and the segmentation mask, we can also back-project I^t

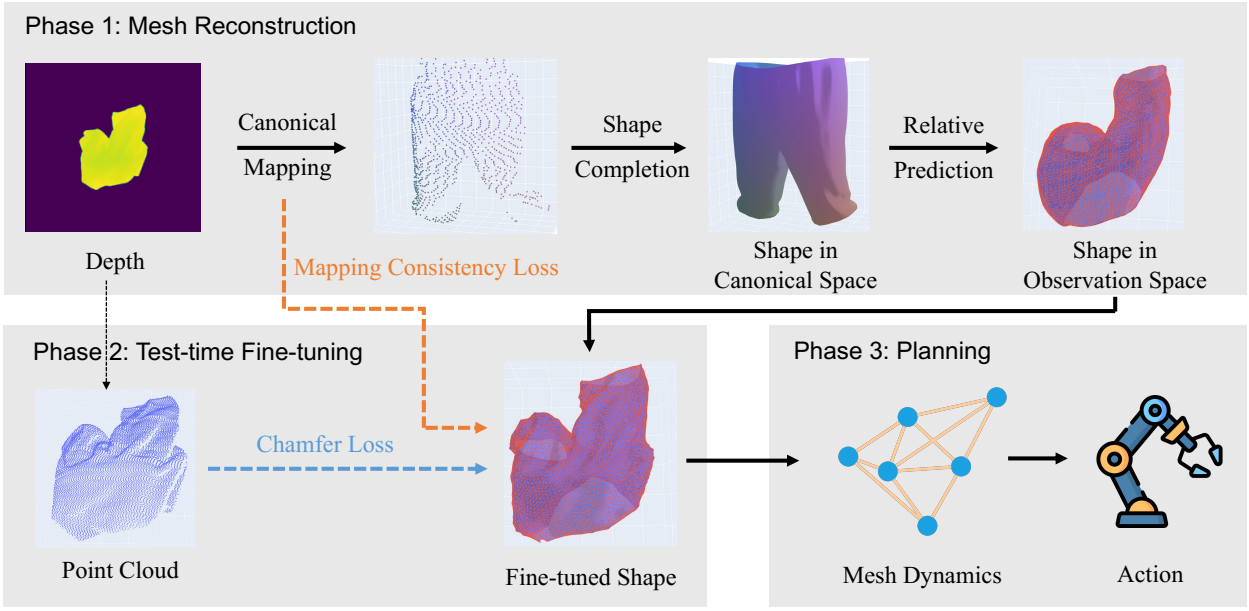


Fig. 2: System overview: First, we obtain an initial estimate of the full shape of an observed instance of a cloth from a depth image. In this phase, we obtain an estimate of the cloth shape in both canonical space and observation space. Then, we conduct test-time finetuning to improve the prediction to better match the observation. Last, we plan with the predicted mesh using a learned mesh-based dynamics model.

to a partial point cloud of the cloth. The observation is captured by a top-down camera mounted on the robot end-effector in our setting. As in prior works [34, 3, 5], we use pick-and-place action primitives for performing cloth manipulation.

B. GarmentNets

GarmentNets [12] is a previous work that performs categorical cloth reconstruction from a partial point cloud. It contains three steps: first, a normalized canonical space (NOCS [32]) is defined for each cloth instance by simulating the cloth worn by a human in a T-pose. A canonicalization network (using a PointNet++ architecture [43]) is trained to map an observed partial point cloud to the canonical space (see Figure 2, top). Second, a 3D CNN performs volumetric feature completion to obtain a dense feature grid. An MLP is then used to predict the winding number [44] for each point, which is used for surface extraction. In the last step, GarmentNets samples points on the extracted surface and trains a warp field network to map each point in the canonical space back to its location in the observation space.

GarmentNets requires a multi-view depth image to reduce occlusions; it also requires the cloth to be grasped in the air to limit the set of possible poses of the cloth. In contrast, we consider the more general setup where the cloth is on the table in an arbitrary configuration.

IV. Approach

Our goal is to build a general cloth manipulation system that can reason about the occluded cloth regions of a cloth from a partial observation and plan robot actions based on these estimates. To do so, we propose a three-part approach: (1) First, given only a partial observation of a crumpled cloth, we train a model to generate a complete mesh of the cloth; (2) Due to the difficulty of occlusion reasoning, the neural network often fails to accurately reconstruct the cloth mesh. Therefore, we design a test-time finetuning scheme to adapt the predicted mesh to match the observation. (3) Last, we plan with the predicted mesh using a learned dynamics model to find optimal actions for the manipulation task.

A. Estimating the pose of a cloth

Reconstruction of the full cloth structure is fundamentally challenging due to the high-dimensional state space and the ambiguity induced by self-occlusion. As discussed in Section III-B, GarmentNets [12] simplifies the problem by using a robot to grasp and lift the cloth and capture 4 observations from different views. In contrast, we tackle the harder problem of estimating the cloth state from a crumpled configuration on the table, so that the cloth state can be re-estimated throughout a manipulation sequence.

To tackle this challenge of cloth pose estimation from crumpled configurations, we make several modifications to GarmentNets that greatly improve its performance:

HRNet: First, since the cloth is not lifted up, we represent the cloth as a depth image captured by a single top-down camera instead of as a point cloud merged from 4 observations. In this case, the Pointnet++ [43]

architecture is no longer able to provide reasonable estimates of the cloth configuration, as we will show. Instead, we use an architecture designed for depth images. In particular, we find that PointNet++ is not able to distinguish whether a piece of cloth is folded above or folded below the rest of the cloth. Differentiating these two cases requires noticing the subtle depth changes at the boundary where two layers meet. Therefore, we replace Pointnet++ with a High Resolution Network (HRNet) [45] which is a convolutional architecture that specializes in producing a high-resolution and spatially precise representation.

Relative predictions: We found that the reconstruction model sometimes inaccurately estimates the configuration of the cloth in observation space, but it often predicts the canonical shape reasonably well (see Figure 2 for a visualization of these two spaces). Thus, we learn to predict the delta between the position of a point in canonical space and its corresponding location in observation space. Specifically, given the predicted mesh in canonical space, $\tilde{M}^c = (\tilde{V}^c, \tilde{E}^c), \tilde{V}^c = \{\tilde{v}_i\}_{i=1...n}$, where the coordinate of each vertex in canonical space is $\tilde{x}_i^c$, we predict a 3-dimensional residual vector $\tilde{f}_i$ for each point i. The predicted coordinates in the observation space $\tilde{x}_i^o$ are obtained by

$$\tilde{x}_i^o = \tilde{x}_i^c + \tilde{f}_i \tag{1}$$

As shown in the ablation experiments in Table I, both modifications described above are important for the performance of the method.

B. Test-time finetuning

Estimating the complete structure of cloth is inherently challenging due to the high degrees of freedom and the ambiguity induced by occlusion. As a result, the network described above still has significant prediction errors, as shown in Fig. 3.

To tackle this issue, we design a test-time finetuning scheme that further optimizes the predicted mesh using self-supervised losses that can be computed without knowledge of the ground-truth cloth state. We deform the predicted mesh by optimizing the location of each vertex to optimize an objective consisting of the sum of two loss terms: unidirectional Chamfer loss and mapping consistency loss.

Unidirectional Chamfer loss. The first self-supervised loss term penalizes any deviations between the predicted mesh and the observed cloth surface (depth image) so that geometric details are preserved. Since the observation only contains information about the visible surface, optimizing the mesh with a standard bi-directional Chamfer loss will result in undesirable results, i.e., the predicted mesh will move entirely to the visible surface and no part of the mesh will remain in the occluded region. Therefore, we use a unidirectional Chamfer loss as described below.

Suppose at timestep t, the point cloud observation of the cloth is $P^t = \{p_i\}_{i=1..L}$ and the predicted mesh in observation space is $\tilde{M}^t = (\tilde{V}^t, \tilde{E}^t)$. The coordinate of each vertex is specified by a 3-dimensional vector. Then the loss term is formulated as:

$$\mathcal{L}_{\mathcal{C}}(\tilde{V}^t; P^t) = \frac{1}{|P^t|} \sum_{p_i \in P^t} \min_{\tilde{v}_j \in \tilde{V}^t} d(p_i, \tilde{v}_j) \tag{2}$$

where $d(\cdot, \cdot)$ can be any distance metric, and we use Euclidean distance. In other words, for each point in the observed point cloud, we find the distance to the nearest point in the predicted mesh and minimize the sum of such distances.

Mapping consistency loss. Since we don't know the exact correspondence between the predicted mesh and the partial point cloud, only optimizing the uni-directional Chamfer loss above may lead to a local minimum. To alleviate this issue, we observe that the mapping from the observation space to canonical space and then back to observation space (see Figure 2, top) creates a cycle; thus we add a loss that, for each visible point, this cycle should end at the location where it started.

Let P^t be the point cloud observation of the cloth; let f be the learned mapping from each observation point to a location in the canonical space; and let g be a learned mapping from each location in the canonical space to a position back in the observation space (shown in Figure 2, top). Note that g operates on the predicted completed cloth surface which includes both observed and occluded points. The mapping consistency loss term can be expressed as:

$$\mathcal{L}_{\mathcal{M}}(P^t) = \frac{1}{|P^t|} \sum_{p_i \in P^t} d(g(f(p_i)), p_i) \tag{3}$$

In other words, this loss penalizes the distance between the original location of each observed point p_i and its predicted location $g(f(p_i))$.

Optimization. We use gradient descent with the Adam optimizer [46] to optimize the losses above. The optimization is divided into two phases. In the first 50 steps, we optimize the mesh using the Chamfer loss together with mapping consistency loss; then we optimize the mesh using the Chamfer loss alone for another 50 iterations.

We do not perform a joint optimization throughout the optimization process because multiple pixels in the depth image might be mapped to the same voxel in the canonical space. Enforcing mapping consistency throughout the optimization process will create implausible meshes whose vertices converge to a set of clusters. Instead, we use the mapping consistency loss to provide a good initialization (beyond the initial network prediction), and

then we use the uni-directional Chamfer loss to refine th geometric details. As shown in the ablation in Table I, th two-stage optimization is critical for the performance.

C. Planning with GNN-based dynamics model

Once we have reconstructed the cloth mesh, we us the reconstructed mesh to plan the robot actions usin a dynamics model. One option would be to use physical simulator as a dynamics model. However, our experiments show that using a learned dynamics model can be more accurate and faster than a simulator when planning with a reconstructed mesh; these results (shown in Sec. V-C) are consistent with previous papers [5]. We suspect that a learned dynamics model can be more robust to errors in the mesh reconstruction compared to a physics-based simulator.

As a first step towards learning a dynamics model, we downsample the mesh by vertex clustering [47] for faster computation. We apply vertex clustering in the canonical space (defined as the pose of the cloth when worn by a human in a T-pose) because the cloth surfaces are well separated and thus will avoid undesirable artifacts such as merging different layers.

Given a down-sampled mesh from the simulator $\bar{M}^t = (\bar{V}^t, \bar{E}^t)$, we train a dynamics Graph Neural Network (GNN) [14, 13]. A GNN encodes the input feature on the nodes and on the edges and then conducts multiple message passing steps between the nodes and edges [48]. The decoder will decode the latent features of each node into the predicted acceleration for that node. The GNN dynamics model that we use is the same as the dynamics model in VCD [5]: the input feature on each node is the historic particle velocities and an indicator of whether the node is picked by the gripper or not. The edge features includes the distance vector of connected vertices $(x_j - x_k)$, its norm $||x_j - x_k||$, and the current displacement from the rest position $||x_j - x_k|| - r_{jk}$. We use Euler integration to obtain the states of the cloth in the next time step. The action is encoded to the dynamics model by directly modifying the position and velocity of the grasped point on the cloth. We refer the reader to previous work [5] for details on the graph dynamics model. Once we have the dynamics model, we use random shooting to plan over different pick-and-place actions and pick the action with the highest predicted reward. At each time step, we plan over a horizon of one, i.e., one pick-and-place action. We train a single dynamics model on Trousers and then apply it to multiple categories in our experiments.

D. Implementation details

All models are trained in simulation and data are generated by Nvidia Flex wrapped in Softgym [49]. To obtain a diverse dataset with garments of different sizes and shapes, we port the CLOTH3D dataset [50] into Softgym. We choose five categories of garments from CLOTH3D: Trousers, T-shirts, Dress, Skirt and Jumpsuit. Each category contains 400-2000 different meshes and covers a wide range of variations, such as shirts with short and long sleeves, with and without an opening in front. We divide the CLOTH3D dataset into a train set and test set in a 9:1 ratio.

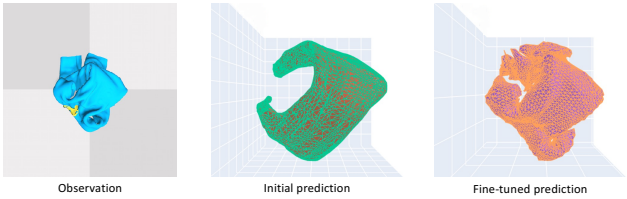


Fig. 3: An example showing the benefits of test-time finetuning: The network is able to correctly estimate the main cloth structure, but the initial predicted mesh may be overly smooth (center). Test-time finetuning significantly improves the quality of predicted mesh (right). For more examples, see our project website.

a) Mesh reconstruction model: Initial crumpled cloth configurations are generated by a random drop or random pick-and-place actions from flattened states, with a ratio of 1:1. The mesh reconstruction model is trained in a category-dependent manner, i.e. one mesh reconstruction model per category, similar to GarmentNets [12]. For each category, the training set contains 20,000 observations.

b) Dynamics model: The GNN dynamics model is trained only on Trousers and is evaluated across all object categories. The dynamics model is trained on a dataset with 5,000 pick-and-place actions, which equals 500,000 intermediate timesteps. The training of the mesh reconstruction model and the dynamics model each take around 3 days.

At test-time, the mesh reconstruction model and test-time finetuning take around 2.5 and 3 seconds per mesh respectively. For planning, we randomly sample 500 pick-and-place actions and rollout each action with the GNN dynamics model, which takes around 100 seconds. For additional implementation details, please refer to the supplementary materials.

V. Experiments

To evaluate the effectiveness of our method, we conduct comprehensive experiments on the tasks of cloth flattening and canonicalization (described below). To demonstrate the generalizability and robustness of our method, we evaluate in both the real world and in simulation on 5 different categories of garments. Through the experiments, we would like to answer the following questions:

1) Does explicit occlusion reasoning improve the performance of cloth manipulation? How does

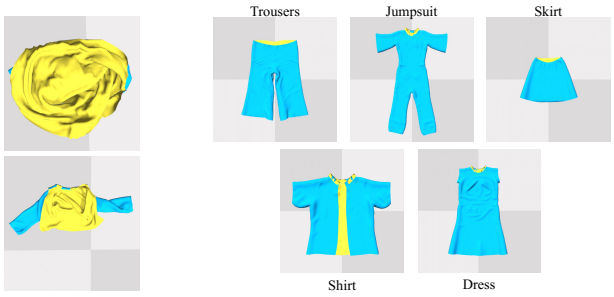


Fig. 4: Flattening (i.e. maximizing the covered area of the cloth) may not always give us a good starting point for folding. Left: Undesirable results of actions that optimize for the flattening task (maximizing the coverage). Right: Examplar goal poses for canonicalization of each category.

it compare to methods that operate only on the visible points?

2) Does test-time finetuning improve the quality of predicted mesh as well as the performance in cloth manipulation tasks?
3) Can our method work on a physical robot?

A. Tasks

Flattening. Our goal is to flatten a crumpled cloth, that is, spreading it on the table. Following prior works [5], we compute the coverage of the cloth as the objective for planning and evaluation.

Canonicalization. Usually, flattening is the first step of a cloth manipulation pipeline [9, 10, 11, 8, 51], which makes the subsequent tasks such as folding easier. However, for certain types of clothing, such as skirts or unbuttoned shirts, the flattening objective can produce undesirable results, as shown in Figure 4a, in which the robot maximizes the area covered by the cloth in a manner that is not conducive for downstream folding.

Therefore, we also evaluate our method on a task that we call "cloth canonicalization," where the goal is to manipulate the cloth and align it with the flattened canonical pose, as shown in Figure 4b. To account for the ambiguity due to rotation and reflection symmetries, we define a set of symmetries for each type of cloth. Using these symmetries, we define a canonical goal set of flattened poses $\mathcal{G} = \{G_i^{N\times 3}\}_{i=1\ldots A}$ for each cloth instance, where A is the number of valid canonical poses. For example, for Trousers, we can rotate the canonical pose by 180 degrees to obtain another valid goal. The cost is computed as the minimum of the average pairwise distance to each of the possible canonical poses. Suppose that the current configuration of the cloth in the simulator is $V \in \mathbb{R}^{N\times 3}$, where N is the number of vertices. Then the cost is computed as

$$Cost_{canon} = \min_{G_i \in \mathcal{G}} \frac{1}{N} \sum_{j=0}^{N-1} (g_j - v_j)^2 \qquad (4)$$

Note that G_i and V have the same number of vertices because both refer to the simulated cloth in different configurations. In this work, we allow for our method to canonicalize the cloth without penalizing for errors with respect to a rigid transformations; this is because, for the task of cloth folding, the rotation and translation of the cloth is of lesser importance. To evaluate this, we first align the goal with the current state by computing an optimal rigid transformation using the Kabsch algorithm [52], and then we compute the cost by Equation 5.

B. Simulation Experiments

1) Baselines: We compare our method to 4 baselines, including two state-of-the-art cloth manipulation methods:

- **VisuoSpatial Foresight (VSF)** [3]. This baseline learns a visual dynamics model in the RGB-D observation space. It is trained on each category separately.
- **Visible Connectivity Graph (VCD) [5]**. Similar to our method, VCD learns a particle-based dynamics model. However, unlike our method, VCD only operates on the visible points on the cloth, without explicitly reasoning about occlusions. An edge GNN is used to infer the mesh structure on the partial point cloud. To make a fair comparison with our method, the edge GNN is trained in a category-specific manner, while the dynamics model is only trained on Trousers (similar to our approach).
- **GarmentNets [12]**. In this baseline, we use the original implementation of GarmentNets for mesh reconstruction, which processes a partial point cloud with PointNet++ [43] and doesn't use relative prediction. For planning, we evaluate this baseline with the same mesh-based dynamics model as our method.
- **MEDOR-no-finetuning**. This is a variant of our method that removes the test-time finetuning step (Section IV-B).
- **MEDOR**. This is our full method, which is essentially a modified version of GarmentNets with test-time finetuning.

For VSF and VCD, we provide the ground-truth RGB-D image and the ground-truth mesh (respectively) in the canonical pose to use for the reward computation. Note that our method does not have access to this information. More details on the baselines can be found in the Supplement.

2) Results: For each task, we show the normalized improvement (NI) of each method, where 0 indicates no change from the initial state and 1 is the best possible performance. For flattening, we use the normalized improvement metric defined in previous work [5]; for

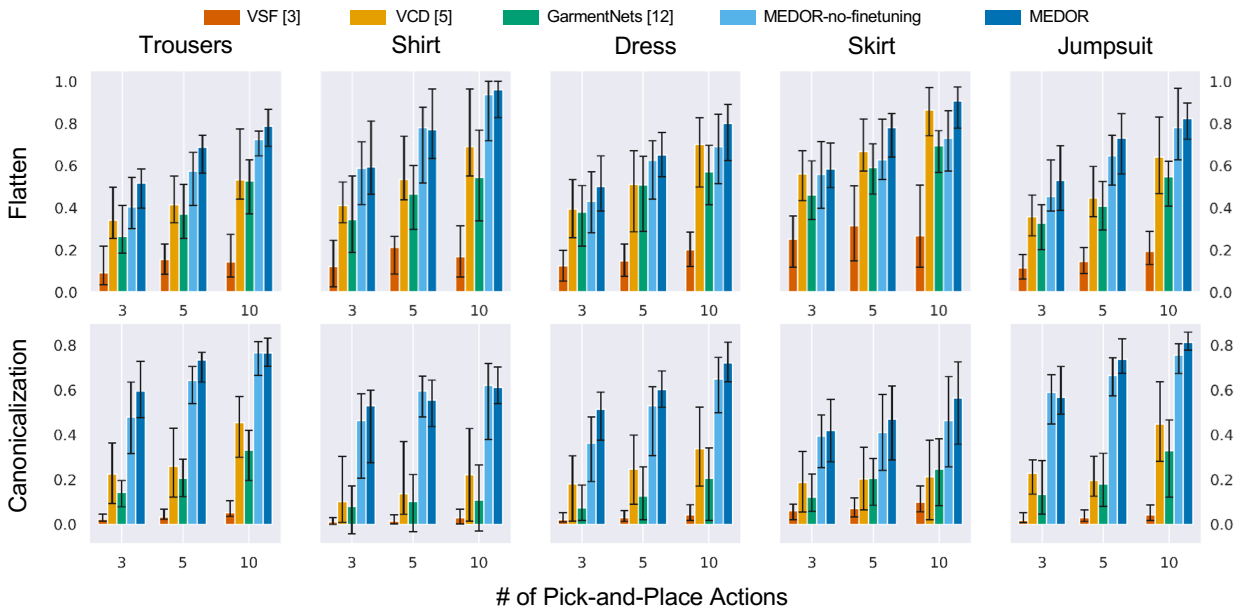


Fig. 5: Normalized improvements on 2 tasks and 5 different categories of cloths. The height of the bar represents the median and the error bars show the 25 and 75 percentile of the performance.

canonicalization, we compute the normalized improvement as $\text{NI}_{canon} = \frac{cost_{init} - cost_{cur}}{cost_{init}}$ where $cost_{init}$, $cost_{cur}$ are the costs of initial and current cloth configurations computed by Eq. 5. A trajectory is terminated after ten picks or when NI>0.95.

For each category, we generate 40 initial configurations by random drops. As shown in Figure 5, compared to methods without explicit occlusion reasoning (*VCD* [5] or *VSF* [3]), our methods MEDOR and MEDOR-no-finetuning both achieve competitive performance in all categories and both tasks. This shows the benefits of occlusion reasoning in cloth manipulation and the benefits of recovering the full configurations explicitly. Comparing *MEDOR* against *MEDOR-no-finetuning*, we can see the importance of test-time finetuning to adapt the mesh to better fit the observation. Qualititative examples are shown in Figure 3 as well as on the website, which show the differences in the mesh prediction before and after test-time finetuning.

For *VSF* (orange) and *VCD* (yellow), we see that the canonicalization task remains challenging even after being provided with the ground-truth RGB-D image or mesh for reward computation. This demonstrates the importance of planning with the completed cloth shape instead of planning only over the visible parts of the cloth.

We can also see that without our modifications, the original GarmentsNets [12] (green) has poor performance. The variant of our method *MEDOR-no-finetuning* includes the modifications to GarmentNets described in Section IV-A: HRNet and Relative Prediction. The huge performance gap between *GarmentNets* and *MEDOR-no-finetuning* demonstrates that these modifications lead to large performance benefits.

C. Ablations

To further examine each component and design choice in the paper, we conduct the following ablations on both the flattening and canonicalization tasks. Table I shows the normalized improvements averaged over 3, 5, and 10 picks, averaged over the flattening and canonlicalization tasks, and averaged over all 5 garment categories.

Why is occlusion reasoning beneficial to cloth manipulation? Explicit occlusion reasoning improves the performance of our framework on cloth manipulation in two aspects: (1) Using the reconstructed mesh helps with the reward computation, and (2) Using the reconstructed mesh helps with the dynamics model. We differentiate these benefits in the following experiments:

(1) In the 8th row *Ours w/ Partial Reward*, we use only the visible portion of the cloth for reward computation, while the dynamics uses the full reconstruction (visible + occluded regions). Compared to our full method, the performance drops by 29%, showing the benefits of occlusion reasoning for the reward computation.

(2) Using the reconstructed mesh improves the accuracy of the dynamics model: In VCD, the GNN dynamics model is trained and tested on the visible portion of the mesh. We compute the open loop rollout error of the VCD dynamics model on the visible mesh, and we likewise compute the rollout error of our method using the reconstructed mesh; we find that the rollout error

Method	Normalized Improvement
GarmentNets [53]	0.320 ± 0.146
No Mesh Reconstruction (VCD [5])	0.391 ± 0.174
No Finetuning and no Relative Prediction	0.560 ± 0.163
No Finetuning	0.585 ± 0.171
Joint Optimization	0.614 ± 0.157
No Consistency Loss	0.623 ± 0.148
Replace GNN by GT Dynamics	0.631 ± 0.161
Ours w/ Partial Reward	0.462 ± 0.210
Ours (full method)	**0.651 ± 0.138**
GT Mesh + Learned Dynamics	0.800 ± 0.096
GT Mesh + GT Dynamics	0.870 ± 0.076

TABLE I: Ablation experiments.

of the VCD dynamics model (on the partial mesh) is 64.2% higher than the rollout error on the reconstructed mesh. This demonstrates the importance of using the reconstructed mesh for accurate cloth dynamics.

To further this analysis, we also perform an experiment in which we use the full reconstructed mesh for the dynamics model but use only the partial mesh for the reward computation (*Ours w/ Partial Reward*). If we compare the performance of this version to the 2nd row *(No Mesh Reconstruction (VCD [5]))* we see the benefits of using the full reconstructed mesh for the dynamics model instead of the partial mesh (0.462 vs 0.391).

How much does test-time finetuning help? Are both losses necessary? When other components remain unchanged, we see that finetuning with only the Chamfer loss (*No Consistency Loss*) already improves the performance over no finetuning (*No Finetuning*) by 6.5%. Adding the Mapping Consistency Loss further boosts the performance from 0.623 (*No Consistency* 0.651 (*Ours (full method)*); the combined imp is 11.3%. Also, we find that without the 2- timization scheme (*Joint Optimization*), the consistency loss hurts the performance (com *Consistency Loss* vs *Joint Optimization*).

HRNet [45] vs PointNet++ [43]: Looking the only difference between the methods in th third row is the use of HRNet instead of Pc The huge difference in performance shows th of the HRNet architecure for this task.

Does Relative Prediction help? Comparin formance of *No Finetuning and No Relative* with *No Finetuning*, we see that this simple mc (described in Section IV-A) improves perform

Do we need to learn the dynamics mod of using the ground-truth dynamics model simulator? Once we reconstruct the full cloth option is to plan using the physics-based dy Nvidia Flex simulator, as similarly done in ablation is shown as *Replace GNN by GT Dy* Table I. We can see that this ablation yield performance drop compared to our full me speculate that this is because the analytical dynamics model is more sensitive to mesh prediction errors than the learned dynamics model. As an additional point, planning with a simulator is 1.4 times slower than using a GNN dynamics model even after heavy parallelization (247 seconds vs. 103 seconds for 500 rollouts).

Where are the remaining gaps in performance? As we can see in the last two rows, using the ground-truth mesh (instead of a learned mesh reconstruction) improves the performance by 23.1%. We can improve performance another 7% by also using the ground-truth dynamics model. The remaining errors come from the sampling-based planner itself, which might fail to sample good actions.

D. Physical Experiments

We also evaluate our method in the real world by deploying it on a 7-DOF Franka Emika Panda robot. We mount an Azure Kinect depth sensor on the end-effector of the robot. When taking the depth image, the end-effector will move to be centered above the cloth. To obtain a valid plan for pick and place actions, we use MoveIt! [54].

We evaluate our method on Trousers (3 instances) and Dress (2 instances) on the cloth flattening task. For each article of clothing, we run 5 trajectories with at most 10 pick-and-place actions each. The trajectories will be terminated if the normalized improvements exceed 95%. We obtain random configurations by performing a random drop three times. We compare our method with two baselines. *Random* is a heuristic policy that performs

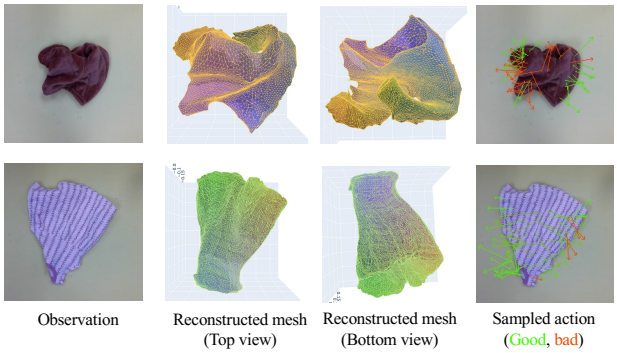
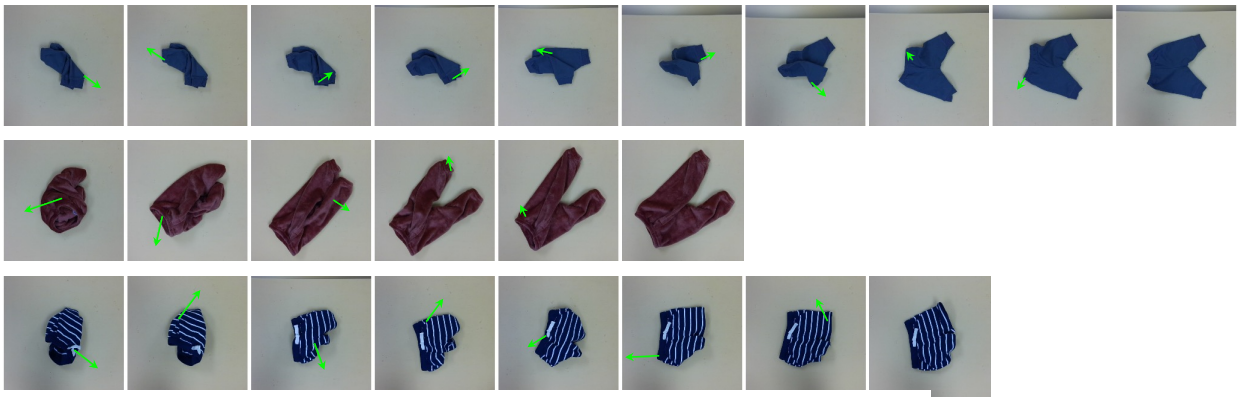
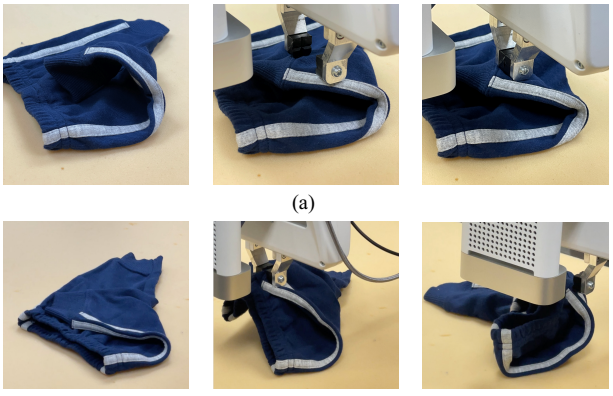


Fig. 6: Example reconstruction results in the real-world, after test-time fine-tuning. In the 1st row, the trousers are successfully reconstructed, including the occluded legs and wrinkles on the surface. The planner (right column) is able to distinguish good actions from bad ones given the reconstructed mesh. In the 2nd row, our model failed to capture the left-bottom corner which is folded under the visible layer. As such, the planner failed to choose actions to reveal the occluded part.



for videos.



(a)

(b)

Fig. 8: Grasping failures: In (a), the cloth is deformed when the gripper moves down, resulting in missed grasping. In (b), the robot is supposed to grasp the upper layer and unfold it, but it mistakenly grasps the bottom layer as well.

	Random	VCD [5]	MEDOR (Ours)
Trousers	0.044	0.562	**0.647**
Dress	0.036	0.361	**0.468**

TABLE II: Results of physical robot experiments.

model can efficiently smooth the clothes by only a few pick-and-place actions. The performance of our method compared to the baselines is shown in Table II.

We do observe a gap between the performance in simulation and in the real-world. The first source of error is the reconstruction error. Since the model is trained in simulation, it suffers from a distribution shift resulting from the difference in modeling the cloth physics and material properties, e.g., stiffness, thickness. In Figure 6, we show a successful and a failed reconstruction result. Another source of error comes from the grasp execution, which mostly occur when multiple layers of cloth are stacked together. There are two main failure modes: (1) The robot gripper deforms the cloth, causing a failed grasp; (2) The robot grasps the wrong number of cloth layers. Figure 8 illustrates the two cases.

VI. Conclusions

We introduce a cloth manipulation system that explicitly reasons about the occluded regions of cloth. At test-time, we optimize the predicted mesh with self-supervised losses. Then we use a learned mesh-based dynamics model to plan over the predicted mesh and find optimal actions for cloth manipulation. We compare against state-of-the-art cloth manipulation methods that do not account for partial observability and show significant improvements from explicit occlusion reasoning and test-time fine-tuning. We also demonstrate the efficacy of our method in real world experiments.

Acknowledgments

This work was supported by LG Electronics, National Science Foundation (NSF) CAREER Award (IIS-2046491) and NSF Smart and Autonomous Systems Program (IIS-1849154).

References

[1] Li Sun, Gerarado Aragon-Camarasa, Paul Cockshott, Simon Rogers, and J Paul Siebert. A heuristic-based approach for flattening wrinkled clothes. In *Conference Towards Autonomous Robotic Systems*, pages 148–160. Springer, 2013. 1, 2

[2] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE, 2010. 1, 2

[3] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Robotics: Science and Systems (RSS)*, 2020. 1, 2, 3, 6, 7, 14

[4] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning (CoRL)*, 2020. 1, 2

[5] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022. 1, 2, 3, 5, 6, 7, 8, 9, 14, 17

[6] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. Clothes state recognition using 3d observed data. In *2009 IEEE International Conference on Robotics and Automation*, pages 1220–1225. IEEE, 2009. 1, 2

[7] Yinxiao Li, Chih-Fan Chen, and Peter K Allen. Recognition of deformable object category and pose. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 5558–5564. IEEE, 2014. 1, 2

[8] Ioannis Mariolis, Georgia Peleka, Andreas Kargakos, and Sotiris Malassiotis. Pose and category recognition of highly deformable objects using deep learning. In *2015 International conference on advanced robotics (ICAR)*, pages 655–662. IEEE, 2015. 1, 2, 6

[9] Yinxiao Li, Yan Wang, Yonghao Yue, Danfei Xu, Michael Case, Shih-Fu Chang, Eitan Grinspun, and Peter K Allen. Model-driven feedforward prediction for manipulation of deformable objects. *IEEE Trans. Autom. Sci. Eng.*, 15(4):1621–1638, October 2018. 1, 2, 6

[10] Yinxiao Li, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter K Allen. Folding deformable objects using predictive simulation and trajectory optimization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6000–6006. IEEE, 2015. 1, 2, 6

[11] Yinxiao Li, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1046–1052. IEEE, 2014. 1, 2, 6

[12] Cheng Chi and Shuran Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. *ICCV*, 2021. 1, 2, 3, 5, 6, 7, 14, 15

[13] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. 1, 5

[14] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020. 1, 5, 17

[15] P Jiménez. Visual grasp point localization, classification and state recognition in robotic manipulation of cloth: An overview. *Rob. Auton. Syst.*, 92:107–125, June 2017. 2

[16] Eiichi Ono, N Kits, and Shigeyuki Sakane. Unfolding folded fabric using outline information with vision and touch sensors. *Journal of Robotics and Mechatronics*, 10:235–243, 1998. 2

[17] Bryan Willimon, Stan Birchfield, and Ian Walker. Model for unfolding laundry using interactive perception. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4871–4876. IEEE, 2011. 2

[18] Jianing Qian, Thomas Weng, Luxin Zhang, Brian Okorn, and David Held. Cloth region segmentation for robust grasp selection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9553–9560. IEEE, 2020. 2

[19] Stephen Miller, Mario Fritz, Trevor Darrell, and Pieter Abbeel. Parametrized shape models for clothing. In *2011 IEEE International Conference on Robotics and Automation*, pages 4861–4868. IEEE, 2011. 2

[20] Jan Stria, Daniel Prusa, and Vaclav Hlavac. Polygonal models for clothing. In *Conference Towards Autonomous Robotic Systems*, pages 173–184. Springer, 2014. 2

[21] Arnau Ramisa, Guillem Alenya, Francesc Moreno-Noguer, and Carme Torras. Learning rgb-d descrip-

tors of garment parts for informed robot grasping. *Engineering Applications of Artificial Intelligence*, 35:246–258, 2014. 2

[22] Boyi Jiang, Juyong Zhang, Yang Hong, Jinhao Luo, Ligang Liu, and Hujun Bao. Bcnet: Learning body and cloth shape from a single image. In *European Conference on Computer Vision*, pages 18–35. Springer, 2020. 2

[23] R Daněřek, Endri Dibra, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment: 3d garment shape estimation from a single image. In *Computer Graphics Forum*, volume 36, pages 269–280. Wiley Online Library, 2017. 2

[24] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7365–7375, 2020. 2

[25] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J Black. Scanimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2886–2897, 2021. 2

[26] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics (ToG)*, 36(4):1–15, 2017. 2

[27] Fangzhou Hong, Liang Pan, Zhongang Cai, and Ziwei Liu. Garment4d: Garment reconstruction from point cloud sequences. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[28] Zhaoqi Su, Tao Yu, Yangang Wang, and Yebin Liu. Deepcloth: Neural garment representation for shape and style editing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[29] Cheng Chi and Dmitry Berenson. Occlusion-robust deformable object tracking without physics simulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6443–6450. IEEE, 2019. 2

[30] Yixuan Wang, Dale McConachie, and Dmitry Berenson. Tracking partially-occluded deformable objects while enforcing geometric constraints. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14199–14205. IEEE, 2021. 2

[31] Te Tang and Masayoshi Tomizuka. Track deformable objects from point clouds with structure preserved registration. *The International Journal of Robotics Research*, page 0278364919841431, 2018. 2

[32] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas 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*, pages 2642–2651, 2019. 2, 3

[33] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. *Conference on Robot Learning (CoRL)*, 2018. 2

[34] Wilson Wu, Yilin adn Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *Robotics Science and Systems (RSS)*, 2020. 2, 3

[35] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2

[36] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022. 2

[37] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pages 192–202. PMLR, 2022. 2

[38] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017. 2

[39] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 2

[40] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021. 2

[41] Siwei Chen, Xiao Ma, Yunfan Lu, and David Hsu. Ab initio particle-based object manipulation. *arXiv preprint arXiv:2107.08865*, 2021. 2, 8

[42] Jhony Kaesemodel Pontes, James Hays, and Simon

Lucey. Scene flow from point clouds with or without learning. In *2020 International Conference on 3D Vision (3DV)*, pages 261–270. IEEE, 2020. 2

[43] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 3, 6, 8

[44] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 3, 15

[45] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 4, 8, 15

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[47] Kok-Lim Low and Tiow-Seng Tan. Model simplification using vertex-clustering. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 75–ff, 1997. 5, 17

[48] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 5

[49] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, 2020. 5, 13, 16

[50] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: Clothed 3d humans. In *European Conference on Computer Vision*, pages 344–359. Springer, 2020. 5, 13, 14, 16, 17

[51] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph Gonzalez, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics. In *ICRA*, 2021. 6

[52] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987. 6

[53] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 8

[54] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012. 8

[55] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 15

[56] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 16

[57] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 16

[58] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994. 17

VII. Experiments Setup

A. Simulation Setup

We conduct all simulation experiments in Softgym [49], a simulation environment for deformable objects built on the particle-based simulator, Nvidia Flex. We model a flying gripper as a spherical picker that can move freely in the 3D space. When a cloth particle is "picked", it will move rigidly with the gripper. The simulation parameters of softgym can be found in Table. III. We obtained the 3D cloth models from CLOTH3D dataset [50]. Considering the physical experiments, we rescale the cloth models so that they could fit in the workspace of our real robot. Also, to make the GNN dynamics computationally feasible, we create a downsampled version of the cloth models. During data collection, we still use the original dense mesh in softgym, but register the downsampled mesh onto the dense mesh by finding the nearest neighbor of each vertex. Therefore, we obtain the trajectory of downsampled mesh and use that for dynamics learning. The detailed specifications of the preprocessed CLOTH3D dataset can be found in Table. V. We use a top-down camera and it's placed at a fixed height of 0.65 m. The valid workspace is 0.53 m $\times$ 0.53 m.

Flattening For flattening task, the goal is to maximize the coverage of the cloth in the current configuration. To compute the coverage, we treat each node on the graph as a sphere with a radius of 0.005 and compute the covered area when projected to the ground plane.

Canonicalization In CLOTH3D, the canonical pose of the cloth is defined to be the pose of cloth when wore by a T-pose human. However, since we are considering cloth manipulation on a planar surface, it is not achievable and cannot be used as the goal pose directly. Therefore, we use gravity (10 times the gravity on Earth) to obtain a flattened version of the canonical pose. Another thing that needs to be handled is the ambiguity caused by reflection symmetry and rotation symmetry. For example, trousers are approximately 180° rotation symmetry, which means that the shapes before and after rotating around the center axis for 180° are the same. Therefore, during the evaluation of cloth canonicalization, we define a canonical goal set $\mathcal{G} = \{G_i^{N\times 3}\}_{i=1\dots A}$, for each type of cloth. A is the number of valid canonical poses. The cost is computed as the minimum of average pairwise distance to each of the possible canonical poses. Suppose that the current configuration of the cloth is $V^{N\times 3}$, where N is the number of vertices, g_j and v_j is the j-th vertex of the mesh, the cost is computed as

$$Cost_{canon} = \min_{G_i \in \mathcal{G}} \sum_{j=n}^{N} \frac{(g_j - v_j)^2}{N} \tag{5}$$

The rotation symmetry of each category is described in Table. IV. It should be noted that since we maintain a discrete set of plausible goal pose, although skirt has infinite order of rotation symmetry, we cannot iterate over all possible cases. Instead, we make an approximation that the order of rotation symmetry of skirt is 12. The order of rotation symmetry is the times the shape fits onto itself when rotating for 360 degrees. An order of 2 means that the shape remains unchanged when rotating for 180 or 360 degrees.

Simulation parameters	Value
Camera view	Top-down
Camera position	[0, 0.65 m, 0]
Field of view	90
Picker radius	0.01 m
Picker threshold	0.00625 m
dt	0.01 second
Damping	1
Dynamic friction	1.2
Particle friction	1
Stiffness (strech, bend, shear)	[1.2, 0.6, 1]
Mass	0.0003
Particle radius	0.005
Gravity	-9.8

TABLE III: Hyper-parameters of softgym.

Trousers	Shirt	Dress	Skirt	Jumpsuit
2	1	2	12	2

TABLE IV: Order of rotation symmetry of different types of cloth. The order of rotation symmetry is the times the shape fits onto itself when rotating for 360 degrees. An order of 2 means that the shape remains unchanged when rotating for 180 or 360 degrees. We use it to construct goal sets for cloth canonicalization task.

VIII. Additional Results

A. Simulation Experiments

The full results of simulation are shown in the Table. VI, best performance within each category is bolded. We also show several trajectories of cloth canonicalization in Fig. 9.

B. Physical Experiments

Some additional results of physical experiments are shown in Fig. 10 and Fig. 11.

IX. Implementation Details of MEDOR

A. GarmentNets-style Mesh Reconstruction Model

As described in the main paper, we employ a GarmentNets-style model to reconstruct the mesh from depth image. GarmentNets formulates the pose estimation problem of clothes as a shape completion task in the canonical space. By doing so, the model learns meaningful correspondence between There are several advantages about GarmentNets: 1) it reconstructs the occluded part of the clothes due to self-occlusion; 2) it estimates the

	Trousers	Shirt	Dress	Skirt	Jumpsuit
No. of Meshes	1691	1111	2037	468	2279
Avg. No. of Vertices	7050 ± 1954	5308 ± 996	8030 ± 2159	4643 ± 1225	10686 ± 2572
Avg. No. of Vertices (downsampled)	278 ± 87	214 ± 59	291 ± 99	190 ± 80	215 ± 58
Rescaling Factor	0.42	0.36	0.29	0.28	0.28
Avg. X (m)	0.29 ± 0.03	0.32 ± 0.10	0.24 ± 0.07	0.20 ± 0.05	0.19 ± 0.08
Avg. Y (m)	0.12 ± 0.02	0.11 ± 0.02	0.15 ± 0.05	0.15 ± 0.05	0.09 ± 0.02
Avg. Z (m)	0.27 ± 0.08	0.19 ± 0.04	0.31 ± 0.06	0.18 ± 0.05	0.31 ± 0.06

TABLE V: The statistics of the CLOTH3D dataset [50] after pre-processing.

	Task	Flattening			Canonicalization		
	Number of Pick-and-Place	1	2	3	1	2	3
Trousers	VSF [3]	0.09 ± 0.13	0.15 ± 0.07	0.14 ± 0.13	0.02 ± 0.02	0.03 ± 0.04	0.05 ± 0.05
	VCD [5]	0.34 ± 0.15	0.41 ± 0.14	0.53 ± 0.24	0.22 ± 0.14	0.26 ± 0.17	0.46 ± 0.16
	GarmentNets [12]	0.27 ± 0.15	0.37 ± 0.14	0.53 ± 0.16	0.14 ± 0.06	0.21 ± 0.09	0.33 ± 0.14
	MEDOR (no fine-tuning)	0.41 ± 0.14	0.57 ± 0.16	0.72 ± 0.08	0.48 ± 0.16	0.64 ± 0.10	0.77 ± 0.10
	MEDOR	$\mathbf{0.52 \pm 0.12}$	$\mathbf{0.69 \pm 0.12}$	$\mathbf{0.79 \pm 0.10}$	$\mathbf{0.59 \pm 0.13}$	$\mathbf{0.73 \pm 0.10}$	$\mathbf{0.77 \pm 0.07}$
Shirt	VSF [3]	0.12 ± 0.07	0.15 ± 0.08	0.20 ± 0.09	0.01 ± 0.02	0.01 ± 0.03	0.03 ± 0.04
	VCD [5]	0.39 ± 0.14	0.51 ± 0.23	0.70 ± 0.20	0.10 ± 0.20	0.14 ± 0.23	0.22 ± 0.21
	GarmentNets [12]	0.34 ± 0.21	0.47 ± 0.17	0.55 ± 0.22	0.08 ± 0.12	0.10 ± 0.14	0.11 ± 0.16
	MEDOR (no fine-tuning)	$\mathbf{0.59 \pm 0.17}$	$\mathbf{0.78 \pm 0.26}$	0.94 ± 0.22	0.46 ± 0.26	$\mathbf{0.60 \pm 0.12}$	$\mathbf{0.62 \pm 0.24}$
	MEDOR	0.59 ± 0.22	0.77 ± 0.19	$\mathbf{0.96 \pm 0.13}$	$\mathbf{0.53 \pm 0.26}$	0.56 ± 0.12	0.61 ± 0.09
Dress	VSF [3]	0.12 ± 0.07	0.15 ± 0.08	0.20 ± 0.09	0.02 ± 0.03	0.03 ± 0.03	0.04 ± 0.04
	VCD [5]	0.39 ± 0.14	0.51 ± 0.23	0.70 ± 0.20	0.18 ± 0.17	0.25 ± 0.16	0.34 ± 0.19
	GarmentNets [12]	0.38 ± 0.16	0.51 ± 0.22	0.57 ± 0.16	0.07 ± 0.10	0.13 ± 0.13	0.21 ± 0.19
	MEDOR (no fine-tuning)	0.43 ± 0.15	0.63 ± 0.19	0.69 ± 0.18	0.36 ± 0.17	0.53 ± 0.22	0.65 ± 0.15
	MEDOR	$\mathbf{0.50 \pm 0.14}$	$\mathbf{0.65 \pm 0.11}$	$\mathbf{0.80 \pm 0.17}$	$\mathbf{0.51 \pm 0.14}$	$\mathbf{0.60 \pm 0.08}$	$\mathbf{0.72 \pm 0.09}$
Skirt	VSF [3]	0.25 ± 0.13	0.32 ± 0.19	0.27 ± 0.24	0.06 ± 0.04	0.07 ± 0.05	0.10 ± 0.07
	VCD [5]	0.56 ± 0.13	0.67 ± 0.15	0.87 ± 0.12	0.19 ± 0.14	0.20 ± 0.14	0.21 ± 0.19
	GarmentNets [12]	0.46 ± 0.16	0.59 ± 0.13	0.70 ± 0.13	0.12 ± 0.10	0.20 ± 0.12	0.25 ± 0.16
	MEDOR (no fine-tuning)	0.56 ± 0.16	0.63 ± 0.19	0.73 ± 0.16	0.39 ± 0.14	0.41 ± 0.17	0.46 ± 0.21
	MEDOR	$\mathbf{0.58 \pm 0.12}$	$\mathbf{0.78 \pm 0.14}$	$\mathbf{0.91 \pm 0.13}$	$\mathbf{0.42 \pm 0.14}$	$\mathbf{0.47 \pm 0.18}$	$\mathbf{0.56 \pm 0.21}$
Jumpsuit	VSF [3]	0.12 ± 0.06	0.15 ± 0.07	0.19 ± 0.10	0.02 ± 0.03	0.03 ± 0.03	0.04 ± 0.05
	VCD [5]	0.36 ± 0.10	0.45 ± 0.15	0.64 ± 0.19	0.23 ± 0.09	0.19 ± 0.11	0.45 ± 0.19
	GarmentNets [12]	0.33 ± 0.12	0.41 ± 0.12	0.55 ± 0.14	0.13 ± 0.15	0.18 ± 0.14	0.33 ± 0.21
	MEDOR (no fine-tuning)	0.45 ± 0.17	0.65 ± 0.14	0.78 ± 0.19	$\mathbf{0.59 \pm 0.14}$	0.67 ± 0.09	0.76 ± 0.08
	MEDOR	$\mathbf{0.53 \pm 0.17}$	$\mathbf{0.73 \pm 0.17}$	$\mathbf{0.82 \pm 0.10}$	0.57 ± 0.14	$\mathbf{0.74 \pm 0.09}$	$\mathbf{0.81 \pm 0.05}$

TABLE VI: Normalized Improvement (NI) of cloth flattening and cloth canonicalization, for varying numbers of allowed pick and place actions.

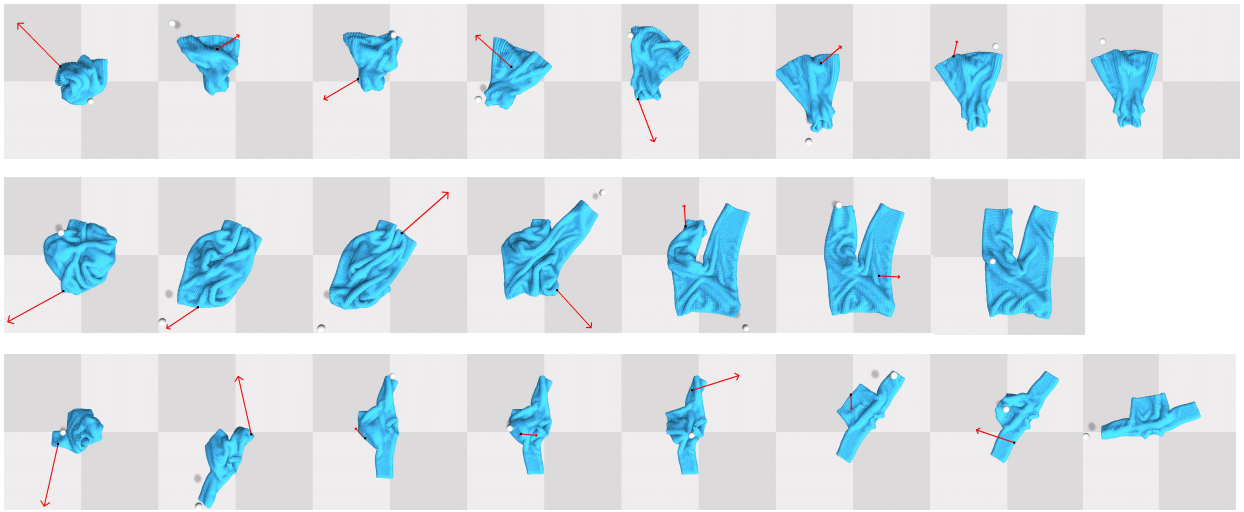


Fig. 9: Examplar trajectories of canonicalization task in simulation. As we can see, our method is able to quickly unfold the cloths from extremely crumpled configurations in a few steps.

correspondence between clothes in canonical space and observation space; 3) it estimates the pose of the clothes (per-vertex location).

1) Model Architecture: We build our mesh reconstruction model based on GarmentNets [12] with some decisive modifications to make it fit in our setup. Here,

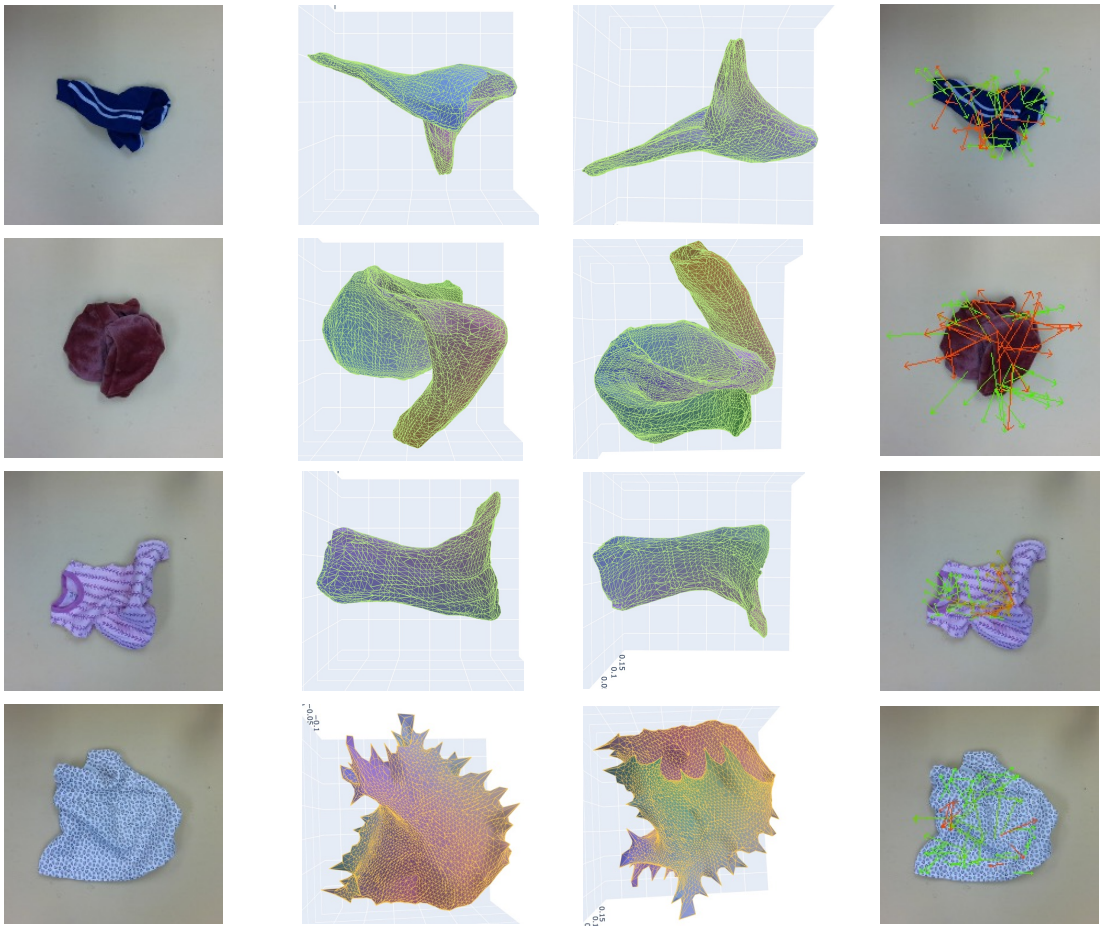


Fig. 10: Reconstruction results in real world.

we give a brief description of the reconstruction pipeline and the modifications we made. For details, please refer to GarmentNets [12].

Canonicalization Given the observation at current state, we first use *canonicalization model* to map it to canonical space by conducting pixel-wise canonical coordinate prediction. We don't pick up the clothes for state estimation, because it may disrupt the configuration of partially folded or almost smooth clothes. We use depth images captured by a top-down camera as the observation and High-Resolution Network (HRNet-32) [45] as the backbone of canonicalization model. HRNet is a convolutional architecture that specializes in producing high-resolution and spatially precise representations. It uses smaller convolutional kernels and avoids overly downsampling the feature map. This is critical for our task because our model needs to infer the structure of crumpled clothes by subtle changes at the contour of different layers. The architecture change improves the performance on cloth smoothing and canonicalization tasks for 75%. Following GarmentNets [12], we formulate the prediction problem as a classification task by dividing each axis into 64 bins and use Cross-entropy loss for training. **Feature Scattering** Given the predicted canonical coordinate, we scattered the features of each pixel to corresponding locations in the canonical space. The aggregated feature volume is further transformed by a 3D UNet [55], which is trained by shape completion and flow prediction.

Shape Completion Before estimating the pose of occluded surface, we first perform volumetric shape completion in the canonical space, which helps capture the shared structure within the same category. Since the structure of clothes are thin and non-watertight, GarmentNets [12] proposes to use Winding Number Field [44] as the shape representation. The shape com-

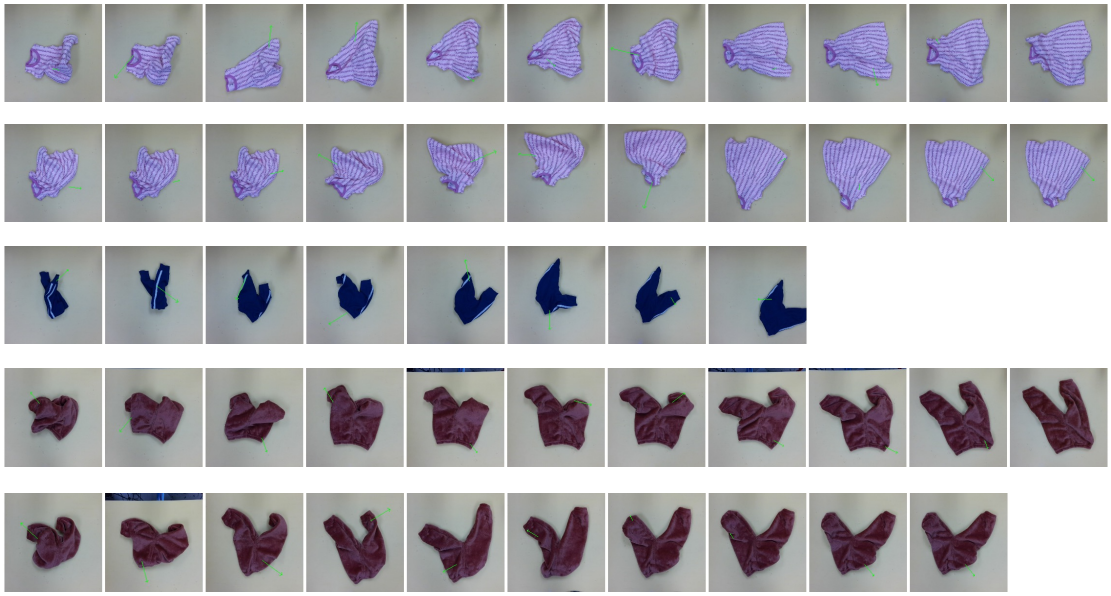


Fig. 11: Rollout of cloth flattening in real world.

pletion network is instantiated as an implicit network. It takes the dense feature produced by 3D UNet and a canonical coordinate and output the winding number field in that coordinate.

Predicting pose in the observation space After we complete the shape of clothes in the canonical space, we estimate the pose of the clothes in observation space. We cast it as a 3D flow prediction problem by predicting per-vertex flow that transforms the clothes from canonical space to observation space.

$$\tilde{x}_i^o = \tilde{x}_i^c + \tilde{f}_i \tag{6}$$

2) Training and Testing Details: . Now we describe how we train and test the model.

Dataset collection. Our model is category-specific, so we collect a dataset for each category separately in Softgym [49]. We obtained 3D clothes models from CLOTH3D dataset [50].

Each dataset contains 4,000 trajectories of length 5 for training and 400 for testing. The At the beginning of each trajectory, we randomly sample a clothes mesh and initialize it by random drop or flattened pose with equal probability. Then we disrupt the clothes with random pick-and-place actions. Random actions are biased towards picking corners (obtained by Harris corner detection[56]) 90% of the time, otherwise it is sampled uniformly on the clothes. The distance between the pick point and the place point is uniformly sampled between [25, 150] pixels.

Training details The model is trained in two-stage. In the first stage, we train the canonicalization network with Cross-entropy loss till convergence. In the second stage, we freeze the canonicalization network and train the rest of the models for shape completion and flow prediction by Mean-square error.

Model parameter	Value
Canonicalization Network	
Backbone	HRNet-32
Dimension of output feature	489
3D CNN	
Backbone	3D Unet
Level	4
Feature maps	32
Dimension of output feature	128
Implicit Shape Completion Network	
Backbone	MLP
Number of hidden layers	3
Size of hidden layers	512
Implicit Shape Completion Network	
Backbone	MLP
Number of hidden layers	3
Size of hidden layers	512
Training parameters	**Value**
Optimizer	Adam
Learning rate	0.0001
Gaussian noise std	0.005
Random rotation	[-180, 180]

TABLE VII: Hyper-parameters of mesh reconstruction model

Inference details At test-time, given a depth image, we first use canonicalization network and 3D UNet to obtain dense feature volume in the canonical space. Then we discretize the canonical space into 128x128x128 grid and evaluate shape completion network at every cell. To retrieve the mesh, we compute the Gaussian derivatives for the predicted winding number field and run Marching Cube algorithm [57].

B. Dynamics Model

Similar to VCD [5], we use a learned GNN-based dynamics model proposed in GNS [14]. The difference between VCD and ours is that we don't have a GNN edge model because edges are estimated by our mesh reconstruction model. The original mesh models in CLOTH3D [50] (see Table. V) are too dense that the rollout becomes computationally infeasible. Therefore, we downsample the mesh by using Vertex Clustering [47] with a voxel size of 0.025m. For the complete list of hyperparameters of the GNN dynamics model, please refer to Table VIII.

C. Planning

The planning algorithm is outlined in Algorithm 1. We plan in the space of pick-and-place primitive with horizon equal to 1. To simulate the effect of each pick-and-place, we divide them into low-level actions and roll out by the dynamics model in parallel. Following [5], the action is encoded into the input mesh by directly modifying the position and picked point, and the displacement will be propagated to the rest of mesh during message passing.

Action sampling during planning For both cloth flattening and canonicalization, we bias the actions sampling toward the contour of the cloth. More specifically, we first obtain the bounding box of the cloth in current observation and expand it by 30 pixels in each direction. Then we randomly sample picked points within the the bounding box region. For points that are not on the cloth, we map them to the nearest point on the cloth. The place direction is uniformly sampled in all directions and place distance is sampled uniformly from [0.05, 0.2]. A dummy action which corresponds to "no action" is added to the list of candidate actions. We sample 500 pick-n-place actions at each timestep.

Reward computation For flattening, we treat each mesh vertex as a sphere of radius 0.01, and the total reward is the covered area of the projection of all vertices to the ground plane. For canonicalization, we rotate the predicted canonical pose according to the predefined rotation symmetry (see Table. IV). For each valid canonical pose, we input it into the simulator to flatten by gravity, which constitutes a predicted goal set. The reward is computed as negative of smallest distance to goals in goal set. We use pairwise l2 distance.

X. Implementation Details of Baselines

A. VisuoSpatial Forsight (VSF)

We use the official codebase of VSF[1]. Image: we train the model with RGB-D images of size 56 x56 pixels, according to the original paper. To reproduce the performance of the original paper, we collected 7115 trajectories with 15 pick-and-place actions for each category. In total, it amounts to 100,000 environment steps, which is 5 times the data compared to our methods.

Action sampling We use a similar action sampling strategy as MEDOR for VSF, that is, bounding box sampling IX-C. We use a smaller padding size (6 pixels) because of the smaller image size.

Reward computation For flattening, we use color thresholding to compute the coverage of cloth. For canonicalization, we project the predicted and goal RGB-D image into 3D rgb point cloud. The RGB values are scaled to be similar to the coordinate values. Then we run ICP [58] for 5 iterations to align the predicted point cloud with the goal point cloud.

B. Visible Connectivity Dynamics (VCD)

We use the official code of VCD, with modifications so that it works well on our dataset.

Given point cloud and mesh, the original VCD conduct bipartite matching to map point cloud to mesh nodes. If the corresponding mesh nodes are connected by mesh edges, we also construct mesh edges for the point cloud points. We found that this approach is highly sensitive to density of point cloud and mesh. Imagine the mesh is denser than the point cloud, there might be many mesh vertices between two adjacent points on point cloud. Thus they are not connected although they should.

To solve this issue, we design a more robust approach for mesh edges construction that is agnostic to the density of mesh. First, for each point on the point cloud, we find the nearest mesh vertex. Then we compute the distance between neighboring points on point cloud by the pairwise geodesic distance of corresponding mesh vertices. A mesh edge is constructed if the geodesic distance is below a threshold.

Training To makes a fair comparison, we train the edge GNN dataset of different categories separately. Each dataset contains 20,000 envrironment steps, which is same as the dataset used for training the mesh reconsctruction model of MEDOR. For dynamics model, similar to MEDOR, we train a single model on Trousers dataset but use it for all categories at test-time.

Planning Same as MEDOR, except that we run ICP [58] for 5 iterations to align the predicted point cloud with the goal point cloud before computing the cost function for canonicalization task.

[1] https://github.com/ryanhoque/fabric-vsf

Model parameter	Value
Encoder(same for both node encoder and edge encoder)	
Number of hidden layers	3
Size of hidden layers	128
Processor	
Number of message passing steps	10
Number of hidden layers in each edge/node update MLP	3
Size of hidden layers	128
Decoder	
Number of hidden layers	3
Size of hidden layers	128
Training parameters	**Value**
Number of trajectories	5000
Learning rate	0.0001
Batch size	16
Training epoch	120
Optimizer	Adam
Beta1	0.9
Beta2	0.999
Weight decay	0
Others	**Value**
dt	0.05 second
Particle radius	0.005 m
Vertex clustering voxel size	0.025 m
Neighbor radius R	0.036 m

TABLE VIII: Hyper-parameters of GNN dynamics model.

Algorithm 1: Planning pipeline of MEDOR

input : Depth Image D, partial point cloud P, mesh reconstruction Model ϕ, dynamics GNN G_{dyn}, number of sampled actions K

output: pick-and-place action $a = \{x_{pick}, x_{place}\}$

Estimate the full mesh of clothes by mesh reconstruction model: $\tilde{M}_{init} = \phi(D)$

Perform test-time fine-tuning: $\tilde{M}^0_{tuned} = finetune(\tilde{M}_{init}) = (\tilde{V}^0, \tilde{E}_M)$.

for $i \leftarrow 1$ ***to*** K **do**

  Sample a pick-and-place action x_{pick}, x_{place}

  Compute low-level actions $\Delta x_1, ..., \Delta x_H$

  Get picked point v_{picked} from x_{pick}

  Pad historic velocities with 0: $\mathbf{x}_0 \leftarrow \tilde{V}^0, \dot{\mathbf{x}}_{-m...0} \leftarrow \mathbf{0}$

  for $t \leftarrow 0$ ***to*** H **do**

    Build collision edges E^t_C with $\mathbf{x}_t$

    Move picked point according to gripper movement by :

    $x_{u,t} \leftarrow x_{u,t} + \Delta x_t, \;\; \dot{x}_{u,t} \leftarrow \Delta x_t / \Delta t$

    Predict accelerations using G_{dyn}: $\ddot{\mathbf{x}}_t \leftarrow G_{dyn}(\mathbf{x}_t, \dot{\mathbf{x}}_{t-m...t}, E_M, E^t_C)$

    Update point cloud predicted positions & velocities:

    $\dot{\mathbf{x}}_{t+1} = \dot{\mathbf{x}}_t + \ddot{\mathbf{x}}_t \Delta t, \;\; \mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_{t+1} \Delta t$

    Readjust picked point according to gripper movement by

    $x_{u,t} \leftarrow x_{u,t} + \Delta x_t, \;\; \dot{x}_{u,t} \leftarrow \Delta x_t / \Delta t$

  end

  Compute reward r based on final mesh nodes position $\mathbf{x}_H$

end

return *pick and place action with maximal reward*