Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics

Aditya Ganapathi[1], Priya Sundaresan[1], Brijen Thananjeyan[1], Ashwin Balakrishna[1], Daniel Seita[1], Jennifer Grannen[1], Minho Hwang[1], Ryan Hoque[1], Joseph E. Gonzalez[1], Nawid Jamali[2], Katsu Yamane[2], Soshi Iba[2], Ken Goldberg[1]

***Abstract*— Robotic fabric manipulation is challenging due to the infinite dimensional configuration space, self-occlusion, and complex dynamics of fabrics. There has been significant prior work on learning policies for specific fabric manipulation tasks, but comparatively less focus on algorithms which can perform many different tasks. We take a step towards this goal by learning point-pair correspondences across different fabric configurations in simulation. Then, given a single demonstration of a new task from an initial fabric configuration, these correspondences can be used to compute geometrically equivalent actions in a new fabric configuration. This makes it possible to define policies to robustly imitate a broad set of multi-step fabric smoothing and folding tasks. The resulting policies achieve 80.3% average task success rate across 10 fabric manipulation tasks on two different physical robotic systems. Results also suggest robustness to fabrics of various colors, sizes, and shapes. See `https://tinyurl.com/fabric-descriptors` for supplementary material and videos.**

I. Introduction

Robot fabric manipulation has applications in folding laundry [4, 17, 25, 46], bed making [37], surgery [8, 38, 42, 43], and manufacturing [27, 45]. However, while robots are able to learn policies to manipulate a variety of rigid objects with increasing reliability [7, 14, 20, 22, 28], learning such policies for manipulating deformable objects remains an open problem due to difficulties in sensing and control. While there is significant prior work on geometric [1, 23, 35, 46] and learning based approaches [36, 37, 47] for fabric manipulation, these approaches often involve designing or learning task-specific manipulation policies, making it difficult to efficiently reuse information across tasks.

In this work, we leverage recent advances in dense keypoint learning [7] to learn visual point-pair correspondences across fabric in different configurations. Then, given a single offline demonstration of a fabric manipulation task from a given configuration, we utilize the learned correspondences to compute geometrically equivalent actions to complete the task on a similar fabric in a different configuration. For example, a human might provide a sequence of actions that would fold a T-shirt when it is placed neck up in a smoothed configuration. However, when the robot is deployed, it may encounter a different T-shirt whose color, size and pose differ from the T-shirt used for the demonstration. Learning visual correspondences that are invariant across these fabric attributes provides a powerful representation for defining policies that can generalize to these variations.

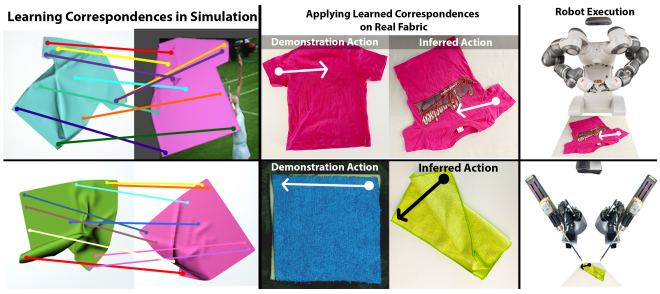

Fig. 1: We use learned visual correspondences across different fabric configurations to perform a variety of fabric manipulation tasks on the ABB YuMi (top) and the da Vinci Research Kit (bottom). Given a single demonstration of smoothing or folding, the robot uses the learned correspondences to compute geometrically equivalent actions for fabric of different color and in different initial configurations. This enables robust one-shot imitation learning of tasks that involve smoothing then folding.

We extend work by Sundaresan *et al.* [41], which leverages dense object descriptors [7] to learn visual correspondences for rope using synthetic depth data in simulation. These correspondences are then used to learn new rope manipulation tasks such as rearrangement or knot tying given a single task demonstration. We find that similar visual correspondence learning methods are also effective for learning correspondences between different fabric configurations using task-agnostic RGB data collected entirely in simulation and can be used to perform fabric manipulation tasks. Precisely, given a user demonstration of the task from a given initial fabric configuration, we leverage the learned visual correspondences to perform the same task from different initial configurations by computing geometrically equivalent actions using the correspondences. This approach has a number of appealing properties. First, visual correspondences can be learned purely in simulation without task-specific data and widely applied to a variety of real fabric manipulation tasks with no further training. Second, training in simulation enables sufficient data variety through domain randomization, making it possible to learn correspondences that generalize to fabrics with different colors, shapes, and configurations. Third, since perception and control are decoupled, the same perception module can be used on different robots with no additional training.

We contribute (1) a framework for learning dense visual correspondences of fabric in simulation using dense object descriptors from [7, 41] and applying them to manipulation

[1]AUTOLab at the University of California, Berkeley, USA
[2]Honda Research Institute, USA
Correspondence to Aditya Ganapathi: `avganapathi@berkeley.edu`

tasks on real fabrics with unseen colors, scales, and textures, (2) a data generation pipeline for collecting images of fabrics and clothing in Blender [3] and a testbed to experiment with different manipulation policies on these fabrics in simulation and (3) physical experiments on both the da Vinci Research Kit (dVRK) [15] and the ABB YuMi suggesting that the learned descriptors transfer effectively on two different robotic systems. We experimentally validate the method on 10 different tasks involving 5 T-shirts and 5 square fabrics of varying dimensions and colors and achieve an average task success rate of 80.3%.

II. Related Work

Fabric manipulation is an active area of robotics research [2, 11, 18, 33]. Over the past decade, the research has primarily been focused on three different categories: perception-based manipulation, learning-based algorithms in the real world, and learning-based algorithms in simulation which are then transferred to real robots.

Traditional Vision-Based Algorithms for Fabric Manipulation: Much of the prior work on perception-based deformable object manipulation relies on traditional image processing techniques to estimate fabric state. This state estimation is then used to define geometric controllers which bring the fabric into some desired configuration. However, due to the generalization challenges faced by these algorithms, most prior work makes specific assumptions on the fabric's initial configurations or requires more complex robotic manipulators to bring the fabric into a desired starting configuration. For example, Miller *et al*. [25] demonstrate a robust folding pipeline for clothing by fitting a polygonal contour to the fabric and designing a geometric controller on top of it, but assume that the initial state of the fabric is flat. Sun *et al*. [39, 40] perform effective fabric smoothing by estimating the wrinkles in the fabric, but condition on a near-flat starting fabric. Other work relies on "vertically smoothing" fabrics using gravity [4, 16, 17, 23, 26] to standardize the initial configuration and to expose fabric corners before attempting the task, which is difficult for large fabrics or single-armed robots.

Learning-Based Algorithms in the Real World: More recent approaches have focused on end-to-end learning of fabric manipulation policies directly on a real system, but these approaches can fail to generalize to a variety of fabrics and tasks due to the high volume of training data required. For example, Ebert *et al.* [5] use model-based reinforcement learning to learn fabric manipulation policies which generalize to many tasks, but require several days of continuous data collection on a real physical system and perform relatively low precision tasks. Jia *et al*. [12, 13] show impressive collaborative human-robot cloth folding under the assumption that fabric has already been grasped and is in a particular starting configuration, and Schulman *et al.* [35] demonstrate deformable object manipulation while requiring task-specific kinesthetic demonstrations. In follow-up work, Lee *et al.* [19] consider many of the same tasks as in this paper and demonstrate that policies can be learned to fold fabric using reinforcement learning with only one hour of experience on a real robot. In contrast, we learn entirely in simulation and decouple perception from control, making it easier to generalize to different fabric colors and shapes and flexibly deploy the learned policies on different robots without further learning.

Sim-to-Real Learning-Based Algorithms: Due to the recent success of sim-to-real transfer [32, 44], many recent papers leverage simulation to collect large amounts of training data, which is used to learn fabric manipulation policies. Seita *et al*. [36, 37] and Wu *et al.* [47] address the smoothing task from [39] but generalize to a wider range of initial fabric states using imitation learning (DAgger [31]), and reinforcement learning (Soft Actor-Critic [9]) respectively. Similarly, Matas *et al.* [24] learn fabric folding policies by using deep reinforcement learning augmented with task-specific demonstrations. However, these works learn policies that are specialized only to fabric smoothing [36, 37] and folding [47] respectively. In follow-up and concurrent work, Hoque *et al.* [10] and Yan *et al.* [48] use simulation to train fabric manipulation policies using model-based reinforcement learning for multiple tasks. In contrast, we leverage simulation to learn visual representations of fabric to capture its geometric structure without task-specific data or a model of the environment and then use this representation to design intuitive policies for several tasks from different starting configurations.

Dense Object Descriptors: We learn visual representations for fabric by using dense object descriptors [7, 34], which were shown to enable task oriented manipulation of various rigid and slightly deformable objects [7]. This approach uses a deep neural network to learn a representation which encourages corresponding pixels in images of an object in different configurations to have similar representations in embedding space. Such descriptors can be used to design geometrically structured manipulation policies for grasping [7], assembly [49], or for learning from demonstrations [6]. Sundaresan *et al.* [41] extend this idea to manipulation of ropes, and demonstrate that deformation-invariant dense object descriptors can be learned for rope using synthetic depth data in simulation and then transferred to a real physical system. Sundaresan *et al.* [41] then use the learned descriptors to imitate offline demonstrations of various rope manipulation tasks. In this work, we apply the techniques from [41] to learn descriptors which capture geometric correspondence across different fabric configurations from synthetic RGB images and use them for 2D fabric manipulation.

III. Problem Definition

A. Assumptions

We assume a deformable object is on a planar workspace in initial configuration ξ_1 with overhead RGB image observation $I_1 := I_1(\xi_1) \in \mathbb{R}^{W \times H \times 3}$. As in prior work [36, 47], we focus on fabric manipulation tasks that can be completed by a sequence of pick and place actions. Precisely, each action involves grasping the fabric at a *pick point*, pulling to a *place point* without changing the orientation of the end-effector,

and releasing the fabric. We assume that the pick point and place point are both visible in the camera frame and that the camera intrinsics and extrinsics are known at test-time. We additionally assume access to a single demonstration of each task in the form of a sequence of pick and place actions from some arbitrary initial fabric configuration ξ_1. These demonstrations can be collected offline, such as through a GUI where a user clicks on an image of fabric to indicate pick and place point pixels. However, the fabric used to create the instruction does not have to be of the same color, the same size or in the same initial configuration as the fabric the robot sees at test time. The only requirement is that the fabric be of a similar geometry. For example, T-shirts can be compared to other instances of T-shirts, but not to pants or long-sleeved shirts.

B. Task Definition

Define the action at step j as

$$\mathbf{a}_j = ((x_g, y_g)_j, (x_p, y_p)_j) \tag{III.1}$$

where $(x_g, y_g)_j$ and $(x_p, y_p)_j$ are the pixel coordinates of a grasp point on the fabric and place point respectively in image I_j at time j. The robot grasps the world coordinate associated with the grasp point and then moves to the world coordinate associated with the place point without changing the end effector orientation. This causes the fabric located at $(x_g, y_g)_j$ in the image to be placed on top of the world coordinate associated with $(x_p, y_p)_j$ with the same surface normals as before. In future work, we will investigate how to execute more complex actions that result in reversed surface normals, which requires a rotation motion during the action. We are given a sequence of actions $(\mathbf{a}_j)_{j=1}^n$ executed on a fabric starting in configuration ξ_1 and corresponding observations $(I_j)_{j=1}^n$ where I_j is the observation of the fabric before action $\mathbf{a}_j$ is taken. Then at test-time, a similar object is dropped onto the surface in a previously unseen configuration and the goal is to generate a corresponding sequence of actions for a fabric in some previously unseen configuration. Specifically, the robot generates a new sequence of actions:

$$\left(\mathbf{a}'_j\right)_{j=1}^n = \left(d_{I_j \to I'_j}(x_g, y_g)_j,\ d_{I_j \to I'_j}(x_p, y_p)_j\right)_{j=1}^n \tag{III.2}$$

for $j \in \{1, \ldots, n\}$ where $d_{I_j \to I'_j} : \mathbb{R}^2 \to \mathbb{R}^2$ is a function which estimates the corresponding point $(x', y')_j$ in I'_j given a point $(x, y)_j$ in I_j. This function is difficult to compute directly from images in general, and even more so for images of highly deformable objects due to their infinite degrees of freedom and tendency to self-occlude. Thus, we leverage dense object descriptors [7] to approximate $d_{I_j \to I'_j}$ for any I_j and I'_j, as described in Sections V and VI.

IV. Simulator

We use Blender 2.8, an open-source simulation and rendering engine [3] released in mid-2019, to both create large synthetic RGB training datasets and model the fabric dynamics for simulated experiments using its in-built fabric solver based on [29, 30]. We simulate T-shirts and square fabrics, each of which we model as a polygonal mesh made up of 729 vertices, a square number we experimentally tuned to trade-off between fine-grained deformations and reasonable simulation speed. See Figure 2 for an illustration. Each vertex on the mesh has a global coordinate which we can query directly through Blender's API, allowing for easily available ground truth information about various locations on the mesh and their pixel counterparts. We can also simulate finer-grained manipulation of the mesh including grasps, pulls, and folds. See the supplement for further details on how we perform manipulation and experiments in simulation.

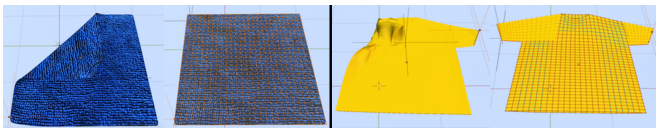

Fig. 2: **Fabric Meshes:** Examples of the meshes generated in Blender for both square cloth (left) and t-shirts (right). The ground-truth vertices are highlighted in the second and fourth columns.

V. Dense Shape Descriptor Training

A. Dense Object Descriptor Training Procedure

We consider an environment with a deformable fabric on a flat workspace and learn policies that perform smoothing and folding tasks. The policies are defined using learned point-pair correspondences between overhead images of the fabric in different configurations. We generate deformation-invariant correspondences by training dense object descriptors [7, 41] on synthetically generated images of the fabric in different configurations.

In Florence *et al.* [7], an input image I is mapped to a descriptor volume $Z = f_\theta(I) \in \mathbb{R}^{W \times H \times D}$, where each pixel (i, j) has a corresponding descriptor vector $Z_{i,j} \in \mathbb{R}^D$. Descriptors are generated by a Siamese network f_θ and are guided closer together for corresponding pixels in images and pushed apart by at least some margin M for non-corresponding pairs by minimizing a pixel-wise contrastive loss function during training [7]. Corresponding pairs of pixels represent the same point on an object. In this work, we also train a Siamese network to cluster corresponding pixel pairs and seperate non-corresponding pixel pairs in descriptor space. Since ground-truth pixel correspondences are difficult to obtain in images across deformations of a real fabric, we train the network on synthetic RGB data from Blender (see Section IV), where perfect information about the pixel correspondences is available through the global coordinates of the fabric mesh's vertices. Note that during training, the sampled image inputs to the Siamese network are enforced to be of the same fabric type to ensure valid correspondences. That is, two different images of T-shirts can be passed into the network, but not a T-shirt and square fabric. Figure 3 demonstrates the pipeline for predicting descriptors for correspondence generation. The learned descriptors can then be used to approximate the correspondence function $d_{I \to I'}$ described in Section III by (1) computing the top k pixel matches based on their distance in descriptor space and (2) computing the geometric median of

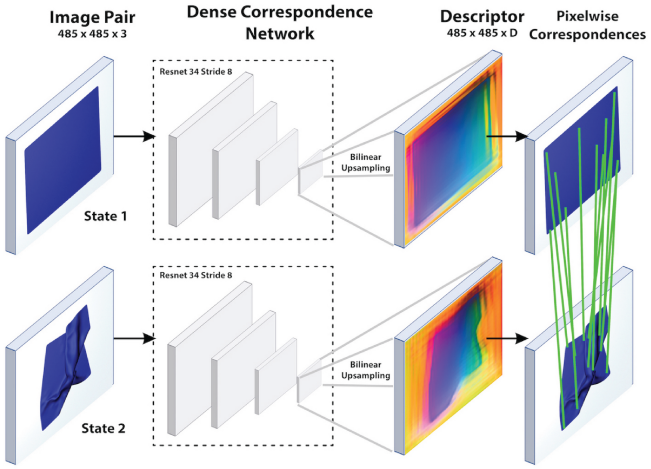

Fig. 3: **Learning Visual Correspondences:** pipeline for training dense object nets for robot fabric manipulation. Left: we train a dense correspondence network on pairs of simulated fabric images to learn pixel-wise correspondences using a pixel-wise contrastive loss. Right: we use the learned descriptors for policy optimization. We can use correspondence to map a reference action to a new fabric configuration. For example, we show an image of a wrinkled fabric in "State 2," and we can use descriptors to figure out the action needed to smooth the fabric from "State 2" to "State 1."

these matches in pixel space:

$$\left((i_l'', j_l'')\right)_{l=1}^{k} = \underset{(i_1', j_1')\ldots(i_k', j_k')}{\arg\min} \sum_{l=1}^{k} \|f_\theta(I)_{i,j} - f_\theta(I')_{i_l', j_l'}\|_2$$
$$\text{s.t. } (i_n', j_n') \neq (i_m', j_m') \; \forall m, n \in [k]$$
$$d_{I \to I'}(i,j) = \underset{(i',j')}{\arg\min} \sum_{l=1}^{k} \|(i',j') - (i_l'', j_l'')\|_2$$

In experiments we find $k = 20$ gives the most robust predictions.

B. Dataset Generation and Domain Randomization

To enable generalization of the learned descriptors to a range of fabric manipulation tasks, we generate a diverse dataset of initial fabric configurations. The first step simulates dropping the fabric onto the planar workspace while executing similar pinning actions to those described in Section IV on an arbitrary subset of vertices, causing some vertices to fall due to gravity while others stay fixed. We then release the pinned vertices 30 frames later so that they collapse on top of the fabric. This allows us to create realistic deformations in the mesh. We then export RGB images which serve as inputs to the Siamese network, pixel-wise annotations which gives us correspondences, and segmentation masks which allow us to sample matches on the fabric.

Simulating soft-body animations is in general a computationally time-consuming process which makes it difficult to render large datasets in short periods of time. We take steps toward mitigating this issue by rendering 10 images per drop, allowing us to collect 10x as much data in the same time period. In simulation, we found that the test time pixel match error was unaffected when including these unsettled images of the fabric in the dataset. We additionally make use of domain randomization [32, 44] by rendering images of the scene while randomizing parameters including mesh size, lighting, camera pose, texture, color and specularity (see supplement for further details). We also restrict the rotation about the z-axis to be between $(-\pi/4, \pi/4)$ radians to reduce ambiguity during descriptor training due to the natural symmetry of fabrics such as squares. To randomize the image background, we sample an image from MSCOCO [21] and "paste" the rendered fabric mask on top. For experiments, we generate one (domain-randomized) dataset, including both T-shirts and square fabric, and train a *single model* which we use for all experiments in Section VII. For reference, generating a single dataset of 7,500 images, half T-shirts and half square cloth, with 729 annotations per image takes approximately 2 hours on a 2.6GHz 6-core Intel Core i7 MacBook Pro.

VI. Descriptor-Parameterized Policies

As discussed in Section III-B, the robot receives a demonstration of the task consisting of actions $(\mathbf{a}_j)_{j=1}^n$ and observations $(I_j)_{j=1}^n$. At execution time, the robot starts with the fabric in a different configuration, and the fabric itself may have a different texture or color. At time $j \in [n]$, the robot observes I_j' then executes $\pi_j(I_j') = \left(d_{I_j \to I_j'}(x_g, y_g)_j,\ d_{I_j \to I_j'}(x_p, y_p)_j\right)$ where $d_{I_j \to I_j'}$ is defined in Section V-A. We train a single descriptor network for a variety of tasks and use it to identify correspondences in different fabric configurations from those in the supplied in demonstrations. π_j then uses these correspondences to identify semantically relevant pixels in I_j' to generate actions that manipulate these keypoints.

For example, one step of a task could involve grasping the top-right corner of the fabric and taking an action to place it in alignment with the bottom-left corner, thereby folding the fabric. The robot could receive an offline demonstration of this task on an initially flat fabric, but then be asked to perform the same task on a crumpled, rotated fabric. To do this, the robot must be able to identify the corresponding points in the new fabric configuration (top-right and bottom-left corners) and define a new action to align them. π_j computes correspondences for the pick and place points across the demonstration frame and the new observation to generate a corresponding action for the new configuration.

A. Fabric Smoothing

In the square fabric smoothing task, the robot starts with a crumpled fabric and spreads it into a smooth configuration on a planar workspace as in Seita *et al.* [36]. To complete this task, we use the approach from [36] and iterate over fabric corners, pulling each one to their target locations on an underlying plane. However, while [36] design a policy to do this using ground-truth knowledge of the fabric in simulation, we alternatively locate corners on the crumpled fabric using a learned descriptor network and a source image of a flat fabric where the corners are labeled. For the T-shirt smoothing task, we apply a similar method, but instead iterate over the corners of the sleeves and the base of the T-shirt.

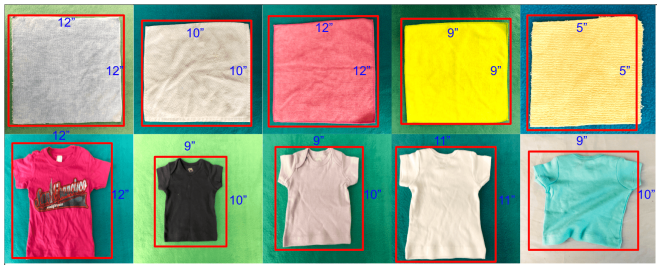

Fig. 4: **Fabric Specifications:** Images and dimensions of the square fabrics and shirts we use in experiments.

B. Fabric Folding

The fabric folding task involves executing a sequence of folds on a fairly smooth starting configuration. For each folding task, we use a single offline demonstration containing up to 4 pick and place actions collected by a human through a simple GUI. The descriptor-parameterized controller is then executed in an open-loop manner.

VII. Experiments

We experimentally evaluate (1) the quality of the learned descriptors and their sensitivity to training parameters and (2) the performance of the descriptor-parameterized policies from Section VI across 10 different fabric manipulation tasks on two physical robotic systems, the da Vinci Research Kit (dVRK) [15] and the ABB YuMi. Results suggest that the learned descriptors and the resulting policies are robust to changes in fabric configuration and color.

A. Tasks

We consider 10 fabric manipulation tasks executed on a set of 5 T-shirts and 5 square fabrics in the real world:

1) *Single Fold (SF):* A single fold where one corner is pulled to its opposing corner.
2) *Double Inward Fold (DIF):* Two opposing corners are folded to the center of the fabric.
3) *Double Triangle Fold (DTF):* Two sets of opposing corners are aligned with each other.
4) *Double Straight Fold (DSF):* The square cloth is folded in half twice, first along the horizontal bisector and then along the vertical bisector.
5) *Four Corners Inward Fold (FCIF):* All four corners are sequentially folded to the center of the cloth.
6) *T-Shirt Sleeves Fold (TSF):* The two sleeves of a t-shirt are folded to the center of the shirt.
7) *T-Shirt Sleeve to Sleeve Fold (TSTSF):* The left sleeve of a T-shirt is folded to the right sleeve of the T-shirt.
8) *Smoothing (S):* Fabric is flattened from a crumpled state.
9) *Smoothing + Double Triangle Fold (SDTF):* Fabric is smoothed then the DTF is executed.
10) *Smoothing + Sleeve to Sleeve Fold (SSTSF):* T-shirt is smoothed then TSTSF is executed.

All fabrics are varied either in dimension or color according to Figure 4. Additionally, we execute a subset of these tasks in simulation. A single visual demonstration consisting of up to 4 actions is provided to generate a policy which the robot then tries to emulate in the same number of actions.

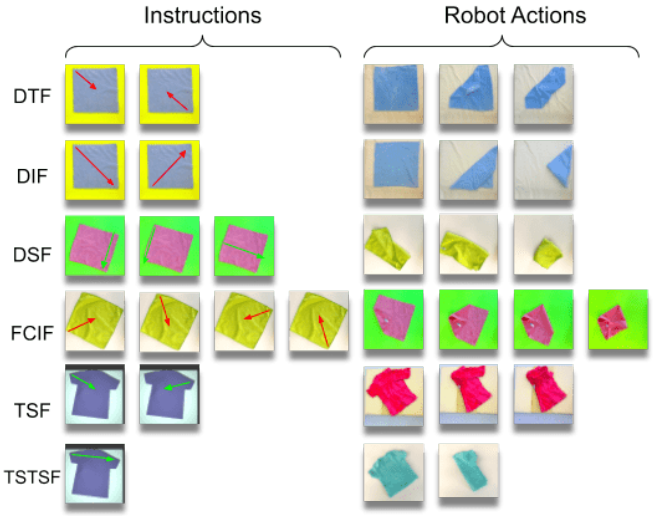

Fig. 5: **Policy Rollouts:** We visualize policy execution on the YuMi for tasks 2, 3, 4, 5, 6 and 7 as described in Section VII-A. The first four columns show the folding instructions on some initial fabric and the last four columns show the corresponding folds executed on novel starting configurations for a different fabric.

B. Experimental Setup

We execute fabric folding and smoothing experiments on the dVRK [15] and ABB YuMi robot. The dVRK is equipped with the Zivid OnePlus RGBD sensor that outputs 1900×1200 pixel images at 13 FPS at depth resolution 0.5 mm. The workspace of the dVRK is only $5" \times 5"$, so we use only square fabric of the same dimension while varying the color according to Figure 4. Manipulating small pieces of fabric into folds is challenging due to the elasticity of the fabric, so we add weight to the fabric by dampening it with water. Additionally, we place a layer of 1 inch foam rubber below the fabric to avoid damaging the gripper. The YuMi has a $36" \times 24"$ workspace, and since only one arm is utilized resulting in a more limited range of motion, we only manipulate at most $12" \times 12"$ pieces of fabric which we do not dampen. In this setup we use a 1080p Logitech webcam to collect overhead color images. For the YuMi, we use both T-shirts and square fabric of varying dimension and color but go no lower than $9" \times 9"$ fabrics due to its larger gripper. Finally, for both robots, we use a standard pixel to world calibration procedure to get the transformation from pixel coordinates to planar workspace coordinates.

For both robots, we follow the same experimental protocol. We manually place the fabric in configurations similar to those shown in Figure 2 and deform them by pulling at multiple locations on the fabric. To obtain image input for the descriptor networks, we crop and resize the overhead image to be 485×485 such that the fabric is completely contained within the image. Although lighting conditions, camera pose and workspace dimensions are significantly different between the two robotic systems, no manual changes are made to the physical setup. We find that the learned descriptors are sufficiently robust to handle this environmental variability.

We evaluate the smoothing task by computing the coverage of the cropped workspace before and after execution. For the folding tasks, as in Lee *et al.* [19], we consider an outcome

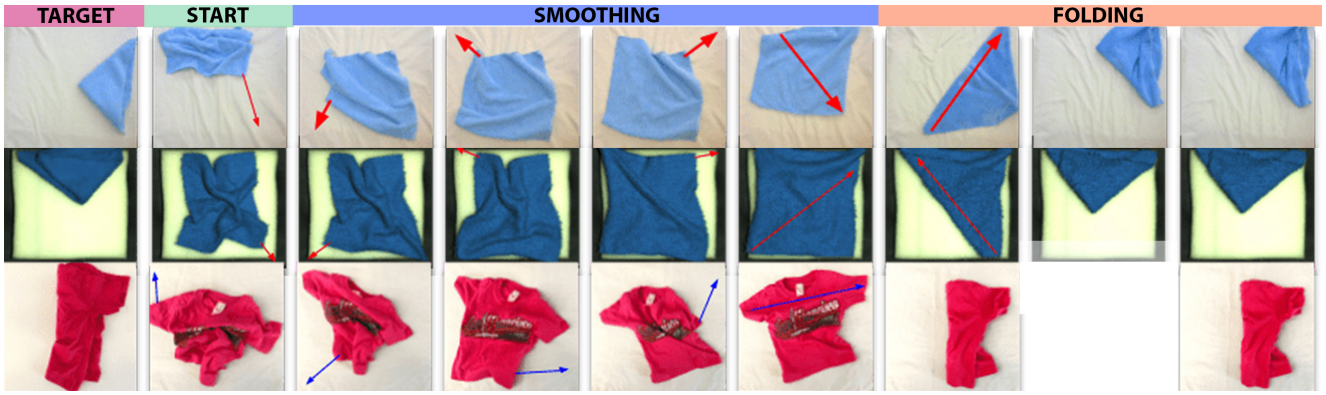

Fig. 6: **Full Folding Sequence:** The first and second row is a time-lapse of a sequence of 6 actions taken by the YuMi and dVRK respectively, and with actions overlaid by red arrows, to successively smooth a wrinkled fabric and then fold it according to task 3 in Section VII-A. The third row is a time-lapse of a sequence of 5 actions taken by the YuMi to complete task 10 in Section VII-A. Here, robot actions are overlaid with blue arrows.

a success if the final state is visually consistent with the goal image. Conventional quantitative metrics such as intersection of union between the final state and a target image provide limited diagnostic information when starting configurations are significantly different as in the presented experiments.

C. Results

We evaluate the smoothing and folding policies on both the YuMi and dVRK on square fabrics and T-shirts. Table II shows the success rates of our method on all proposed tasks in addition to a breakdown of the failure cases detailed in Table III. We observe that the descriptor-parameterized controller is able to successfully complete almost all folding tasks at least 75% of the time, and the smoothing policies are able to increase coverage of the cloth to over 83% (Table I). The execution of the smoothing policy followed by the double triangle folding policy results in successful task completion $6/10$ and $8/10$ times on the YuMi and dVRK respectively. We find that the most frequent failure mode is an unsuccessful grasp of the fabric which is compounded for tasks that require more actions. Though this is independent of the quality of the learned descriptors, it highlights the need for more robust methods to grasp highly deformable objects.

Task	Robot	Avg. Start Coverage	Avg. End Coverage
S	YuMi	71.4 ± 6.2	83.2 ± 8.1
S	dVRK	68.4 ± 4.4	86.4 ± 5.2

TABLE I: **Physical Fabric Smoothing Experiments:** We test the smoothing policies designed in Section VI on the YuMi and the dVRK. Both robots achieve an average increase in coverage of $11-22$ percent.

VIII. Discussion and Future Work

We present an approach for multi-task fabric manipulation by learning dense visual correspondences entirely in simulation. Experiments suggest that the learned correspondences are robust to different fabric colors, shapes, textures, and sizes and make it possible to efficiently learn 10 different fabric smoothing and folding tasks on two different physical robotic systems with no training in the real world. In future work, we plan to explore hierarchical fabric manipulation policies, where visual correspondences can be used to define coarse action plans while a closed loop controller can be learned to realize these plans. We will also explore more complex fabric manipulation tasks, such as wrapping rigid objects, in which reasoning about fabric dynamics is critical. Finally, we will also explore the use of a new inverted tweezer gripper that is more reliable for grasping fabric and addresses the common Type A error that occurs in this work.

Task	Robot	# Actions	Success	Error A	Error B	Error C
SF	YuMi	1	18/20	2	0	0
SF	dVRK	1	20/20	0	0	0
DIF	YuMi	2	16/20	3	0	1
DIF	dVRK	2	20/20	0	0	0
DTF	YuMi	2	14/20	3	2	1
DTF	dVRK	2	18/20	0	2	0
TSF	YuMi	2	15/20	3	0	2
SDTF	YuMi	6	6/10	2	1	1
SDTF	dVRK	6	8/10	0	2	0
DSF	YuMi	3	15/20	1	1	3
DSF	dVRK	3	17/20	1	0	2
FCIF	YuMi	4	13/20	5	1	1
FCIF	dVRK	4	18/20	0	1	1
TSTSF	YuMi	1	17/20	2	0	1
SSTSF	YuMi	5	6/10	2	0	2

TABLE II: **Physical Fabric Folding Experiments:** We test the folding policies from Section VI on the YuMi and the dVRK. We observe both robots are able to perform almost all folding tasks at least 75 percent of the time. The YuMi is able to perform the smoothing then folding task $6/10$ times and the dVRK is able to do so $8/10$ times.

Error	Description
A	Gripper picks up more than one layer of fabric or fabric slips out of gripper
B	Pick or drop correspondence error greater than 30 pixels (10% of cloth width) or pick correspondence not on fabric mask
C	Unintended physics: resulting fold does not hold due to variable stiffness of the fabric, friction of the fabric, or friction of the underlying plane

TABLE III: **Failure Mode Categorization**

IX. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, and with UC Berkeley's Center for Automation and Learning for Medical Robotics (Cal-MR). The authors were supported in part by donations from SRI International, Toyota Research Institute, Honda, Intel, and Intuitive Surgical. The dVRK was supported by the National Science Foundation, via the National Robotics Initiative (NRI), as part of the collaborative research project "Software Framework for Research in Semi-Autonomous Teleoperation" between The Johns Hopkins University (IIS 1637789), Worcester Polytechnic Institute (IIS 1637759),

and the University of Washington (IIS 1637444). Ashwin Balakrishna is supported by an NSF GRFP and Daniel Seita is supported by a Graduate Fellowships for STEM Diversity (GFSD).

REFERENCES

[1] B. Balaguer and S. Carpin, "Combining Imitation and Reinforcement Learning to Fold Deformable Planar Objects", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[2] J. Borras, G. Alenya, and C. Torras, "A Grasping-centered Analysis for Cloth Manipulation", *arXiv:1906.08202*, 2019.

[3] B. O. Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[4] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014.

[5] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control", *arXiv:1812.00568*, 2018.

[6] P. Florence, L. Manuelli, and R. Tedrake, "Self-Supervised Correspondence in Visuomotor Policy Learning", in *IEEE Robotics & Automation Letters*, 2020.

[7] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation", in *Conf. on Robot Learning (CoRL)*, 2018.

[8] J. Grannen*, P. Sundaresan*, B. Thananjeyan, J. Ichnowski, A. Balakrishna, M. Hwang, V. Viswanath, M. Laskey, J. E. Gonzalez, and K. Goldberg, "Learning robot policies for untangling dense knots in linear deformable structures", 2020.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", in *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

[10] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation", in *Proc. Robotics: Science and Systems (RSS)*, 2020.

[11] R. Jangir, G. Alenya, and C. Torras, "Dynamic Cloth Manipulation with Deep Reinforcement Learning", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[12] B. Jia, Z. Hu, J. Pan, and D. Manocha, "Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.

[13] B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha, "Cloth Manipulation Using Random-Forest-Based Imitation Learning", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[14] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation", Jun. 2018.

[15] P. Kazanzides, Z. Chen, A. Deguet, G. Fischer, R. Taylor, and S. DiMaio, "An Open-Source Research Kit for the da Vinci Surgical System", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014.

[16] Y. Kita, T. Ueshiba, E. S. Neo, and N. Kita, "A Method For Handling a Specific Part of Clothing by Dual Arms", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[17] ——, "Clothes State Recognition Using 3D Observed Data", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009.

[18] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms", *arXiv:1907.03146*, 2019.

[19] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, "Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience", in *Conf. on Robot Learning (CoRL)*, 2020.

[20] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[21] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar, "Microsoft COCO: Common Objects in Context", *arXiv:1405.0312*, 2014.

[22] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies", in *Science Robotics*, 2019.

[23] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010.

[24] J. Matas, S. James, and A. J. Davison, "Sim-to-Real Reinforcement Learning for Deformable Object Manipulation", *Conf. on Robot Learning (CoRL)*, 2018.

[25] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A Geometric Approach to Robotic Laundry Folding", in *Int. Journal of Robotics Research (IJRR)*, 2012.

[26] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of Massive Laundry and Classification Types by Dual Manipulator", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 11, no. 5, 2007.

[27] J. K. Parker, R. Dubey, F. W. Paul, and R. J. Becker, "Robotic Fabric Handling for Automating Garment Manufacturing", *Journal of Manufacturing Science and Engineering*, vol. 105, 1983.

[28] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016.

[29] X. Provot *et al.*, "Deformation constraints in a mass-spring model to describe rigid cloth behaviour", in *Graphics interface*, Canadian Information Processing Society, 1995, pp. 147–147.

[30] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments", in *Computer Animation and Simulation'97*, Springer, 1997, pp. 177–189.

[31] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning", in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[32] F. Sadeghi and S. Levine, "CAD2RL: Real Single-Image Flight without a Single Real Image", in *Proc. Robotics: Science and Systems (RSS)*, 2017.

[33] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey", in *Int. Journal of Robotics Research (IJRR)*, 2018.

[34] T. Schmidt, R. A. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence", *IEEE Robotics and Automation Letters*, vol. 2, pp. 420–427, 2017.

[35] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from Demonstrations Through the Use of Non-Rigid Registration", in *Int. S. Robotics Research (ISRR)*, 2013.

[36] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. Canny, and K. Goldberg, "Deep

Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[37] D. Seita, N. Jamali, M. Laskey, R. Berenstein, A. K. Tanwani, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making", in *Int. S. Robotics Research (ISRR)*, 2019.

[38] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[39] L. Sun, G. Aragon-Camarasa, P. Cockshott, S. Rogers, and J. P. Siebert, "A Heuristic-Based Approach for Flattening Wrinkled Clothes", *Towards Autonomous Robotic Systems. TAROS 2013. Lecture Notes in Computer Science, vol 8069*, 2014.

[40] L. Sun, G. Aragon-Camarasa, S. Rogers, and J. P. Siebert, "Accurate Garment Surface Analysis using an Active Stereo Robot Head with Application to Dual-Arm Flattening", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015.

[41] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg, "Learning Rope Manipulation Policies using Dense Object Descriptors Trained on Synthetic Depth Data", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[42] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg, "Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks", *IEEE Robotics & Automation Letters*, 2020.

[43] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral Surgical Pattern Cutting in 2D Orthotropic Gauze with Deep Reinforcement Learning Policies for Tensioning", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017.

[44] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[45] E. Torgerson and F. Paul, "Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1987.

[46] B. Willimon, S. Birchfield, and I. Walker, "Model for Unfolding Laundry using Interactive Perception", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[47] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations", *RSS*, 2020.

[48] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation", in *Conf. on Robot Learning (CoRL)*, 2020.

[49] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2Fit: Learning Shape Priors for Generalizable Assembly from Disassembly", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

X. Appendix

The appendix is organized as follows:

- Appendix A contains additional details on the fabric simulator
- Appendix B contains additional details on the experiments conducted in simulation
- Appendix C contains results for simulation experiments
- Appendix D contains details on the pixel-wise contrastive loss method
- Appendix E shows visualizations of the learned descriptor mappings.
- Appendix F contains images from additional physical trials executed on the dVRK and YuMi.
- Appendix G conducts a detailed study on the effect of various hyperparameters on the quality of the learned visual correspondences.

A. Fabric Simulator Details

We use Blender 2.8 to both create dynamic cloth simulations and to render images of the fabric in different configurations. As can be seen in Figure 2, we are able to retrieve the world coordinates of each vertex via Blender's API which we then use to find ground truth pixel correspondences through an inverse camera to world transformation. This allows us to create dense pixel-vertex annotations along the surface of the fabric which we feed to into the descriptor training procedure. Figure 9 is a visualization of the learned descriptors and Figure 7 contains examples of the domain randomized training data we generate through Blender.

1) Fabric Model: To generate the square cloth in Blender, we first import a default square mesh and subdivide it three times to create a grid of 27×27 grid of vertices. We found that this number of square vertices resulted in a visually realistic animation in comparison to our real fabrics. We additionally add 0.02 meter thickness to the cloth to increase its weight which creates more realistic collision physics. In order to apply Blender's in-built cloth physics to the mesh, we simply make use of the cloth physics modifier through which we are able modify the parameters shown in Table IV. Internally, Blender simulates fabric physics for polygonal meshes with gravitational forces, damping, stiffness, and by interconnecting the mesh vertices with four types of virtual springs: tension springs, compression springs, shear springs, and angular bending springs. Each vertex also exerts repulsive forces within a self-contained virtual sphere on vertices both within fabric and in surrounding objects, to simulate self-collisions and collisions with other objects. We visually tune the simulator by replaying a fabric folding action while varying parameters, most notably the friction coefficients and spring elasticity constants. From observing videos of the folding actions, we settle on the parameter values specified in Table IV. A visualization of these steps can be seen in the top row of Figure 8. To generate the t-shirt mesh, we similarly import a default square mesh and subdivide it three times, but also delete all vertices that do not lie in a predefined t-shirt cutout of the square mesh which results in the bottom right image of Figure 2.

TABLE IV: Blender Cloth Simulation Parameters

Parameter	Explanation	Value
Quality Steps	quality of cloth stability and collision response	5.0
Speed Multiplier	how fast simulation progresses	1.0
Cloth Mass (kg)	–	0.3
Air Viscosity	air damping	1.0
Tension Springs	tension damping/stretching	5.0
Compression Springs	compression damping/stretching	5.0
Shear Springs	damping of shear behavior	5.0
Bending Springs	damping of bending behavior	0.5
Friction	friction with self-contact	5
Self-Collision Distance (m)	per-vertex spherical radius for repulsive forces	0.015

2) Manipulation with Hook Objects: To implement the action space defined in Section III, we first deproject the pixel corresponding to the pick point and map it to the vertex whose global coordinates are closest in $\mathbb{R}^3$ to the pixel's deprojected coordinates. We then directly manipulate this vertex by pinning it and translating it over a sequence of 30 frames. We utilize hook objects to take actions in the Blender simulator. A hook object attaches to a mesh vertex and exerts a proportional sphere of influence over the selected vertex and those in its vicinity, pulling the fabric in the direction of movement. We simulate a grasp, drag, and drop of the fabric by assigning a hook object to a fabric vertex, moving this hook over a series of frames to the deprojected pixel drop location, and removing the hook object assignment to release the cloth.

3) Starting Configurations and Actions: To generate varied starting configurations, we simulate dropping the fabric from 0.2 meters above the workspace while pinning it an arbitrary subset of the 729 vertices. After 30 frames in the animation, the pinned vertices are released and are allowed to settle for another 30 frames. This creates natural deformation in the cloth and introduces a wide range of starting configurations to the training dataset. A sequence of these steps is shown in the second row of Figure 8. When running simulated experiments, taking pick and place actions requires manipulating the cloth via hook objects as defined in Section IV. A sequence of frames throughout the course of an action using a hook object as well as the corresponding rendered frames are shown in the last two rows of Figure 8.

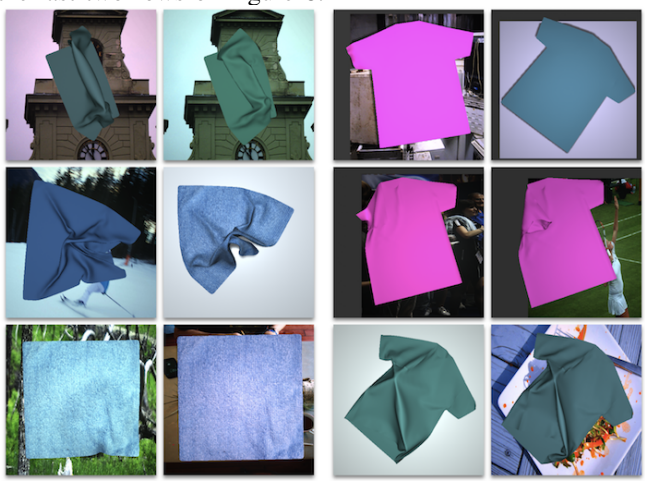

Fig. 7: Examples of domain-randomized images of the starting fabric states encountered in the dataset generation phase described in Section V-B. The first two columns show examples of images with a square fabric, and the last two columns show similar examples but with a t-shirt.

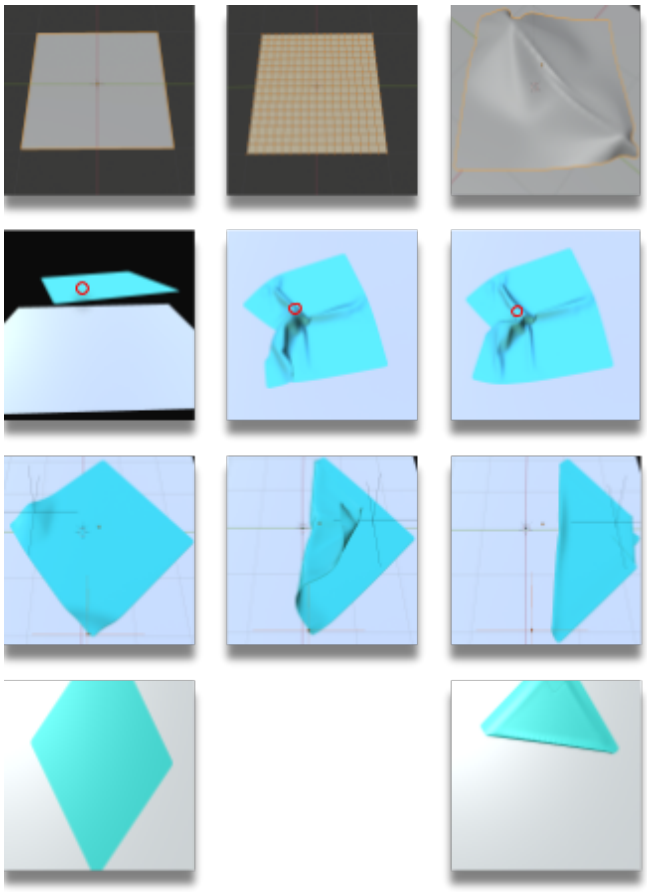

Fig. 8: The top row illustrates the the process of creating cloth in Blender from a default square mesh. The second row is an example of a starting configuration generated by dropping the cloth from a fixed height and pinning a single arbitrary vertex. The pinned vertex is labeled by the red circle. The third row illustrates frames from a folding action in the simulator and the last row shows the corresponding rendered images of the settled cloth before and after the action.

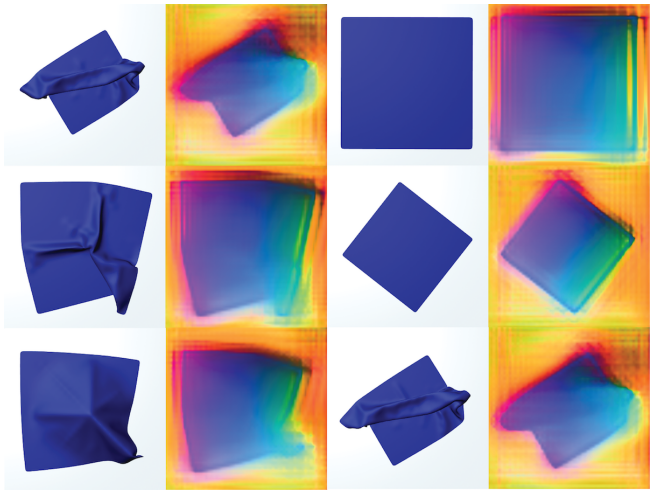

Fig. 9: Visualization of the 3-dimensional descriptors learned via the training procedure described in Section V by mapping each pixel's descriptor vector to an RGB vector. Thus, similar colors across the images of columns two and four represent corresponding points on the square cloth.

B. Simulation Experiment Details

In simulation, we conduct 50 trials of the first 4 folding tasks described in VII-A on a domain randomized test set generated as described in V-B. We consider an outcome a success if the final state is visually consistent with the target image. We additionally declared a failure when the planned pick and drop pixels were more than 50 pixels away from their correct ground truth locations which we had access to in Blender. Note that this is neither a sufficient nor necessary condition for a successful fold, but nevertheless serves as a decent heuristic. While we considered more quantitative metrics such as structural similarity between the target image and the final state and summed distance between corresponding vertices on the mesh, these metrics are insufficient when the test time starting configuration is significantly different from the demonstration configuration.

C. Simulation Experiment Results

We evaluate the folding policies designed in Section VI in the simulated fabric environment. The policies successfully complete the tasks 84 to 96 percent of the time (Table V).

Task	Success Rate
Single Fold	46/50
Double Inward Fold	48/50
Double Triangle Fold	42/50
T-Shirt Sleeves Fold	44/50

TABLE V: **Simulated Fabric Folding Experiments:** We observe that the system is able to successfully complete the tasks 84 to 96 percent of the time in simulation. Success is determined by visual inspection of the cloth after the sequence of actions is executed. Example simulation rollouts are shown in Figure 14.

D. Pixel-wise Contrastive Loss

Here we touch upon the details of the pixel-wise contrastive method from [7] used to train the descriptor networks. A neural network f maps I_a to a D-dimensional descriptor volume: $f : \mathbb{R}^{W\times H\times 3} \longmapsto \mathbb{R}^{W\times H\times D}$. During training, a pair of images and sets of both matching pixels and non-matching pixels are sampled between the image pair. The following contrastive loss minimizes descriptor distance between matching pixels and pushes descriptors for non-matching pixels apart by a fixed margin M:

$$L(I_a, I_b, u_a, v_a, u_b, v_b) =$$

$$\begin{cases} ||f(I_b)[u_b, v_b] - f(I_a)[u_a, v_a]||_2^2 & \text{match} \\ \max(0, M - ||f(I_b)[u_b, v_b] - f(I_a)[u_a, v_a]||_2)^2 & \text{non-match} \end{cases}$$

where $(u_a, v_a), (u_b, v_b)$ in equation (1) is a correspondence pair and where $(u_a, v_a), (u_b, v_b)$ in equation (2) is not a correspondence pair.

E. Descriptor Mapping Visualizations

We present the descriptor volumes produced by a model trained to output 3-dimensional descriptors. We coarsely visualize the volumes by presenting them as RGB images (Figure 9), and observe that corresponding pixels of the cloth map to similar colors in the descriptor volumes across configurations.

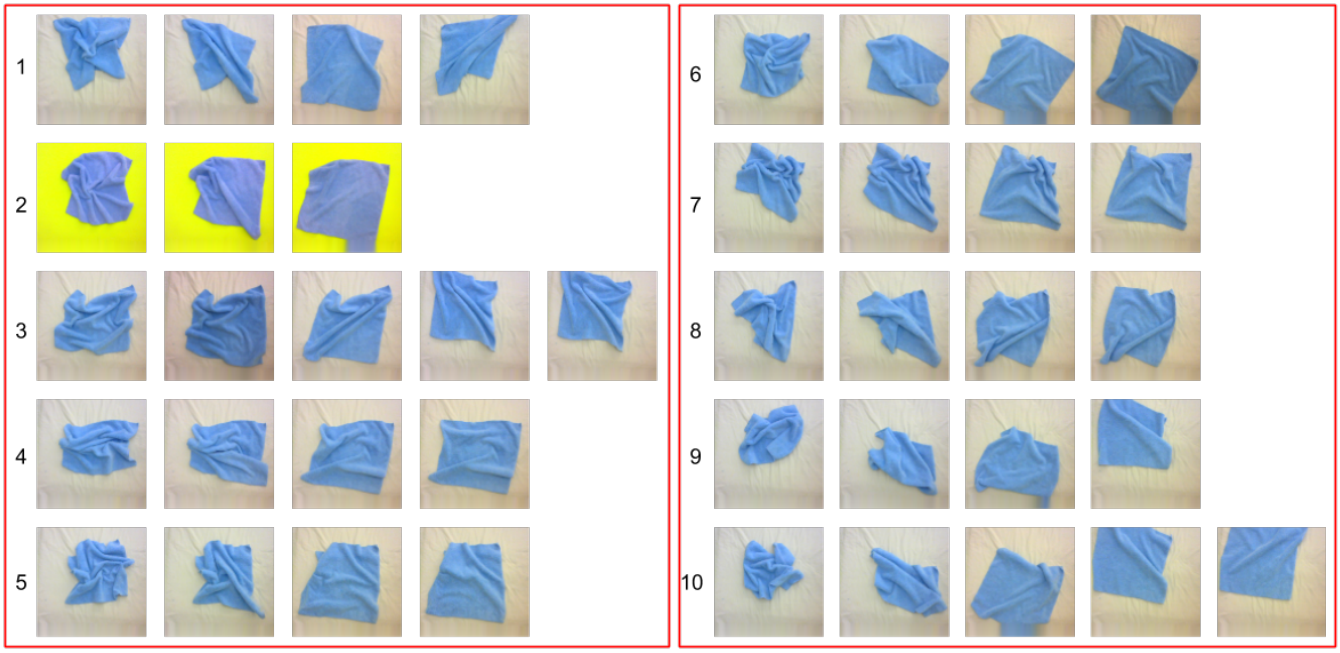

Fig. 10: Additional rollouts of the smoothing task from randomly chosen starting configurations. The learned descriptors are used to locate the corners of the fabric and successively pull them to a reference location in an image of the flat cloth.

Fig. 11: Additional rollouts of the smoothing task from randomly chosen starting configurations. The learned descriptors are used to locate the corners of the fabric and successively pull them to a reference location in an image of the flat cloth.

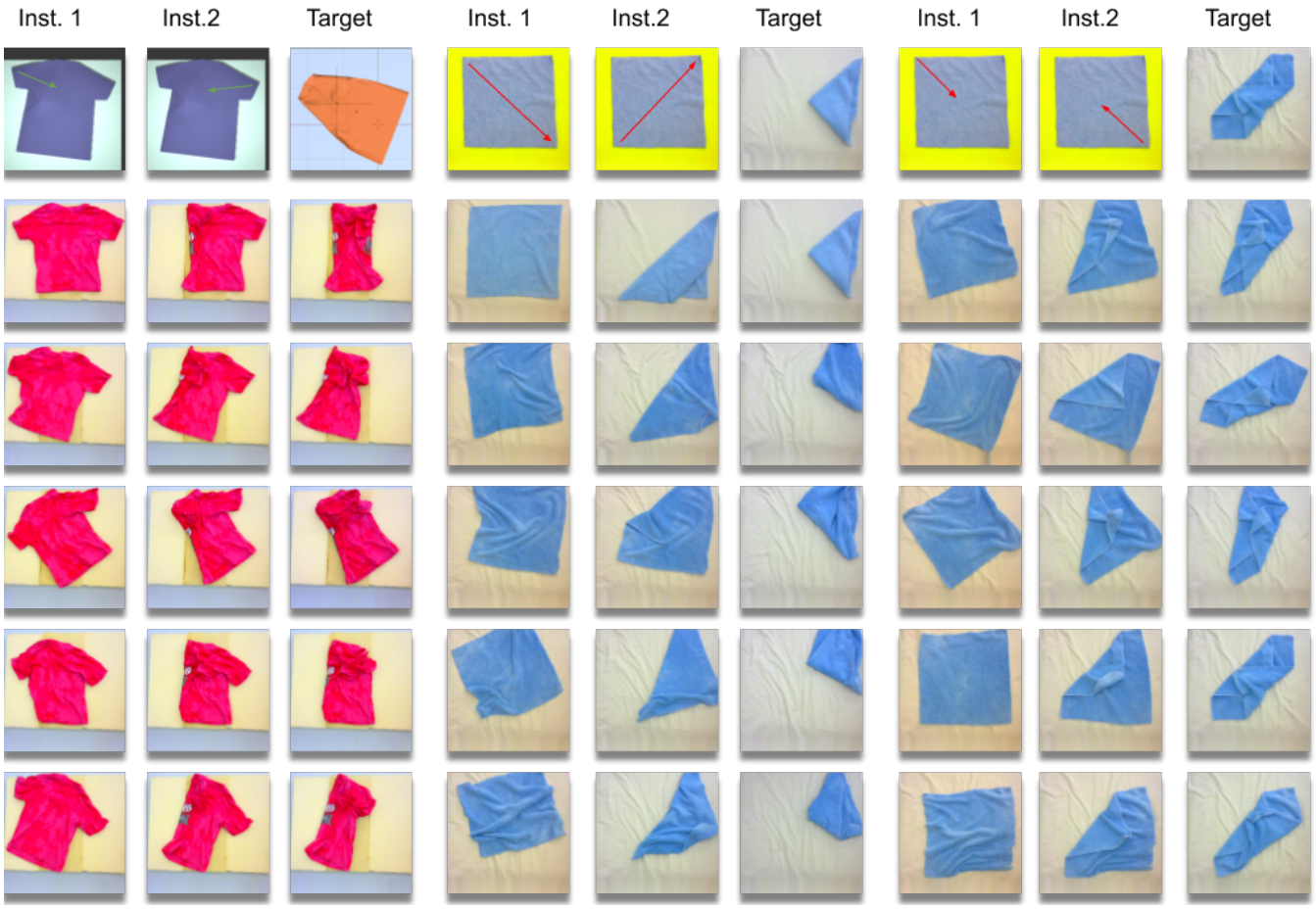

Fig. 12: Additional rollouts of the folding tasks described in Section VII-A from arbitrary starting configurations. The left column, center column and right column contain results for tasks 4, 3 and 2 respectively.

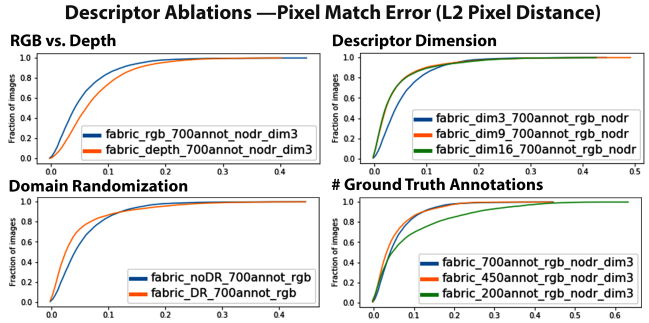

Fig. 13: **Ablation studies:** We study the sensitivity of the learned dense object descriptors as described in Sections V and X-G to training parameters. Starting from top left, and proceeding clockwise, we test the effect of testing on RGB vs depth images, on the descriptor dimension (either 3, 9, or 16), on the number of ground truth annotations, and whether domain randomization is used. All results are evaluated using pixel match error on a held-out set of image pairs.

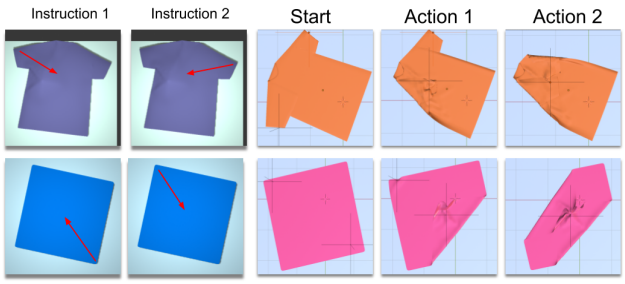

Fig. 14: **Simulation Policy Visualization:** Visualization of the policy executed in simulation (with Blender) using learned descriptors for folding tasks 2 and 4 described in VII-A. The first two columns show the corresponding folding instructions from a web interface (pick-and-place actions shown with red arrows) for tasks 2 and 4. The third column shows images of the previously unseen initial configurations of fabrics before the actions, while the last two columns show the result of executing descriptor-parameterized actions. Results suggest that the learned descriptors can be used to successfully perform a variety of folding tasks from varying initial configurations.

F. Physical Trial Trajectories

In this section, we present additional trials of the physical experiments conducted using the descriptor-based policies for smoothing (Figure 10, Figure 11) and folding (Figure 12).

G. Descriptor Quality Ablations

To investigate the quality of the learned descriptors with training process described in Section V, we perform four sets of ablation studies. We evaluate the quality of learned descriptors in a manner similar to Sundaresan *et al.* [41] by evaluating the ℓ_2 pixel distance of the pixel match error on a set of 100 pairs of held-out validation set images, where for each we sample 100 pixel pairs. We study the effect of training descriptors on (1) RGB or depth images, (2) using descriptor dimension 3, 9, or 16, (3) using 200, 450, or 700 ground-truth annotated images, and (4) whether domain randomization is

used or not. Results suggest that the learned descriptors are best with RGB data, with descriptor dimension between 3 and 16 and with domain randomization, though the performance is generally insensitive to the parameter choices, suggesting a robust training procedure. Based on these results, we use RGB images with domain randomization, and with descriptor dimension 3 for all simulated experiments for both the t-shirt and square fabric. We use RGB, domain-randomized, 9-dimensional descriptors for real fabric experiments. See Figure 13 for plots.