Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images

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 and complex dynamics. In this paper, we learn visual representations of deformable fabric by training dense object descriptors that capture correspondences across images of fabric in various configurations. The learned descriptors capture higher level geometric structure, facilitating design of explainable policies. We demonstrate that the learned representation facilitates multi-step fabric smoothing and folding tasks on two real physical systems, the da Vinci surgical robot and the ABB YuMi given high level demonstrations from a supervisor. The system achieves a 78.8% average task success rate across six fabric manipulation tasks. See `https://tinyurl.com/fabric-descriptors` for supplementary material and videos.**

I. Introduction

Robot fabric manipulation has applications in folding laundry [4, 16, 23, 45], bed making [35], surgery [36, 41, 42], and manufacturing [25, 44]. However, while robots are able to learn general purpose policies to manipulate a variety of rigid objects with increasing reliability [6, 13, 18, 20, 26], learning reliable policies for manipulating deformable objects remains an open problem due to difficulties in sensing and control. While there is significant prior work both on geometric [1, 21, 33, 45] and learning based approaches [34, 35, 46] for fabric manipulation, these approaches often involve designing or learning task-specific manipulation policies, making it difficult to efficiently reuse information for different tasks.

In this work, we decouple perception from control and learn a dense geometrically structured visual representation for fabric without using any task-specific data. Then, this representation can be used to design intuitive policies for different fabric smoothing and folding tasks at test-time. To do this, we build on work by Sundaresan *et al.* [39], which leverages dense object descriptors [6] to learn visual representations for rope using synthetic depth data in simulation and then uses these representations to tie knots and achieve various rope configurations. We experimentally demonstrate that the algorithms for descriptor learning in [39] can be used to learn correspondences between different fabric configurations using task-agnostic data collected entirely in simulation. We then show that these correspondences can be used to perform a variety of tasks at test-time to imitate supplied demonstrations by designing geometric controllers built on top of the learned correspondences. This may, for example, enable a robot to manipulate fabric from web-based instructions for folding tasks. See Figure 1 for an overview.

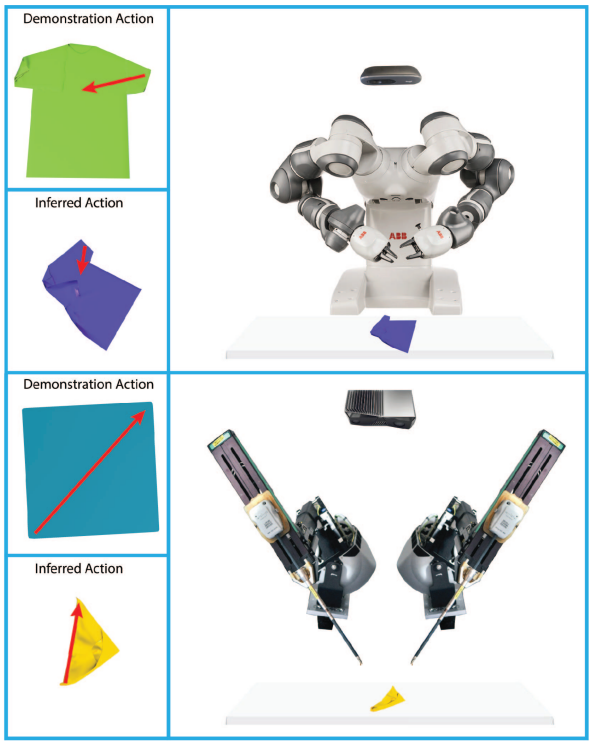

Fig. 1: **Fabric manipulation experiments:** We use the ABB YuMi (top) and the da Vinci Research Kit (bottom) for fabric manipulation experiments with dense object descriptors that produce interpretable actions for smoothing or folding even for fabric configurations dissimilar from those in demonstrations.

This paper makes 3 contributions: (1) applying dense object descriptors from [6, 39] to fabric from synthetically generated data, (2) simulation experiments demonstrating that the learned descriptors can be used to design fabric manipulation policies for smoothing and folding which are robust to unseen fabric configurations and colors and (3) physical experiments on both the da Vinci Research Kit (dVRK) and the ABB YuMi suggesting that the learned descriptors transfer effectively on two different robotic systems.

II. Related Work

Learning fabric manipulation policies is an active area of robotics research [2, 17, 31]. To make fabric manipulation tractable, most prior work makes specific assumptions on the fabric's initial configurations or are tailored to particular tasks. For example, Sun *et al.* [37, 38] perform effective

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

fabric smoothing conditioned on a near-flat starting fabric, and Seita *et al*. [34, 35] followed-up by generalizing to a wider range of initial fabric states, but tailor policies specifically for smoothing. Jia *et al*. [11, 12] show impressive human-robot fabric manipulation under the assumption that fabric has already been grasped, and Schulman *et al.* [33] demonstrate deformable object manipulation while requiring task-specific kinesthetic demonstrations. Other work relies on "vertically smoothing" fabrics using gravity [4, 15, 16, 21, 24] to standardize the initial configuration and to expose fabric corners before attempting the task, which is difficult for large fabrics or single-armed robots. 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.

Due to the recent success of sim-to-real transfer [30, 43], many recent papers leverage simulation to learn fabric manipulation policies. Seita *et al.* [34] and Wu *et al.* [46] use imitation learning (DAgger [29]) and reinforcement learning (Soft Actor-Critic [8]), respectively, for fabric smoothing. Similarly, Matas *et al.* [22] and Jangir *et al.* [10] learn fabric folding policies by using deep reinforcement learning augmented with task-specific demonstrations. These works use simulation to optimize fabric manipulation policies for specific tasks. In follow-up and concurrent work, Hoque *et al.* [9] and Yan *et al.* [47] 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 captures its geometric structure without task-specific data or a model of the environment and then use this representation to design intuitive controllers for a several fabric manipulation tasks from different starting configurations.

We learn visual representations for fabric by using dense object descriptors, which were introduced by [6, 32] and shown to enable task oriented manipulation of various rigid and slightly deformable objects [6]. Here, a deep neural network is used 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 [6], assembly [48], or for learning from demonstrations [7]. Sundaresan *et al.* [39] 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. We apply the techniques from [39] for 2-D fabric manipulation.

III. Problem Definition

A. Assumptions

In this work, we assume a robot manipulates a deformable fabric with initial configuration ξ_1 on a planar workspace with corresponding RGB image $I_1 \in \mathbb{R}^{W \times H \times 3}$. We consider two fabric shapes: t-shirts and squares. We assume the tasks can be completed by a sequence of actions where each includes grasping at a *pick point*, pulling to a *place point* without changing the orientation of the end-effector, and releasing at the place point. This is equivalent to the action space in Seita *et al.* [34] and Wu *et al.* [46]; we will investigate more complex action primitives such as pushing and rotation in future work. We consider smoothing and folding tasks and assume the availability of folding and smoothing instructions in the form of a sequence of pick and place actions from some initial fabric configuration. These demonstrations can be collected offline, such as through a web interface where a user clicks on an image of fabric to indicate pick and place point pixels. These demonstrations can then be used to perform the desired task from different initial fabric configurations by learning correspondences between different fabric configurations. Note that even if the initial fabric configuration were held fixed between the provided demonstration and at test-time, if the robot simply executes the actions in the demonstration, this may result in different configurations due to control errors and variations in fabric physics (e.g. stiffness, inertial effects, friction).

B. Task Definition

Let an action at step j be denoted 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.

Given a sequence of actions $(\mathbf{a}_j)_{j=1}^n$ and a fabric in configuration ξ_1, the goal is to generate a corresponding sequence of actions for a fabric in some previously unseen configuration ξ_2. Specifically, we will do this by generating a new sequence of actions $\left(\mathbf{a}'_j\right)_{j=1}^n$:

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

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, so we leverage dense object descriptors [6] 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, a new open-source simulation and rendering engine [3] released in mid-2019, to both create large synthetic RGB training datasets and to model the fabric dynamics for simulated experiments using its in-built fabric

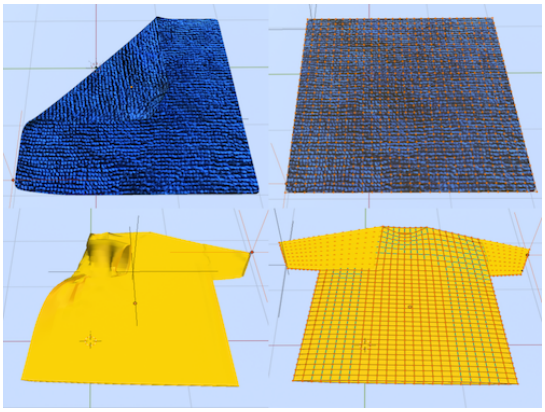

Fig. 2: **Fabric Meshes:** Examples of the meshes generated in Blender for both square cloth (top row) and t-shirts (bottom row). The vertices in the top right and bottom right images are highlighted.

solver based on [27, 28]. 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. These meshes can be easily constructed in Blender by starting with a planar grid of vertices and then removing vertices and edges to create desired shapes.

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. 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 I.

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. Each vertex also exerts repulsive forces within a self-contained virtual sphere on vertices both within cloth and in surrounding objects, to simulate self-collisions and collisions with other objects.

In addition to arbitrary drops, we can simulate finer-grained manipulation of the mesh including grasps, pulls, and folds. 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 to a hook object. 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. This generalized form of manipulation allows us to easily execute experiments in simulation and to generate oracle demonstrations for both real and simulated experiments. See Section VI for more details on these demonstrations.

TABLE I: 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

V. Dense Shape Descriptor Training

A. Dense Object Descriptor Training Procedure

We consider an environment with a deformable fabric sheet on a flat tabletop and learn policies that perform smoothing and folding tasks. The policies we train use point-pair correspondences that are generated between overhead images of the fabric in different configurations. We generate deformation invariant correspondences by training dense object descriptors [6, 39] on synthetically generated images of the fabric in different configurations.

In Florence *et al.* [6], input images are mapped to descriptor volumes, where each pixel has a corresponding descriptor vector. Descriptors are generated by a Siamese network and are guided closer together for corresponding pixels in images and guided apart by at least some margin M for non-corresponding pairs by minimizing a pixel-wise contrastive loss function during training [6]. Corresponding pairs of pixels represent the same point on an object. In Sundaresan *et al.* [39], this descriptor training procedure is applied to synthetic depth images of rope in different configurations, and the learned representation is used to design policies for rope manipulation tasks such as knot-tying on a real robot. In this work, we take a similar approach by training a Siamese network to output descriptors that are close for corresponding pairs of pixels and separated for non-corresponding pixel pairs for overhead images of fabric in different configurations. 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 mesh's vertices. Figure 3 demonstrates the pipeline for training descriptors for policy design. The learned descriptors can then be used to approximate correspondence function $d_{I \to I'}$ as defined in Section III, which makes it possible to generate explainable geometric controllers (Section VI).

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 starting 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 other 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 export RGB images,

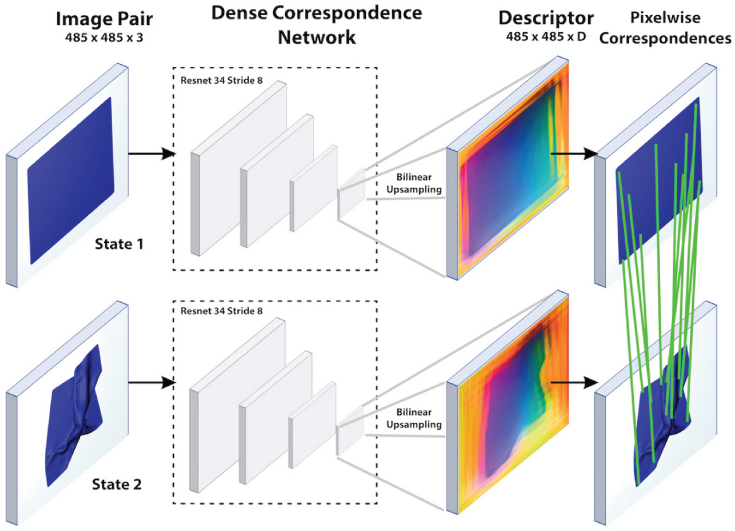

Fig. 3: **System Overview:** pipeline for training dense object nets and then using them for robot fabric manipulation. Left: we train a dense object network on pairs of simulated fabric images to learn pixel-wise correspondences using a pixelwise contrastive loss. Right: we use the learned descriptors 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."

pixel-wise annotations, segmentation masks and depth images using Blender's Z-buffer output. We then make use of domain randomization [30, 43] by rendering images of the scene while randomizing parameters including mesh size, lighting, camera pose, texture, color and specularity. 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 [19] and "paste" the rendered fabric mask on top.

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. We found no change in performance when including these unsettled images of the fabric in the dataset. We generated two (domain-randomized) datasets, one including only t-shirts and one including only square fabric, and trained two separate models on them which we used for the experiments in Section VII. For reference, generating a single dataset of 3,500 images with 729 annotations per image takes approximately 2 hours on a 2.6GHz 6-core Intel Core i7 MacBook Pro.

VI. Descriptor-Parameterized Policy Design

Once descriptors are learned as described in Section V, we use the geometric structure of the learned visual representations to design algorithms based on the descriptors to perform specific tasks. As in [39], the learned descriptors are used to identify semantically relevant pixels in the current frame, and then actions are designed to manipulate these keypoints for a specific task. For example, a descriptor parameterized 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. Another example of an intuitive descriptor-parameterized task would be to grasp the right sleeve of a t-shirt and place it at the chest area to emulate a high level folding instruction. In later settings, even if the fabric is in an unseen configuration, if the descriptor vector for the grasp point and place point are recorded for one example of the above folding task, a new grasp point and place point can be found by finding the closest points in descriptor space in the new configuration to the grasp and place points in the original configuration respectively. With a single high-level demonstration of a task such as folding a specific type of fabric, we ask the robot to repeat the task on a series of unseen configurations and instances of the fabric.

A. Fabric Smoothing

In the fabric smoothing task, the robot starts with a crumpled fabric and spreads it into a smooth configuration on the planar workspace as in Seita *et al.* [34]. To complete this task, we execute the simulated algorithmic supervisor used in [34] with ground truth information that iterates over the fabric corners and pulls each to target locations on an underlying plane. However, as we lack ground-truth corner positions, we use the descriptor mapping to identify the current corner in the image as a grasp point by finding the closest match in descriptor space to the descriptor corresponding to the corner in a saved image. The place point is a pre-specified target location in the workspace and not a position defined by a point on the cloth.

B. Fabric Folding

In the fabric folding task, we record a sequence of grasp and place points by clicking on an image of a flat fabric. We execute this sequence of instructions on the fabric, varying its

initial configuration and color. As described in Sections III-B and VI, we use the dense object descriptors to compute corresponding actions on the real fabric, even though its configuration was not seen in the demonstration.

VII. Experiments

We experimentally evaluate (1) the quality of the learned descriptors and their sensitivity to training parameters, (2) the performance of the descriptor-based policies from Section VI in simulation, and (3) the performance of the policies on two physical robotic systems. Results suggest that the learned descriptors and the resulting policies are robust to changes in fabric configuration and color. Simulated experiments are performed in Blender 2.8 [3] in the simulated environment described in Section IV while physical experiments are performed on the da Vinci Research Kit (dVRK) [14] and the ABB YuMi. See Section VII-C for more details on how we conduct the physical experiments.

A. Tasks

We consider a total of 6 fabric manipulation tasks executed on a set of 3 t-shirts and 3 square fabrics in the real world to evaluate the proposed descriptor based manipulation approach. Additionally, we execute a subset of these tasks in simulation. See Section VII-B for more details on the simulation experiments. A single visual demonstration consisting of either 1 or 2 actions is provided to generate a policy which the robot then tries to emulate in the same number of actions. We separate tasks into the following categories:

1) *Single Fold:* A single fold where one corner is pulled to its opposing corner.
2) *Double Inward Fold:* Two opposing corners are folded to the center of the fabric.
3) *Double Triangle Fold:* Two sets of opposing corners are aligned with each other.
4) *T-Shirt Sleeves Fold:* The two sleeves of a t-shirt are folded to the center of the shirt.
5) *Smoothing:* Fabric is flattened from a crumpled state.
6) *Smoothing + Double Triangle Fold:* Fabric is smoothed then the double triangle folds are executed.

See Sections VII-B and VII-C for more details on how we evaluate performance on these tasks.

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 good 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. Physical Experiment Details

We execute the physical fabric folding and smoothing experiments on the da Vinci Research Kit (dVRK) [14] and ABB YuMi robot. The dVRK is equipped with the Zivid OnePlus RGBD sensor that outputs a 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 fabrics of the same dimension. 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 which allows us to manipulate $12" \times 12"$ pieces of fabric. In this setup we use a 1080p Logitech webcam to collect overhead color images. 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 comparable to those shown in Figure 2 and deform it by informally 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.

Like in the simulation experiments, we consider an outcome 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. The smoothing task is evaluated by computing the coverage of the cropped workspace before and after execution.

D. Descriptor Quality Evaluation

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.* [39] 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

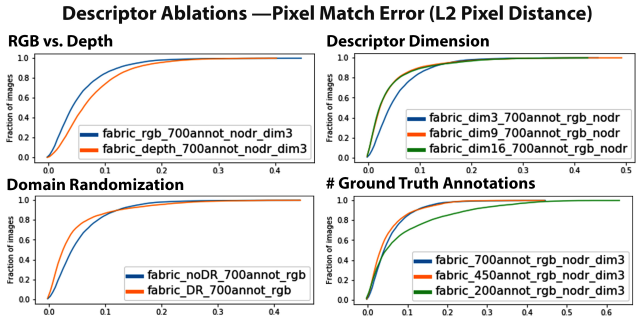

Fig. 4: **Ablation studies:** We study the sensitivity of the learned dense object descriptors as described in Sections V and VII-D to training parameters. Starting from top left, and proceeding clockwise, we test the effect of testing on RGB vs depth images, on the descriptor dimension (either 3, 9, 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.

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.

E. 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 II).

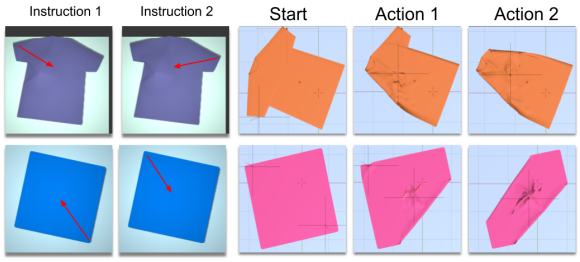

Fig. 5: **Simulation Policy Visualization:** Visualization of the policy executed in simulation (with Blender) using learned descriptors for folding tasks 2 and 4 described in VII-A. The first two columns show folding instructions from a web interface (pick-and-place actions shown with red arrows). The third column shows images of the starting state of fabrics before the actions, while the last two columns show the result of executing actions. Results suggest that the learned descriptors can be used to successfully perform a variety of folding tasks from varying initial configurations.

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

TABLE II: **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.

F. Physical Experiment Results

We evaluate the smoothing and folding policies on both the YuMi and dVRK on square fabrics and t-shirts. We observe that the descriptor-parameterized policies are able to successfully complete almost all folding tasks at least 80% of the time, and the smoothing policies are able to increase coverage of the cloth to over 83% (Table IV). The execution of the smoothing policy followed by the double triangle folding policy results in successful task completion $3/5$ and $4/5$ times on the YuMi and dVRK respectively.

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

TABLE III: **Physical Fabric Smoothing Experiments:** We test the smoothing policies designed in Section VI on the YuMi and the dVRK robots. Both robots are able to increase coverage during the smoothing task by $11-22$ percent on average.

Task	Robot	Success Rate
Single Fold	YuMi	8/10
Single Fold	dVRK	9/10
Double Inward Fold	YuMi	8/10
Double Inward Fold	dVRK	9/10
Double Triangle Fold	YuMi	6/10
Double Triangle Fold	dVRK	8/10
T-Shirt Sleeves Fold	YuMi	8/10
Smoothing + Double Triangle Fold	YuMi	3/5
Smoothing + Double Triangle Fold	dVRK	4/5

TABLE IV: **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 80 percent of the time. The YuMi is able to perform the smoothing then folding task $3/5$ times and the dVRK is able to do so $4/5$ times.

VIII. Discussion and Future Work

This paper leverages dense object descriptors to present an approach for multi-task fabric manipulation. Experiments suggest that these dense descriptors are capable of learning rich visual representations of highly deformable fabrics allowing for explainable manipulation policies without explicitly modeling fabric dynamics. In future work, we will explore using the learned descriptors as a representation for imitation and reinforcement learning and investigate ways to combine the presented approach with image-level descriptors for segmenting videos of fabric manipulation like in [40]. We are also interested in hierarchical fabric manipulation policies, where the policies described in this paper provide high level commands, and a lower level controller is trained to executed the desired commands. Finally, we will apply descriptors to more complex fabric manipulation tasks, such as wrapping rigid objects or dealing with multiple fabrics simultaneously.

IX. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, Berkeley Deep Drive (BDD), the Real-Time Intelligent Secure Execution (RISE) Lab, the CITRIS "People and Robots" (CPAR) Initiative, 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, Siemens, Google, 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 an NPSC Fellowship.

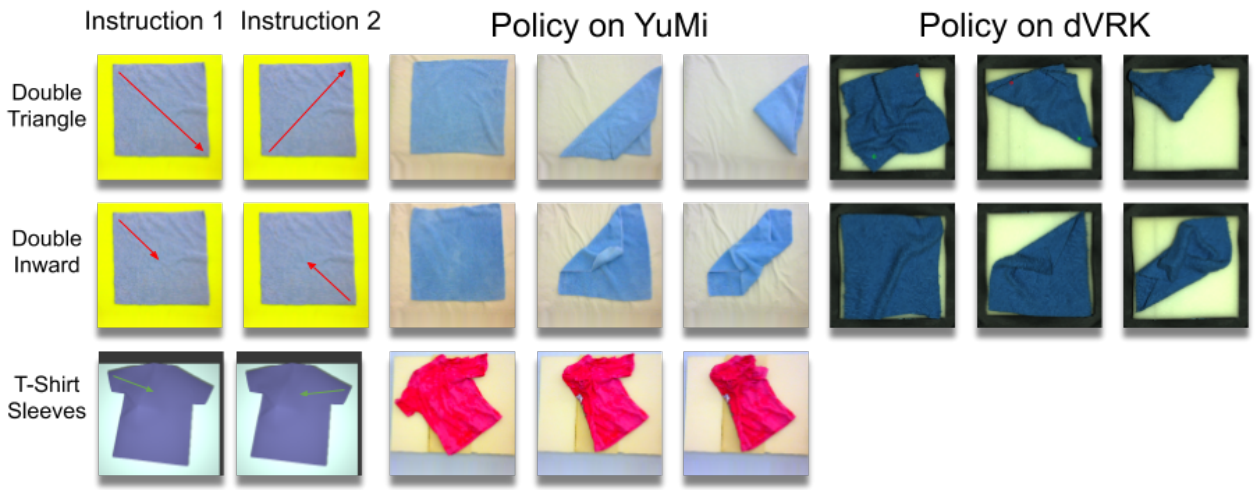

Fig. 6: **Physical Policy Visualization:** We visualize policy execution on both the YuMi and the dVRK for tasks 2, 3, and 4 described in Section VII-A. Similar to Figure 5, the first two columns show folding instructions from a human supervisor, the third and sixth columns show the starting configuration of the fabrics in the YuMi and dVRK workspaces, respectively. Columns 4 through 5, and 7 through 8, show the results of the executed action by both robots.

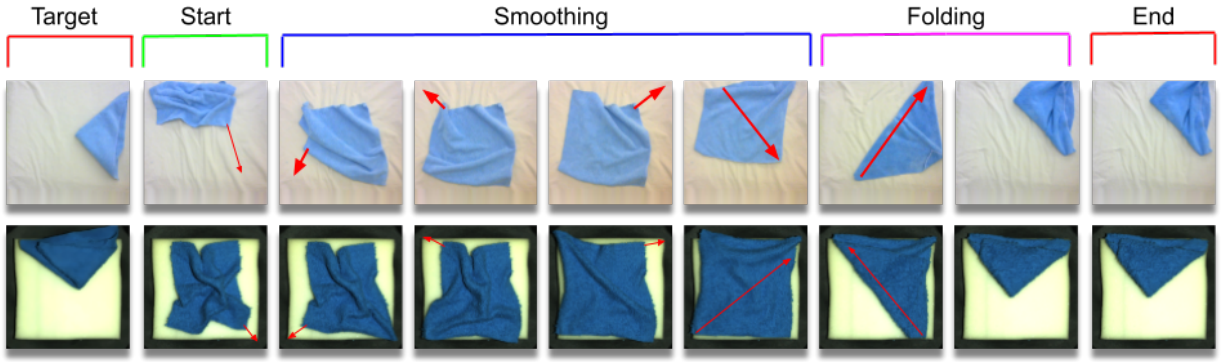

Fig. 7: **Full Folding Sequence:** Time-lapse of a sequence of 6 actions taken by the YuMi (top row) and dVRK (bottom row), with actions overlaid by red arrows, to successively smooth a wrinkled fabric and then fold it according to task 3 in Section VII-A.

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. 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.

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

[8] 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.

[9] 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", *arXiv:2003.09044*, 2020.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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", *arXiv:1910.04854*, 2019.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] A. K. Tanwani, P. Sermanet, A. Yan, R. Anand, M. Phielipp, and K. Goldberg, "Motion2Vec: Semi-Supervised Representation Learning from Surgical Videos", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations", *arXiv:1910.13439*, 2019.

[47] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation", *arXiv:2003.05436*, 2020.

[48] 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 images from additional physical trials executed on the dVRK and YuMi.

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 10 is a visualization of the learned descriptors and Figure 8 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 I. A visualization of these steps can be seen in the top row of Figure 9. 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.

Fig. 8: 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.

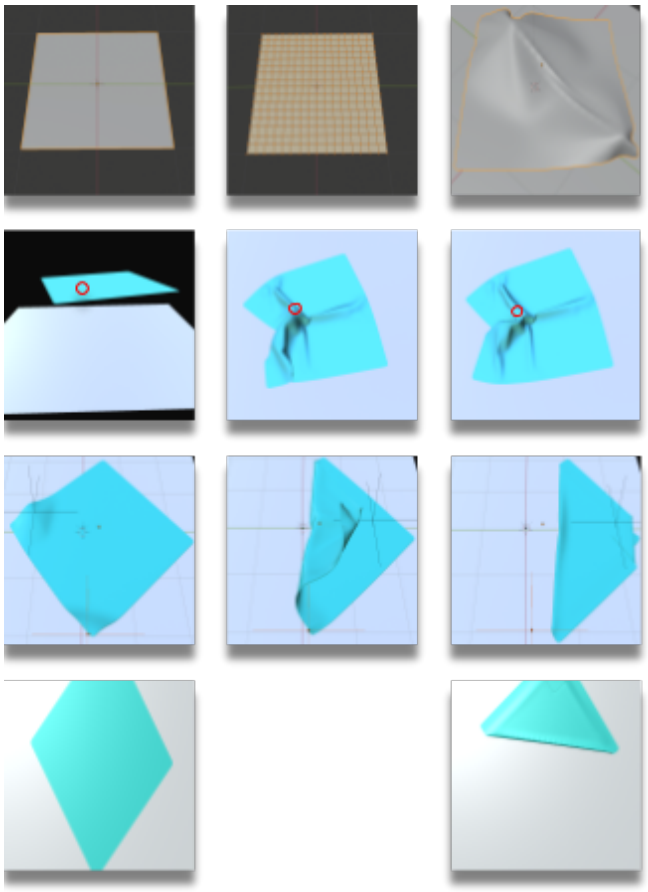

Fig. 9: The top row illustrates the the process of creating cloth in Blender from a default square mesh. The second row is an example of a starting configuration generated by dropping the cloth from a fixed height and pinning a single arbitrary vertex. The pinned vertex is labeled by the red circle. 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.

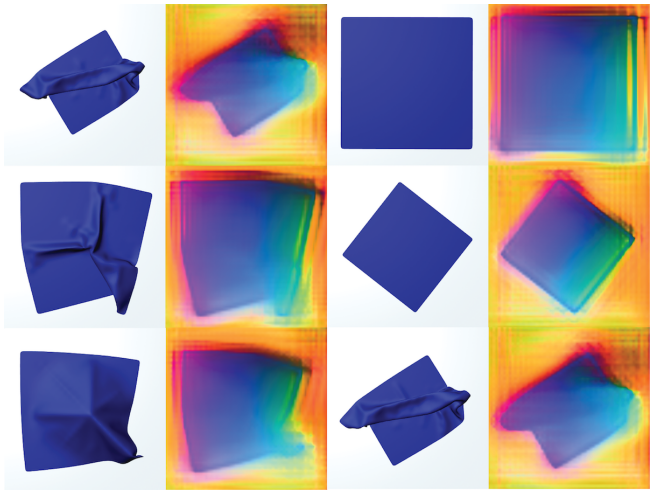

Fig. 10: Visualization of the 3-dimensional descriptors learned via the training procedure described in Section V by mapping each pixel's descriptor vector to an RGB vector. Thus, similar colors across the images of columns two and four represent corresponding points on the square cloth.

2) Starting Configurations and Actions: To generate varied starting configurations, we simulate dropping the fabric from

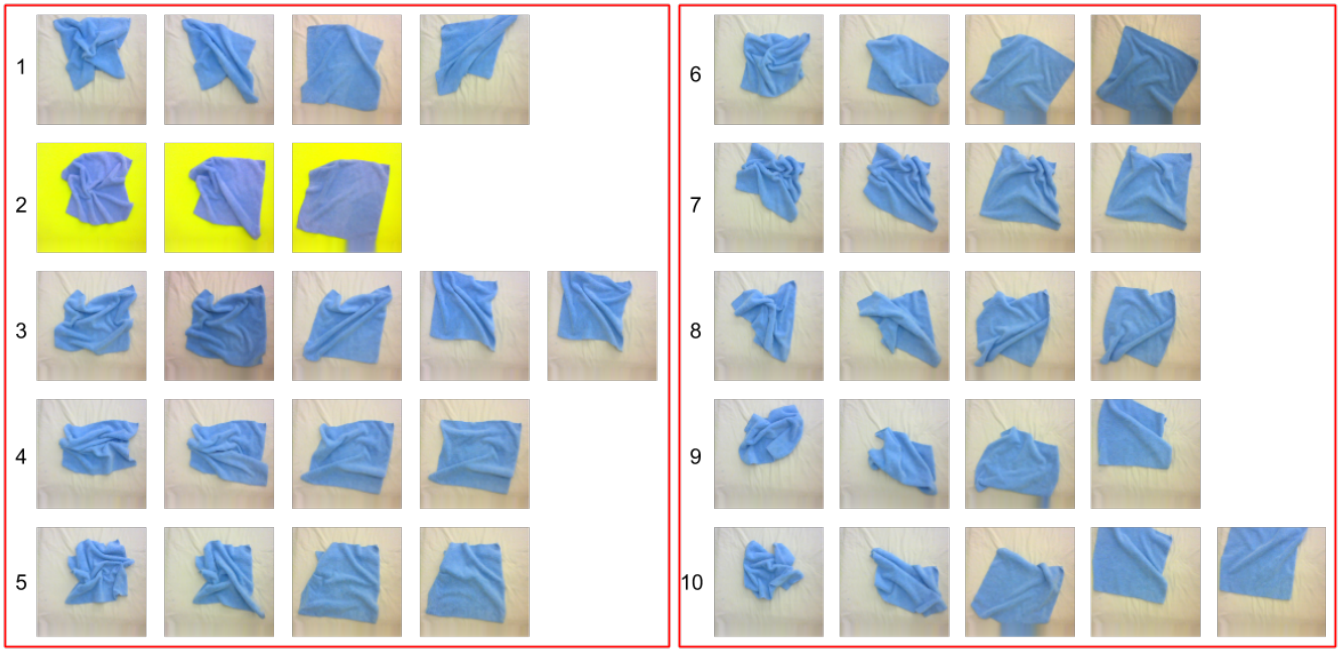

Fig. 11: Additional rollouts of the smoothing task from randomly chosen starting configurations. The learned descriptors are used to locate the corners of the fabric and successively pull them to a reference location in an image of the flat cloth.

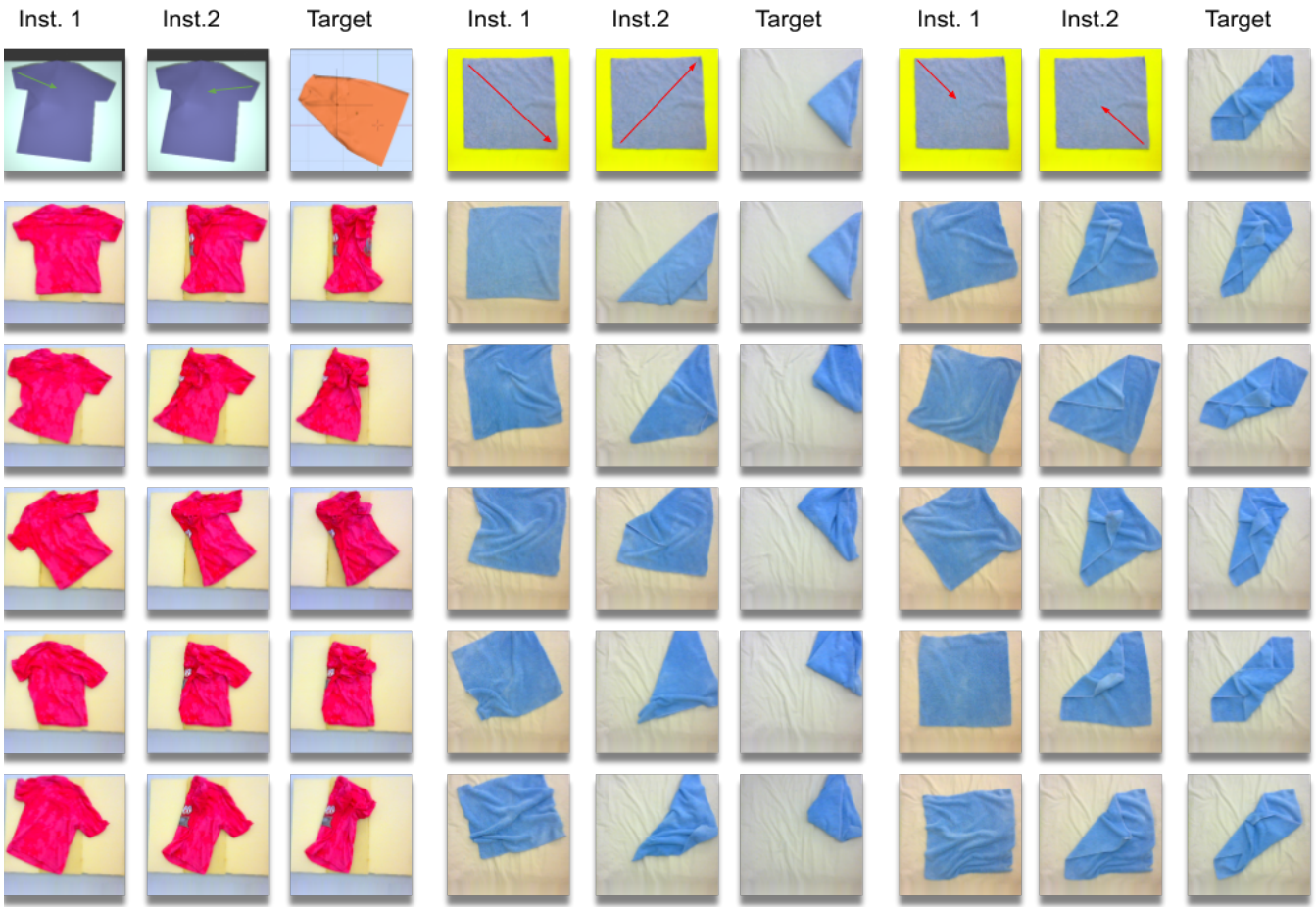

Fig. 12: Additional rollouts of the folding tasks described in Section VII-A from arbitrary starting configurations. The left column, center column and right column contain results for tasks 4, 3 and 2 respectively.

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 9. 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 9.

B. 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 10), and observe that corresponding pixels of the cloth map to similar colors in the descriptor volumes across configurations.

C. Physical Trial Trajectories

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