Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor

Daniel Seita[1], Aditya Ganapathi[1], Ryan Hoque[1], Minho Hwang[1], Edward Cen[1], Ajay Kumar Tanwani[1], Ashwin Balakrishna[1], Brijen Thananjeyan[1], Jeffrey Ichnowski[1], Nawid Jamali[2], Katsu Yamane[2], Soshi Iba[2], John Canny[1], Ken Goldberg[1]

***Abstract*— Sequential pulling policies to flatten and smooth fabrics have applications from surgery to manufacturing to home tasks such as bed making and folding clothes. Due to the complexity of fabric states and dynamics, we apply deep imitation learning to learn policies that, given color (RGB), depth (D), or combined color-depth (RGBD) images of a rectangular fabric sample, estimate pick points and pull vectors to spread the fabric to maximize coverage. To generate data, we develop a fabric simulator and an algorithmic supervisor that has access to complete state information. We train policies in simulation using domain randomization and dataset aggregation (DAgger) on three tiers of difficulty in the initial randomized configuration. We present results comparing five baseline policies to learned policies and report systematic comparisons of RGB vs D vs RGBD images as inputs. In simulation, learned policies achieve comparable or superior performance to analytic baselines. In 180 physical experiments with the da Vinci Research Kit (dVRK) surgical robot, RGBD policies trained in simulation attain coverage of 83% to 95% depending on difficulty tier, suggesting that effective fabric smoothing policies can be learned from an algorithmic supervisor and that depth sensing is a valuable addition to color alone. Supplementary material is available at `https://sites.google.com/view/fabric-smoothing`.**

I. Introduction

Robot manipulation of fabric has applications in senior care and dressing assistance [13], [14], [15], sewing [39], ironing [23], laundry folding [24], [28], [42], [54], fabric upholstery manufacturing [31], [48], and handling gauze in robotic surgery [46]. However, fabric manipulation is challenging due to its infinite dimensional configuration space and unknown dynamics.

We consider the task of transforming fabric from a rumpled and highly disordered starting configuration to a smooth configuration via a series of grasp and pull actions. We explore a deep imitation learning approach based on a Finite Element Method (FEM) fabric simulator with an algorithmic supervisor and use DAgger [36] to train policies. Using color and camera domain randomization [37], [47], learned policies are evaluated in simulation and in physical experiments with the da Vinci Research Kit (dVRK) surgical robot [19]. Figure 1 shows examples of learned smoothing episodes in simulation and the physical robot.

This paper contributes: (1) an open-source simulation environment and dataset for evaluation of fabric smoothing with three difficulty tiers of initial fabric state complexity, (2) deep imitation learning of fabric smoothing policies from an algorithmic supervisor using a sequence of pick and pull actions, and (3) transfer to physical experiments on a da Vinci surgical robot with comparisons of coverage performance using color (RGB), depth (D), or RGBD input images.

[1]AUTOLAB at the University of California, Berkeley, USA.
[2]Honda Research Institute, USA.
Correspondence to `seita@berkeley.edu`

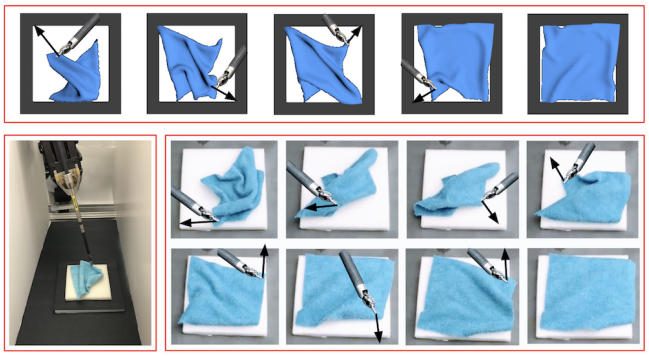

Fig. 1: Learned policies executed in simulation and with a physical da Vinci surgical robot, with actions indicated from the overlaid arrows. Policies are learned in simulation using DAgger with an algorithmic supervisor that has full state information, using structured domain randomization with color and/or depth images. The 4-action smoothing episode in simulation (top) increases coverage from 43% to 95%. The 7-action episode on the physical da Vinci robot (bottom) increases coverage from 49% to 92%.

II. Related Work

Well-known research on robotic fabric manipulation [5], [38], [8] uses bilateral robots and gravity to expose corners. Osawa et al. [30] proposed a method of iteratively re-grasping the lowest hanging point of a fabric to flatten and classify fabrics. Subsequently, Kita et al. [20], [21] used a deformable object model to simulate fabric suspended in the air, allowing the second gripper to grasp at a desired point. Follow-up work generalized to a wider variety of initial configurations of new fabrics. In particular, Maitin-Shepard et al. [26], Cusumano-Towner et al. [9], and Doumanoglou et al. [11] identified and tensioned corners to fold laundry or to bring clothing to desired positions. These methods rely on gravity to reveal corners of the fabric. We consider the setting where a single armed robot adjusts a fabric strewn across a surface without lifting it entirely in midair, which is better suited for larger fabrics or when robots have a limited range of motion.

A. Reinforcement Learning for Fabric Manipulation

Reinforcement Learning (RL) [45] is a promising method for training policies that can manipulate highly deformable objects. In RL applications for folding, Matas et al. [27] assumed that fabric is flat, and Balaguer et al. [2] began with fabric gripped in midair to loosen wrinkles. In contrast, we consider the problem of bringing fabric from a highly rumpled configuration to a flat configuration. Using model-based RL, Ebert et al. [12] were able to train robots to fold pants and fabric. This approach requires executing a physical robot for many thousands of actions and then training a video prediction model. In surgical robotics, Thananjeyan et al. [46] used RL to learn a tensioning policy to cut gauze, with one arm pinching at a pick point to let the other arm cut. We focus on fabric smoothing without tensioning, and additionally consider cases where the initial fabric state may be highly rumpled and disordered. In concurrent and independent work, Jangir et al. [18] used deep reinforcement learning with demonstrations to train a policy using fabric state information for simulated dynamic folding tasks.

B. Fabric Smoothing

In among the most relevant prior research on fabric smoothing, Willimon et al. [51] present an algorithm that pulls at eight fixed angles, and then uses a six-step stage to identify corners from depth images using the Harris Corner Detector [16]. They present experiments on three simulated trials and one physical robot trial. Sun et al. [43] followed up by attempting to explicitly detect and then pull at wrinkles. They measure wrinkledness as the average absolute deviation in a local pixel region for each point in a depth map of the fabric [35] and apply a force perpendicular to the largest wrinkle. Sun et al. evaluate on eight fixed, near-flat fabric starting configurations in simulation. In subsequent work, Sun et al. [44] improved the detection of wrinkles by using a shape classifier as proposed in Koenderink and van Doorn [22]. Each point in the depth map is classified as one of nine shapes, and they use contiguous segments of certain shapes to define a wrinkle. While Sun et al. were able to generalize the method beyond a set of hard-coded starting states, it was only tested on nearly flat fabrics in contrast to the highly rumpled configurations we explore.

In concurrent and independent work, Wu et al. [53] trained an image-based policy for fabric smoothing in simulation using deep reinforcement learning, and then applied domain randomization to transfer it to a physical PR2 robot.

This paper extends prior work by Seita et al. [40] that only estimated a pick point and pre-defined the pull vector. In contrast, we learn the pull vector and pick point simultaneously. Second, by developing a simulator, we generate far more training data and do not need to run a physical robot. This enables us to perform systematic experiments comparing RGB, D, and RGBD image inputs.

III. Problem Statement

Given a deformable fabric and a flat fabric plane, each with the same rectangular dimensions, we consider the task of manipulating the fabric from a start state to one that maximally covers the fabric plane. We define an *episode* as one instance of the fabric smoothing task.

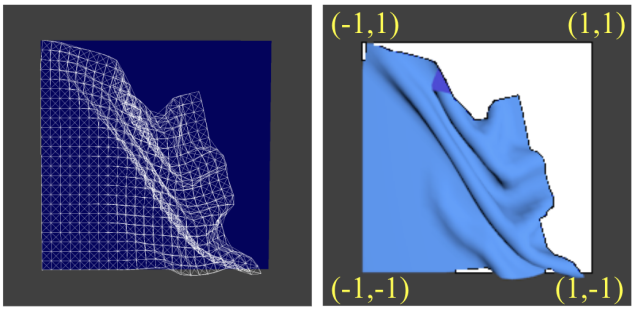

Fig. 2: FEM fabric simulation. Left: a wireframe rendering, showing the 25×25 grid of points and the spring-mass constraints. Right: the corresponding image with the white fabric plane, rendered using Blender and overlaid with coordinates. The coverage is 73%, measured as the percentage of the fabric plane covered.

Concretely, let ξ_t be the full state of the fabric at time t with positions of all its points (described in Section IV). Let $\mathbf{o}_t \in O$ be the *image observation* of the fabric at time t, where $O = \mathbb{R}^{H \times W \times c}$ represents the space of images with $H \times W$ pixels, and $c = 1$ channels for depth images, $c = 3$ for color images, or $c = 4$ for combined color and depth (i.e., RGBD) images. Let A be the set of actions the robot may take (see Section IV-A). The task performance is measured with *coverage* $C(\xi_t)$, or the percentage of the fabric plane covered by ξ_t.

We frame this as imitation learning [1], where a supervisor provides data in the form of paired observations and actions $\mathcal{D} = \{(\mathbf{o}_t, \mathbf{a}_t)\}_{t=1}^{N}$. From $\mathcal{D}$, the robot's goal is to learn a policy $\pi : O \rightarrow A$ that maps an observation to an action, and executes sequentially until a coverage threshold or iteration termination threshold is reached.

IV. Fabric and Robot Simulator

We implement a Finite Element Method (FEM) [4] fabric simulator and interface with an OpenAI gym environment design [6]. The simulator is open source and available on the project website. Alternative fabric simulators exist, such as from Blender [7], which is a popular open-source computer graphics software toolkit. Since 2017, Blender has had significant improvements in physics and realism of fabrics, but these changes are only supported in Blender 2.80, which does not support headless rendering of images and therefore meant we could not run massive data collection. We downgraded to an older version of Blender, 2.79, which supports headless rendering, to create images generated from the proposed custom-built fabric simulator. MuJoCo 2.0 provides another fabric simulator, but did not support OpenAI-gym style fabric manipulation environments until concurrent work [53].

The fabric (Figure 2) is represented as a grid of 25×25 point masses, connected by three types of springs [34]:

- *Structural*: between a point mass and the point masses to its left and above it.

- *Shear*: between a point mass and the point masses to its diagonal upper left and diagonal upper right.
- *Flexion*: between a point mass and the point masses two away to its left and two above it.

Each point mass is acted upon by both an external gravitational force which is calculated using Newton's Second Law and a spring correction force

$$F_s = k_s \cdot (\|q_a - q_b\|_2 - \ell), \tag{1}$$

for each of the springs representing the constraints above, where k_s is a spring constant, $q_a \in \mathbb{R}^3$ and $q_b \in \mathbb{R}^3$ are positions of any two point masses connected by a spring, and ℓ is the default spring length. We update the point mass positions using Verlet integration [49]. Verlet integration computes a point mass's new position at time $t+\Delta_t$, denoted with $p_{t+\Delta_t}$, as:

$$p_{t+\Delta_t} = p_t + v_t\Delta_t + a_t\Delta_t^2, \tag{2}$$

where $p_t \in \mathbb{R}^3$ is the position, $v_t \in \mathbb{R}^3$ is the velocity, $a_t \in \mathbb{R}^3$ is the acceleration from all forces, and $\Delta_t \in \mathbb{R}$ is a timestep. Verlet integration approximates $v_t\Delta_t = p_t - p_{t-\Delta_t}$ where $p_{t-\Delta_t}$ is the position at the last time step, resulting in

$$p_{t+\Delta_t} = 2p_t - p_{t-\Delta_t} + a_t\Delta_t^2 \tag{3}$$

The simulator adds damping to simulate loss of energy due to friction, and scales down v_t, leading to the final update:

$$p_{t+\Delta_t} = p_t + (1-d)(p_t - p_{t-\Delta_t}) + a_t\Delta_t^2 \tag{4}$$

where $d \in [0, 1]$ is a damping term, which we tuned to 0.02 based on visually inspecting the simulator.

We apply a constraint from Provot [34] by correcting point mass positions so that spring lengths are at most 10% greater than ℓ at any time. We also implement fabric-fabric collisions following [3] by adding a force to "separate" two points if they are too close.

The simulator provides access to the full fabric state ξ_t, which contains the exact positions of all 25×25 points, but does not provide image observations $\mathbf{o}_t$ which are more natural and realistic for transfer to physical robots. To obtain image observations of a given fabric state, we create a triangular mesh and render using Blender.

A. Actions

We define an action at time t as a 4D vector which includes the pick point (x_t, y_t) represented as the coordinate over the fabric plane to grasp, along with the pull direction. The simulator implements actions by grasping the top layer of the fabric at the pick point. If there is no fabric at (x_t, y_t), the grasp misses the fabric. After grasping, the simulator pulls the picked point upwards and towards direction $\Delta x_t \in [-1, 1]$ and $\Delta y_t \in [-1, 1]$, deltas in the x and y direction of the fabric plane. In summary, actions $\mathbf{a}_t \in A$ are defined as:

$$\mathbf{a}_t = \langle x_t, y_t, \Delta x_t, \Delta y_t \rangle \tag{5}$$

representing the pick point coordinates (x_t, y_t) and the pull vector $(\Delta x_t, \Delta y_t)$ relative to the the pick point.

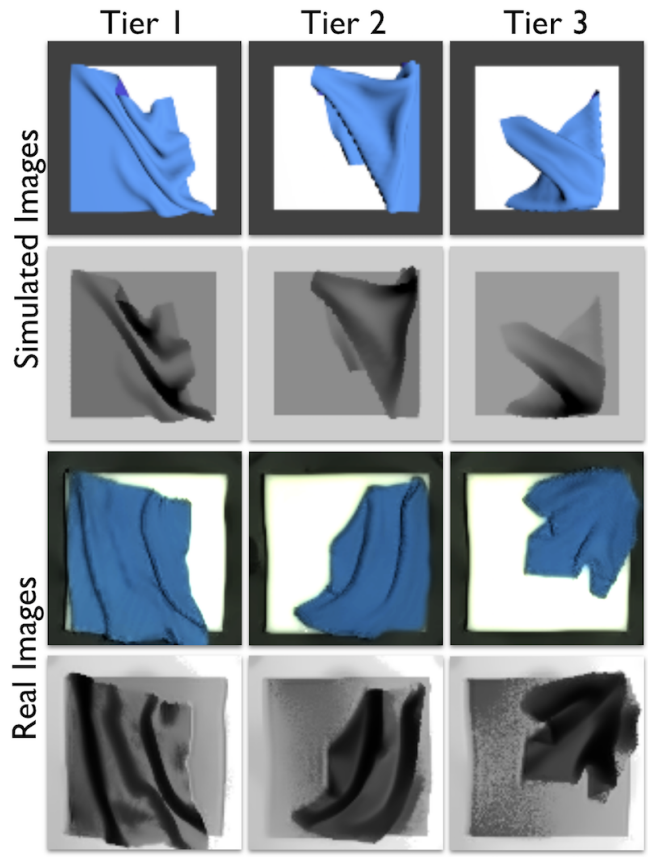

Fig. 3: Initial fabric configurations drawn from the distributions specified in Section IV-B, with tiers grouped by columns. The first two rows show representative simulated color (RGB) and depth (D) images, respectively, while the last two rows show examples of real images from a mounted Zivid One Plus camera, after smoothing and de-noising, which we then pass as input to a neural network policy. Domain randomization is not applied on the simulated images shown here.

B. Starting State Distributions

The performance of a smoothing policy, or more generally any fabric manipulation policy, depends heavily on the distribution of starting fabric states. We categorize episodes as belonging to one of three custom difficulty tiers. For each tier, we randomize the starting state of each episode. The starting states, as well as their average coverage based on 2000 simulations, are generated as follows:

- **Tier 1,** 78.3 ± 6.9**% Coverage (High)**: starting from a flat fabric, we make two short, random pulls to slightly perturb the fabric. All fabric corners remain visible.
- **Tier 2,** 57.6 ± 6.1**% Coverage (Medium)**: we let the fabric drop from midair on one side of the fabric plane, perform one random grasp and pull across the plane, and then do a second grasp and pull to cover one of the two fabric corners furthest from its plane target.
- **Tier 3,** 41.1 ± 3.4**% Coverage (Low)**: starting from a flat fabric, we grip at a random pick point and pull high in the air, drag in a random direction, and then drop, usually resulting in one or two corners hidden.

Figure 3 shows examples of color and depth images of

TABLE I: Results from the five baseline policies discussed in Section V. We report final coverage and the number of actions per episode. All statistics are from 2000 episodes, with tier-specific starting states. Both oracle policies (in bold) perform the best.

Tier	Method	Coverage	Actions
1	Random	25.0 +/- 14.6	2.43 +/- 2.2
1	Highest	66.2 +/- 25.1	8.21 +/- 3.2
1	Wrinkle	91.3 +/- 7.1	5.40 +/- 3.7
1	Oracle	**95.7 +/- 2.1**	**1.76 +/- 0.8**
1	Oracle-Expose	**95.7 +/- 2.2**	**1.77 +/- 0.8**
2	Random	22.3 +/- 12.7	3.00 +/- 2.5
2	Highest	57.3 +/- 13.0	9.97 +/- 0.3
2	Wrinkle	87.0 +/- 10.8	7.64 +/- 2.8
2	Oracle	**94.5 +/- 5.4**	**4.01 +/- 2.0**
2	Oracle-Expose	**94.6 +/- 5.0**	**4.07 +/- 2.2**
3	Random	20.6 +/- 12.3	3.78 +/- 2.8
3	Highest	36.3 +/- 16.3	7.89 +/- 3.2
3	Wrinkle	73.6 +/- 19.0	8.94 +/- 2.0
3	Oracle	**95.1 +/- 2.3**	**4.63 +/- 1.1**
3	Oracle-Expose	**95.1 +/- 2.2**	**4.70 +/- 1.1**

fabric initial states in simulation and real physical settings for all three tiers of difficulty. The supplementary material contains additional examples of images.

V. Baseline Policies

We propose five baseline policies for fabric smoothing.

1) Random: As a naive baseline, we test a random policy that uniformly selects random pick points and pull directions.

2) Highest (Max z): This policy, tested in Seita et al. [40] grasps the highest point on the fabric. We get the pick point by determining p, the highest of the $25^2 = 625$ points from ξ_t. To compute the pull vector, we obtain the target coordinates by considering where p's coordinates would be if the fabric is perfectly flat. The pull vector is then the vector from p's current position to that target. Seita et al. [40] showed that this policy can achieve reasonably high coverage, particularly when the highest point corresponds to a corner fold on the uppermost layer of the fabric.

3) Wrinkle: Sun et al. [43] propose a two-stage algorithm to first identify wrinkles and then to derive a force parallel to the fabric plane to flatten the largest wrinkle. The process repeats for subsequent wrinkles. We implement this method by finding the point in the fabric of largest local height variance. Then, we find the neighboring point with the next largest height variance, treat the vector between the two points as the wrinkle, and pull perpendicular to it.

4) Oracle: This policy uses complete state information from ξ_t to find the fabric corner furthest from its fabric plane target, and pulls it towards that target. When a corner is occluded and underneath a fabric layer, this policy will grasp the point directly above it on the uppermost fabric layer, and the resulting pull usually decreases coverage.

5) Oracle-Expose: When a fabric corner is occluded, and other fabric corners are not at their targets, this policy picks above the hidden corner, but pulls away from the fabric plane target to reveal the corner for a subsequent action.

VI. Simulation Results for Baseline Policies

We evaluate the five baseline fabric smoothing policies by running each for 2000 episodes in simulation. Each episode draws a randomized fabric starting state from one of three difficulty tiers (Section IV-B), and lasts for a maximum of 10 actions. Episodes can terminate earlier under two conditions: (1) if a pre-defined coverage threshold is obtained, or (2) the fabric is out of bounds over a certain threshold. For (1) we use 92% as the threshold, which produces visually smooth fabric (e.g., see the last few images in Figure 4) and avoids supervisor data being dominated by taking actions of short magnitudes at the end of its episodes. For (2) we define a fabric as out of bounds if it has any point which lies at least 25% beyond the fabric plane relative to the full distance of the edge of the plane. This threshold allows the fabric to go slightly off the fabric plane which is sometimes unavoidable since a perfectly smoothed fabric is the same size as the plane. We do not allow a pick point to lie outside the plane.

Table I indicates that both oracle policies attain nearly identical performance and have the highest coverage among the baseline policies, with about 95% across all tiers. The wrinkles policy is the next best policy in simulation, with 91.3%, 87.0%, and 73.6% final coverage for the three respective tiers, but requires substantially more actions per episode.

One reason why the oracle policy still performs well with occluded corners is that the resulting pulls can move those corners closer to their fabric plane targets, making it easier for subsequent actions to increase coverage. Figure 4 shows an example episode from the oracle policy on a tier 3 starting state. The second action pulls at the top layer of the fabric above the corner, but the resulting action still moves the occluded corner closer to its target.

VII. Imitation Learning with DAgger

We use the oracle (not oracle-expose) policy to generate supervisor data and corrective labels. For each tier, we generate 2000 episodes from the supervisor and use that as offline data. We train a fabric smoothing policy in simulation using imitation learning on synthetic images. When behavior cloning on supervisor data, the robot's policy will learn the supervisor's actions on states in the training data, but generalize poorly outside the data distribution [32]. To address this, we use Dataset Aggregation (DAgger) [36], which requests the supervisor to label the states the robot encounters when running its learned policy. A limitation of DAgger is the need for continued access to the supervisor's policy, rather than just offline data. During training, the oracle corner-pulling supervisor is able to efficiently provide an action to each data point encountered by accessing the underlying fabric state information, so in practice this does not cause problems.

A. Policy Training Procedure

We use domain randomization [47] during training. For RGB images, we randomize the fabric color by selecting RGB values uniformly at random across intervals that include shades of blue, purple, pink, red, and gray. For RGB images, we additionally randomize the brightness with gamma corrections [33] and vary the shading of the fabric plane. For depth (D) images, we make the images slightly darker

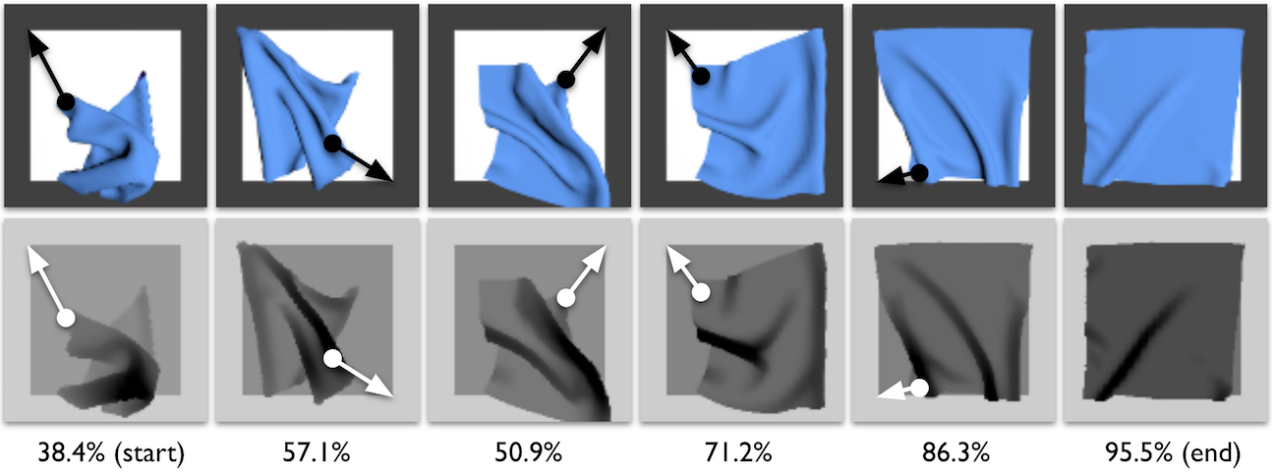

Fig. 4: Example simulated episode of the oracle corner supervisor policy, from left to right, drawn from a Tier 3 starting state with 38.4% coverage. The policy uses the exact corner location from the fabric state (not the images) and pulls the one furthest from its target on the fabric plane. For visualization purposes, we show matching color and depth images from the episode without domain randomization. Overlaid circles and arrows represent the action taken after the given state. In the second action, the fabric corner furthest from the target is slightly underneath the fabric, and the supervisor pulls at the fabric's top layer. Nonetheless, the underlying corner is still moved closer to its target position, and subsequent pulls are able to achieve high coverage. The oracle policy took five actions before getting to 95.5% coverage and triggering the 92% threshold in the rightmost images.

to more closely match real depth images. For both RGB and D, we randomize the camera pose with independent Gaussian distributions for each of the position and orientation components. The supplementary material has examples of images after domain randomization.

The policy neural network architecture is similar to the one in Matas et al. [27]. As input, the network consumes images with dimension $(100 \times 100 \times c)$, where the number of channels is $c = 1$ for D, $c = 3$ for RGB, or $c = 4$ for RGBD images. It passes the input image through four convolutional layers, each with 32 filters of size 3×3, followed by four dense layers of size 256 each, for a total of about 3.4 million parameters. The network produces a 4D vector with a hyperbolic tangent applied to make components within $[-1, 1]$. We optimize using Adam with learning rate 10^{-4} and use L_2 regularization of 10^{-5}.

The imitation learning code uses OpenAI baselines [10] to make use of its parallel environment support. We run the fabric simulator in ten parallel environments, which helps to alleviate the major time bottleneck when training, and pool together samples in a shared dataset. We first train with a "behavior cloning (BC) phase" where we minimize the L_2 error on the offline supervisor data, and then use a "DAgger phase" which rolls out the agent's policy and applies DAgger. We use 500 epochs of behavior cloning based on when the network's L_2 error roughly converged on a held-out validation dataset. The DAgger phase runs until the agent collectively performs 50,000 total steps. Further training details are in the supplementary material.

B. Simulation Experiments

For all simulated training runs, we evaluate on 50 new tier-specific starting states that are not seen during training. Figure 5 shows results across all tiers, suggesting that after behavior cloning, DAgger improves final coverage performance by 6.1% when averaging over the nine experimental conditions (three image input modalities across three tiers). In addition, RGB policies attain better coverage in simulation than D policies with gains of 10.8%, 8.3%, and 10.9% across respective tiers, which may be due to high color contrast between the fabric and fabric plane in the RGB images, as opposed to the D images (see Figure 3). The RGBD policies perform at a similar level as the RGB-based policies.

In all difficulty tiers, the RGB policies get higher final coverage performance than the wrinkles policy (from Table I): 94.8% over 91.3%, 89.6% over 87.0%, and 91.2% over 73.6%, respectively, and gets close to the corner pulling supervisor despite only having access to image observations rather than underlying fabric state. Similarly, the RGBD policies outperform the wrinkles policy across all tiers. The depth-only policies outperform the wrinkles policy on tier 3, with 80.3% versus 73.6% coverage.

VIII. Physical Experiments

The da Vinci Research Kit (dVRK) surgical robot [19] is a cable-driven surgical robot with imprecision as reviewed in prior work [25], [41]. We use a single arm with an end effector that can be opened to 75°, or a gripper width of 10mm. We set a fabric plane at a height and location that allows the end-effector to reach all points on it. To prevent potential damage to the grippers, the fabric plane is foam rubber, which allows us to liberally set the gripper height to be lower and avoids a source of height error present in [27]. For the fabric, we cut a 5x5 inch piece from a Zwipes 735 Microfiber Towel Cleaning Cloth with a blue color within the distribution of domain randomized fabric colors. We mount a Zivid One Plus RGBD camera 0.9 meters above the workspace, which is used to obtain color and depth images.

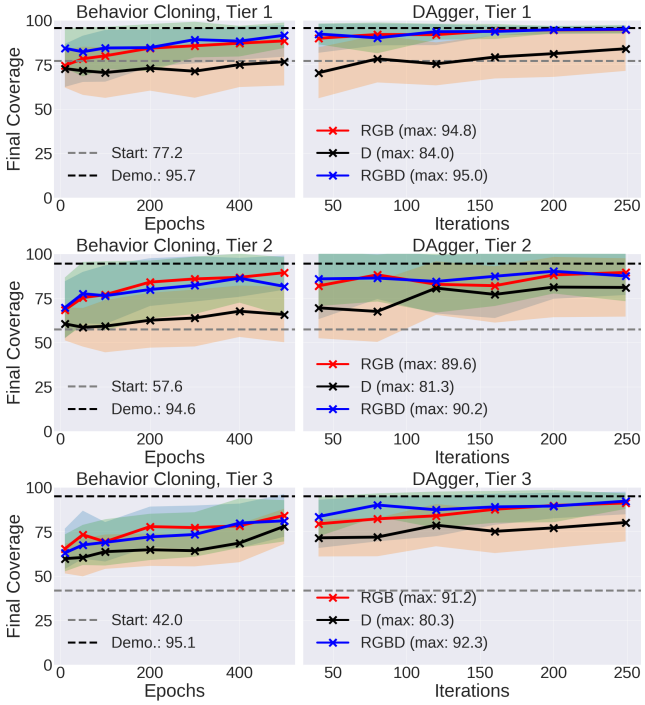

Fig. 5: Coverage over 50 simulated episodes at checkpoints (shown with "X") during behavior cloning (left) and DAgger (right), which begins right after the last behavior cloning epoch. Results, from top to bottom, are for tier 1, 2, and 3 starting states. We additionally annotate with dashed lines the average starting coverage and the supervisor's average final coverage. Results suggest that the RGB and RGBD policies attain strong coverage performance in simulation. Shaded regions represent one standard deviation range.

A. Physical Experiment Protocol

We manually create starting fabric states similar to those in simulation for all tiers. Given a starting fabric, we randomly run one of the RGB or D policies for one episode for at most 10 steps (as in simulation). Then, to make comparisons fair, we "reset" the fabric to be close to its starting state, and run the other policy. After these results, we then run the RGBD baseline to combine RGB and D images, again manipulating the fabric starting state to be similar among comparable episodes. To facilitate this process, we save all real images encountered and use them as a guide to creating the initial fabric configurations.

During preliminary trials, the dVRK gripper would sometimes miss the fabric by 1-2 mm, which is within the calibration error. To counter this, we measure structural similarity [50] of the image before and after an action to check if the robot moved the fabric. If it did not, the next action is adjusted to be closer to the center of the fabric plane, and the process repeats until the robot touches fabric.

B. Physical Experiment Results

We run 20 episodes for each combination of input modality (RGB, D, or RGBD) and tiers, resulting in 180 total episodes as presented in Table II. We report starting coverage, ending coverage, maximum coverage across the episode after the initial state, and the number of actions. The maximum coverage allows for a more nuanced understanding of performance, because policies can take strong initial actions that achieve high coverage (e.g., above 80%) but a single counter-productive action at the end can substantially lower coverage.

TABLE II: Physical experiments. We run 20 episodes for each of the tier 1 (T1), tier 2 (T2), and tier 3 (T3) fabric conditions, with RGB, D, and RGBD policies, for a total of $20 \times 3 \times 3 = 180$ episodes. We report: (1) starting coverage, (2) final coverage, with the highest values in each tier in bold, (3) maximum coverage at any point after the start state, and (4) the number of actions per episode. Results suggest that RGBD is the best policy with harder starting fabric configurations.

	(1) Start	**(2) Final**	**(3) Max**	**(4) Actions**
T1 RGB	78.4 +/- 4	**96.2 +/- 2**	96.2 +/- 2	1.8 +/- 1
T1 D	77.9 +/- 4	78.8 +/- 24	90.0 +/- 10	5.5 +/- 4
T1 RGBD	72.5 +/- 4	95.0 +/- 2	95.0 +/- 2	2.1 +/- 1
T2 RGB	58.5 +/- 6	87.7 +/- 13	92.7 +/- 4	6.3 +/- 3
T2 D	58.7 +/- 5	64.9 +/- 20	85.7 +/- 8	8.3 +/- 3
T2 RGBD	55.0 +/- 5	**91.3 +/- 8**	92.7 +/- 6	6.8 +/- 3
T3 RGB	46.2 +/- 4	75.0 +/- 18	79.9 +/- 14	8.7 +/- 2
T3 D	47.0 +/- 3	63.2 +/- 9	74.7 +/- 10	10.0 +/- 0
T3 RGBD	41.7 +/- 2	**83.0 +/- 10**	85.8 +/- 6	8.8 +/- 2

Results suggest that, despite not being trained on real images, the learned policies can smooth physical fabric. All policies improve over the starting coverage across all tiers. Final coverage averaged across all tiers is 86.3%, 69.0%, and 89.8% for RGB, D, and RGBD policies, respectively, with net coverage gains of 25.2%, 7.8%, and 33.4% over starting coverage. In addition, the RGB and RGBD policies deployed on tier 1 starting states each achieve the 92% coverage threshold 20 out of 20 times. While RGB has higher final coverage on tier 1 starting states, the RGBD policies appear to have stronger performance on the more difficult starting states without taking considerably more actions.

Qualitatively, the RGB and RGBD-trained policies are effective at "fine-tuning" by taking several short pulls to trigger at least 92% coverage. For example, Figure 6 shows an episode taken by the RGBD policy trained on tier 3 starting states. It is able to smooth the highly wrinkled fabric despite several corners hidden underneath fabric layers. The depth-only policies do not perform as well, but this is in large part because the depth policy sometimes takes counterproductive actions after several reasonable actions. This may be in part due to uneven texture on the fabric we use, which is difficult to replicate in simulated depth images.

C. Failure Cases

The policies are sometimes susceptible to performing highly counter-productive actions. In particular, the depth-only policies can fail by pulling near the center of the fabric for fabrics that are already nearly smooth, as shown in Figure 7. This results in poor coverage and may lead to cascading errors where one poor action can lead fabric to reach states that are not seen in training.

One cause may be that there are several fabric corners that are equally far from their targets, which creates ambiguity in which corner should be pulled. One approach to mitigate this issue, in addition to augmenting depth with RGB-based

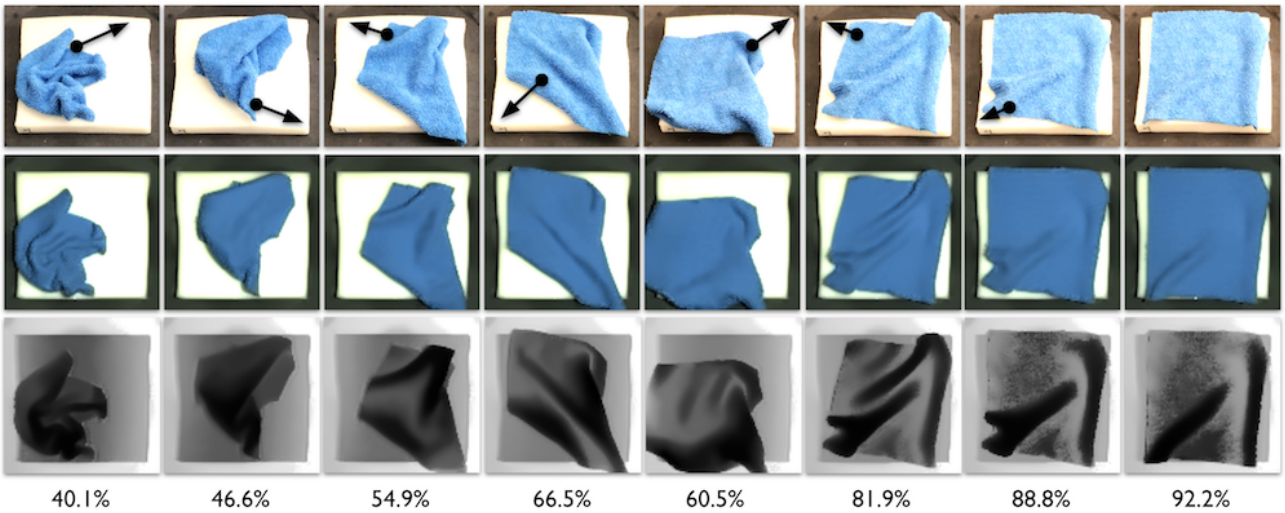

Fig. 6: An example episode taken by a learned policy trained on RGBD images from tier 3 starting states. The top row shows screen captures from the camera view used to record videos. The middle and bottom row show the processed RGB and D images that are passed into the neural network policy. The leftmost images show the starting state of the fabric, set to be highly wrinkled with at least the bottom left fabric corner hidden. The policy takes seven actions shown here, with pick points and pull vectors indicated by the overlaid black arrows in the top row. Despite the highly wrinkled starting state, along with hidden fabric corners (as shown in the first four columns) the policy is able to smooth fabric from 40.1% to 92.2% coverage as shown at the rightmost images.

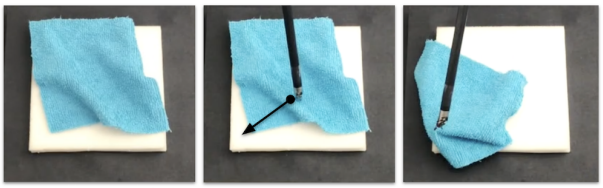

Fig. 7: A poor action from a policy trained on only depth images. Given the situation shown in the left image, the learned policy erroneously picks a point near the center of the fabric. The resulting pick, followed by the pull to the lower left, causes a major decrease in coverage and makes it hard for the robot to recover.

data as in the RGBD policies we train, is to formulate corner picking with a mixture model to resolve this ambiguity.

IX. Conclusion and Future Work

We investigate baseline and learned policies for fabric smoothing. Using a low fidelity fabric simulator and a custom environment, we train policies in simulation using DAgger with a corner pulling supervisor. We use domain randomization to transfer policies to a surgical robot. When testing on fabric of similar color to those used in training, RGB-based and RGBD-based policies achieve higher coverage than D-based policies. On the harder starting fabric configurations, the combined RGBD-based policies get the highest coverage among the tested policies, suggesting that depth is beneficial.

In future work, we will test on fabric shapes where corner pulling policies may get poor coverage. We plan to apply state-of-the-art deep reinforcement learning methods to learn richer policies that can explicitly reason over multiple time steps and varying geometries. To improve sample efficiency, we will consider model-based approaches such as Visual Foresight [12] and future image similarity [52], along with approaches that predict physics properties [17]. We will utilize higher-fidelity fabric simulators such as ARCSim [29]. Finally, we would like to extend the method beyond fabric coverage to tasks such as folding and wrapping, and will apply it to ropes, strings, and other deformable objects.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with Honda Research Institute USA, the Berkeley AI Research (BAIR) Lab, Berkeley Deep Drive (BDD), the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS "People and Robots" (CPAR) Initiative, and by the Scalable Collaborative Human-Robot Learning (SCHooL) Project, NSF National Robotics Initiative Award 1734633. The authors were supported in part by Siemens, Google, Amazon Robotics, Toyota Research Institute, Autodesk, ABB, Samsung, Knapp, Loccioni, Intel, Comcast, Cisco, Hewlett-Packard, PhotoNeo, NVidia, and Intuitive Surgical. Daniel Seita is supported by an NPSC Fellowship. We thank Jackson Chui, Michael Danielczuk, Shivin Devgon, and Mark Theis.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning From Demonstration," *Robotics and Autonomous Systems*, vol. 57, 2009.

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

[3] D. Baraff and A. Witkin, "Large Steps in Cloth Simulation," in *ACM SIGGRAPH*, 1998.

[4] K. Bathe, *Finite Element Procedures*. Prentice Hall, 2006. [Online]. Available: https://books.google.com/books?id=rWvefGICfO8C

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

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[7] B. O. Community, *Blender – a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[8] E. Corona, G. Alenya, A. Gabas, and C. Torras, "Active Garment Recognition and Target Grasping Point Detection Using Deep Learning," in *Pattern Recognition*, 2018.

[9] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing Clothing Into Desired Configurations with Limited Perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[10] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI Baselines," https://github.com/openai/baselines, 2017.

[11] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[12] 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.

[13] Z. Erickson, H. Clever, G. Turk, K. Liu, and C. Kemp, "Deep Haptic Model Predictive Control for Robot-Assisted Dressing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[14] Z. Erickson, M. Collier, A. Kapusta, and C. Kemp, "Tracking Human Pose During Robot-Assisted Dressing using Single-Axis Capacitive Proximity Sensing," in *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[15] Y. Gao, H. J. Chang, and Y. Demiris, "Iterative Path Optimisation for Personalised Dressing Assistance using Vision and Force Information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[16] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *In Proceedings of the Fourth Alvey Vision Conference*, 1988.

[17] A. Howard and G. Bekey, "Intelligent Learning for Deformable Object Manipulation," in *Autonomous Robtoics*, 2000.

[18] R. Jangir, G. Alenya, and C. Torras, "Dynamic Cloth Manipulation with Deep Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[19] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

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

[21] ——, "Clothes State Recognition Using 3D Observed Data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[22] J. J. Koenderink and A. J. Van Doorn, "Surface shape and curvature scales," *Image and vision computing*, vol. 10, no. 8, 1992.

[23] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen, "Multi-Sensor Surface Analysis for Robotic Ironing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[24] Y. Li, Y. Yue, D. X. E. Grinspun, and P. K. Allen, "Folding Deformable Objects using Predictive Simulation and Trajectory Optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[25] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel, and K. Goldberg, "Learning Accurate Kinematic Control of Cable-Driven Surgical Robots Using Data Cleaning and Gaussian Process Regression." in *IEEE Conference on Automation Science and Engineering (CASE)*, 2014.

[26] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

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

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

[29] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive Anisotropic Remeshing for Cloth Simulation," in *ACM SIGGRAPH Asia*, 2012.

[30] 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*, 2007.

[31] 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.

[32] D. A. Pomerleau, "Alvinn: An Autonomous Land Vehicle in a Neural Network," Carnegie-Mellon University, Tech. Rep., 1989.

[33] C. Poynton, *Digital Video and HDTV Algorithms and Interfaces*, 1st ed. Morgan Kaufmann Publishers Inc., 2003.

[34] X. Provot, "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior," in *Graphics Interface*, 1995.

[35] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Using Depth and Appearance Features for Informed Robot Grasping of Highly Wrinkled Clothes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[36] 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.

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

[38] 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 *International Journal of Robotics Research (IJRR)*, 2018.

[39] J. Schrimpf and L. E. Wetterwald, "Experiments Towards Automated Sewing With a Multi-Robot System," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[40] 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 *International Symposium on Robotics Research (ISRR)*, 2019.

[41] D. Seita, S. Krishnan, R. Fox, S. McKinley, J. Canny, and K. Goldberg, "Fast and Reliable Autonomous Surgical Debridement with Cable-Driven Robots Using a Two-Phase Calibration Procedure," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[42] S. Shibata, T. Yoshimi, M. Mizukawa, and Y. Ando, "A Trajectory Generation of Cloth Object Folding Motion Toward Realization of Housekeeping Robot," in *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2012.

[43] 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*, 2014.

[44] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[45] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[46] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[47] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[48] E. Torgerson and F. Paul, "Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1987.

[49] L. Verlet, "Computer Experiments on Classical Fluids: I. Thermodynamical Properties of LennardJones Molecules," *Physics Review*, vol. 159, no. 98, 1967.

[50] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *Trans. Img. Proc.*, Apr. 2004.

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

[52] A. Wu, A. Piergiovanni, and M. Ryoo, "Model-based Behavioral Cloning with Future Image Similarity Learning," in *Conference on Robot Learning (CoRL)*, 2019.

[53] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations," in *Robotics: Science and Systems (RSS)*, 2020.

[54] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," in *IEEE Robotics and Automation Letters (RA-L)*, 2017.