Robotic Fabric Manipulation with Deep Imitation Learning and Reinforcement Learning in Simulation

Ryan Hoque
Ken Goldberg, Ed.
Pieter Abbeel, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

May 27, 2020

Robotic Fabric Manipulation with Deep Imitation Learning and Reinforcement Learning in Simulation

by

Ryan Hoque

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the
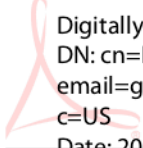
University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Pieter Abbeel

Spring 2020

The thesis of Ryan Hoque, titled Robotic Fabric Manipulation with Deep Imitation Learning and Reinforcement Learning in Simulation, is approved:

Chair _____

Digitally signed by Ken Goldberg
DN: cn=Ken Goldberg, o, ou,
email=goldberg@berkeley.edu,
c=US
Date: 2020.05.27 14:14:53 -08'00'

Date _____27 May 2020_____

_____

Date _____27-MAY-2020_____

University of California, Berkeley

Robotic Fabric Manipulation with Deep Imitation Learning and Reinforcement Learning in Simulation

Abstract

Robotic Fabric Manipulation with Deep Imitation Learning and Reinforcement Learning in Simulation

by

Ryan Hoque

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

While work on robotic manipulation of rigid objects has come a long way, the manipulation of deformable objects remains an open problem in robotics. Highly deformable structures such as rope, fabric and bags are very tricky to model and reason about due to their infinite-dimensional state space and complex dynamics. Meanwhile, techniques for deep learning, reinforcement learning, depth sensing, and simulation continue to advance and offer powerful tools for representing these complex objects. We study the problem of fabric manipulation, which has applications in the home (laundry, dressing assistance, bed-making, etc.) as well as industry (surgical gauze handling, textile and upholstery manufacturing, carbon fiber molding, etc.).

First, we study the problem of fabric smoothing, i.e. finding the sequence of pick-and-place actions that maximize coverage of the underlying plane. We train a policy on RGB and depth (RGBD) image observations with imitation learning of an algorithmic supervisor that has access to fabric state in simulation. To evaluate our approach, we transfer the policy to a physical da Vinci Research Kit (dVRK) surgical robot and achieve 95% final coverage on simple starting configurations and 83% final coverage on highly crumpled starting states.

Next, we generalize the problem from smoothing to goal-conditioned fabric manipulation, in which we wish to learn a policy to manipulate toward some arbitrary goal image. To do this, we propose a technique we call VisuoSpatial Foresight (VSF), which decouples learning a visual dynamics model on simulated RGBD data from planning over that model. In our experiments we demonstrate that a single learned policy can both rival the imitation learning agent on the smoothing task and manipulate toward images of singly folded and doubly folded fabric configurations without any task demonstrations at training or test time.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

I would not have been able to do this project without the support of many individuals. First, a big thank you to my advisor Prof. Ken Goldberg, who gave me an opportunity to prove myself at the AUTOLAB two years ago. With his constant support, helpful feedback, and research acumen, I have gone from an undergrad knowing virtually nothing about robotics research to an incoming PhD student with publications under my belt and a passion for research. Another big thank you to Daniel Seita, who advised me on a lower level and collaborated with me throughout much of the past two years. I am extremely grateful for all the invaluable research guidance, hard work, patience, and faith in my abilities. It's no understatement to say that you two have fundamentally changed the trajectory of my career and life toward the pursuit of science.

Thanks to Prof. Pieter Abbeel for valuable feedback on this project and inspiring lectures during my time in his Advanced Robotics class. Thanks also to Ashwin Balakrishna, Adi Ganapathi, and the rest of my talented peers in the AUTOLAB. Your gusto for robotics and AI research has been contagious.

Thank you to my good friends Chester Leung and Rohan Taori, without whom my undergraduate and graduate years at Berkeley so far would be far less memorable. Watching you two go through your individual research trajectories as I pursued my own has been both instructive and motivational.

Finally, a special thanks to my family. Undoubtedly I would not have been able to get to where I am today without your sacrifices and endless support.

Chapter 1

Introduction

As the ongoing COVID-19 pandemic spread around the world, endangering the lives of our loved ones and overwhelming many of our institutions, it exposed a need for automation. From our supply chain to our hospitals, automation can step in where it's far too dangerous for humans. Unfortunately, while great progress has been made in robotics and machine learning research, many open problems remain to be solved before we can safely deploy reliable and intelligent robots in the real world. One such problem is manipulation of highly deformable structures such as fabric, which, for example, could be helpful in making hospital beds [50] and caring for seniors in nursing homes [19]. Fabric manipulation also has applications in sewing [48], ironing [29], laundry folding [36, 30, 68, 52], manufacturing upholstery [60], handling surgical gauze [57], and more.

In Chapter 3, we consider learning a policy for the task of *fabric smoothing*: sequentially maximizing coverage of fabric in highly crumpled initial configurations. Fabric smoothing is both a starting point for further fabric manipulation and an interesting problem in its own right, as it standardizes the setup of fabric for subsequent tasks like folding. This chapter covers [49], work done by myself, Daniel Seita, Adi Ganapathi, Professor Goldberg and others. The paper contributes an open source fabric simulator, a smoothing policy learned via imitation of an algorithmic supervisor, and experimental validation on a physical robotic system with comparison of color and depth modalities.

In Chapter 4, we generalize the problem to *goal-conditioned fabric manipulation*: sequentially manipulating fabric toward some goal image. This chapter is based on [22], work done by myself, Daniel Seita, Ashwin Balakrishna and Professor Goldberg scheduled to appear at Robotics: Science and Systems (RSS) 2020. This paper presents "VisuoSpatial Foresight" (VSF), a model-based reinforcement learning algorithm for planning over visual dynamics from simulated RGBD data. Results indicate that a single learned policy can reach a variety of goal images such as smooth, folded, and doubly folded without any task demonstrations.

We conclude the report with a brief summary of where we are now in robotic fabric manipulation and where to go from here.

Chapter 2

Related Work

2.1   Fabric Manipulation

Manipulating fabric and other deformable objects is a long-standing challenge in robotics research. Smoothing helps standardize the configuration of the fabric, enabling reliable completion of subsequent tasks such as folding fabric [7, 47]. A popular approach in prior work is to first hang fabric in the air and allow gravity to "vertically smooth" it [41, 27, 28, 13], which has led to results such as Maitin-Shepard et al. [34], which reported a 100% success rate in single-towel folding. In contrast, the approach we present is targeted towards larger fabrics like blankets [50] and for single-armed robots which may have a limited range of motion, making such "vertically smoothing" infeasible. Similar work on fabric smoothing and folding, such as by Balaguer et al. [3] and Jia et al. [24, 25] assume the robot is initialized with the fabric already grasped, whereas we require the robot to begin manipulating without touching the fabric. Other work on fabric smoothing, including Willimon et al. [65] and Sun et al. [54, 55], address a similar problem setting as we do, but the initial fabric configurations are much closer to fully smoothed than those we consider. For the smoothing task, we consider three different tiers of initial state complexity, the hardest of which covers less than half of the plane and has 1-2 corners occluded.

While captivating demonstrations of laundry folding in commercial robots and research [36, 34] may initially lead us to believe that we have all but solved fabric manipulation, these policies come with strong assumptions and are highly engineered for that specific use case. Accurate state estimation for deformable objects remains intractable. Thus, we seek to learn robust, intelligent policies that can reason about the fabric state despite uncertainty and generalize to related manipulation tasks, especially in Chapter 4.

2.2   Learning from Fabric Simulation

There has been recent interest in learning sequential fabric manipulation policies with fabric simulators, in which we can take advantage of access to the underlying fabric state

and generate large training datasets efficiently. Examples of fabric simulators include those from ARCSim [39], MuJoCo [59], PyBullet [10], Blender [9], and the simulator we built in [49]. Matas et al. [35] simulate fabric in PyBullet and train a policy with the Deep Deterministic Policy Gradient (DDPG) algorithm [31]. They focus on the folding task and assume an initially smooth configuration. In independent work conducted concurrently with [49] (the focus of Chapter 3), Wu et al. [66] and Jangir et al. [23] also learn fabric smoothing from simulation using Model-Free Reinforcement Learning (MFRL) with Soft Actor-Critic [21] and DDPG with demonstrations [62] respectively.

   While these algorithms achieve impressive results, they are designed or trained for specific fabric manipulation tasks (such as folding or smoothing), and do not reuse learned structure to generalize to a wide range of tasks. This motivates learning fabric dynamics to enable more general purpose fabric manipulation strategies, which we consider in [22] (the focus of Chapter 4). Another option is learning the visual structure (as opposed to dynamical structure) of the fabric by finding correspondences between pairs of observations. In subsequent work in our lab, Ganapathi et al. [18] explores this approach using dense object descriptors [17] in simulation and achieves similarly promising results on real smoothing and folding tasks.

2.3   Visual Foresight

In Chapter 4 we consider learning a visual dynamics model and planning over it in a model-predictive control (MPC) framework. MPC is a popular approach for leveraging learned dynamics for robotics control that has shown success in learning robust closed-loop policies even with substantial dynamical uncertainty [56, 4, 15, 53]. However, many of these prior works require knowledge or estimation of underlying system state, which can often be infeasible or inaccurate. Finn et al. [16] and Ebert et al. [14] introduce *visual foresight* and demonstrate that MPC can be successfully combined with a learned visual dynamics model to accomplish a variety of robotic tasks, including deformable object manipulation such as folding pants, by planning over models trained on raw visual input of real fabric. However, the trajectories shown are limited to a single pick and pull, while we focus on longer horizon sequential tasks, aided by a pick-and-pull action space better suited for fabric manipulation. Furthermore, while prior work on visual foresight learns general purpose policies which can manipulate a variety of different objects, the tasks performed have a wide range of valid goal images, such as covering a utensil with a towel or moving a pant leg upwards. In contrast, we focus on achieving precise goal configurations via multi-step interaction with the fabric. Prior work on visual foresight [16, 14, 11] also generally collects data for training visual dynamics models in the real world, which is impractical and unsafe for robots such as the dVRK surgical robot we use due to the sheer volume of data required for the technique (on the order of 100,000 to 1 million actions, often requiring weeks of physical interaction data [11]). One recent exception is the work of Nair et al. [38] which trains in simulation for matching Tetris blocks. Prior work also uses RGB color images for training, while we

find that combining RGB images and depth maps in video prediction enables more effective planning for achieving a variety of smoothed and folded fabric configurations. The use of depth in this work is motivated by how depth cameras are available at ever decreasing cost as well as increasing resolution and frame rate.

In subsequent work, Yan et al. [67] and Lippi et al. [32] also train goal-conditioned fabric manipulation policies, but they learn dynamics and plan in latent space rather than image space. They demonstrate that learning a latent space can help with sample complexity and task performance, and they show generalization to planar rope manipulation and a simulated block-stacking task respectively. However, the fabric manipulation experiments are limited to smoothing in the former and folding in the latter.

Chapter 3

Learning Fabric Smoothing from an Algorithmic Supervisor

3.1 Introduction

This chapter is based on [49], in which we wish to learn a policy to smooth fabric from highly crumpled initial configurations with a single robot arm. To do this, we build a fabric simulator and collect smoothing episodes performed by a corner-pulling algorithmic demonstrator with access to the ground-truth mesh of the fabric. Such a technique could not be performed on real data as we would not have access to the same privileged information. We proceed to train a policy with DAgger [45] on this data and evaluate the learned policy against various baselines in simulation. Finally, we transfer the policy to a physical da Vinci Research Kit (dVRK) surgical robot by leveraging domain randomization [58] and compare performance among policies trained on RGB, depth, and RGBD modalities. See Figure 3.1 for an overview of the project.

3.2 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 some start state to one that maximally covers the fabric plane. We define an *episode* as one instance of the fabric smoothing task.

Concretely, let ξ_t be the full state of the fabric at time t with (x, y, z) positions of all points in the mesh. 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 channels, where $c = 1$ for depth images, 3 for color images, and 4 for combined color and depth (RGBD) images. Let A be the space of actions the robot may take (see Section 3.3). 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
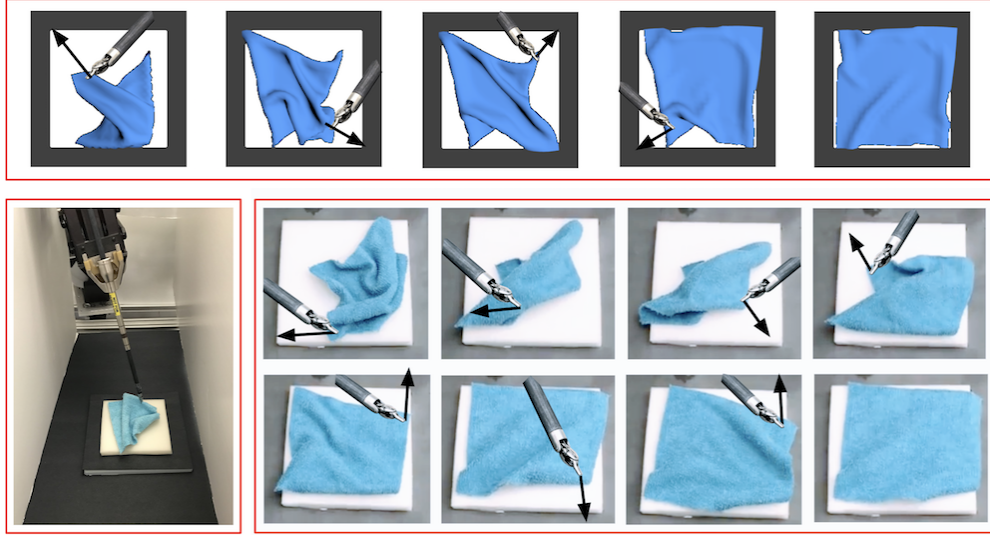
Figure 3.1: Learned smoothing policies executed in simulation and with a physical dVRK surgical robot, where pick-and-pull actions are indicated with the overlaid arrows. Policies are learned in simulation using DAgger [45] with an algorithmic supervisor that has full state information, using structured 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%.

policy $\pi : O \rightarrow A$ that maps an observation to an action, which it then executes sequentially until achieving sufficient coverage, pulling the fabric out of bounds, or reaching some limit on the number of iterations.

3.3   Fabric Simulator

We implement a Finite Element Method (FEM) [6] fabric simulator and interface with an OpenAI gym environment design [8]. The simulator is open source and available on the project website. Alternative fabric simulators exist, such as from Blender [9], which is a popular open-source computer graphics software toolkit. Since 2017, Blender has had significant improvements in physics and realism of fabrics, but did not have allow for massive data collection and low-level interaction with the simulated fabric until more recent work [18]. MuJoCo 2.0 provides another fabric simulator, but did not support OpenAI-gym style fabric manipulation environments until concurrent work [66]. ARCSim [39] is a third option that provides highly accurate fabric physics, but it was too computationally expensive for us to generate sufficient training data at a reasonable rate.
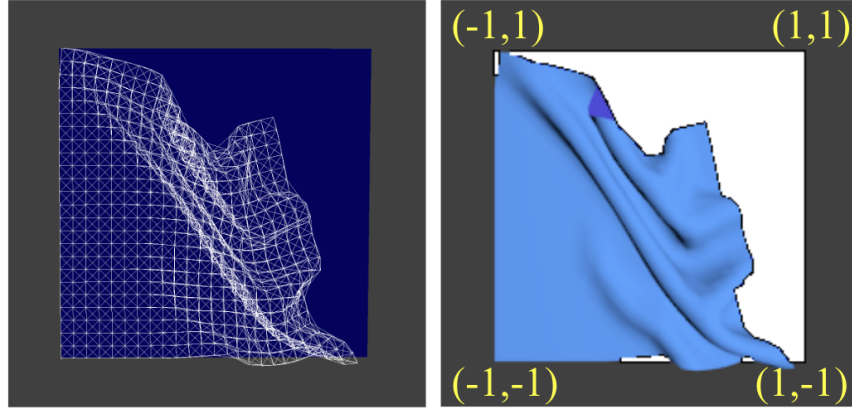
Figure 3.2: FEM fabric simulation. Left: a wireframe rendering, showing the 25×25 grid of points and the spring-mass constraints. Right: the corresponding image with the white fabric plane, rendered using Blender and overlaid with coordinates. The coverage is 73%, measured as the percentage of the fabric plane covered.

Inspired by course material from UC Berkeley's CS 184/284A: Computer Graphics class, the fabric (Figure 3.2) is represented as a grid of 25×25 point masses, connected by three types of springs [44]:

- *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 $F = m \cdot a$ and a spring correction force

$$F_s = k_s \cdot (\|q_a - q_b\|_2 - \ell), \tag{3.1}$$

for each of the springs representing the constraints above, where k_s is a spring constant (a hyperparameter we tune for realistic simulation), $q_a \in \mathbb{R}^3$ 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 (which we also tune). We update the point mass positions using Verlet integration [63]. Verlet integration computes a point mass's new position at time $t + \Delta_t$, denoted with $p_{t+\Delta_t}$, as:

$$p_{t+\Delta_t} = p_t + v_t \Delta_t + a_t \Delta_t^2, \tag{3.2}$$

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

$$p_{t+\Delta_t} = 2p_t - p_{t-\Delta_t} + a_t \Delta_t^2. \tag{3.3}$$

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

$$p_{t+\Delta_t} = p_t + (1 - d)(p_t - p_{t-\Delta_t}) + a_t \Delta_t^2 \tag{3.4}$$

where $d \in [0, 1]$ is a damping term that we tuned to 0.02 based on visual inspection of the simulator.

We apply a constraint from [44] by correcting point mass positions so that spring lengths are at most 10% greater than ℓ at any time. We also implement fabric self-collisions as suggested in [5] by adding a force to encourage separation between two points if they are too close.

The simulator provides access to the full fabric state ξ_t, which contains the exact (x, y, z) positions of all 25×25 points. However it does not provide corresponding image observations $\mathbf{o}_t$, which are necessary for transfer to physical robots. To obtain image observations of a given fabric state, we create a triangular mesh from ξ_t and render it using Blender 2.79.

Actions

We define an action $\mathbf{a}_t \in A$ at time t as a 4D vector

$$\mathbf{a}_t = \langle x_t, y_t, \Delta x_t, \Delta y_t \rangle \tag{3.5}$$

which includes the pick point (x_t, y_t) represented as the coordinate over the fabric plane to grasp as well as with the pull vector $(\Delta x_t, \Delta y_t)$ relative to the pick point. 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 lifts the fabric at the pick point, drags it towards the point $(x + \Delta x_t, y + \Delta y_t)$ in the fabric plane, and releases, allowing the fabric to settle.

Starting State Distributions

The performance of a smoothing policy depends heavily on the distribution of starting fabric states, which we vary more widely than prior work. We categorize episodes as belonging to one of three custom difficulty tiers. For each tier, we randomize the starting state of each episode. We report average coverage over 2000 simulations for each tier as well as how to generate each tier:
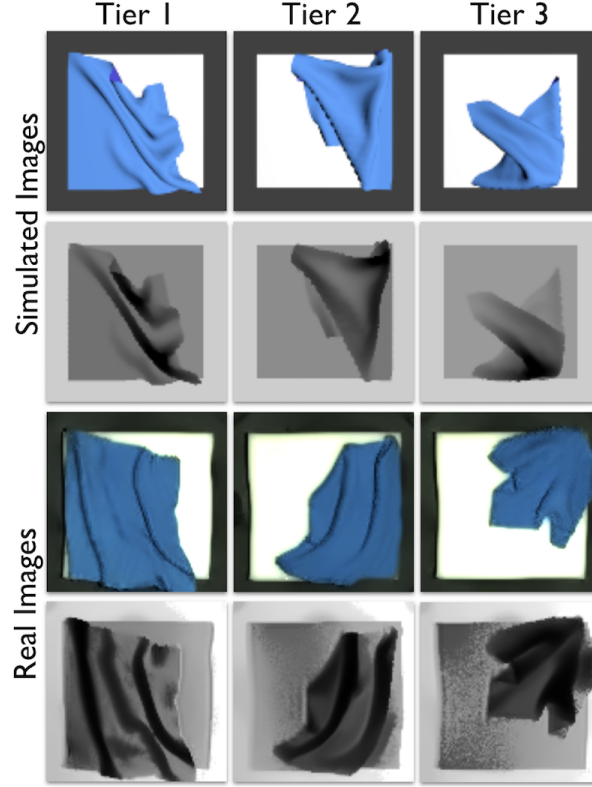
Figure 3.3: Initial fabric configurations drawn from the distributions specified in Section 3.3, with tiers grouped by columns. The first two rows show representative simulated color (RGB) and depth (D) images, respectively, while the last two rows show examples of real images from a mounted Zivid One 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.

- **Tier 1, 78.3 $\pm$ 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 $\pm$ 6.1% Coverage (Medium)**: We orient the fabric vertically and let the fabric drop from midair on one side of the fabric plane. We 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 target.

- **Tier 3, 41.1 $\pm$ 3.4% Coverage (Low)**: Starting from a flat fabric, we grip at a random pick point, pull high in the air, drag in a random direction, and then drop, usually resulting in one or two corners hidden.

Figure 3.3 shows examples of color and depth images of fabric initial states in simulation and real physical settings for all three tiers of difficulty.

3.4 Imitation Learning

Algorithmic Supervisor

A clear advantage of using a fabric simulator for training data is access to the ground truth state of the fabric ξ_t. Meticulously collecting this state information from real world examples by hand would be not only impractical but also inaccurate. Moreover, an algorithmic supervisor can generate demonstrations much more efficiently and cheaply than a human.

We call the fabric smoothing algorithm we use the *oracle policy*. Using its state information, it repeatedly finds the fabric corner furthest from its target position (its corresponding position in a fully smooth configuration) and pulls until it reaches the target. If the corner is occluded, the policy picks the top layer of the fabric. While this doesn't always do exactly what we want, we find that the policy performs very well on all difficulty tiers. Defining episode termination as reaching 92% coverage, we obtain the following results with the oracle policy when averaging over 2000 simulations:

- **Tier 1:** Final Coverage 95.7% +/- 2.1%, Number of Actions 1.8 +/- 0.8.

- **Tier 2:** Final Coverage 94.5% +/- 5.4%, Number of Actions 4.0 +/- 2.0.

- **Tier 3:** Final Coverage 95.1% +/- 2.3%, Number of Actions 4.6 +/- 1.1.

Imitation Learning with DAgger

We use the oracle policy to generate supervisor data. For each tier, we generate 2000 episodes from the supervisor and aggregate it offline into a dataset $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{a}_i)\}_{i=1}^{N}$, where N is the sum of the number of actions across all episodes. We train a fabric smoothing policy $\pi_\theta : O \to A$ on this data. As is standard in the imitation learning literature, we first train in a Behavior Cloning (BC) phase that minimizes the Euclidean distance between our policy's output and the supervisor's labels $\{\mathbf{a}_i\}$. Since BC has been shown to generalize poorly outside the data distribution [42], we proceed with a second phase of Dataset Aggregation (DAgger) [45], which requests the supervisor to label the states the robot encounters when running its learned policy. Again, since we are using an algorithmic supervisor, we can efficiently query the oracle online during this phase.

Since we are training on image data, we use a convolutional neural network for our policy. The network consumes images with dimensions $(100 \times 100 \times c)$, where the number of channels is $c = 1$ for D, $c = 3$ for RGB, or $c = 4$ for RGBD images. The network architecture and number of parameters are as follows:

```
policy/convnet/c1    864 params (3, 3, 3, 32)
policy/convnet/c2    9216 params (3, 3, 32, 32)
policy/convnet/c3    9216 params (3, 3, 32, 32)
policy/convnet/c4    9216 params (3, 3, 32, 32)
policy/fcnet/fc1     3276800 params (12800, 256)
```

Table 3.1: Hyperparameters for the main DAgger experiments.

Hyperparameter	Value
Parallel environments	10
Steps per env, between gradient updates	20
Gradient updates after parallel steps	240
Minibatch size	128
Supervisor (offline) episodes	2000
Policy learning rate	1e-4
Policy L_2 regularization parameter	1e-5
Behavior Cloning epochs	500
DAgger steps after Behavior Cloning	50000

```
policy/fcnet/fc2    65536 params (256, 256)
policy/fcnet/fc3    65536 params (256, 256)
policy/fcnet/fc4    1024 params (256, 4)
Total model parameters: 3.44 million
```

The imitation learning code uses OpenAI baselines [12] to make use of its parallel environment support. We run the fabric simulator in ten parallel environments to speed up training, and we pool together the samples in a shared dataset that includes both DAgger episodes and the offline BC data. See Table 3.1 for various training hyperparameters. After dataset aggregation our final dataset has approximately 110,000 $(\mathbf{o}_i, \mathbf{a}_i)$ transitions.

We use domain randomization [58] during training to facilitate sim-to-real transfer. 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 also randomize the brightness with gamma corrections [43] and vary the shading of the fabric plane. For depth (D) images, we make the images slightly darker 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. See Figure 3.4 for example images.

3.5 Simulated Experiments

We evaluate our learned policy in simulation by comparing final coverage against the following four baselines, all of which have access to ground truth state ξ_t:

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

2. *Highest:* This policy grasps the point p in the mesh ξ_t with the largest z coordinate. To compute the pull vector, we obtain the target coordinates by considering where
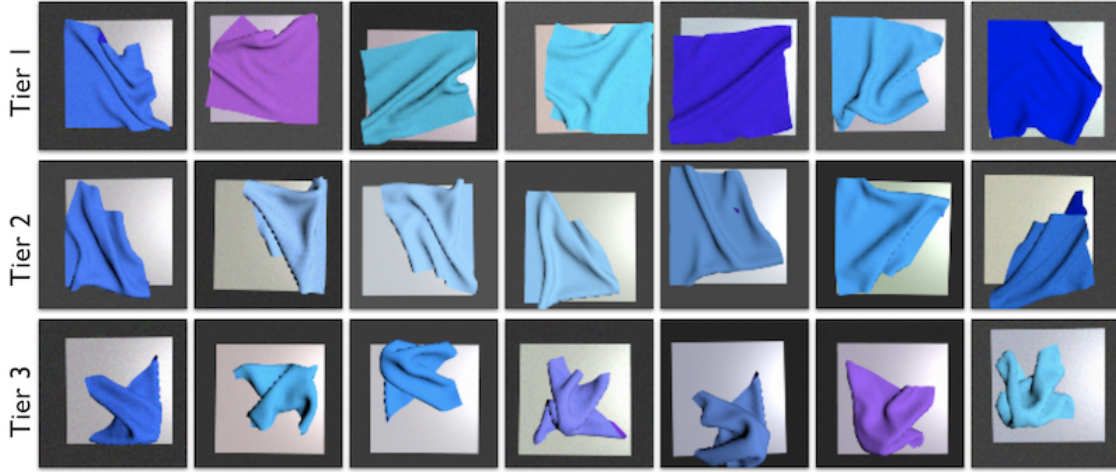
Figure 3.4: Simulated RGB domain-randomized images with starting states drawn from the distributions specified in Section 3.3. All images are of dimension $100 \times 100 \times 3$. Top row: Tier 1. Middle row: Tier 2. Bottom row: Tier 3. Domain randomization is applied on the fabric color, the shading of the white background plane, the camera pose, and the overall image brightness, and we apply uniform random noise to each pixel.

p's coordinates would be if the fabric were perfectly smooth. The pull vector is then the delta between p's current position and the target. Seita et al. [50] 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. [54] propose a smoothing algorithm that iteratively identifies the largest wrinkle and then attempts to smooth it by applying a force parallel to the fabric plane and perpendicular to the wrinkle. 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:* As in Section 3.4, we repeatedly pull the corner farthest from its target position. This is the policy we use for the algorithmic supervisor.

We report final coverage and number of actions for our learned policy as well as each of these analytic baselines in Table 3.2. Episodes end when at least one of the following conditions is met: (1) the fabric is sufficiently smooth (which we define to be 92% coverage), (2) the fabric is more than 25% out of bounds relative to the size of the fabric plane, or (3) we have already taken 15 actions. Note that the numbers for the oracle policy are the same as in Section 3.4 and that they are an upper bound on policy performance. The learned

Table 3.2: Results from the four baseline policies described in Section 3.5 and the imitation learning (IL) policy. We report final coverage and the number of actions per episode. All baselines statistics are from 2000 episodes of each tier, and learned policy statistics are from 200 episodes of each tier.

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	IL	94.3 ± 2.3	3.3 ± 3.1
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	IL	92.8 ± 7.0	5.7 ± 4.0
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	IL	88.6 ± 11.5	10.1 ± 3.9

policy is trained and evaluated on RGBD domain-randomized images; we postpone modality comparison to Section 3.6.

As we can see in Table 3.2, our learned policy performs nearly as well as the oracle and outperforms all other baselines on all three tiers of difficulty. We conclude that imitation learning from an algorithmic supervisor is sufficient for strong policy performance on fabric smoothing in simulation. See Figure 3.5 for an example of a successful episode at test time.

3.6 Physical Experiments

The da Vinci Research Kit (dVRK) [26] is a cable-driven surgical robot with imprecision as reviewed in prior work [33, 51]. 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 [35]. For the fabric, we cut a 5" × 5" 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 to obtain color and depth images.
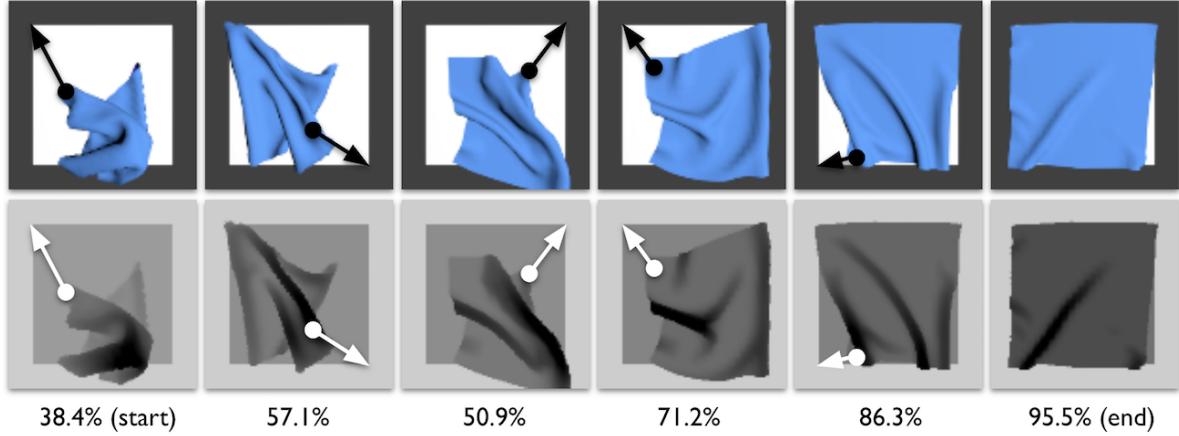
Figure 3.5: Example simulated episode of our learned policy, from left to right, drawn from a Tier 3 starting state with 38.4% coverage. For visualization purposes, we show matching color and depth images from an episode without domain randomization. Overlaid circles and arrows represent the action taken on the given observation. The policy tends to pull at corners, as it was trained on a corner-pulling supervisor. The policy takes five actions before getting to 95.5% coverage, which exceeds our 92% threshold.

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, D, and RGBD policies for one episode for at most 10 steps. Then, to make comparisons fair, before each of the other policies we "reset" the fabric to be close to its starting state. 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 robot's calibration error. To counter this, we measure structural similarity [64] of the image before and after an action to check if the robot has 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 the fabric.

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 3.3. 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, as policies can sometimes take strong initial actions that achieve high coverage (e.g. above 80%) but suffer from a single counter-productive action at the end that substantially lowers coverage.
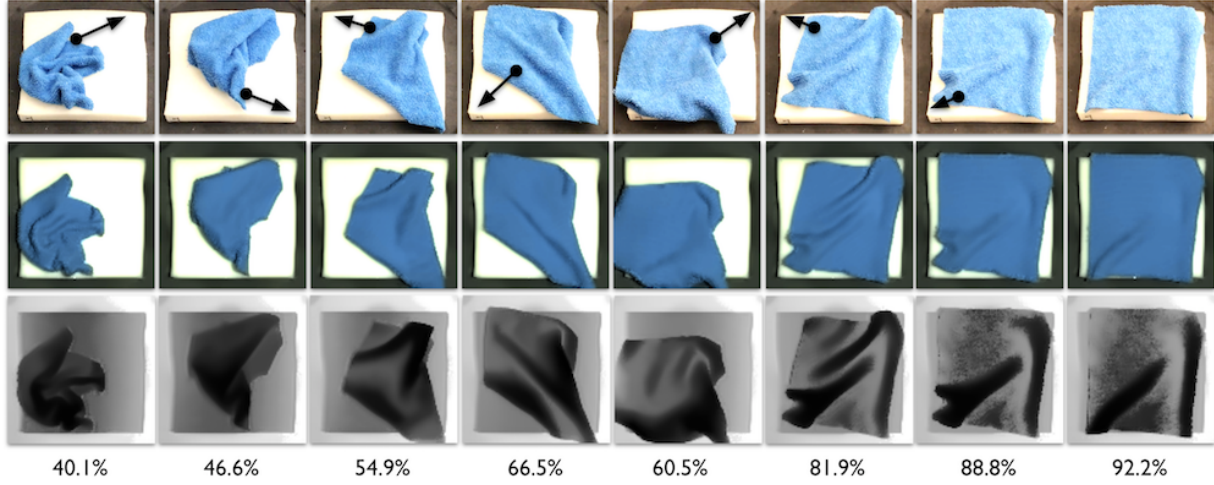
Figure 3.6: An example episode taken by a learned policy trained on RGBD images from Tier 3 starting states. The top row shows screen captures from the camera view used to record videos. The middle and bottom row show the processed RGB and D images that are passed to the neural network policy. The leftmost 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 the 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 and hidden fabric corners, the policy is able to smooth fabric from 40.1% to 92.2% coverage.

Results suggest that despite not being trained on real images, the learned policies can smooth physical fabric. All policies improve on 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 has stronger performance on the more difficult starting states without taking considerably more actions.

Qualitatively, the RGB and RGBD policies are effective at "fine-tuning" by taking several short pulls to trigger at least 92% coverage. Figure 3.6 shows an episode taken by the RGBD policy 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.

Table 3.3: 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
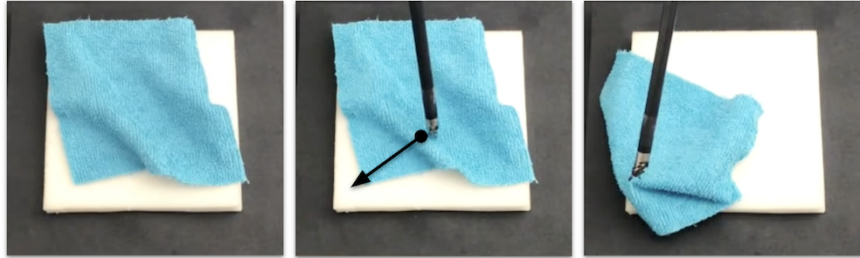
Figure 3.7: A poor action from a policy trained on only depth images. Given the situation shown in the left image, the policy picks a point near the center of the fabric. The resulting pick, followed by the pull to the lower left, causes a major decrease in coverage and makes it hard for the robot to recover.

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 3.7. This results in poor coverage and may lead to cascading errors where one poor action can lead fabric to reach states outside the training distribution.

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 could be to formulate corner picking with a mixture model to resolve this ambiguity.

Chapter 4

Goal-Conditioned Fabric Manipulation with VisuoSpatial Foresight

4.1 Introduction

This chapter is based on [22], in which we generalize the problem from *fabric smoothing* to *goal-conditioned fabric manipulation* with a single robot arm. To this end, we propose a Model-Based Reinforcement Learning (MBRL) algorithm called VisuoSpatial Foresight (VSF) to learn visual fabric dynamics in simulation and reuse the learned structure across related manipulation tasks like smoothing and folding. We demonstrate that a single policy can reach a variety of goal images in simulation and transfer the policy to a dVRK surgical robot. See Figure 4.1 for an overview.

4.2 Problem Statement

We consider learning goal-conditioned fabric manipulation policies that enable planning to specific fabric configurations given a goal image of the fabric in the desired configuration. We define the fabric configuration at time t as ξ_t, represented as a mass-spring system with an $N \times N$ grid of point masses subject to gravity and Hookean spring forces. However, due to the difficulties of state estimation for highly deformable objects such as fabric we consider overhead RGBD observations $\mathbf{o}_t \in \mathbb{R}^{56 \times 56 \times 4}$, which consist of three-channel RGB and single-channel depth images.

We assume tasks have a finite task horizon T and can be achieved with a sequence of actions from a single robot arm which involve grasping a specific point on the fabric and pulling it in a particular direction, which holds for common manipulation tasks such as
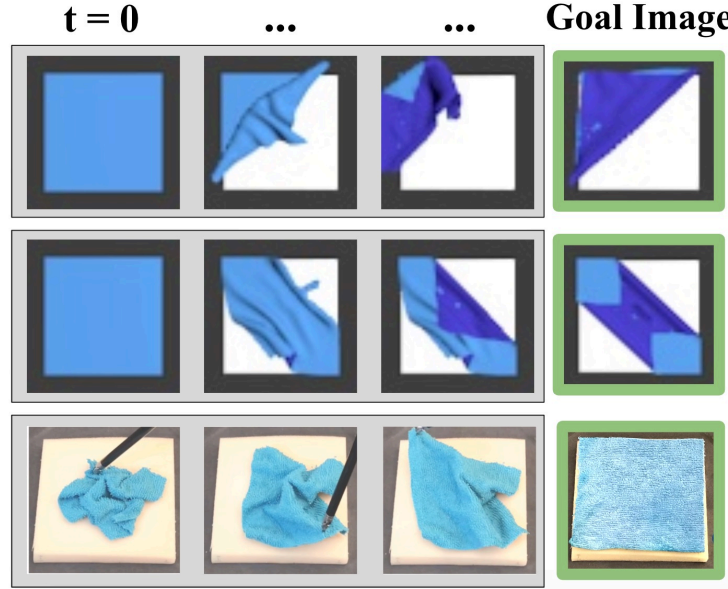
Figure 4.1: Using VSF on domain randomized RGBD data, we learn a goal-conditioned fabric manipulation policy in simulation without any task demonstrations. We evaluate the same policy on several goal images in simulation (top two rows) and on the physical da Vinci surgical robot (bottom row). The rows display the RGB portions of subsampled frames from episodes for folding, double folding, and smoothing, respectively, toward the goal images in the right column.

folding and smoothing. As in Chapter 3, we consider four-dimensional actions

$$\mathbf{a}_t = \langle x_t, y_t, \Delta x_t, \Delta y_t \rangle. \tag{4.1}$$

Each action $\mathbf{a}_t$ at time t involves grasping the top layer of the fabric at coordinate (x_t, y_t) with respect to an underlying background plane, lifting, translating by $(\Delta x_t, \Delta y_t)$, and then releasing and letting the fabric settle. When appropriate, we omit the time subscript t for brevity.

The objective is to learn a goal-conditioned policy which minimizes some goal conditioned cost function defined on realized or predicted interaction episodes $c_g(\tau)$ with goal g and episode τ, where the latter in this work consists of one or more images.

4.3 VisuoSpatial Foresight

Visual Foresight

Visual foresight, also called *visual MPC* (model-predictive control), is a framework presented by Finn et al. [16] for vision-based robotic manipulation tasks. It consists of two phases: (1) learning a visual dynamics model directly from image observations and (2) using the model to plan a trajectory with low cost according to some planning cost function. In Phase 1, a deep convolutional neural network is trained on thousands of episodes, where an episode is a complete trajectory of observations and actions from some starting state to a terminal state. Usually these episodes consist of mostly random actions and some on-policy actions relevant to the task(s) under consideration so that the network can train on enough data from the distribution encountered at test time. The network $f_\theta(\cdot)$ learns to map current observation and a sequence of actions to a sequence of predicted observations, i.e.

$$f_\theta(\mathbf{o}_t, \mathbf{a}_{t:t+H-1}) = (\hat{\mathbf{o}}_{t+1:t+H}) \tag{4.2}$$

where $\mathbf{o}_t$ is the image observation of the underlying state ξ_t at time step t, $\mathbf{a}_t$ is the action taken on ξ_t at time step t, and H is the "horizon," a hyperparameter that specifies how many observations to predict. Thus we get the name *visual foresight*: the model is in a sense "predicting the future" by generating "imagined" trajectories predicted to be the result of a given sequence of actions. Images closer to the end of the imagined trajectory are typically more blurry due to increased uncertainty.

Phase 2 occurs at test time when we are actually rolling out our policy. We specify some planning cost function $c(\tau)$ evaluated over H-length trajectories and use our visual dynamics model to find the optimal action sequence in terms of this cost. Since finding the actual optimal sequence is intractable, we approximate with random shooting (generating many random sequences and taking the one with lowest cost) or the cross-entropy method (CEM, repeatedly alternating between sampling action sequences from some normal distribution and re-fitting that distribution onto the lowest cost trajectories). Finally, we mitigate inaccuracies that propagate with time by using model-predictive control (MPC), i.e. executing only the first action in our approximately optimal sequence and then replanning. Concretely, in Phase 2, at every time step during policy execution we:

1. Generate random action sequences $\mathbf{a}_{t:t+H-1}$ by sampling from some $\mathcal{N}(\mu, \Sigma)$.

2. Pass each action sequence $\mathbf{a}_{t:t+H-1}$ along with our current observation $\mathbf{o}_t$ to our trained dynamics model $f_\theta(\cdot)$ to get a predicted trajectory τ for each action sequence.

3. Evaluate each τ with our cost function $c(\tau)$ to find the 10% or 20% of trajectories with the lowest cost. (The percentage value here is a CEM hyperparameter.)

4. (CEM) Fit $\mathcal{N}(\mu, \Sigma)$ to these "elite" trajectories and redo steps (1)-(3) for some fixed number of iterations.

5. (MPC) Execute the first action in μ in our environment to get some resulting ξ_{t+1} and $\mathbf{o}_{t+1}$.

Approach Overview

To learn goal-conditioned policies, we extend the visual foresight framework discussed in the previous section with simulated depth data to learn a model of fabric dynamics. We then use this model to plan over an H-step MPC horizon to compute action sequences to optimize a goal-conditioned cost function. Since depth maps give us an additional "spatial" component to the dynamics model, we call the resulting algorithm *VisuoSpatial Foresight* (VSF).

To represent the dynamics of the fabric, we train a deep recurrent convolutional neural network [20] to predict a sequence of RGBD output frames conditioned on a sequence of RGBD context frames and a sequence of actions. This visuospatial dynamics model, denoted as f_θ, is trained on thousands of self-supervised simulated episodes of interaction with the fabric, where an episode consists of a contiguous trajectory of observations and actions. We use Stochastic Variational Video Prediction [2] for the model architecture.

For planning, we utilize a goal-conditioned planning cost function $c_g(\hat{\mathbf{o}}_{t+1:t+H})$ with goal g, which in the proposed work is a target image $\mathbf{o}^{(g)}$. The cost is evaluated over the H-length sequence of predicted images $\hat{\mathbf{o}}_{t+1:t+H}$ sampled from f_θ conditioned on the current observation $\mathbf{o}_t$ and some proposed action sequence $\hat{\mathbf{a}}_{t:t+H-1}$. One example of a cost function optimized for smoothing is one minus coverage, where coverage is the percentage of a background plane covered by a fabric of the same size. While there are a variety of cost functions that can be used for visual foresight [14], we utilize the Euclidean pixel distance between the final RGBD images at timestep t and the goal image $\mathbf{o}^{(g)}$ across all 4 channels as this works well in practice for the tasks we consider. Precisely, we define the planning cost as follows:

$$c_g(\hat{\mathbf{o}}_{t+1:t+i*}) = \|\mathbf{o}^{(g)} - \hat{\mathbf{o}}_{t+i*}\|_2 \tag{4.3}$$

where

$$i* = \min\{H, T - t\} \tag{4.4}$$

represents the time horizon. As in prior work [16, 14, 11], we utilize the cross entropy method [46] to plan action sequences to minimize $c_g(\hat{\mathbf{o}}_{t+1:t+H})$ over a receding H-step horizon at each time t.

There are four main components of this approach: (1) fabric simulation for policy learning in a simulator, (2) data collection, (3) learning a visuospatial dynamics model, and (4) generating plans over this dynamics model. See Figure 4.2 for a summary of the approach.

Fabric Simulator

VSF requires a large amount of training data to predict full-resolution RGBD images. Since getting real data is cumbersome and imprecise, we use a fabric simulator to generate data
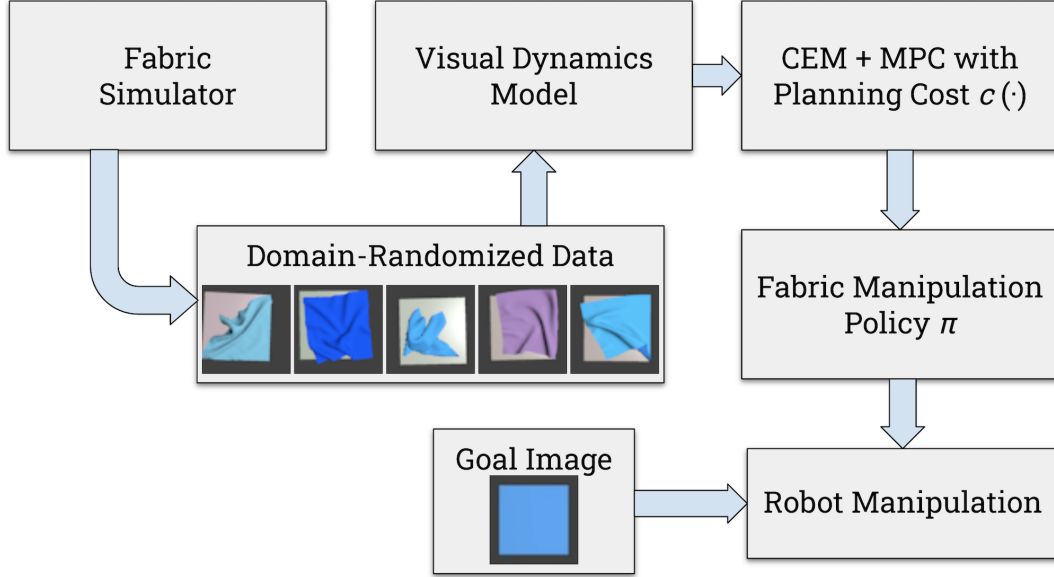
Figure 4.2: Overview of the approach. The fabric simulator generates domain-randomized RGBD images. The data is used to train the SV2P visual dynamics model. At test time, given some choice of planning cost function, we plan with the trained dynamics model to get the goal-conditioned fabric manipulation policy π. Given some goal image, we take actions to reach that observation.

quickly and efficiently. We use the open-source fabric simulator from Seita et al. [49] (discussed in Section 3.3) which was shown to be sufficiently accurate for reasonable fabric smoothing policies and sim-to-real transfer, and we use the same physics-based hyperparameters. We briefly recap the most relevant aspects of the simulator with emphasis on the changes from the original.

The fabric is represented as a mass-spring system with a 25×25 grid of point masses [44]. Each point mass has springs connecting it to its neighbors. Verlet integration [63] is used to update point mass positions by approximating velocity as the difference in position between discrete time steps. The simulator implements self-collision by adding a repulsive force to points that are too close [5] and applies damping to simulate friction. As in [49], the simulator is wrapped in an OpenAI Gym environment [8], with actions as four-dimensional vectors as described in Section 3.3. For the pick point, $x \in [-1, 1]$ and $y \in [-1, 1]$. The only change in the action space from Seita et al. [49] is that we truncate Δx and Δy to be within $[-0.4, 0.4]$, rather than $[-1, 1]$ as earlier. Restricting the action space may make it easier to learn visual dynamics models since larger deltas drastically alter the shape of the fabric.

We use the open-source software Blender to render (top-down) image observations $\mathbf{o}_t$ of the fabric. We make a couple changes for the observations relative to [49]. First, we reduce the size of observations to 56×56 from 100×100, to make training visual dynamics models more computationally tractable. Second, to enable transfer of policies trained in

simulation to the real-world, we adjust the domain randomization [58] techniques so that color, brightness, and positional hyperparameters are fixed *per episode* to ensure that the video prediction model learns to only focus on predicting changes in the fabric configuration rather than changes due to domain randomization.

Data Generation

We collect 7,115 episodes of length 15 each for a total of 106,725 $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$ transitions for training the visuospatial dynamics model. The episode starting states are sampled from three tiers of difficulty with equal probability. These tiers are the same as those in [49] and differ in initial *coverage*, or the amount that a fabric covers the underlying plane. All data was generated from a random policy: execute a randomly sampled action, but resample if the grasp point (x, y) is not within the bounding box of the 2D projection of the fabric, and take the additive inverse of Δx and/or Δy if $(x + \Delta x, y + \Delta y)$ is out of bounds (off the plane by more than 25% of its width, causing episode termination).

VisuoSpatial Dynamics Model

Due to the inherent stochasticity in fabric dynamics, we utilize an implementation of Stochastic Variational Video Prediction (SV2P) model from Babaeizadeh et al. [2], a state-of-the-art model for action-conditioned video prediction. Here, the video prediction model f_θ is designed as a latent variable model, enabling the resulting posterior distribution to capture different modes in the distribution of predicted frames. Specifically, [2] trains a generative model which predicts a sequence of n output frames conditioned on a context vector of m frames and a sequence of actions starting from the most recent context frame. Since the stochasticity in video prediction is often a consequence of latent events not directly observable in the context frames as noted in [2], predictions are conditioned on a vector of latent variables $z_{t+m:t+m+n-1}$, each sampled from a fixed prior distribution $p(z)$. This gives rise to the following generative model.

$$p_\theta(\hat{\mathbf{o}}_{t+m:t+m+n-1} | \hat{\mathbf{a}}_{t+m-1:t+m+n-2}, \mathbf{o}_{t:t+m-1}) =$$
$$p(z_{t+m}) \prod_{t'=t+m}^{t+m+n-1} p_\theta(\hat{\mathbf{o}}_{t'} | \hat{\mathbf{o}}_{t:t'-1}, z_{t'}, \hat{\mathbf{a}}_{t'-1}). \tag{4.5}$$

Here $\mathbf{o}_{t:t+m-1}$ are image observations from time t to $t+m-1$, $\hat{\mathbf{a}}_{t+m-1:t+m+n-2}$ is a candidate action sequence at timestep $t+m-1$, and $\hat{\mathbf{o}}_{t+m:t+m+n-1}$ is the sequence of predicted images. Since the generative model is trained in a recurrent fashion, it can be used to sample an H-length sequence of predicted images $\hat{\mathbf{o}}_{t+m:t+m+H-1}$ for any $m > 0, H > 0$ conditioned on a current sequence of image observations $\mathbf{o}_{t:t+m-1}$ and an H-length sequence of proposed actions taken from $\mathbf{o}_{t+m-1}$, given by $\hat{\mathbf{a}}_{t+m-1:t+m+H-2}$.

During training, we set the number of context frames $m = 3$ and number of output frames $n = 7$, i.e. the SV2P model learns to predict 7 frames of an episode from the preceding 3

Table 4.1: Cross Entropy Method (CEM) hyperparameters for VSF. The variance reported here is the diagonal of the covariance matrix.

Hyperparameter	Value
Number of Iterations	10
Population Size	2000
Number of Elites	400 (20%)
α	0.1
Planning Horizon	5
Initial Mean μ	(0, 0, 0, 0)
Initial Variance Σ (Smoothing)	(0.25, 0.25, 0.04, 0.04)
Initial Variance Σ (Folding)	(0.67, 0.67, 0.24, 0.24)

frames. We train on domain-randomized RGBD data, where we randomize fabric color (in a range centered around blue), shading of the plane, image brightness, and camera pose. At test time, we utilize only one context frame $m = 1$ and a planning horizon of $n = 5$ output frames. This yields the model $p_\theta(\hat{\mathbf{o}}_{t+1:t+5}|\hat{\mathbf{a}}_{t:t+4}, \mathbf{o}_t)$. See Babaeizadeh et al. [2] for more details on the model architecture and training procedure. For this work, we utilize the SV2P implementation provided in [61].

VisuoSpatial Foresight

We combine the RGBD dynamics model of the previous section and planning with the Cross-Entropy Method (CEM) [46] into the VSF framework. As described in the Visual Foresight section, CEM repeatedly samples action sequences from a multivariate Gaussian distribution, uses the learned dynamics model to predict output frames for each action sequence, finds the "elite" sequences with the lowest cost according to our cost function, and refits a Gaussian distribution to these elite sequences. For our pixel distance cost function, we remove the 7 pixels on each side of the image to get rid of the impact of the dark border, i.e. we turn our 56x56 frame into a 42x42 frame. See Table 4.1 for tuned CEM hyperparameters. We use slightly different settings of CEM variance for smoothing and folding, as we found that different values yield better performance for each task. To mitigate compounding model error we leverage Model-Predictive Control (MPC), which takes only the first action in the action sequence given by CEM and then replans.

4.4 Simulated Experiments

VisuoSpatial Dynamics Prediction Quality

One advantage of VSF (and visual foresight more generally) is that we can inspect the model to see if its predictions are accurate, as the model provides some visual interpretability.
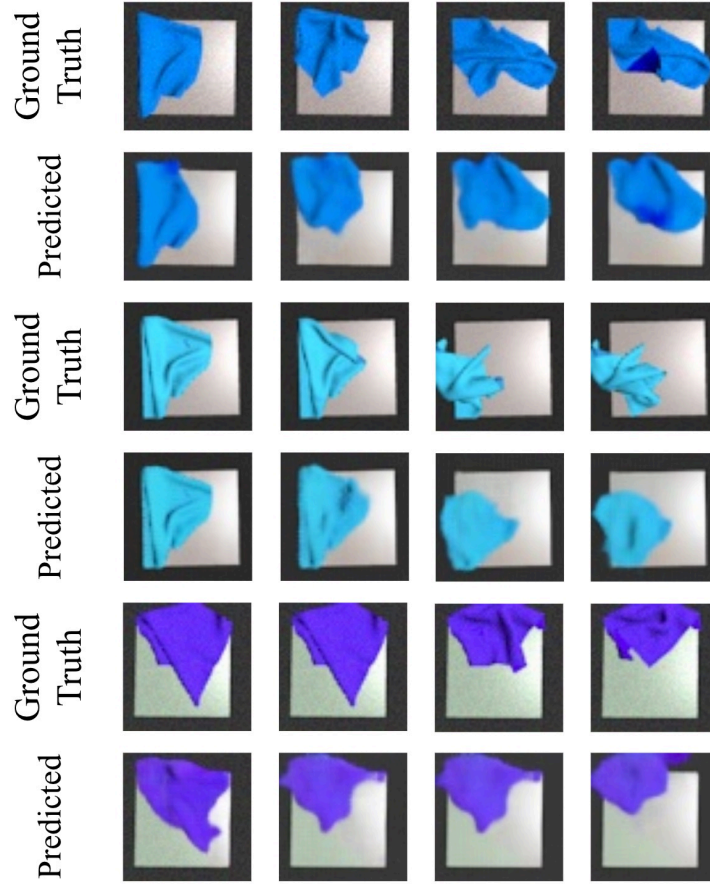
Figure 4.3: Three pairs of a sequence of four simulated image observations in ground truth compared against a sequence of the corresponding predictions from the visuospatial dynamics model. Here we show the RGB portion of the observations. Each episode is a test example and has randomized camera angle, fabric color, brightness, and shading. Prediction quality varies and generally gets blurrier across time, but is sufficient for planning.

Figure 4.3 shows three examples of predicted image sequences within an episode compared to ground truth images from the actual episode. All episodes are from test time episodes of a random policy, meaning that these ground truth images did not appear in the training data for the visuospatial dynamics model.

The predictions are reasonably robust to domain randomization, as the model is able to produce fabrics of roughly the appropriate color and can appropriately align the angle of the light gray background plane. For a more quantitative measure of prediction quality, we calculate the average Structural SIMilarity (SSIM) index [64] over corresponding image pairs in 200 predicted sequences against 200 ground truth sequences to be 0.701. A higher $SSIM \in [-1,1]$ corresponds to higher image similarity.
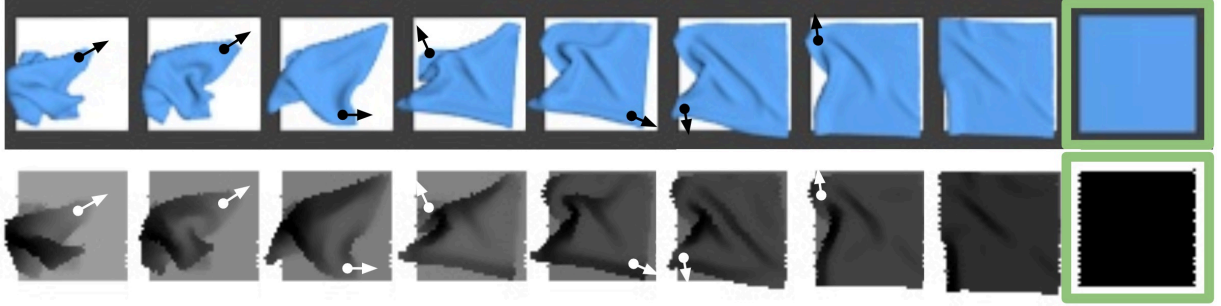
Figure 4.4: A simulated episode executed by the VSF policy on a Tier 3 starting state, given a smooth goal image (shown in the far right). The first row shows RGB images and the second shows the corresponding depth maps. In this example, the policy is able to successfully cross the coverage threshold of 92% after executing 7 actions. Actions are visualized with the overlaid arrows.

Fabric Smoothing Simulations

We first experiment on the smoothing task: maximizing fabric coverage, which we measure as the percentage of an underlying plane covered by the fabric. The plane is the same size of a fabric that is fully smooth. We evaluate smoothing on the three tiers of difficulty as reviewed in Section 4.3. Each episode lasts for a maximum of $t = 15$ time steps. Following [49], episodes can terminate earlier if a threshold of 92% coverage is triggered, or if any fabric point falls sufficiently outside of the fabric plane.

To see how the general VSF policy performs against existing smoothing techniques, for each difficulty tier, we execute 200 VSF episodes and compare against the baselines in Chapter 3, including the imitation learning expert. We also compare against DDPG with demonstrations [62], a state-of-the-art model-free reinforcement learning algorithm. The L2 cost in VSF is taken with respect to a smooth goal image (see Figure 4.4). Note that VSF does *not* explicitly optimize for coverage, and only optimizes the cost function from Equation 4.3, which measures pixel distance to a target image. To recap, we compare with the following baselines:

Random Randomly sample pick point and pull direction.

Highest Using ground truth state information, pick the point mass with the maximum z-coordinate and set pull direction to point to where the point mass would be if the fabric were perfectly smooth.

Wrinkle As in Sun et al. [55], find the largest wrinkle using ground truth state information and then pull perpendicular to it at the edge of the fabric to smooth it out.

Table 4.2: DDPG hyperparameters.

Hyperparameter	Value
Parallel environments	20
Steps per env, between gradient updates	10
Gradient updates after parallel steps	240
Minibatch size	128
Discount factor γ	0.99
Demonstrator (offline) samples	6697
Actor learning rate	1e-4
Actor L_2 regularization parameter	1e-5
Critic learning rate	1e-3
Critic L_2 regularization parameter	1e-5
Pre-training epochs	200
DDPG steps after pre-training	110000

Imitation Learning (IL) As in Seita et al. [49], train an imitation learning agent using DAgger [45] with a simulated "corner-pulling" demonstrator that picks and pulls at the fabric corner furthest from its target. For a fair comparison, VSF is trained on approximately the same amount of data (about 110,000 transitions) as the IL agent, although the datasets are distinct.

Model-Free RL We run DDPG [31] and extend it to use demonstrations and a pre-training phase as suggested in Vecerik et al. [62]. We also use the Q-filter from Nair et al. [37]. We train with a similar number of data points as IL and VSF for a reasonable comparison. We design a reward function for the smoothing task that, at each time step, provides reward equal to the change in coverage between two consecutive states. Inspired by OpenAI et al. [40], we also provide a +5 bonus for triggering a coverage success, and −5 penalty for pulling the fabric out of bounds. We use the same CNN architecture as the IL policy (see Section 3.4), and training hyperparameters are given in Table 4.2.

We report results in Table 4.3, reusing the results from Table 3.2 where appropriate. The data indicates that VSF significantly outperforms the analytic and model-free reinforcement learning baselines. Despite being trained solely on random interaction data, VSF performs similarly to the IL agent, a "smoothing specialist" that rivals the performance of the corner pulling demonstrator it imitates. See Figure 4.4 for an example Tier 3 VSF episode. VSF, however, requires quite a few more actions than the DAgger approach, especially on Tier 1, with 8.3 actions per episode compared to 3.3 per episode. However, we hypothesize that this leads to VSF being more conservative and reliable. Table 4.3 suggests that VSF has lower variance in its performance on Tier 2 and Tier 3 settings.

Table 4.3: Simulated smoothing experimental results for the six policies in Section 4.4. We report final coverage and number of actions per episode, averaged over 200 simulated episodes per tier for the learned policies and 2000 per tier for the analytic policies. VSF attains similar final coverage as the IL agent from [49] and outperforms the other baselines despite not seeing any smoothing demonstrations.

Tier	Method	Coverage	Actions
1	Random	25.0 ± 14.6	2.4 ± 2.2
1	Highest	66.2 ± 25.1	8.2 ± 3.2
1	Wrinkle	91.3 ± 7.1	5.4 ± 3.7
1	DDPG and Demos	87.1 ± 10.7	8.7 ± 6.1
1	Imitation Learning	94.3 ± 2.3	3.3 ± 3.1
1	VisuoSpatial Foresight	92.5 ± 2.5	8.3 ± 4.7
2	Random	22.3 ± 12.7	3.0 ± 2.5
2	Highest	57.3 ± 13.0	10.0 ± 0.3
2	Wrinkle	87.0 ± 10.8	7.6 ± 2.8
2	DDPG and Demos	82.0 ± 14.7	9.5 ± 5.8
2	Imitation Learning	92.8 ± 7.0	5.7 ± 4.0
2	VisuoSpatial Foresight	90.3 ± 3.8	12.1 ± 3.4
3	Random	20.6 ± 12.3	3.8 ± 2.8
3	Highest	36.3 ± 16.3	7.9 ± 3.2
3	Wrinkle	73.6 ± 19.0	8.9 ± 2.0
3	DDPG and Demos	67.9 ± 15.6	12.9 ± 3.9
3	Imitation Learning	88.6 ± 11.5	10.1 ± 3.9
3	VisuoSpatial Foresight	89.3 ± 5.9	13.1 ± 2.9

Fabric Folding Simulations

Here we demonstrate that VSF is not only able to successfully smooth fabric, but it also directly generalizes to folding tasks. We use the same video prediction model, trained only with random interaction data, and we leave all planning parameters the same besides the initial CEM variance. We change only the goal image to the triangular, folded shape in Figure 4.5. We also change the initial state to the smooth state (which can be interpreted as the result of the smoothing policy from the previous subsection). The two sides of the fabric are shaded differently, and the darker shade in the goal image indicates only the bottom side of the fabric is visible. Due to the action space bounds (Δx and Δy are limited to $[-0.4, 0.4]$), getting to this goal state directly is not possible in under three actions and requires a precise sequence of picks.

We visually inspect the final states in each episode and classify them as successes or failures. See Figure 4.5 for a typical success case, and Table 4.4 for quantitative results. We compare performance on L2 cost taken on RGB, depth, and RGBD channels. RGBD loss significantly outperforms the other modes, which correspond to Visual Foresight and "Spatial Foresight" respectively, suggesting the usefulness of augmenting Visual Foresight with depth maps.

Table 4.4: Folding results in simulation. VisuoSpatial Foresight is run with a smooth starting state and the goal image in Figure 4.5 for 20 episodes where L2 is taken on the depth, RGB, and RGBD channels. The results suggest that adding depth allows us to significantly outperform RGB-only Visual Foresight on this task.

Cost Function	Successes	Failures	% Success
L2 Depth	0	20	0%
L2 RGB	10	10	50%
L2 RGBD	**18**	**2**	**90%**
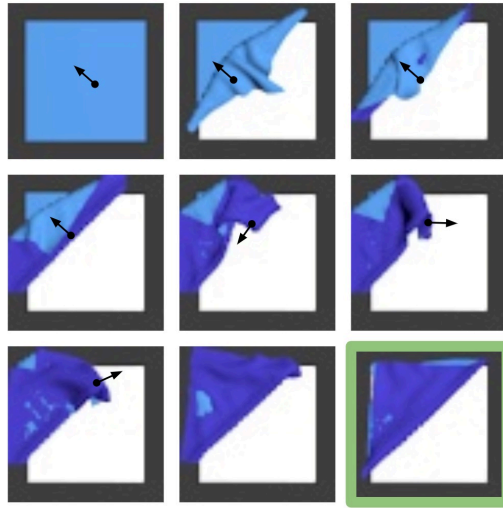
Figure 4.5: Simulated RGB observations of a successful folding episode. The RGB portion of the goal image is displayed in the bottom right corner. The rest is an 8-step episode (left-to-right, top-to-bottom) from smooth to approximately folded. There are several areas of the fabric simulator which have overlapping layers due to the difficulty of accurately modeling fabric-fabric collisions in simulation, which explain the light blue patches in the figure.

We also attempt to reach a more complex goal image with two overlapping folds for a total of three layers at the center of the fabric, again with the same VSF policy. Here 5 out of 10 trials succeed, and 4 of the 5 successes arrange the two folds in the correct ordering. Folding two opposite corners in the right order may be difficult to do solely with RGB data, since the bottom layer of the fabric has a uniform color. The depth maps, in contrast, provide information about one fold lying on top of another. See Figure 4.6 for an example of a successful rollout. Failure cases are often characterized by the robot picking at a point that just misses a fabric corner, but where the SV2P model predicts that the resulting pull will behave as if the robot had picked at the fabric corner.
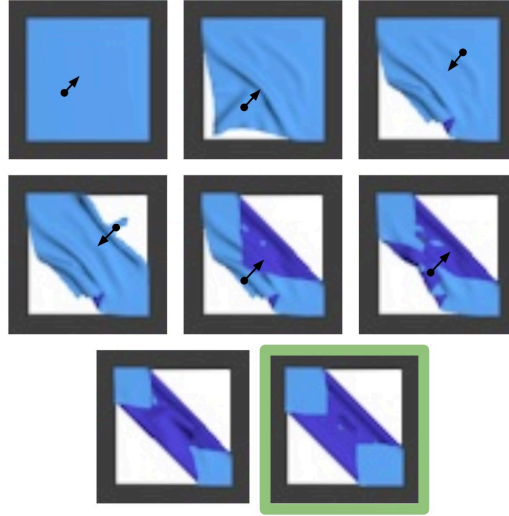
Figure 4.6: An example of a successful double folding episode. The RGB portion of the goal image is displayed in the bottom row. The rest is an 7-step episode (left-to-right, top-to-bottom) from smooth to approximately matching the goal image. The two folds are stacked in the correct order.

Table 4.5: Physical smoothing robot experiment results for Imitation Learning (IL) and VisuoSpatial Foresight (VSF). For both methods, we choose the policy snapshot with highest performance in simulation, and each are applied on all tiers (T1, T2, T3). We report results across 10 episodes of IL per tier and 5 episodes of VSF per tier. Results suggest that VSF gets final coverage results comparable to or exceeding that of IL.

Tier	(1) Start	(2) Final	(3) Max	(4) Num. Actions
1 IL	74.2 ± 5	92.1 ± 6	92.9 ± 3	4.0 ± 3
1 VSF	78.3 ± 6	$\mathbf{93.4 \pm 2}$	93.4 ± 2	8.2 ± 4
2 IL	58.2 ± 3	84.2 ± 18	86.8 ± 15	9.8 ± 5
2 VSF	59.5 ± 3	$\mathbf{87.1 \pm 9}$	90.0 ± 5	12.8 ± 3
3 IL	43.3 ± 4	75.2 ± 18	79.1 ± 14	12.5 ± 4
3 VSF	41.4 ± 3	$\mathbf{75.6 \pm 15}$	76.9 ± 15	15.0 ± 0

4.5 Physical Experiments

We next deploy the domain-randomized policies on a physical robot. We use the da Vinci surgical robot [26] and use the same experimental setup as Seita et al. [49], including the calibration procedure to map pick points (x, y) from actions into positions and orientations with respect to the robot's base frame. This is a challenging task due to the robot's imprecision [51] and the fine-grained precision that is necessary to manipulate fabrics. We use a Zivid One Plus camera for RGBD images, which we mount 0.9 meters above the workspace.
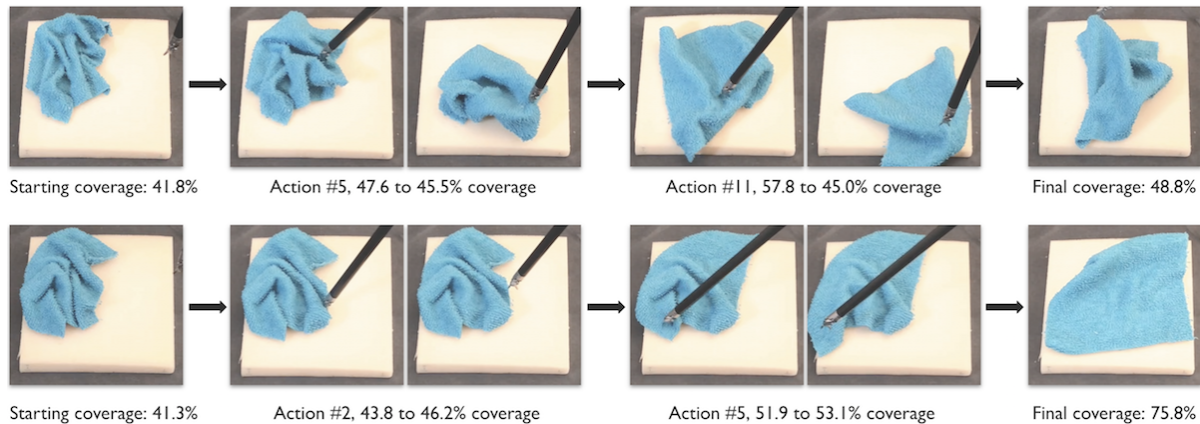
Figure 4.7: A qualitative comparison of physical smoothing episodes with an IL policy (top row) and a VSF policy (bottom row). The rows show screen captures taken from the third-person video view for recording episodes and do not represent the top-down input to the neural networks. To facilitate comparisons among IL and VSF, we manually make the starting fabric state as similar as possible. Over the course of several actions, the IL policy sometimes takes actions that are highly counter-productive, such as the 5th and 11th actions above. In contrast, VSF takes shorter pulls on average, with representative examples shown above for the 2nd and 5th actions. At the end, the IL policy gets just 48.8% coverage (far below its usual performance), whereas VSF gets 75.8%.

Experiment Protocol

We test with the RGBD IL and VSF policies as measured in simulation and reported in Table 4.3. We do not test with the model-free DDPG policy baseline, as DDPG performed significantly worse than the other two methods in simulation. We reiterate that VSF uses a video prediction model that is trained on *entirely random data*, requiring no labeled data in contrast with IL, which requires a demonstrator with privileged fabric state information during training.

To match the simulation setup, we limit each episode to a maximum of 15 actions. On the smoothing task, we perform episodes in which the fabric is initialized in highly rumpled states which mirror those from the simulated tiers. We run ten episodes per tier for IL and five episodes per tier for VSF, for a total 45 episodes. Within each tier, we attempt to make starting fabric states reasonably comparable between IL and VSF episodes (as in Figure 4.7).

Physical Fabric Smoothing

We present quantitative results in Table 4.5. We note that these are new IL trials, so results are different from Table 3.3. Results suggest that VSF gets final coverage results comparable to that of IL, despite not being trained on the corner-pulling demonstration data. However, it sometimes requires more actions to complete an episode and takes an order of magnitude
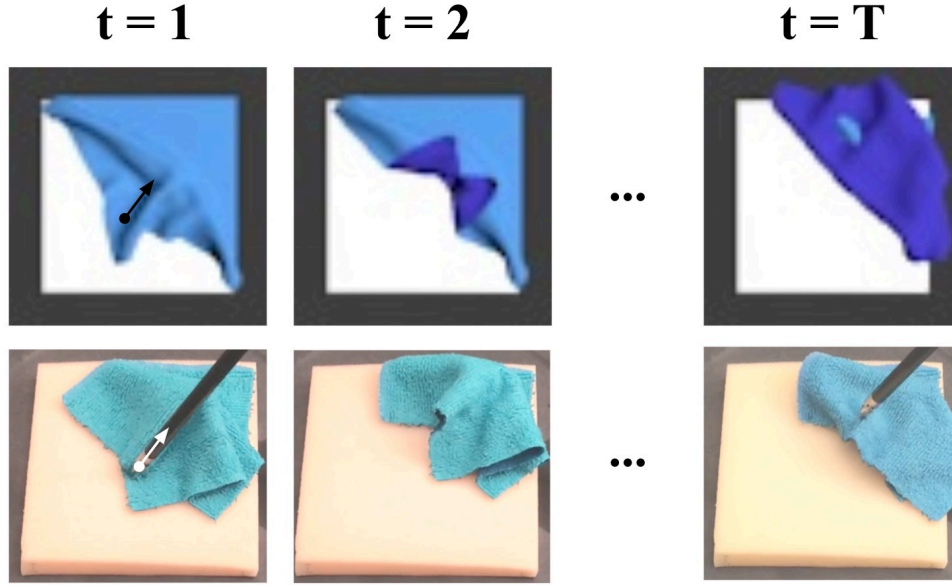
Figure 4.8: Physical folding policy executed in the simulator and on the surgical robot, with actions determined from *real* image input only. Despite this, the actions are able to fold in simulation. The difference in dynamics is apparent from $t = 1$ to $t = 2$, where the simulated fabric's bottom left corner is overturned from the action, but the corresponding corner on the real fabric is not.

more computation time per action (around 20 seconds versus around 1 second), as CEM requires thousands of forward passes through a neural network while IL needs only a single forward pass.

The actions for VSF usually have smaller deltas, which helps to take more precise actions. This is likely a result of the initialization of the mean and variance used to fit the Gaussians during CEM, which in our experiments we found to hurt performance if excessively increased. One qualitative effect of IL taking relatively longer actions is that it may be susceptible to highly counter-productive actions. The fabric manipulation tasks we consider require high precision, and a small error in the pick point region coupled with a long pull direction may cover a corner or substantially decrease coverage. As an example, Figure 4.7 shows a time lapse of a subset of actions for one episode from IL and VSF. On the fifth action, the IL policy has a pick point that is slightly north of the ideal spot. The pull direction to the lower right fabric plane corner is reasonable, but due to the length of the pull and the slightly suboptimal pick point, the lower right fabric corner gets covered.

Physical Fabric Folding

Finally, we attempt to apply the *same* VSF policy, but on a fabric *folding* task, starting from a smooth state. We conduct four goal-conditioned physical folding episodes, with one

episode for each of the four possible diagonal folds. We run each episode for 15 time steps. Qualitatively, the robot tends to move the fabric in the right direction based on the target corner, but does not get the fabric in a clean two-layer fold. To evaluate the quality of the policy, we took the actions generated for a *real episode* on the physical system and ran them open-loop in the fabric simulator. See Figure 4.8 for a comparison. Since the *same actions* are able to fold reasonably well in simulation, we conclude that the difference is due to a dynamics mismatch between simulated and real environments, compounded with pick point inaccuracies common in cable-driven robots such as the dVRK [51]. We have taken steps to address this mismatch and hypothesize that we can improve policy performance, but due to the current public health concerns we have not yet been able to test these hypotheses on the physical system. (We hope everyone is staying safe!)

Chapter 5

Conclusions and Future Work

In this work, we cover two novel learning-based techniques for robotic fabric manipulation in simulation. The work done for these projects spans everything from building the fabric simulator from scratch to developing learning algorithms and deploying learned policies on a physical robotic system. In Chapter 3, we present a procedure for learning a fabric smoothing policy via deep imitation learning from an algorithmic supervisor that has access to ground truth state in simulation. In Chapter 4, we generalize our setup to consider goal-conditioned fabric manipulation, and we learn a model-based reinforcement learning policy for both smoothing and folding given a single RGBD goal image. In practice, there are tradeoffs between the two algorithms. For instance, while VisuoSpatial Foresight gives us generality, Imitation Learning is much faster at execution time. Depending on the setting, we may desire a hybrid approach that first standardizes the setup with the fast smoothing policy before deploying VSF toward a desired configuration.

We are encouraged by the learned policies' ability to manipulate fabric toward a desired state despite low room for error due to the sequential nature of the tasks we consider. We observe the critical role of depth sensing, learning, and simulation in our work, and we suspect these techniques will be instrumental in future work on fabric manipulation as well as deformable object manipulation in general, including one-dimensional and three-dimensional structures such as ropes, cables, and bags. We are excited by several avenues for future work:

1. Augmenting the action space: Since our single-arm, pick-and-place action space limits the space of realizable manipulation, we wish to lift these constraints by training bilateral policies for dual-armed robots as well as learning 6 degree-of-freedom grasps instead of a single pick point, which will allow for more subtle motions such as gripping the fabric edge from both sides at once.

2. Increasing task complexity: Given an augmented action space, we can consider motion primitives beyond smoothing and folding such as shaking, rolling, lofting, and wrapping the fabric, which will almost certainly require explicit modeling of dynamical structure as we do in VSF. We can theoretically compose these primitives to accomplish very complex fabric manipulation.

3. Increasing reliability: We hope to boost policy performance with techniques such as learning additional structure given our access to ground truth state in simulation, explicitly modeling of contact dynamics between the end effector and the fabric, and allowing a human in the loop for particularly challenging scenarios.

While robotics research in this area has come a long way and recent work is encouraging [49, 22, 18, 32, 67], deformable object manipulation is still very much an open problem. Fundamental difficulties in robotic perception, planning, and control lie between present capabilities and truly intelligent, dexterous manipulation. I hope to continue working on such exciting problems during my PhD program at UC Berkeley.

Bibliography

[1]   Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. "A Survey of Robot Learning From Demonstration". In: *Robotics and Autonomous Systems* 57 (2009).

[2]   Mohammad Babaeizadeh, Chelsea Finn, Dumitri Erhan, Roy H. Campbell, and Sergey Levine. "Stochastic Variational Video Prediction". In: *International Conference on Learning Representations (ICLR)*. 2018.

[3]   Benjamin Balaguer and Stefano Carpin. "Combining Imitation and Reinforcement Learning to Fold Deformable Planar Objects". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[4]   Ashwin Balakrishna, Brijen Thananjeyan, Jonathan Lee, Arsh Zahed, Felix Li, Joseph E. Gonzalez, and Ken Goldberg. "On-Policy Robot Imitation Learning from a Converging Supervisor". In: *Conference on Robot Learning (CoRL)*. 2019.

[5]   David Baraff and Andrew Witkin. "Large Steps in Cloth Simulation". In: *ACM SIG-GRAPH*. 1998.

[6]   K.J. Bathe. *Finite Element Procedures*. Prentice Hall, 2006. URL: https://books.google.com/books?id=rWvefGICfO8C.

[7]   Julia Borras, Guillem Alenya, and Carme Torras. "A Grasping-centered Analysis for Cloth Manipulation". In: *arXiv:1906.08202* (2019).

[8]   Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". In: *arXiv:1606.01540* (2016).

[9]   Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[10]   Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2019.

[11]   Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. "RoboNet: Large-Scale Multi-Robot Learning". In: *Conference on Robot Learning (CoRL)*. 2019.

[12] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. *OpenAI Baselines.* `https://github.com/openai/baselines`. 2017.

[13] Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, and Sotiris 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.

[14] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. "Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control". In: *arXiv:1812.00568* (2018).

[15] Zackory Erickson, Henry M. Clever, Greg Turk, C. Karen Liu, and Charles C. Kemp. "Deep Haptic Model Predictive Control for Robot-Assisted Dressing". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[16] Chelsea Finn and Sergey Levine. "Deep Visual Foresight for Planning Robot Motion". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[17] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. "Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation". In: *Conference on Robot Learning (CoRL)*. 2018.

[18] Adi Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph Gonzalez, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. "Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images". In: *arXiv:2003.12698* (2020).

[19] Yixing Gao, Hyung Jin Chang, and Yiannis 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.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning (ICML)*. 2018.

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

[23] Rishabh Jangir, Guillem Alenya, and Carme Torras. "Dynamic Cloth Manipulation with Deep Reinforcement Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[24] Biao Jia, Zhe Hu, Jia Pan, and Dinesh Manocha. "Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[25] Biao Jia, Zherong Pan, Zhe Hu, Jia Pan, and Dinesh Manocha. "Cloth Manipulation Using Random-Forest-Based Imitation Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[26] 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.

[27] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki 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.

[28] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. "Clothes State Recognition Using 3D Observed Data". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009.

[29] Yinxiao Li, Xiuhan Hu, Danfei Xu, Yonghao Yue, Eitan Grinspun, and Peter K. Allen. "Multi-Sensor Surface Analysis for Robotic Ironing". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[30] Yinxiao Li, Yonghao Yue, Danfei Xu Eitan Grinspun, and Peter K. Allen. "Folding Deformable Objects using Predictive Simulation and Trajectory Optimization". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015.

[31] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous Control with Deep Reinforcement Learning". In: *International Conference on Learning Representations (ICLR)*. 2016.

[32] Martin Lippi, Petra Poklukar, Michael Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. "Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation". In: *arXiv:2003.08974* (2020).

[33] 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.

[34] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. "Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010.

[35] Jan Matas, Stephen James, and Andrew J. Davison. "Sim-to-Real Reinforcement Learning for Deformable Object Manipulation". In: *Conference on Robot Learning (CoRL)* (2018).

[36] Stephen Miller, Jur van den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. "A Geometric Approach to Robotic Laundry Folding". In: *International Journal of Robotics Research (IJRR)*. 2012.

[37] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[38] Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. "Time Reversal as Self-Supervision". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[39] Rahul Narain, Armin Samii, and James F. O'Brien. "Adaptive Anisotropic Remeshing for Cloth Simulation". In: *ACM SIGGRAPH Asia*. 2012.

[40] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. "Learning Dexterous In-Hand Manipulation". In: *arXiv:1808.00177* (2018).

[41] Fumiaki Osawa, Hiroaki Seki, and Yoshitsugu Kamiya. "Unfolding of Massive Laundry and Classification Types by Dual Manipulator". In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 11.5 (2007).

[42] Dean A Pomerleau. *Alvinn: An Autonomous Land Vehicle in a Neural Network*. Tech. rep. Carnegie-Mellon University, 1989.

[43] Charles Poynton. *Digital Video and HDTV Algorithms and Interfaces*. 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. ISBN: 1558607927.

[44] Xavier Provot. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior". In: *Graphics Interface*. 1995.

[45] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[46] Reuven Rubinstein. "The Cross-Entropy Method for Combinatorial and Continuous Optimization". In: *Methodology And Computing In Applied Probability* (1999).

[47] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey". In: *International Journal of Robotics Research (IJRR)*. 2018.

[48] Johannes Schrimpf and Lars Erik Wetterwald. "Experiments Towards Automated Sewing With a Multi-Robot System". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012.

[49] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. "Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor". In: *arXiv:1910.04854* (2020).

[50] Daniel Seita, Nawid Jamali, Michael Laskey, Ron Berenstein, Ajay Kumar Tanwani, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. "Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making". In: *International Symposium on Robotics Research (ISRR)*. 2019.

[51] Daniel Seita, Sanjay Krishnan, Roy Fox, Stephen McKinley, John Canny, and Kenneth 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.

[52] Syohei Shibata, Takashi Yoshimi, Makoto Mizukawa, and Yoshinobu 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.

[53] Changyeob Shin, Peter Walker Ferguson, Sahba Aghajani Pedram, Ji Ma, Erik P. Dutson, and Jacob Rosen. "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[54] Li Sun, Gerarado Aragon-Camarasa, Paul Cockshott, Simon Rogers, and J. Paul Siebert. "A Heuristic-Based Approach for Flattening Wrinkled Clothes". In: *Towards Autonomous Robotic Systems. TAROS 2013. Lecture Notes in Computer Science, vol 8069* (2014).

[55] Li Sun, Gerardo Aragon-Camarasa, Simon Rogers, and J. Paul 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.

[56] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. "Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2020).

[57] Brijen Thananjeyan, Animesh Garg, Sanjay Krishnan, Carolyn Chen, Lauren Miller, and Ken 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.

[58] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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.

[59] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[60] Eric Torgerson and Fanget Paul. "Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1987.

[61] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. "Tensor2Tensor for Neural Machine Translation". In: *CoRR* abs/1803.07416 (2018). URL: http://arxiv.org/abs/1803.07416.

[62] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards". In: *arXiv:1707.08817* (2017).

[63] Loup Verlet. "Computer Experiments on Classical Fluids: I. Thermodynamical Properties of LennardJones Molecules". In: *Physics Review* 159.98 (1967).

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

[65] Bryan Willimon, Stan Birchfield, and Ian Walker. "Model for Unfolding Laundry using Interactive Perception". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[66] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. "Learning to Manipulate Deformable Objects without Demonstrations". In: *arXiv:1910.13439* (2019).

[67] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *arXiv:2003.05436* (2020).

[68] Pin-Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning". In: *IEEE Robotics and Automation Letters (RA-L)*. 2017.