Learning to Perceive and Manipulate Deformable Objects by Leveraging Simulation and Cloud Robotics

Aditya Ganapathi
Ken Goldberg, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-153

http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-153.html

May 20, 2022

Learning to Perceive and Manipulate Deformable Objects by Leveraging Simulation and Cloud Robotics

by

Aditya Ganapathi

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 Joseph E. Gonzalez

Spring 2022

Learning to Perceive and Manipulate Deformable Objects by Leveraging Simulation and Cloud Robotics

by Aditya Ganapathi

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Ken Goldberg
Research Advisor

11 May 2022

(Date)

* * * * * * *

Joseph E. Gonzalez

Professor Joseph E. Gonzalez

Second Reader

5/12/22

(Date)

Learning to Perceive and Manipulate Deformable Objects by Leveraging Simulation and Cloud Robotics

Abstract

Learning to Perceive and Manipulate Deformable Objects by Leveraging Simulation and Cloud Robotics

by

Aditya Ganapathi

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

Deformable object manipulation has been an area of high interest in the robotics community due to its myriad applications in manufacturing, supply chain and hospitality services. However, teaching robots to manipulate deformable objects has proven to be a long standing challenge due to their infinite dimensional configuration space, tendency to self occlude, and complex dynamics. Fortunately, recent advances in deep learning, computer vision and simulation-to-reality have opened up exciting new directions to tackle these challenges. In this work, we study the problem of fabric manipulation through a variety of methods ranging from learning-based perception combined with control to fully end-to-end learning techniques.

First, we study the problem of general-purpose fabric smoothing and folding. While there has been significant prior work on learning policies for specific fabric manipulation tasks, less focus has been given to algorithms which can perform many different tasks. We take a step towards this goal by learning point-pair correspondences across different fabric configurations in simulation. Then, given a single demonstration of a new task from an initial fabric configuration, these correspondences can be used to compute geometrically equivalent actions in a new fabric configuration. This makes it possible to define policies to robustly imitate a broad set of multi-step fabric smoothing and folding tasks. The resulting policies achieve 80.3% average task success rate across 10 fabric manipulation tasks on two different physical robotic systems. Results also suggest robustness to fabrics of various colors, sizes, and shapes. We also propose Multi-Modal Gaussian Shape Descriptor (MMGSD), a new visual representation of deformable objects which extends ideas from dense object descriptors to predict all symmetric correspondences between different object configurations.

Next, we present the first systematic benchmarking of fabric manipulation algorithms on physical hardware using Reach, a cloud robotics platform that enables low-latency remote execution of control policies on physical robots. We develop 4 novel learning-based algorithms that model expert actions, keypoints, reward functions, and dynamic motions, and we compare these against 4 learning-free and inverse dynamics algorithms on the task of folding a crumpled T-shirt with a single robot arm. The entire lifecycle of data collection, model training, and policy evaluation is performed remotely without physical access to the robot workcell. Results suggest a new algorithm combining imitation learning with analytic methods achieves 84% of human-level performance on the folding task.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

There are many people without whom this project would not be possible. Above all, I would like to thank Professor Goldberg, who changed the trajectory of my career 4 years ago when he took me into the lab as a freshman with essentially no experience. He has taught me virtually everything I know about conducting good science. He has also taught me the importance of storytelling, and that we should always strive to present the most beautiful versions of our work.

I would like to thank Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita and Jeffrey Ichnowski for being the best mentors a student could ask for. Their patience, wit, and work ethic still stun me to this day. I would also like to thank my lab mates Ryan Hoque, Yahav Avigal, Priya Sundaresan, Harry Zhang, Mark Presten, and many others who not only brought out the best work in me, but also served as amazing friends.

I would certainly not be here today without the support of my best friends. David, Josh, Tim, Kian, Tony, Ryan: you all have been constants in a turbulent and uncertain time of life, and for that I will forever be grateful. I would also like to thank my new friends Valeria, Henar, Justin, Max, Radu, Corn, Alix, Juanita, Emma and Anna. You all have taught me a new way of looking at life, and it is all the more colorful now because of it.

Last but not least, I would like to thank my family. I simply cannot express how much support they have given me along this journey. Dad, Mom, Anuva: you all are the ”why” behind everything I do.

Chapter 1

Introduction

Robot fabric manipulation has applications in folding laundry [79, 38, 13, 49], bed making [68], surgery [75, 76, 70, 23], and manufacturing [53, 78]. However, while robots are able to learn policies to manipulate a variety of rigid objects with increasing reliability [45, 17, 57, 34, 41], learning such policies for manipulating deformable objects remains an open problem due to difficulties in sensing and control. While there is significant prior work on geometric [65, 79, 3, 46] and learning based approaches [81, 66, 68] for fabric manipulation, these approaches often involve designing or learning task-specific manipulation policies, making it difficult to efficiently reuse information across tasks.

In Chapter 3, we leverage recent advances in dense keypoint learning [17] to learn visual point-pair correspondences across fabric in different configurations. Then, given a single offline demonstration of a fabric manipulation task from a given configuration, we utilize the learned correspondences to compute geometrically equivalent actions to complete the task on a similar fabric in a different configuration. For example, a human might provide a sequence of actions that would fold a T-shirt when it is placed neck up in a smoothed configuration. However, when the robot is deployed, it may encounter a different T-shirt whose color, size and pose differ from the T-shirt used for the demonstration. Learning visual correspondences that are invariant across these fabric attributes provides a powerful representation for defining policies that can generalize to these variations. This chapter is based on [21] and contributes (1) a framework for learning dense visual correspondences of fabric in simulation using dense object descriptors from [17, 74] and applying them to manipulation tasks on real fabrics with unseen colors, scales, and textures, (2) a data generation pipeline for collecting images of fabrics and clothing in Blender [11] and a testbed to experiment with different manipulation policies on these fabrics in simulation and (3) physical experiments on both the da Vinci Research Kit (dVRK) [35] and the ABB YuMi suggesting that the learned descriptors transfer effectively on two different robotic systems. We experimentally validate the method on 10 different tasks involving 5 T-shirts and 5 square fabrics of varying dimensions and colors and achieve an average task success rate of 80.3%.

Chapter 4 is based on [22] and is extension of Chapter 3 as it addresses the issues of uncertainty and symmetries, which can cause issues for downstream planning and control.

Consider a robotic agent attempting to identify a corner of towel to fold it according to a video demonstration. The agent could leverage the fabric's inherent symmetry to manipulate the corner of the towel for which its uncertainty in its location is the lowest, since all four corners are viable options. To enable such behaviors, we extend the correspondence learning algorithms from [74, 17, 21] to (1) provide measures of uncertainty in predicted correspondences by formulating a distribution matching objective for correspondence learning inspired by [16] and (2) explicitly predicting symmetric correspondences. Experiments suggest that the learned correspondences for both 1D and 2D deformable objects are more stable and continuous than those used in prior work and are less prone to symmetrical ambiguities and provide uncertainty estimates.

In Chapter 5, we discuss the importance of reproducibility in robotics research and provide a case study that benchmarks a set of fabric smoothing and folding algorithms on Reach, a prototype hardware testbed from Robotics at Google [80] which provides shared access to remote, standardized hardware via the Cloud. Reach consists of several physical robot workcells as well as open source software for remote execution of control policies in real time. Each workcell is configured for a particular benchmark task: one such task is folding a T-shirt with a UR5 robot arm and 3-jaw piSOFTGRIP gripper [55]. This work is based on [29] and was performed in collaboration with Ryan Hoque, Kaushik Shivakumar, Shrey Aeron, Gabriel Deza, myself and Professor Goldberg. The work contributes (1) four novel learning-based algorithms for the folding task, (2) implementation of and comparison with four additional benchmarks, and (3) a case study of robotics research performed exclusively using a remotely managed robot workcell. This paper does *not* contribute the design of the Reach cloud robotics platform, which is being developed by a larger team at Google [80].

Chapter 2

Related Work

2.1 Fabric Manipulation

Fabric manipulation is an active area of robotics research [5, 63, 39, 31]. Over the past decade, the research has primarily been focused on three different categories: perception-based manipulation, learning-based algorithms in the real world, and learning-based algorithms in simulation which are then transferred to real robots.

Traditional Vision-Based Algorithms for Fabric Manipulation

Much of the prior work on perception-based deformable object manipulation relies on traditional image processing techniques to estimate fabric state. This state estimation is then used to define geometric controllers which bring the fabric into some desired configuration. However, due to the generalization challenges faced by these algorithms, most prior work makes specific assumptions on the fabric's initial configurations or requires more complex robotic manipulators to bring the fabric into a desired starting configuration. For example, Miller *et al.* [49] demonstrate a robust folding pipeline for clothing by fitting a polygonal contour to the fabric and designing a geometric controller on top of it, but assume that the initial state of the fabric is flat. Sun *et al.* [72, 73] perform effective fabric smoothing by estimating the wrinkles in the fabric, but condition on a near-flat starting fabric. Other work relies on "vertically smoothing" fabrics using gravity [52, 37, 38, 13, 46] to standardize the initial configuration and to expose fabric corners before attempting the task, which is difficult for large fabrics or single-armed robots.

Learning-Based Algorithms in the Real World

More recent approaches have focused on end-to-end learning of fabric manipulation policies directly on a real system, but these approaches can fail to generalize to a variety of fabrics and tasks due to the high volume of training data required. For example, [14] use model-based reinforcement learning to learn fabric manipulation policies which generalize to many tasks, but require several days of continuous data collection on a real physical system and perform relatively low precision tasks. Jia *et al.* [33, 32] show impressive collaborative human-robot cloth folding under the assumption that fabric has already been grasped and is in a particular starting configuration, and [65] demonstrate deformable object manipulation while requiring task-specific kinesthetic demonstrations. In follow-up work, [40] consider many of the same tasks as in this paper and demonstrate that policies can be learned to fold fabric using reinforcement learning with only one hour of experience on a real robot. In contrast, we learn entirely in simulation and decouple perception from control, making it easier to generalize to different fabric colors and shapes and deploy the learned policies on different robots without further learning.

Sim-to-Real Learning-Based Algorithms

Due to the recent success of sim-to-real transfer [62, 77], many recent papers leverage simulation to collect large amounts of training data, which is used to learn fabric manipulation policies. Seita *et al.* [68, 66] and [81] address the smoothing task from [72] but generalize to a wider range of initial fabric states using imitation learning (DAgger [60]), and reinforcement learning (Soft Actor-Critic [26]) respectively. Similarly, [47] learn fabric folding policies by using deep reinforcement learning augmented with task-specific demonstrations. However, these works learn policies that are specialized only to fabric smoothing [68, 66] and folding [81] respectively. In follow-up and concurrent work, [30] and [84] use simulation to train fabric manipulation policies using model-based reinforcement learning for multiple tasks. In contrast, we leverage simulation to learn visual representations of fabric to capture its geometric structure without task-specific data or a model of the environment and then use this representation to design intuitive policies for several tasks from different starting configurations.

Dense Object Descriptors

We learn visual representations for fabric by using dense object descriptors [17, 64], which were shown to enable task oriented manipulation of various rigid and slightly deformable objects [17]. This approach uses a deep neural network to learn a representation which encourages corresponding pixels in images of an object in different configurations to have similar representations in embedding space. Such descriptors can be used to design geometrically structured manipulation policies for grasping [17], assembly [87], or for learning from demonstrations [15]. [74] extend this idea to manipulation of ropes, and demonstrate that deformation-invariant dense object descriptors can be learned for rope using synthetic depth data in simulation and then transferred to a real physical system. [74] then use the learned descriptors to imitate offline demonstrations of various rope manipulation tasks. In this work, we apply the techniques from [74] to learn descriptors which capture geometric correspondence across different fabric configurations from synthetic RGB images and use them for 2D fabric manipulation.

2.2 Remote Testbeds

The most similar remote robotics testbed initiative is from Bauer et al. [4], who hosted the online "Real Robot Challenge" in 2020 and in 2021 at Neural Information Processing Systems (NeurIPS). Six robotics groups from around the world were able to access their tri-finger robot [83] remotely via the Internet and evaluate their algorithms on the shared infrastructure. Our study differs from this project in the following ways: (1) they consider dexterous manipulation of rigid objects while we consider deformable object manipulation; (2) they use a custom tri-finger robotic system while we use a UR5 robot arm, standard in industrial settings; and (3) the Real Robot Challenge submissions are either learning-free [20, 9, 86] or learned only in simulation [48, 2], while we consider learning algorithms trained on real data. Other remote testbeds that do not consider manipulation are the Robotarium [56] for swarm robotics and Duckietown [54] for autonomous driving.

2.3 Reproducibility in Robotics

Several other approaches have been proposed for facilitating reproducibility in robotics research. One direction is benchmarking in simulation, where evaluation is inexpensive and reproducible. Simulation environments have been developed for robot locomotion [6], household tasks [71], and deformable object manipulation [43]. While researchers have made significant progress on these benchmarks [27, 19], especially using reinforcement learning, such advances do not readily transfer to physical robots [10]. Another initiative for improving reproducibility is development of a low-cost open source platform that can be assembled independently by different labs [1, 85, 83]. A third approach considers benchmarking performance on large offline datasets such as robot grasps on 3D object models, e.g., EGAD [50] and Dex-Net [44]; RGBD scans and meshes of real-world common household objects, e.g., the YCB Object and Model set [7]; and video frames of robot experience, e.g., RoboNet [12]. These datasets have been used to explore and compare algorithms [36], but they limit evaluation to states within the dataset.

Chapter 3

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

3.1 Introduction

This chapter is based on [21], where we leverage recent advances in dense keypoint learning [17] to learn visual point-pair correspondences across fabric in different configurations. Then, given a single offline demonstration of a fabric manipulation task from a given configuration, we utilize the learned correspondences to compute geometrically equivalent actions to complete the task on a similar fabric in a different configuration. For example, a human might provide a sequence of actions that would fold a T-shirt when it is placed neck up in a smoothed configuration. However, when the robot is deployed, it may encounter a different T-shirt whose color, size and pose differ from the T-shirt used for the demonstration. Learning visual correspondences that are invariant across these fabric attributes provides a powerful representation for defining policies that can generalize to these variations. See Figure 3.1 for an overview.

3.2 Problem Statement

Assumptions

We assume a deformable object is on a planar workspace in initial configuration ξ_1 with overhead RGB image observation $I_1 := I_1(\xi_1) \in R^{W \times H \times 3}$. As in prior work [66, 81], we focus on fabric manipulation tasks that can be completed by a sequence of pick and place actions. Precisely, each action involves grasping the fabric at a *pick point*, pulling to a *place point* without changing the orientation of the end-effector, and releasing the fabric. We assume that the pick point and place point are both visible in the camera frame and that the camera

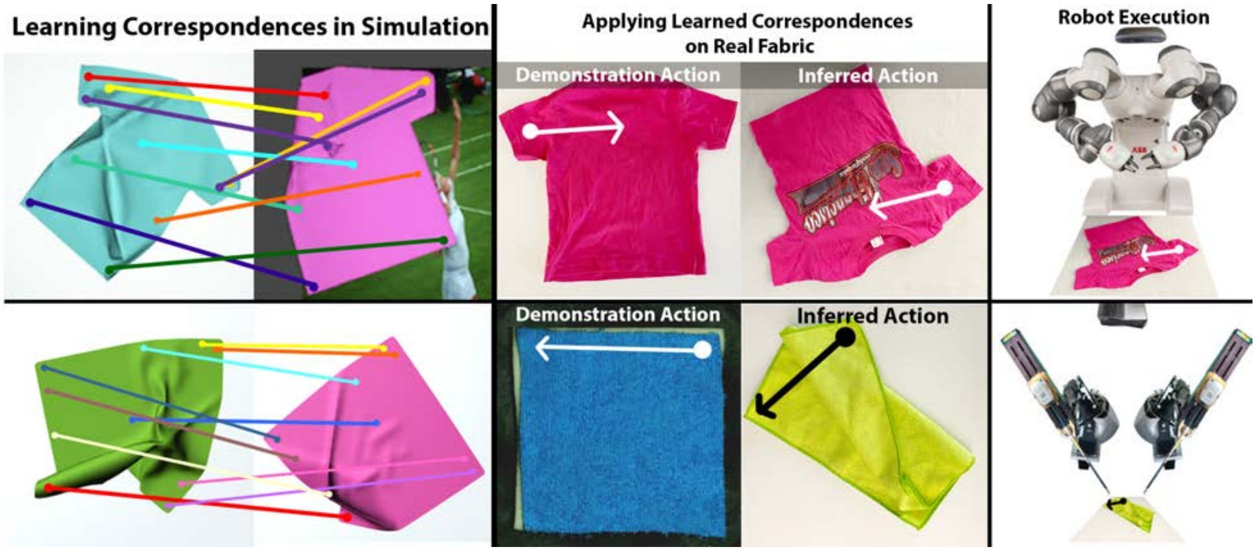

Learning Correspondences in Simulation
Applying Learned Correspondences on Real Fabric
Robot Execution
Demonstration Action
Inferred Action
Demonstration Action
Inferred Action

Figure 3.1: We use learned visual correspondences across different fabric configurations to perform a variety of fabric manipulation tasks on the ABB YuMi (top) and the da Vinci Research Kit (bottom). Given a single demonstration of smoothing or folding, the robot uses the learned correspondences to compute geometrically equivalent actions for fabric of different color and in different initial configurations. This enables robust one-shot imitation learning of tasks that involve smoothing then folding.

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

Task Definition

Define the action at step j as

$$\mathbf{a}_j = ((x_g, y_g)_j, (x_p, y_p)_j) \tag{3.1}$$

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

$$\left(\mathbf{a}'_j\right)_{j=1}^n = \left(d_{I_j \to I'_j}(x_g, y_g)_j, \;\; d_{I_j \to I'_j}(x_p, y_p)_j\right)_{j=1}^n \tag{3.2}$$

for $j \in \{1, \ldots, n\}$ where $d_{I_j \to I'_j} : R^2 \to R^2$ is a function which estimates the corresponding point $(x', y')_j$ in I'_j given a point $(x, y)_j$ in I_j. This function is difficult to compute directly from images in general, and even more so for images of highly deformable objects due to their infinite degrees of freedom and tendency to self-occlude. Thus, we leverage dense object descriptors [17] to approximate $d_{I_j \to I'_j}$ for any I_j and I'_j, as described in Sections 3.4 and 3.5.

3.3 Fabric Simulator

We use Blender 2.8, an open-source simulation and rendering engine [11] released in mid-2019, to both create large synthetic RGB training datasets and model the fabric dynamics for simulated experiments using its in-built fabric solver based on [59, 58]. We simulate T-shirts and square fabrics, each of which we model as a polygonal mesh made up of 729 vertices, a square number we experimentally tuned to trade-off between fine-grained deformations and reasonable simulation speed. See Figure 3.2 for an illustration. Each vertex on the mesh has a global coordinate which we can query directly through Blender's API, allowing for easily available ground truth information about various locations on the mesh and their pixel counterparts. We can also simulate finer-grained manipulation of the mesh including grasps, pulls, and folds. See the supplement for further details on how we perform manipulation and experiments in simulation.

3.4 Dense Shape Descriptor Training

Dense Object Descriptor Training Procedure

We consider an environment with a deformable fabric on a flat workspace and learn policies that perform smoothing and folding tasks. The policies are defined using learned point-pair correspondences between overhead images of the fabric in different configurations. We generate deformation-invariant correspondences by training dense object descriptors [74, 17] on synthetically generated images of the fabric in different configurations.

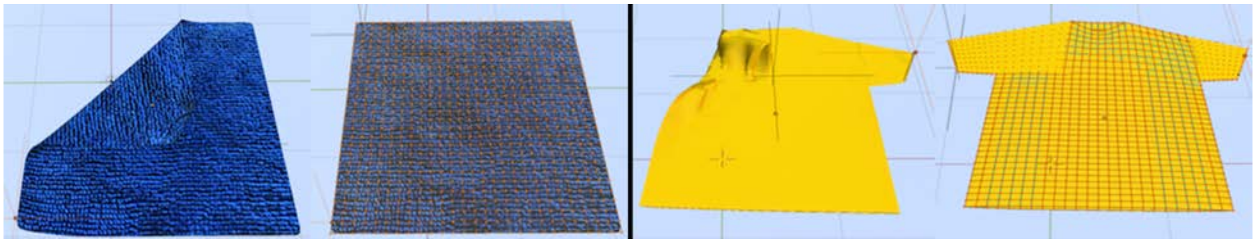


Figure 3.2: **Fabric Meshes:** Examples of the meshes generated in Blender for both square cloth (left) and t-shirts (right). The ground-truth vertices are highlighted in the second and fourth columns.

In [17], an input image I is mapped to a descriptor volume $Z = f_\theta(I) \in R^{W \times H \times D}$, where each pixel (i, j) has a corresponding descriptor vector $Z_{i,j} \in R^D$. Descriptors are generated by a Siamese network f_θ and are guided closer together for corresponding pixels in images and pushed apart by at least some margin M for non-corresponding pairs by minimizing a pixel-wise contrastive loss function during training [17]. Corresponding pairs of pixels represent the same point on an object. In this work, we also train a Siamese network to cluster corresponding pixel pairs and seperate non-corresponding pixel pairs in descriptor space. Since ground-truth pixel correspondences are difficult to obtain in images across deformations of a real fabric, we train the network on synthetic RGB data from Blender (see Section 3.3), where perfect information about the pixel correspondences is available through the global coordinates of the fabric mesh's vertices. Note that during training, the sampled image inputs to the Siamese network are enforced to be of the same fabric type to ensure valid correspondences. That is, two different images of T-shirts can be passed into the network, but not a T-shirt and square fabric. Figure 3.3 demonstrates the pipeline for predicting descriptors for correspondence generation. The learned descriptors can then be used to approximate the correspondence function $d_{I \to I'}$ described in Section 4.2 by (1) computing the top k pixel matches based on their distance in descriptor space and (2) computing the geometric median of these matches in pixel space:

$$
\begin{aligned}
((i''_l, j''_l))_{l=1}^{k} &=_{(i'_1,j'_1)\ldots(i'_k,j'_k)} \sum_{l=1}^{k} \|f_\theta(I)_{i,j} - f_\theta(I')_{i'_l,j'_l}\|_2 \\
&\text{s.t. } (i'_n, j'_n) \neq (i'_m, j'_m)\ \forall m, n \in [k] \\
d_{I \to I'}(i, j) &=_{(i',j')} \sum_{l=1}^{k} \|(i', j') - (i''_l, j''_l)\|_2
\end{aligned}
$$

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

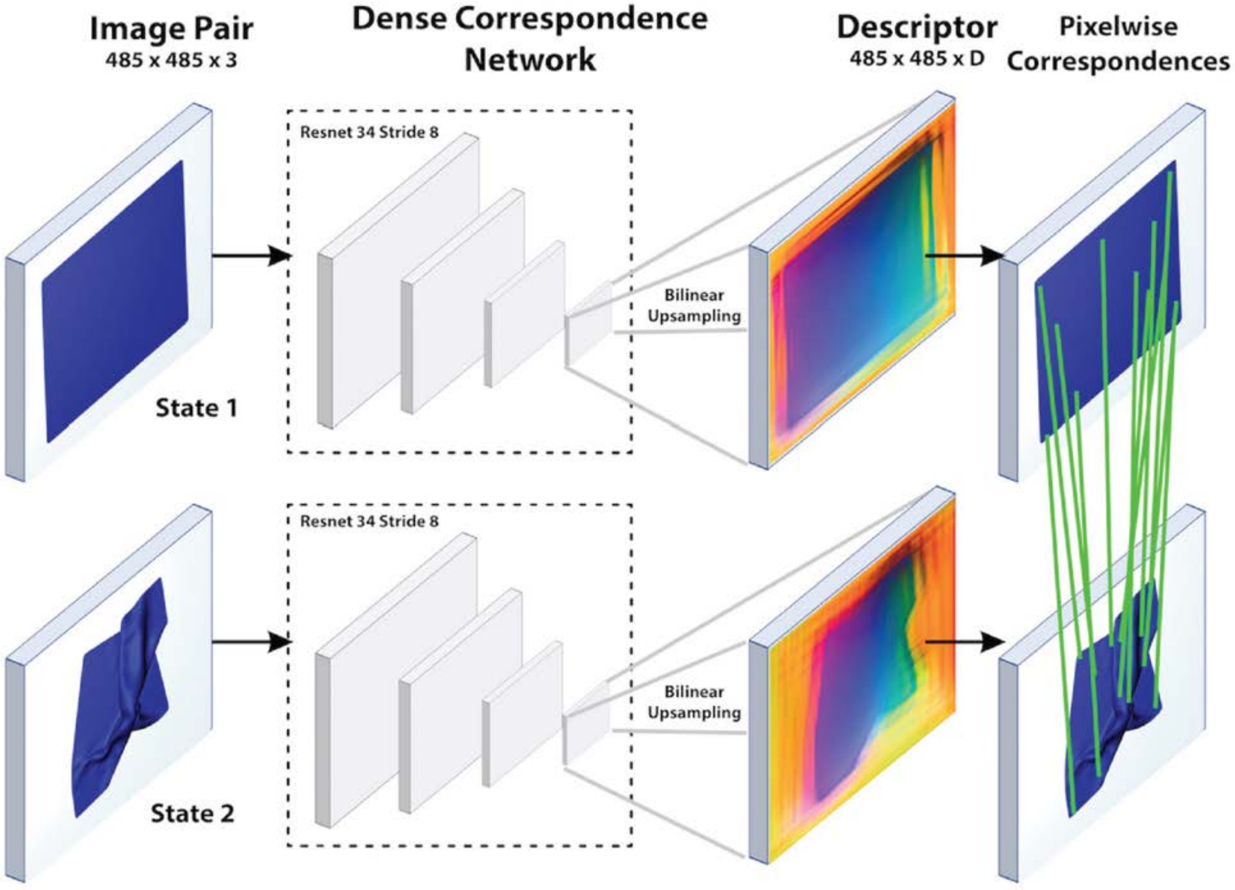


Figure 3.3: **Learning Visual Correspondences:** pipeline for training dense object nets for robot fabric manipulation. Left: we train a dense correspondence network on pairs of simulated fabric images to learn pixel-wise correspondences using a pixel-wise contrastive loss. Right: we use the learned descriptors for policy optimization. We can use correspondence to map a reference action to a new fabric configuration. For example, we show an image of a wrinkled fabric in “State 2,” and we can use descriptors to figure out the action needed to smooth the fabric from “State 2” to “State 1.”

Dataset Generation and Domain Randomization

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

Simulating soft-body animations is in general a computationally time-consuming process which makes it difficult to render large datasets in short periods of time. We take steps toward mitigating this issue by rendering 10 images per drop, allowing us to collect 10x as much data in the same time period. In simulation, we found that the test time pixel match error was unaffected when including these unsettled images of the fabric in the dataset. We additionally make use of domain randomization [62, 77] by rendering images of the scene while randomizing parameters including mesh size, lighting, camera pose, texture, color and specularity (see supplement for further details). We also restrict the rotation about the z-axis to be between $(-\pi/4, \pi/4)$ radians to reduce ambiguity during descriptor training due to the natural symmetry of fabrics such as squares. To randomize the image background, we sample an image from MSCOCO [42] and "paste" the rendered fabric mask on top. For experiments, we generate one (domain-randomized) dataset, including both T-shirts and square fabric, and train a *single model* which we use for all experiments in Section 4.4. For reference, generating a single dataset of 7,500 images, half T-shirts and half square cloth, with 729 annotations per image takes approximately 2 hours on a 2.6GHz 6-core Intel Core i7 MacBook Pro.

3.5 Descriptor-Parameterized Policies

As discussed in Section 3.2, the robot receives a demonstration of the task consisting of actions $(\mathbf{a}_j)_{j=1}^n$ and observations $(I_j)_{j=1}^n$. At execution time, the robot starts with the fabric in a different configuration, and the fabric itself may have a different texture or color. At time $j \in [n]$, the robot observes I'_j then executes $\pi_j(I'_j) = \Big(d_{I_j \to I'_j}(x_g, y_g)_j,\ \ d_{I_j \to I'_j}(x_p, y_p)_j\Big)$ where $d_{I_j \to I'_j}$ is defined in Section 3.4. We train a single descriptor network for a variety of tasks and use it to identify correspondences in different fabric configurations from those in the supplied in demonstrations. π_j then uses these correspondences to identify semantically relevant pixels in I'_j to generate actions that manipulate these keypoints.

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

Fabric Smoothing

In the square fabric smoothing task, the robot starts with a crumpled fabric and spreads it into a smooth configuration on a planar workspace as in [66]. To complete this task, we use the approach from [66] and iterate over fabric corners, pulling each one to their target locations on an underlying plane. However, while [66] design a policy to do this using ground-truth knowledge of the fabric in simulation, we alternatively locate corners on the crumpled fabric using a learned descriptor network and a source image of a flat fabric where the corners are labeled. For the T-shirt smoothing task, we apply a similar method, but instead iterate over the corners of the sleeves and the base of the T-shirt.

Fabric Folding

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

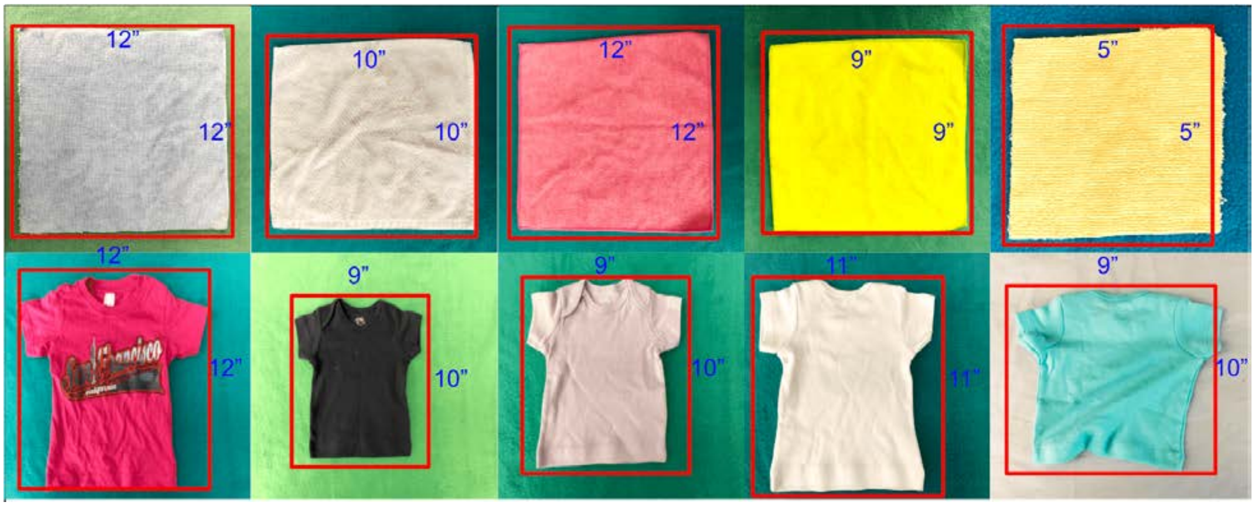


Figure 3.4: **Fabric Specifications:** Images and dimensions of the square fabrics and shirts we use in experiments.

3.6 Experiments

We experimentally evaluate (1) the quality of the learned descriptors and their sensitivity to training parameters and (2) the performance of the descriptor-parameterized policies from Section 3.5 across 10 different fabric manipulation tasks on two physical robotic systems, the da Vinci Research Kit (dVRK) [35] and the ABB YuMi. Results suggest that the learned descriptors and the resulting policies are robust to changes in fabric configuration and color.

Tasks

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

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

All fabrics are varied either in dimension or color according to Figure 3.4. Additionally, we execute a subset of these tasks in simulation. A single visual demonstration consisting of up to 4 actions is provided to generate a policy which the robot then tries to emulate in the same number of actions.

Experimental Setup

We execute fabric folding and smoothing experiments on the dVRK [35] and ABB YuMi robot. The dVRK is equipped with the Zivid OnePlus RGBD sensor that outputs 1900×1200 pixel images at 13 FPS at depth resolution 0.5 mm. The workspace of the dVRK is only 5” × 5”, so we use only square fabric of the same dimension while varying the color according to Figure 3.4. Manipulating small pieces of fabric into folds is challenging due to the elasticity

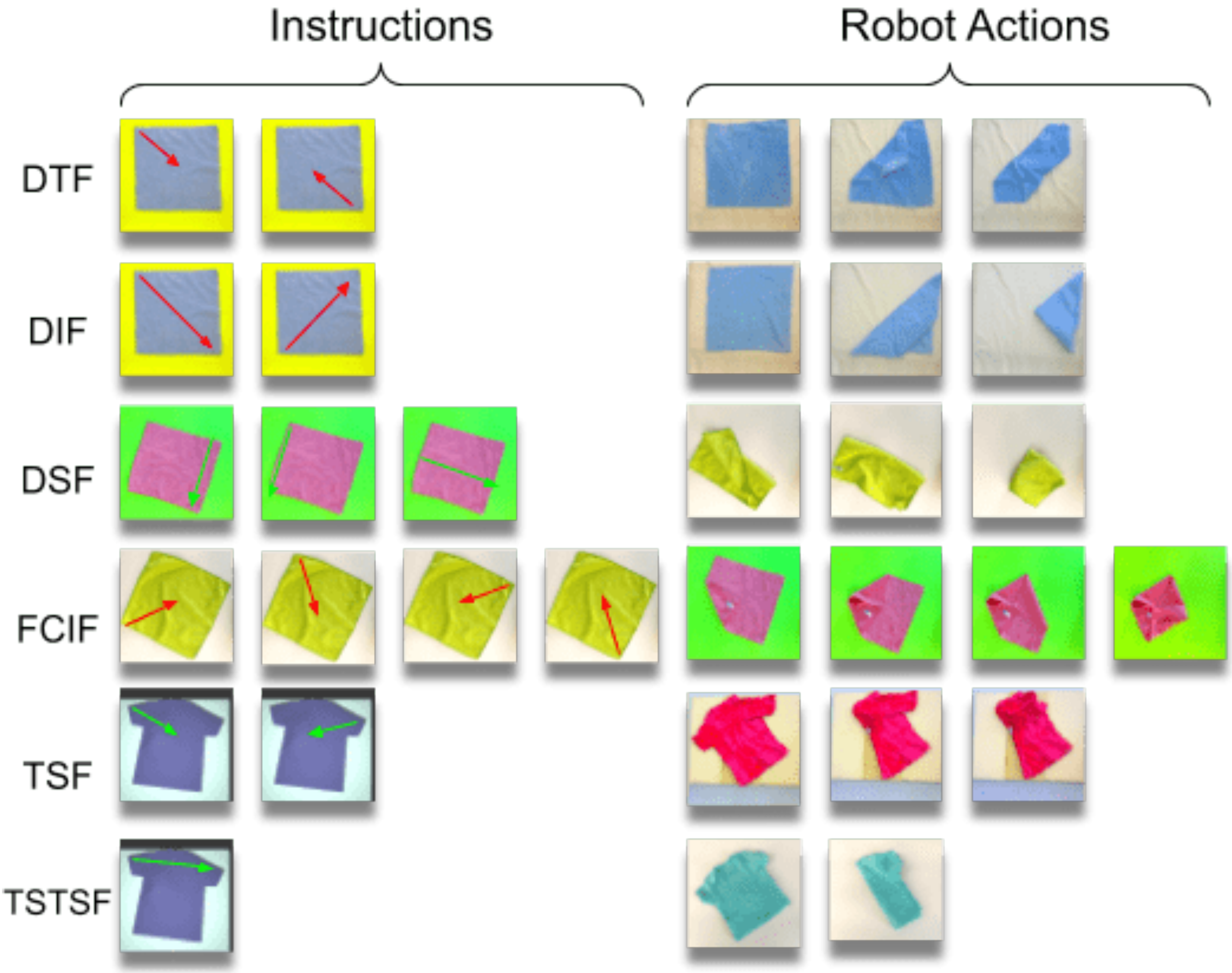


Figure 3.5: **Policy Rollouts:** Policy execution on the YuMi for tasks 2, 3, 4, 5, 6 and 7 as described in Section 3.6. The first four columns show the folding instructions on some initial fabric and the last four show the corresponding folds executed on novel starting configurations for a different fabric.

of the fabric, so we add weight to the fabric by dampening it with water. Additionally, we place a layer of 1 inch foam rubber below the fabric to avoid damaging the gripper. The YuMi has a 36” × 24” workspace, and since only one arm is utilized resulting in a more limited range of motion, we only manipulate at most 12” × 12” pieces of fabric which we do not dampen. In this setup we use a 1080p Logitech webcam to collect overhead color images. For the YuMi, we use both T-shirts and square fabric of varying dimension and color but go no lower than 9” × 9” fabrics due to its larger gripper. Finally, for both robots, we use a standard pixel to world calibration procedure to get the transformation from pixel

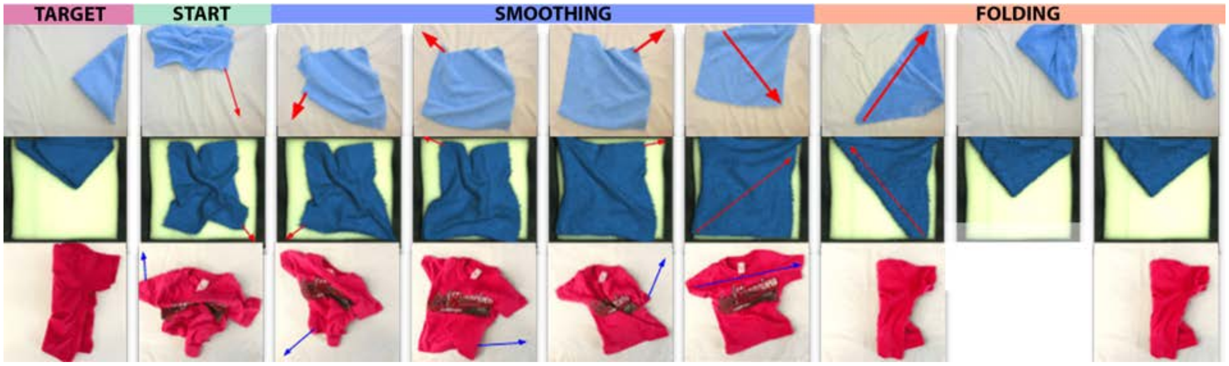


Figure 3.6: **Full Folding Sequence:** The first and second row is a time-lapse of a sequence of 6 actions taken by the YuMi and dVRK respectively, and with actions overlaid by red arrows, to successively smooth a wrinkled fabric and then fold it according to task 3 in Section 3.6. The third row is a time-lapse of a sequence of 5 actions taken by the YuMi to complete task 10 in Section 3.6. Here, robot actions are overlaid with blue arrows.

coordinates to planar workspace coordinates.

For both robots, we follow the same experimental protocol. We manually place the fabric in configurations similar to those shown in Figure 3.2 and deform them by pulling at multiple locations on the fabric. To obtain image input for the descriptor networks, we crop and resize the overhead image to be 485 × 485 such that the fabric is completely contained within the image. Although lighting conditions, camera pose and workspace dimensions are significantly different between the two robotic systems, no manual changes are made to the physical setup. We find that the learned descriptors are sufficiently robust to handle this environmental variability.

We evaluate the smoothing task by computing the coverage of the cropped workspace before and after execution. For the folding tasks, as in [40], we consider an outcome a success if the final state is visually consistent with the goal image. Conventional quantitative metrics such as intersection of union between the final state and a target image provide limited diagnostic information when starting configurations are significantly different as in the presented experiments.

3.7 Results

We evaluate the smoothing and folding policies on both the YuMi and dVRK on square fabrics and T-shirts. Table 3.2 shows the success rates of our method on all proposed tasks in addition to a breakdown of the failure cases detailed in Table 3.3. We observe that the descriptor-parameterized controller is able to successfully complete almost all folding tasks at least 75% of the time, and the smoothing policies are able to increase coverage of the cloth to over 83% (Table 3.1). The execution of the smoothing policy followed by the double triangle folding policy results in successful task completion 6/10 and 8/10 times on the YuMi and dVRK respectively. We find that the most frequent failure mode is an unsuccessful grasp of the fabric which is compounded for tasks that require more actions. Though this is independent of the quality of the learned descriptors, it highlights the need for more robust methods to grasp highly deformable objects.

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

Table 3.1: **Physical Fabric Smoothing Experiments:** We test the smoothing policies designed in Section 3.5 on the YuMi and the dVRK. Both robots achieve an average increase in coverage of $11 - 22$ percent.

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

Table 3.2: **Physical Fabric Folding Experiments:** We test the folding policies from Section 3.5 on the YuMi and the dVRK. We observe both robots are able to perform almost all folding tasks at least 75 percent of the time. The YuMi is able to perform the smoothing then folding task 6/10 times and the dVRK is able to do so 8/10 times.

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

Table 3.3: **Failure Mode Categorization**

Chapter 4

Multi-Modal Gaussian Shape Descriptors for Correspondence Matching in 1D and 2D Deformable Objects

4.1 Introduction

This chapter is based on [22] and is an extension to Chapter 3. Here we address the issue of symmetry that arises in Chapter 3 when an object has multiple correspondences (i.e. the four corners of a square cloth). We extend the correspondence learning algorithms from [74, 17, 21] to (1) provide measures of uncertainty in predicted correspondences by formulating a distribution matching objective for correspondence learning inspired by [16] and (2) explicitly predicting symmetric correspondences. Experiments suggest that the learned correspondences for both 1D and 2D deformable objects are more stable and continuous than those used in prior work and are less prone to symmetrical ambiguities and provide uncertainty estimates. See Figure 4.1 for an overview.

4.2 Problem Statement

Given two images, I_a and I_b, of a deformable object in two different configurations respectively, and a source pixel location (u_a, v_a) (such that the pixel is $I_a[u_a, v_a]$), find its $n(u_a, v_a)$ pixel correspondences $((u_{b_i}, v_{b_i}))_{i=1}^{n(u_a,v_a)}$ in I_b. There may be multiple possible matches due to symmetry, such as when matching a corner of a square cloth in I_a to all four corners of the cloth in I_b. We assume access to a dataset of pairs of images of deformable objects, for which $n(u_a, v_a)$ is known, and a collection of correspondences and non-correspondences between each pair. We use Blender 2.8 [11] to both generate arbitrary configurations of cloth and rope in simulation as well as to render images of these configurations for dataset

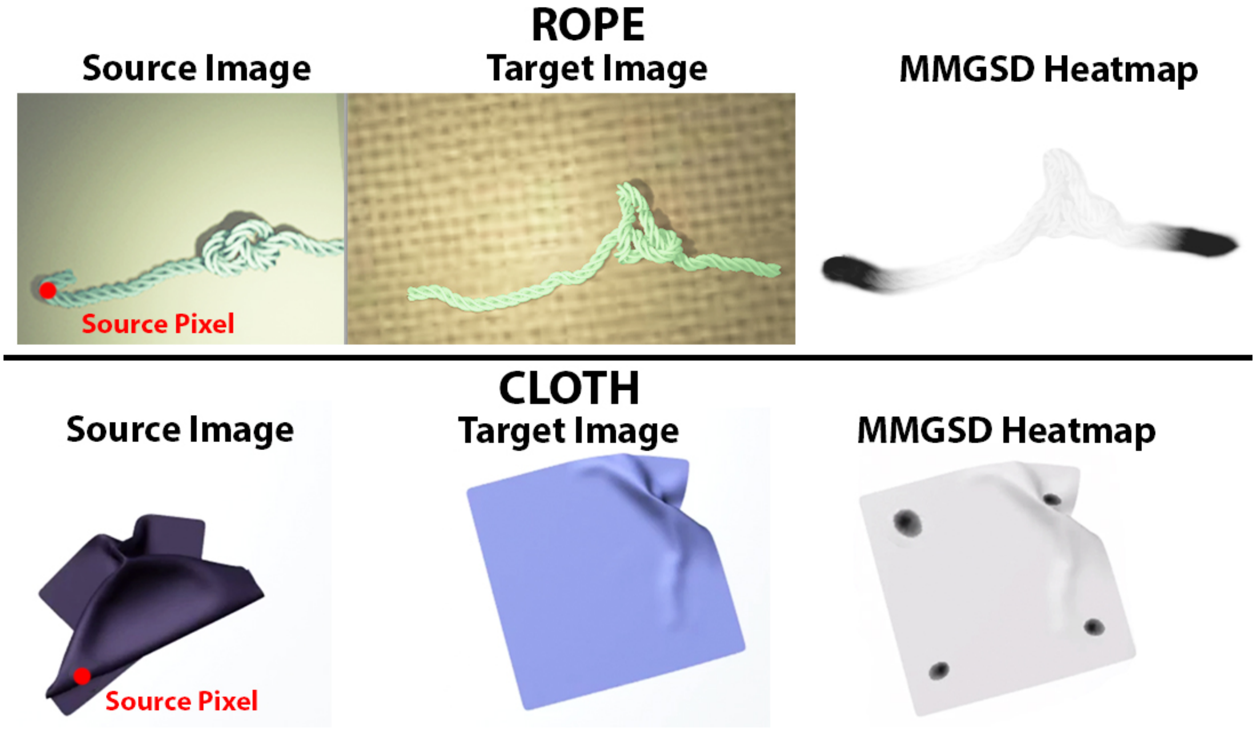


Figure 4.1: Multi-Modal Gaussian Shape Descriptors (MMGSD) learns symmetry-aware pixelwise correspondences for multi-modal semantic keypoints between source and target images of deformable objects. We visualize the results of 2-modal and 4-modal MMGSD correspondence heatmaps for rope and cloth, respectively, relative to the source pixels from column 1.

curation. Blender gives us access to the underlying mesh vertices that these objects are composed of which allows us to densely sample mesh vertex pixel locations at any point.

4.3 Methods

Preliminaries: Pixel-wise Contrastive Loss

We first review the unimodal matching method from [21, 74, 17, 64]. A neural network f maps I_a to a D-dimensional descriptor volume: $f : R^{W\times H\times 3} \longmapsto R^{W\times H\times D}$. During training, a pair of images and sets of both matching pixels and non-matching pixels are sampled between the image pair. The following contrastive loss minimizes descriptor distance between matching pixels and pushes descriptors for non-matching pixels apart by a fixed margin M:

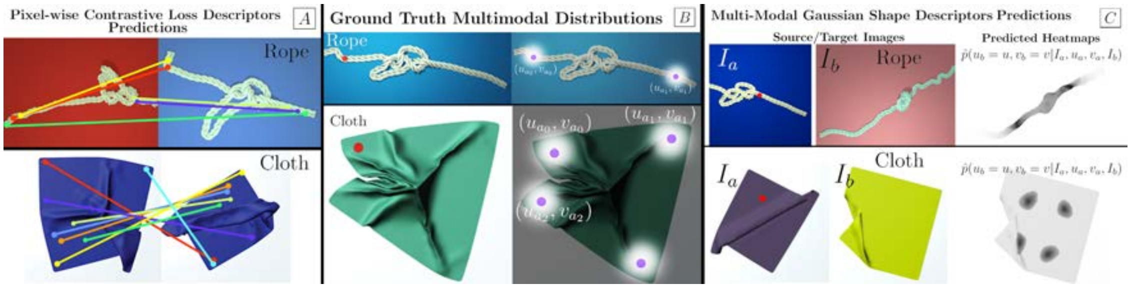

Pixel-wise Contrastive Loss Descriptors Predictions
A
Rope
Cloth
Ground Truth Multimodal Distributions
B
Rope
(u_{a_0}, v_{a_0})
(u_{a_1}, v_{a_1})
Cloth
(u_{a_0}, v_{a_0})
(u_{a_1}, v_{a_1})
(u_{a_2}, v_{a_2})
Multi-Modal Gaussian Shape Descriptors Predictions
C
Source/Target Images
Predicted Heatmaps
I_a
I_b
Rope
$\hat{p}(u_b = u, v_b = v | I_a, u_a, v_a, I_b)$
I_a
I_b
Cloth
$\hat{p}(u_b = u, v_b = v | I_a, u_a, v_a, I_b)$

Figure 4.2: We visualize the multi-modal ground truth distributions (B) and predicted (A,C) correspondences on domain-randomized images of both cloth and rope. Contrastive descriptor-based methods can fail to generalize to objects with inherent symmetries, as in example A. In contrast, the predicted 2-modal and 4-modal MMGSD heatmaps for rope and cloth, respectively, appear to be sensitive to object symmetries. MMGSD exhibits the least uncertainty at object extremities and greater variance as the source pixel moves inward (C), as shown in the probability mass distributed around the rope knot (top heatmap of C).

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

While this method is effective at determining pixel-wise correspondences for both cloth and rope [74, 21], it does not account for inherent symmetry in these objects and therefore is susceptible to symmetric orientation based error as is shown in the 2D cloth example of Figure 4.2B. The authors of [21] address this by limiting the rotation of the training data for a square fabric to be between $(-\frac{\pi}{4}, -\frac{\pi}{4})$, but the model still suffers from symmetric ambiguities at the boundary conditions. The authors of [74] break symmetry by adding a ball to the end of the rope.

Symmetric Pixel-wise Contrastive Loss (SPCL) Baseline

This method extends Section 4.3 to handle multiple matches for the same source pixel (u_a, v_a). Now, we try to match equivalent source pixels $((u_{a_i}, v_{a_i}))_{i=0}^n$ to a set of destination pixels $((u_{b_i}, v_{b_i}))_{i=0}^n$ that are equivalent due to symmetry by adding all pairs of pixels as matches. We use the same loss function as in Section 4.3.

While this method addresses the symmetry issue from method 4.3 by learning to find multiple matches in the destination image for an input pixel, we find that it is unstable and has discontinuity issues due to the contrastive nature of training. During test time, we create a heatmap of the target image by normalizing the descriptor norm differences. We then fit an $n(u_a, v_a)$-modal Gaussian distribution to the heatmap and take the $n(u_a, v_a)$ pixel modes as the predicted symmetric correspondences.

Symmetric Distributional Loss (MMGSD)

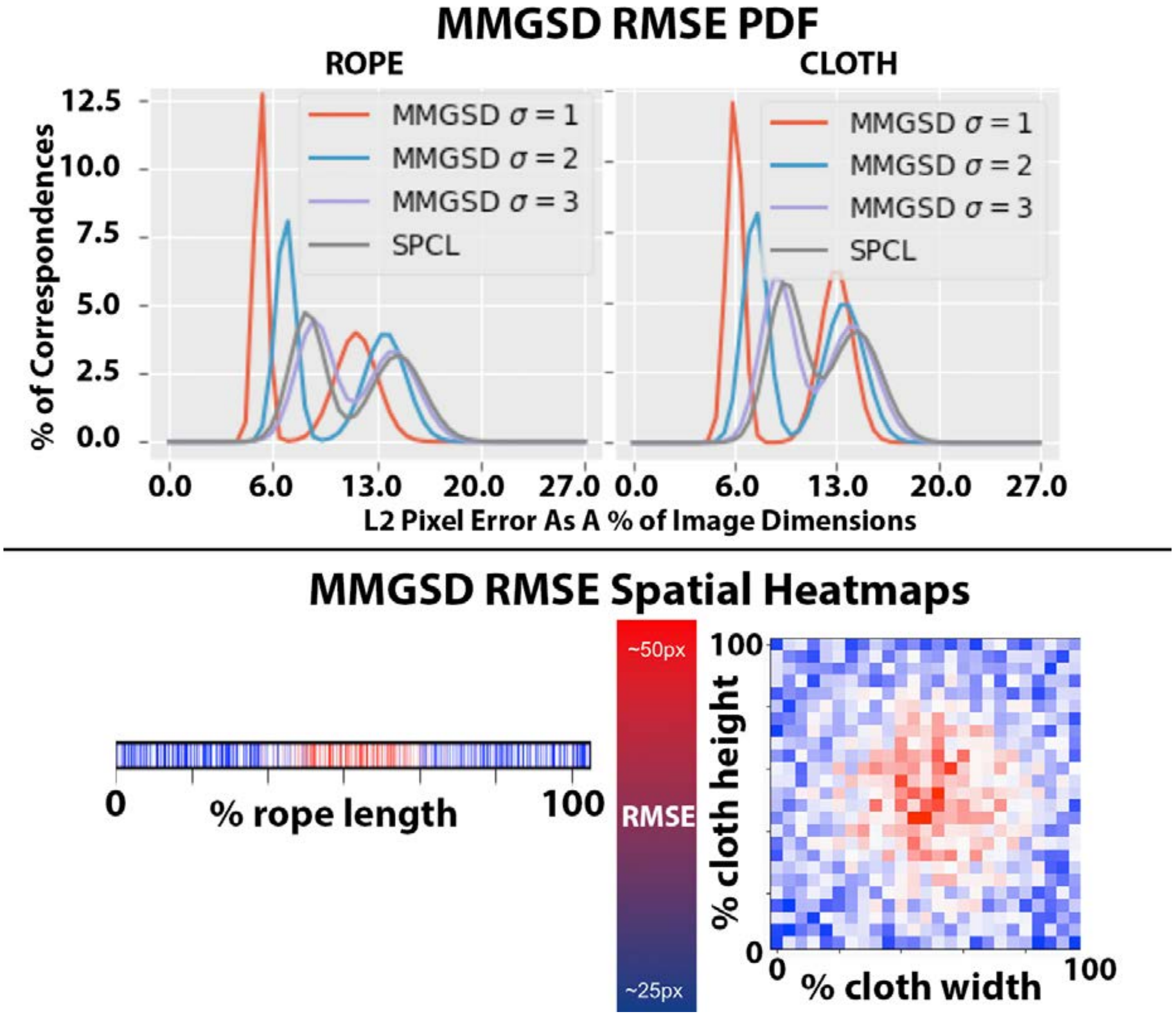
We extend a distributional descriptor network method suggested in [16] to learn an estimator $\hat{p}(u_b = u, v_b = v | I_a, u_a, v_a, I_b)$ that outputs the probability that (u, v) in I_b matches with (u_a, v_a) in I_a. Specifically, we let $\hat{p}(u_b = u, v_b = v | I_a, u_a, v_a, I_b) = \frac{\exp \|f(I_a)[u_a,v_a]-f(I_b)[u,v]\|_2^2}{\sum_{u',v'} \exp \|f(I_a)[u_a,v_a]-f(I_b)[u',v']\|_2^2}$, where f is a neural network with trainable parameters. To fit $\hat{p}$, we use the cross-entropy loss function with respect to a target distribution p that is an isotropic Gaussian mixture model with modes at all the ground truth pixel correspondences in I_b, thus accounting for all symmetric matches. For the ground truth target distributions, σ is empirically fine-tuned to tradeoff spatial continuity in the learned distribution with overlap and collapse of modes. Using this distributional divergence loss function maintains spatial continuity between matches, and we find that this can be more stable than the method in Section 4.3. Additionally, predicting a distribution instead allows uncertainty estimation by computing the entropy of the predicted distribution. This method is similar to [16] but uses a multi-modal target distribution due to the multiple symmetric correspondences. As illustrated in Figure 4.2B and Figure 4.1B, this method is successfully able to place its mass at the multiple possible matches in the target image. We fit an $n(u_a, v_a)$-modal Gaussian distribution to the predicted output distribution $\hat{p}$ and take the $n(u_a, v_a)$ pixel modes as the predicted symmetric correspondences.

4.4 Quantitative Results

We evaluate the quality of the symmetric learned correspondences (methods 4.3 and 4.3) using the root-mean-square error (RMSE) metric. Both the rope and cloth networks are trained on 3,500 training images each and evaluated on a held-out test set of 500 images. All training and testing is carried out with images of a synthetic square cloth and braided synthetic nylon rope. The cloth images are 485 × 485 and the rope images are 640 × 480 in aspect ratio. We compute the $n(u_a, v_a)$ pixel mode predictions and compare them directly to the ground truth pixel locations: $\dfrac{1}{n(u_a, v_a)} \sum_{i=1}^{n(u_a,v_a)} ||[\hat{u}_{b_i}, \hat{v}_{b_i}] - [u_{b_i}, v_{b_i}]||_2^2$ where $[u_{b_i}, v_{b_i}]$ is the ground truth pixel correspondence in I_b for the source pixel $[u_a, v_a]$. We average over 625 source pixel locations in each of 500 test image pairs from simulation (Figure 4.3) using a model trained on 3500 image pairs of cloth and rope each.

In Figure 4.3 we compare MMGSD against SPCL with the probability density function of percentage of correspondences below an L2 pixel threshold (as a percentage of the pixel dimensions of the object). We note that while MMGSD is able to predict multi-modal correspondences more effectively than SPCL, it exhibits high uncertainty and modal collapse for highly occluded regions, such as rope knots (Figure 4.2C), object interiors, or occluded fabric corners. This high degree of variance in the resulting heatmaps is a consequence of MMGSD attempting to preserve spatial continuity, at the expense of concentrating probability mass in isolated symmetric regions. We illustrate this in the bottom half of Figure 4.3

by visualizing the source of high RMSE error on both rope and cloth. The top half of Figure 4.3 also reveals this second mode centered at higher RMSE error.



RMSE of Symmetric Correspondences

Figure 4.3: We find that MMGSD, trained with $\sigma = 1$px, is able to more effectively learn symmetric correspondences over SPCL, evaluated by the PDF of correspondences with respect to L2 pixel error. For all other σ, MMGSD degrades due to intermixing of modes caused by higher variance in the ground truth target distributions. We also visualize the average RMSE in 1D and 2D space for rope and cloth, respectively, noting that MMGSD exhibits the highest error at object interiors due to modal collapse and relatively low RMSE at object extremities. This behavior of MMGSD is also suggested by the bimodal nature of the PDF with low error at object exteriors (first peak) and higher error at object interiors (second peak).

Chapter 5

Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research

5.1 Introduction

This chapter is based on [29], a case study in which we evaluate a series of fabric smoothing and folding algorithms on Reach [80], a prototype hardware testbed from Robotics at Google. This case study not only provides benchmarking between several fabric smoothing and folding algorithms, but also serves as a standard for reproducibility in robotics research where benchmarking is difficult due to interaction with physical environments. This work contributes: (1) four novel learning-based algorithms for the folding task, (2) implementation of and comparison with four additional benchmarks, and (3) a case study of robotics research performed exclusively using a remotely managed robot workcell. This work does *not* contribute the design of the Reach cloud robotics platform, which is being developed by a larger team at Google [80].

5.2 The Google Reach Testbed

In this section, we review the most salient details of the Google Cloud Robotics testbed [80] as it relates to this case study.

Hardware

See Figure 5.1 for an image of the workcell. The robot is a single Universal Robot UR5e arm equipped with a Piab piSOFTGRIP vacuum-driven soft 3-jaw gripper [55]. The workcell is equipped with 4 Intel Realsense D415 cameras which each capture 640 × 360 RGB images at 20 FPS and 640 × 360 depth images at 1 FPS. The worksurface is a bright pink silicone

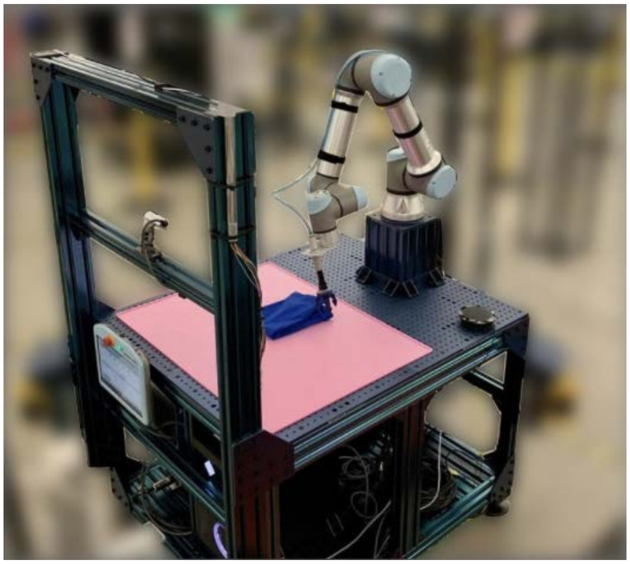


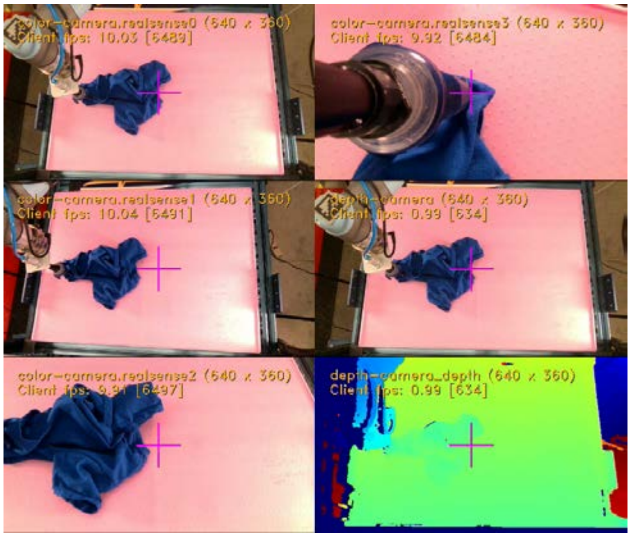
Figure 5.1: A Reach cloud robotics workcell developed by Robotics and Google.

mat and the garment is a blue crew-neck short sleeve T-shirt. The workcell is maintained by lab technicians who are onsite 8 hours a day to reset the robot and troubleshoot.

Software

Reach includes PyReach, an open source Python library developed by Robotics at Google for interfacing with the Reach system. The software includes infrastructure for authenticated users to establish a network connection with the robot server over the Internet, a viewer tool for locally displaying the 4 workcell camera feeds in real time (Figure 5.2), a simulated workcell that mimics the real workcell for safely testing motions prior to deployment on the real system, and utility functions such as a pixel-to-world transform using the depth camera and conversions between different pose representations.

PyReach also includes PyReach Gym, an application programming interface (API) modeled after OpenAI Gym [6]. Remote agents receive observations of the environment and request actions through this interface. In particular, at each time step with frequency up to 10 Hz, a remote agent can receive the joint angles and Cartesian pose of the arm, the binary state of the gripper (closed or open), and camera observations. The agent specifies an action to execute as a desired pose of the arm in either joint or Cartesian space and a desired binary state of the gripper.



```
color-camera.realsense0 (640 x 360)
Client fps: 10.03 [6489]
color-camera.realsense3 (640 x 360)
Client fps: 9.92 [6484]
color-camera.realsense1 (640 x 360)
Client fps: 10.04 [6491]
depth-camera (640 x 360)
Client fps: 0.99 [634]
color-camera.realsense2 (640 x 360)
Client fps: 9.91 [6497]
depth-camera_depth (640 x 360)
Client fps: 0.99 [634]
```

Figure 5.2: The client PyReach viewer, which updates the RGB images from the workcell cameras at about 10 Hz and depth images at 1 Hz. Our algorithms use the overhead RGB images (top left panel).

Garment Folding Case Study: Problem Definition

We assume that the target folded configuration is known beforehand, that training and evaluation are performed in real (not simulation), that the hardware setup is as specified in Section 5.2, and that the garment stays the same during training and evaluation. The task is to iteratively execute two procedures in a loop: (1) crumple the T-shirt and (2) fold the T-shirt. Crumpling is performed via a series of 6 random drops of the T-shirt resulting in an average of 37.5% coverage (Section 5.4), where coverage is the fraction of the maximum 2D area the T-shirt is able to attain. The folding task is to manipulate the T-shirt toward the target configuration in Figure 5.3. We decompose the folding task into two subtasks: (1) *flattening*, i.e., spreading out from an initially crumpled configuration until the garment is smooth, followed by (2) *folding*, i.e., folding the t-shirt from initially flattened until sufficiently close to the target configuration. We measure folding accuracy with a combination of Intersection over Union (IOU) and wrinkle detection (Section 5.4).

5.3 Garment Folding Algorithms

Due to the unique challenges of the flattening and folding subtasks, we benchmark each subtask with its own set of algorithms. Hyperparameter and implementation details for all algorithms are available in the appendix, and notation for this section is defined in Table 5.1. With the exception of Section 5.3, all actions are quasistatic pick-and-place actions from pick

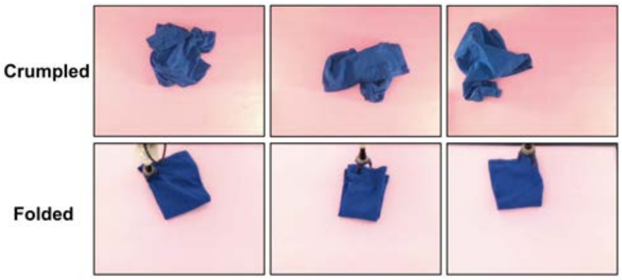


Figure 5.3: Examples of crumpled states (Row 1) and folded states (Row 2).

$\mathbf{o}_t$	Cropped RGB observation of the workcell state from the overhead Realsense camera at timestep t (Figure 5.2).
$\mathbf{a}_t$	The action at time t, expressed as a pick-and-place action (p_0, p_1) in pixel coordinates except in Section 5.3.
$\mathbf{o}_t^m$	Color-thresholded mask of the T-shirt analytically computed from $\mathbf{o}_t$.
$\text{com}(\mathbf{o}_t^m)$	A function that returns the *visual* center of the T-shirt.
$\text{cover}(\mathbf{o}_t^m)$	A function that computes the 2D fabric coverage.
$\mathcal{T}$	A template image of a fully flattened shirt in the workspace.

Table 5.1: Notation for Section 5.3.

point p_0 to place point p_1, where p_0 and p_1 are specified as (x, y) coordinates in pixel space; see Section 5.4 for implementation details.

Flattening: 4 New Algorithms

Learned Pick-Analytic Place (LP$_0$AP$_1$)

Inspired by prior work in imitation learning for fabric manipulation [67, 28], we develop an algorithm to learn pick points from human demonstrations. Since we empirically observe that human-selected pick points combined with analytic placing performs well on flattening, we propose only learning the pick points p_0 and analytically computing place points with

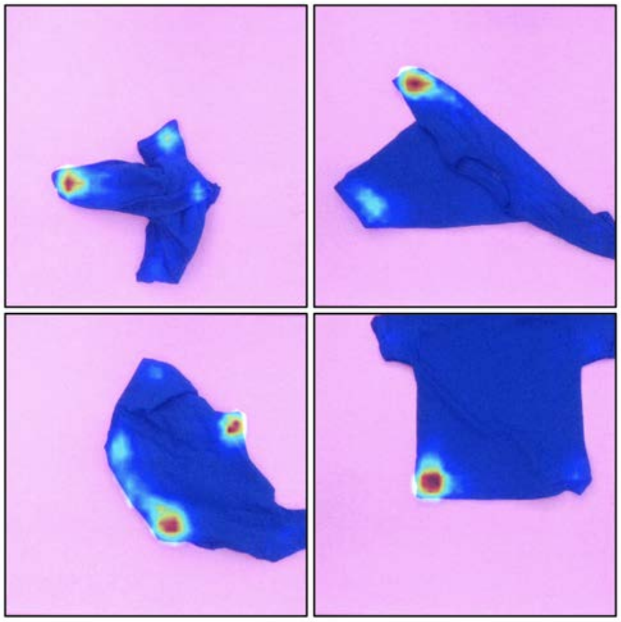


Figure 5.4: LP_0AP_1 pick point predictions on the test set. Bright red and yellow regions correspond to high probability pick points. The output heatmap is able to capture the multimodality in human actions.

the strategy in Section 5.3 to improve sample efficiency. While other work has considered learning a pick-conditioned place policy for fabric manipulation [82], we define analytic placing actions that make the pick-conditioned policy unnecessary. To handle the inherent multimodality in the human policy, we train a fully convolutional network (FCN) [69] to output heatmaps corresponding to probability density instead of regressing to individual actions (Figure 5.4). The FCN can be interpreted as an implicit energy-based model [18, 88] where the state and action pairs are the receptive fields of the network. As in DAgger [61], we reduce distribution shift by iteratively adding on-policy action labels to the dataset.

Learning Keypoints (KP)

This approach separates perception from planning and proposes to only learn the perception component. Specifically, we collect a hand-labeled dataset of images with up to 5 visible keypoints on the fabric corresponding to the collar, 2 sleeves, and 2 base corners (Figure 5.5). While the dataset generation policy is open-ended for this approach, we choose to first train an initial KP policy on random data (Section 5.3) and then augment the dataset with states encountered under the policy to mitigate distribution mismatch similar to DAgger [61]. We train a FCN with 3 output heatmaps to predict each of the 3 classes of keypoints separately. Using keypoint predictions, we propose an analytic corner-pulling policy inspired by [67] that

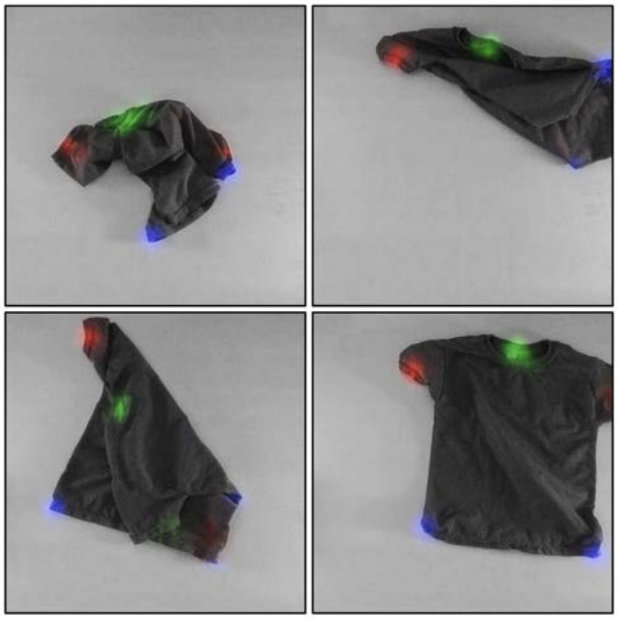


Figure 5.5: KP predictions on the test set. The predicted collar is colored green, the two sleeves are red, and the two base points are blue. Shirt images are shown in grayscale for viewing convenience.

iteratively moves the keypoints from their current positions to their respective locations on a template flattened shirt $\mathcal{T}$. To reduce ambiguity, we compute the rotation and translation of $\mathcal{T}$ that best matches the current state and first move the keypoint farthest from its target location to its destination. To our knowledge, the combination of the FCN for multi-class keypoint prediction, T-shirt template matching, and corner pulling is a novel flattening policy.

Coverage Reward Learning (CRL)

This approach seeks to learn a reward function corresponding to fabric coverage cover$(\cdot)$ from data and execute a policy using this reward function. We learn this reward with self-supervised learning and execute a greedy policy that seeks to maximize the 1-step reward at each time step. Specifically, we fit a Convolutional Neural Network (CNN) $R_\theta(\mathbf{o}_t, \mathbf{a}_t)$ to the scalar change in coverage (i.e., $\text{cover}(\mathbf{o}^m_{t+1}) - \text{cover}(\mathbf{o}^m_t)$) that results from executing action $\mathbf{a}_t$ on $\mathbf{o}_t$. At execution time we randomly sample thousands of pick points on the fabric mask $\mathbf{o}^m_t$ and place points in the workspace and select the action with the highest predicted change in coverage. To our knowledge, greedy planning over a learned model of coverage dynamics for fabric flattening is novel. Once again, the dataset generation policy is a design choice; here, we opt for a random action policy (Section 5.3) to enable large-scale self-supervised data collection and increase data diversity.

Drop (DROP)

Inspired by Ha et al. [25], we investigate whether dynamic motions can leverage aerodynamic effects to accelerate the flattening of the shirt when combined with Approach 5.3. We propose a simple vertical drop primitive that grabs the visual center of mass $\mathrm{com}(\mathbf{o}_t^m)$, lifts the shirt into the air, and releases. We profile the coverage dynamics of the drop and the $\mathrm{LP}_0\mathrm{AP}_1$ pick-and-place and run Q-value iteration to determine which primitive to execute (i.e., drop or pick-and-place) given a discretized version of the current coverage $\mathrm{cover}(\mathbf{o}_t^m)$. Q-value iteration on the following reward function produces a policy that minimizes the total number of actions required to flatten the shirt:

$$r(s = \mathrm{cover}(\cdot), a) = \begin{cases} -1 & \mathrm{cover}(\cdot) < C \\ 0 & \mathrm{cover}(\cdot) \geq C \end{cases}$$

where C is a coverage threshold defined in Section 5.4 and $\mathrm{cover}(\cdot)$ is the discretized current coverage.

Flattening: 4 Baselines

Random (RAND)

As a simple baseline, we implement a random pick-and-place policy that selects p_0 uniformly at random from $\mathbf{o}_t^m$ and p_1 uniformly at random in the workspace within a maximum distance from p_0.

Human Teleoperation (HUMAN)

As an upper bound on performance and action efficiency, a human selects pick and place points through a point-and-click interface (see the appendix for details).

Analytic Edge-Pull (AEP)

We implement a fully analytic policy to explore to what extent learning is required for the T-shirt flattening task. The policy seeks to flatten the shirt by picking the edges and corners and pulling outwards. Formally, we sample p_0 uniformly from the set of points in the shirt mask $\mathbf{o}_t^m$ that are within a distance k from the perimeter of $\mathbf{o}_t^m$, where k is a hyperparameter. Given p_0, we compute p_1 by pulling a fixed distance l in the direction of the average of two unit vectors: (1) away from the visual center of mass $\mathrm{com}(\mathbf{o}_t^m)$ and (2) toward the nearest pixel outside $\mathbf{o}_t^m$.

Learning an Inverse Dynamics Model (IDYN)

A inverse dynamics model $f(\mathbf{o}_t, \mathbf{o}_{t+1})$ produces the action $\mathbf{a}_t$ that causes the input transition from $\mathbf{o}_t$ to $\mathbf{o}_{t+1}$. Here we implement the algorithm proposed by Nair et al. [51], which learns to model visual inverse dynamics. Specifically, we approximate the dynamics with a Siamese CNN $f_\theta(\cdot, \cdot)$ trained on the random action dataset collected in Section 5.3. As in [51], the network factors the action by predicting the pick point p_0 before the pick-conditioned place point p_1 to improve sample efficiency. During policy evaluation, the inputs to the network are the current observation $\mathbf{o}_t$ and the template goal observation $\mathcal{T}$.

Folding Algorithms

Human Teleoperation (HUMAN)

As an upper bound on performance, a human chooses pick and place points for folding through a point-and-click interface.

Analytic Shape-Matching (ASM)

Since the folding subtask is significantly more well-defined than flattening, we investigate whether an open-loop policy computed via shape matching can successfully fold the shirt. We specify a fixed sequence of folding actions with a single human demonstration. During evaluation, we compute rotations and translations of the corresponding template images to find the best match with $\mathbf{o}_t$ and transform the folding actions in the demonstration accordingly.

Learned Pick-Learned Place (LP$_0$LP$_1$)

This approach is identical to Section 5.3 but learns both pick points and place points, as the analytically computed place point is designed for flattening. Since folding demonstrations are difficult to obtain (the garment must be flattened first) and successful folding episodes are short-horizon and visually similar, we collect only two demonstrations and augment the data by a factor of 20 with affine transforms that encourage rotational and translational invariance.

Fully Autonomous Flattening with Analytic Shape-Matching (A-ASM)

The algorithms above are evaluated after the garment is fully flattened via human teleoperation to study the folding subtask in isolation. This approach combines the best performing autonomous flattening algorithm (i.e., LP$_0$AP$_1$) with ASM (Section 5.3) to evaluate the performance of a fully autonomous pipeline for manipulating the garment from crumpled to folded.

5.4 Experiments

Experimental Setup

All actions executed on the robot are either a pick-and-place primitive (p_0, p_1) or a drop primitive (for the DROP algorithm). During flattening, the pick-and-place primitive is a composition of 9 calls to the PyReach Gym API (Section 5.2) that moves the gripper (oriented top-down) to p_0, lowers the gripper to grab the top layer of the fabric at p_0 (computed via a pixel-to-world transform using the depth camera), lifts the gripper, translates to p_1, and releases. During folding, the pick-and-place primitive moves the gripper more slowly; performs a deeper, possibly multi-layer pick; and lowers the gripper at p_1 instead of letting go in the air. The drop primitive uses a similar vertical pick to grab the shirt at $\text{com}(\mathbf{o}_t^m)$, raise it to a predefined height over the center of the workspace, and let go. After executing an action, the arm is commanded to clear the field of view of the camera to prevent occlusion in the image observation. See the appendix or code for exact implementation details. During data collection, actions are chosen either autonomously (e.g., with RAND in Section 5.3) or by a human via a point-and-click graphical user interface. At execution time, actions are specified by trained model outputs.

To improve the performance of the deployed flattening algorithms, we include two additional primitives: (1) a recentering primitive for when the shirt has drifted too far from the center of the workspace, and (2) a recovery primitive that executes a random action when the coverage is stalled for an extended period of time.

Flattening Metrics

We perform 10 trials of all flattening algorithms from an initially crumpled state (Figure 5.3). Crumpling is performed autonomously via a series of 6 actions, each of which grabs the T-shirt at a random point, quickly lifts it into the air, and releases, resulting in an initial coverage of 37.5% ± 14.9% over 45 trials. In Table 5.2 we report maximum coverage as a percentage of the pixel coverage of a fully flattened shirt, i.e. 47,000 pixels in the shirt mask $\mathbf{o}_t^m$. We also report the number of samples used to train the algorithm, the execution time per action, and the number of actions executed, where we allow a maximum of 100 actions but terminate early if a coverage threshold is reached ($C =$ 45,000 pixels or 96% of maximally flattened).

Folding Metrics

We perform 5 trials of all folding algorithms from an initially flattened state. A-ASM initial states are flattened by LP_0AP_1 while all other initial states are flattened via human teleoperation. In Table 5.3 we report the number of actions and execution time per action, and we measure the quality of the final state against a goal configuration (Figure 5.3) according to two metrics: (1) intersection over union (IoU) and (2) a penalty for edges and wrinkles.

IoU is calculated between the shirt mask and the goal template, after rotating and translating the goal to best match the shirt mask. The wrinkle penalty calculates the fraction of pixels in the interior of the shirt mask detected as edges by the Canny edge detector [8]. A high-quality folding episode achieves a high IoU score and low edge penalty; for reference, the scores for a fully folded goal image are provided in Table 5.3 as GOAL.

Flattening Results

See Table 5.2 and Figure 5.6 for results. We find that fully analytical policies such as RAND and AEP are unable to attain high coverage while HUMAN is able to consistently flatten the garment in 11.9 actions on average, suggesting the efficacy of the pick-and-place action primitive and the value of intelligently selecting pick points. Interestingly, we find that despite training an inverse dynamics model on nearly 4,000 real samples, IDYN is unable to outperform RAND. We hypothesize that the fully flattened goal image $\mathcal{T}$ provided as input is too distant from the encountered states, resulting in a data sample outside the training data distribution. While a more fine-grained sequence of subgoal images can mitigate this, such a sequence is not well-defined for flattening, suggesting IDYN is not well-suited for flattening without significant modifications.

CRL is better able to leverage the large self-supervised dataset as it attains higher coverage, though it does require more time per action due to executing thousands of forward passes through the network during planning. However, since the dataset is generated by RAND, which achieves an average maximum coverage of only 55.0%, CRL has trouble producing high-quality actions in the high coverage regime, where it has encountered relatively little data. Modifications to the dataset such as including demonstration data or actively interleaving data collection and training with policy execution could lead to further improvements and is an interesting direction for future work. KP is also able to improve upon RAND but struggles with achieving high coverage, despite having access to hand-labeled data relatively within the distribution of encountered states. While KP achieves a higher maximum coverage than AEP and RAND, it is prone to executing regressive actions that prevent it from maintaining this coverage. Results suggest that KP can be improved by (1) autonomous labeling, e.g. with fiducial markers, to avoid human error on challenging garment states with high self-occlusion, and (2) improvements to the analytic corner-pulling policy, which, for example, can struggle when all visible keypoints are positioned correctly but others are layered underneath the garment.

We find that LP_0AP_1 significantly outperforms all other algorithms, rivaling HUMAN-level performance by consistently reaching the threshold coverage C in less than 3 times the amount of actions as HUMAN. We hypothesize that this is due to increased sample efficiency from analytic placing in conjunction with the modeling power of the FCN, which exhibits equivariance by sharing parameters for pixel predictions and is an implicit energy-based model like other state-of-the-art architectures [18, 88].

Finally, we find that DROP, which converges through Q-iteration to a policy that executes a drop if coverage is below 45% and LP_0AP_1 otherwise, is unable to improve upon LP_0AP_1.

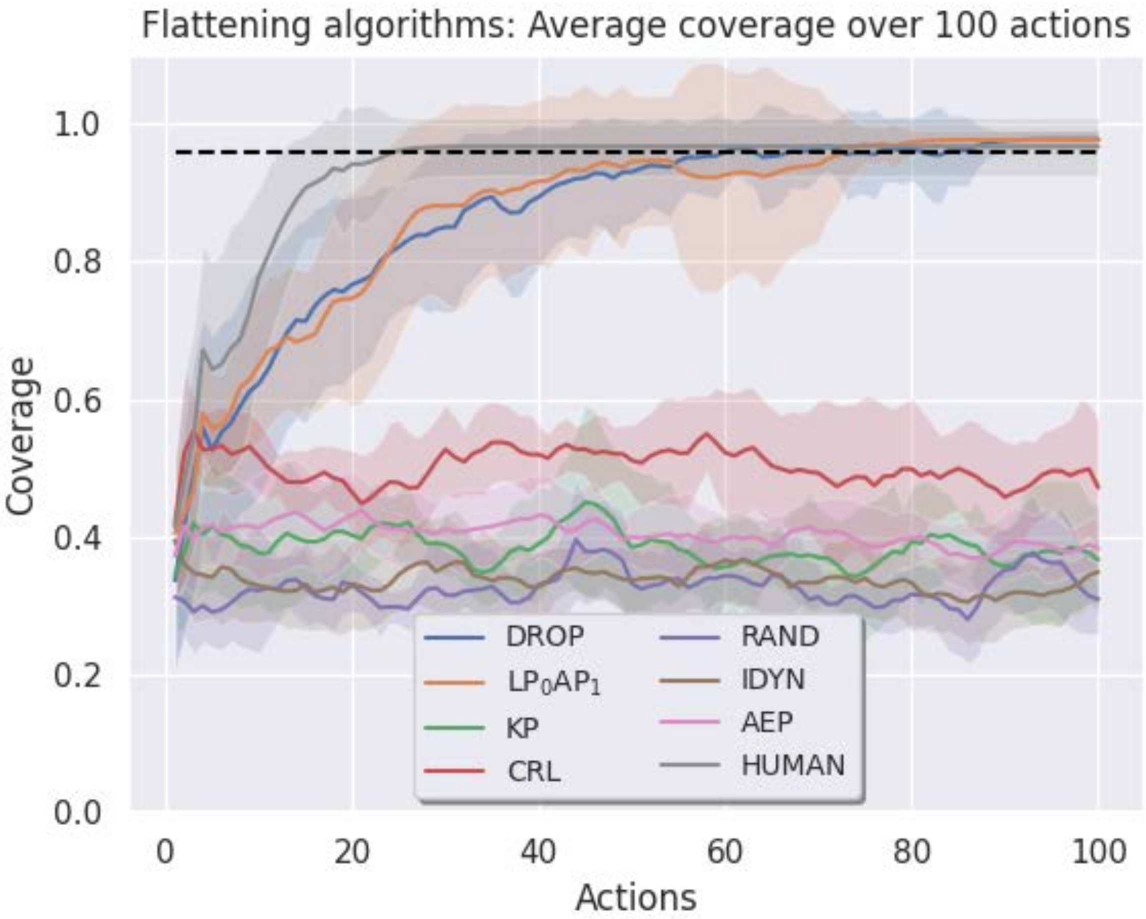


Figure 5.6: Coverage vs. time plot for the various flattening policies that we benchmark on the workcell, averaged across 10 rollouts. Shading represents one standard deviation, and the horizontal dashed line is the flattening success threshold (96%).

This may occur due to our modeling of the coverage dynamics of LP_0AP_1 as the same regardless of the current coverage, whereas in reality, LP_0AP_1 improves coverage faster in lower-coverage states (Figure 5.6). Nevertheless, the DROP framework may be an effective way to combine multiple action primitives given more powerful dynamic primitives, such as bimanual actions that can better leverage aerodynamic effects [25].

Folding Results

See Table 5.3 for results and Figure 5.7 for folding episodes. The folding subtask presents unique challenges: (1) data collection and evaluation require an initially flattened state,

Table 5.2: Flattening results. We report maximum coverage, number of actions, number of samples in the dataset, and evaluation time, where averages and standard deviations are computed over 10 trials.

Algorithm	**% Coverage**	**Actions**	**Dataset**	**Time/Act (s)**
RAND	55.0 ± 6.0	100.0 ± 0.0	N/A	23.9 ± 2.5
HUMAN	**97.7 ± 3.9**	11.9 ± 5.3	N/A	45.1 ± 18.6
AEP	55.3 ± 5.5	100.0 ± 0.0	N/A	24.6 ± 2.0
IDYN	57.0 ± 5.9	100.0 ± 0.0	3936	23.7 ± 3.7
KP	72.4 ± 9.2	100.0 ± 0.0	681	25.7 ± 2.7
CRL	73.8 ± 8.4	100.0 ± 0.0	3936	32.1 ± 5.3
DROP	**97.7 ± 1.3**	38.6 ± 20.6	524	25.7 ± 0.8
LP_0AP_1	**97.7 ± 1.4**	31.9 ± 17.2	524	25.6 ± 0.9

Table 5.3: Folding results. We report intersection over union (IoU), wrinkle penalty, number of actions, and evaluation time, where averages and standard deviations are computed over 5 trials.

Algo.	**IoU (↑)**	**Wrinkle (↓)**	**Actions**	**Time/Act (s)**
GOAL	0.98	0.093	N/A	N/A
HUMAN	**0.74 ± 0.06**	0.088 ± 0.023	4.4 ± 0.5	63.8 ± 14.9
ASM	**0.69 ± 0.08**	**0.087 ± 0.038**	4.0 ± 0.0	35.1 ± 1.9
LP_0LP_1	0.68 ± 0.08	0.112 ± 0.032	4.0 ± 0.0	35.7 ± 1.3
A-ASM	0.62 ± 0.12	0.112 ± 0.038	4.0 ± 0.0	35.5 ± 1.7

which is difficult to attain through a remote interface, (2) slightly incorrect actions can dramatically alter the fabric state, often requiring re-flattening the garment, and (3) the single-arm pick-and-place primitive is not well-suited for the precise manipulation required for crisp garment folding. Indeed, we find that even with folding optimizations to the pick-and-place (Section 5.4), a human teleoperator attains only 76% of the goal IoU on average (Table 5.3). However, we find that both ASM and LP_0LP_1 are able to effectively leverage the primitive to achieve near human-level performance, where ASM performs similarly to LP_0LP_1. We also find that the fully autonomous pipeline A-ASM is able to reach similar performance from an initially *crumpled* state, setting a baseline score for the end-to-end folding task. Although ASM is open-loop and LP_0LP_1 learns from only 2 demonstrations, HUMAN cannot significantly outperform them due to the difficulty of correcting inaccurate actions in folding. Further progress on the folding subtask will likely benefit more from the design of manipulation primitives than from algorithmic innovations.

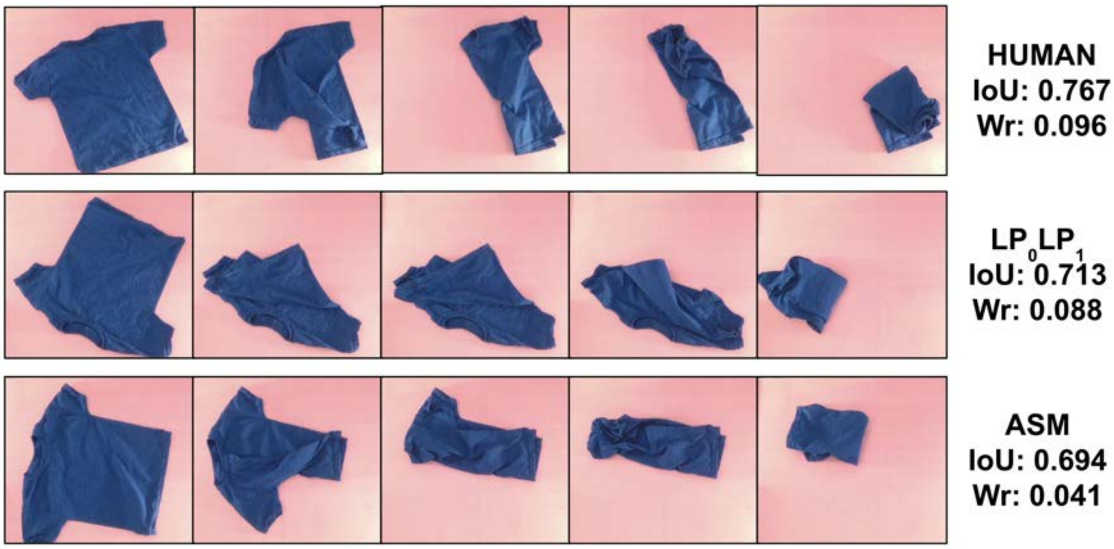


Figure 5.7: Representative episodes of the folding subtask executed by HUMAN (Row 1), $\mathrm{LP}_0\mathrm{LP}_1$ (Row 2), and ASM (Row 3). $\mathrm{LP}_0\mathrm{LP}_1$ and ASM achieve performance competitive with human teleoperation.

Chapter 6

Conclusions and Future Work

In Chapter 3, we present an approach for multi-task fabric manipulation by learning dense visual correspondences entirely in simulation. Experiments suggest that the learned correspondences are robust to different fabric colors, shapes, textures, and sizes and make it possible to efficiently learn 10 different fabric smoothing and folding tasks on two different physical robotic systems with no training in the real world. In future work, we plan to explore hierarchical fabric manipulation policies, where visual correspondences can be used to define coarse action plans while a closed loop controller can be learned to realize these plans. We will also explore more complex fabric manipulation tasks, such as wrapping rigid objects, in which reasoning about fabric dynamics is critical. Finally, we will also explore the use of a new inverted tweezer gripper that is more reliable for grasping fabric and addresses the common Type A error that occurs in this work.

Chapter 4 proposes an extension of dense descriptor-based pixel-wise correspondence that addresses symmetry and uncertainty estimation in deformable object tracking. In future work, we will explore generalizing MMGSD to other types of objects with task-relevant multimodal properties such as sleeves, buttons, drawstrings, or pockets on clothing. We hypothesize that the uncertainty of MMGSD — in object interiors or in occluded parts — would pose a challenge to tasks that involve manipulating all parts of a deformable object, such as untangling a knotted rope or performing consecutive folds on a fabric. However, we will further investigate the limitations of MMGSD and ways to utilize these measures of uncertainty while planning, such as by taking actions to disambiguate a deformable object's state. The framework presented is also applicable to rigid objects containing an axis of symmetry or multimodal properties. Additionally, we will explore learning the dynamics of these correspondences conditioned on robot action sequences. We will also explore 3D representations of deformable objects using geodesic distance as a measure of correspondence.

Finally, Chapter 6 discusses work on benchmarking novel and existing algorithms for T-shirt smoothing and folding tasks. We find that policies that combine learning with analytical methods achieve the highest performance in practice, suggesting the value of future work in this area.

Remote robotics research poses both opportunities and challenges. On the one hand, the ability to access a robot from anywhere, the abstraction of robot operations behind an intuitive API, the setup and maintenance by dedicated staff, and the consistency of the task environment all contributed to quick and convenient experimentation. On the other hand, onsite technicians have limited availability, variable-latency 2D camera projections are at times insufficient for fully understanding the scene, and manual resets (e.g., flattening the T-shirt) become difficult to perform, suggesting the importance of learning self-supervised reset policies [24] or integrating flattening more closely with folding.

In future work, we will (1) further optimize performance on the unimanual folding task, (2) evaluate alternative approaches such as continuous control, reinforcement learning, and different action primitives, and (3) evaluate each algorithm's ability to generalize to other garments with variation in color, shape, size, and material.

Bibliography

[1] Michael Ahn et al. “ROBEL: Robotics Benchmarks for Learning with Low-Cost Robots”. In: *Conf. on Robot Learning (CoRL)*. 2019.

[2] Arthur Allshire et al. “Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger”. In: *arXiv preprint arXiv:2108.09779* (2021).

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

[4] Stefan Bauer et al. “A Robot Cluster for Reproducible Research in Dexterous Manipulation”. In: *arXiv preprint arXiv:2109.10957* (2021).

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

[6] Greg Brockman et al. “OpenAI Gym”. In: *arXiv preprint arXiv:1606.01540* (2016).

[7] Berk Calli et al. “Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set”. In: *IEEE Robotics & Automation Magazine* 22.3 (Sept. 2015), pp. 36–52. ISSN: 1070-9932. DOI: 10.1109/mra.2015.2448951. URL: http://dx.doi.org/10.1109/MRA.2015.2448951.

[8] John Canny. “A computational approach to edge detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1986), pp. 679–698.

[9] Claire Chen et al. “Dexterous Manipulation Primitives for the Real Robot Challenge”. In: *ArXiv preprint arXiv:2101.11597* (2021).

[10] John James Collins, David Howard, and J. Leitner. “Quantifying the Reality Gap in Robotic Manipulation Tasks”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 6706–6712.

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

[12] Sudeep Dasari et al. “RoboNet: Large-Scale Multi-Robot Learning”. In: *Conf. on Robot Learning (CoRL)*. 2019.

[13] Andreas Doumanoglou et al. “Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014.

[14] Frederik Ebert et al. “Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control”. In: *arXiv:1812.00568* (2018).

[15] Peter Florence, Lucas Manuelli, and Russ Tedrake. “Self-Supervised Correspondence in Visuomotor Policy Learning”. In: *IEEE Robotics & Automation Letters*. 2020.

[16] Peter R Florence. PhD thesis. Massachusetts Institute of Technology, 2020. URL: `http://groups.csail.mit.edu/robotics-center/public_papers/Florence19.pdf`.

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

[18] Peter R. Florence et al. “Implicit Behavioral Cloning”. In: *Conf. on Robot Learning (CoRL)*. 2021.

[19] Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2018.

[20] Niklas Funk et al. “Benchmarking Structured Policies and Policy Optimization for Real-World Dexterous Object Manipulation”. In: *IEEE Robotics and Automation Letters* 7.1 (Jan. 2022), pp. 478–485. ISSN: 2377-3774. DOI: `10.1109/lra.2021.3129139`. URL: `http://dx.doi.org/10.1109/LRA.2021.3129139`.

[21] Aditya Ganapathi et al. “Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images”. In: *arXiv preprint arXiv:2003.12698* (2020).

[22] Aditya Ganapathi et al. *MMGSD: Multi-Modal Gaussian Shape Descriptors for Correspondence Matching in 1D and 2D Deformable Objects*. 2020. DOI: `10.48550/ARXIV.2010.04339`. URL: `https://arxiv.org/abs/2010.04339`.

[23] Jennifer Grannen* et al. “Learning Robot Policies for Untangling Dense Knots in Linear Deformable Structures”. In: 2020.

[24] Abhishek Gupta et al. “Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention”. In: *arXiv preprint arXiv:2104.11203* (2021).

[25] Huy Ha and Shuran Song. “FlingBot: The Unreasonable Effectiveness of Dynamic Manipulation for Cloth Unfolding”. In: *Conf. on Robot Learning (CoRL)*. 2021.

[26] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2018.

[27] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2018.

[28] Ryan Hoque et al. “LazyDAgger: Reducing context switching in interactive imitation learning”. In: *International Conference on Automation Sciences and Engineering (CASE)*. 2021.

[29] Ryan Hoque et al. *Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research*. 2022. DOI: `10.48550/ARXIV.2204.10297`. URL: `https://arxiv.org/abs/2204.10297`.

[30] Ryan Hoque et al. “VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation”. In: *Proc. Robotics: Science and Systems (RSS)*. 2020.

[31] Rishabh Jangir, Guillem Alenya, and Carme Torras. “Dynamic Cloth Manipulation with Deep Reinforcement Learning”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020.

[32] Biao Jia et al. “Cloth Manipulation Using Random-Forest-Based Imitation Learning”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019.

[33] Biao Jia et al. “Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018.

[34] Dmitry Kalashnikov et al. “QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation”. In: (June 2018).

[35] P Kazanzides et al. “An Open-Source Research Kit for the da Vinci Surgical System”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014.

[36] Chung Min Kim et al. “Simulation of Parallel-Jaw Grasping using Incremental Potential Contact Models”. In: *arXiv preprint arXiv:2111.01391* (2021).

[37] Yasuyo Kita et al. “A Method For Handling a Specific Part of Clothing by Dual Arms”. In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2009.

[38] Yasuyo Kita et al. “Clothes State Recognition Using 3D Observed Data”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2009.

[39] Oliver Kroemer, Scott Niekum, and George Konidaris. “A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms”. In: *arXiv:1907.03146* (2019).

[40] Robert Lee et al. “Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience”. In: *Conf. on Robot Learning (CoRL)*. 2020.

[41] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436. DOI: `10.1177/0278364917710318`.

[42] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *arXiv:1405.0312* (2014).

[43] Xingyu Lin et al. “SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation”. In: *ArXiv preprint arXiv:2011.07215* (2020).

[44] Jeffrey Mahler et al. “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”. In: *Proc. Robotics: Science and Systems (RSS).* 2017.

[45] Jeffrey Mahler et al. “Learning Ambidextrous Robot Grasping Policies”. In: *Science Robotics.* 2019.

[46] Jeremy Maitin-Shepard et al. “Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA).* 2010.

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

[48] Robert McCarthy et al. “Solving the Real Robot Challenge using Deep Reinforcement Learning”. In: *arXiv preprint arXiv:2109.15233* (2021).

[49] Stephen Miller et al. “A Geometric Approach to Robotic Laundry Folding”. In: *Int. Journal of Robotics Research (IJRR).* 2012.

[50] Douglas Morrison, Peter Corke, and J. Leitner. “EGAD! An Evolved Grasping Analysis Dataset for Diversity and Reproducibility in Robotic Manipulation”. In: *IEEE Robotics and Automation Letters* 5 (2020), pp. 4368–4375.

[51] Ashvin Nair et al. “Combining self-supervised learning and imitation for vision-based rope manipulation”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), pp. 2146–2153.

[52] 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).

[53] J. K. Parker et al. “Robotic Fabric Handling for Automating Garment Manufacturing”. In: *Journal of Manufacturing Science and Engineering* 105 (1983).

[54] Liam Paull et al. “Duckietown: An open, inexpensive and flexible platform for autonomy education and research”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA).* 2017, pp. 1497–1504. DOI: `10.1109/ICRA.2017.7989179`.

[55] *Piab piSOFTGRIP Gripper.* `https://www.piab.com/en-us/suction-cups-and-soft-grippers/soft-grippers/pisoftgrip-vacuum-driven-soft-gripper-/pisoftgrip-/#overview`. Accessed: 2022-02-21.

[56] Daniel Pickem et al. "The Robotarium: A remotely accessible swarm robotics research testbed". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1699–1706. DOI: `10.1109/ICRA.2017.7989200`.

[57] Lerrel Pinto and Abhinav Gupta. "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2016.

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

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

[60] 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.

[61] 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.

[62] Fereshteh Sadeghi and Sergey Levine. "CAD2RL: Real Single-Image Flight without a Single Real Image". In: *Proc. Robotics: Science and Systems (RSS)*. 2017.

[63] Jose Sanchez et al. "Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey". In: *Int. Journal of Robotics Research (IJRR)*. 2018.

[64] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. "Self-Supervised Visual Descriptor Learning for Dense Correspondence". In: *IEEE Robotics and Automation Letters* 2 (2017), pp. 420–427.

[65] John Schulman et al. "Learning from Demonstrations Through the Use of Non-Rigid Registration". In: *Int. S. Robotics Research (ISRR)*. 2013.

[66] Daniel Seita et al. "Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[67] Daniel Seita et al. "Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[68] Daniel Seita et al. "Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making". In: *Int. S. Robotics Research (ISRR)*. 2019.

[69] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2017), pp. 640–651.

[70] Changyeob Shin et al. "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019.

[71] Mohit Shridhar et al. "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 10737–10746.

[72] Li Sun et al. "A Heuristic-Based Approach for Flattening Wrinkled Clothes". In: *Towards Autonomous Robotic Systems. TAROS 2013. Lecture Notes in Computer Science, vol 8069* (2014).

[73] Li Sun et al. "Accurate Garment Surface Analysis using an Active Stereo Robot Head with Application to Dual-Arm Flattening". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2015.

[74] Priya Sundaresan et al. "Learning Rope Manipulation Policies using Dense Object Descriptors Trained on Synthetic Depth Data". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020.

[75] Brijen Thananjeyan et al. "Multilateral Surgical Pattern Cutting in 2D Orthotropic Gauze with Deep Reinforcement Learning Policies for Tensioning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017.

[76] Brijen Thananjeyan et al. "Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks". In: *IEEE Robotics & Automation Letters* (2020).

[77] Josh Tobin et al. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017.

[78] Eric Torgerson and Fanget Paul. "Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1987.

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

[80] Adrian Wong et al. *PyReach - Python Client SDK for Robot Remote Control*. `https://github.com/google-research/pyreach`. 2022.

[81] Yilin Wu et al. "Learning to Manipulate Deformable Objects without Demonstrations". In: *RSS* (2020).

[82] Yilin Wu et al. "Learning to Manipulate Deformable Objects without Demonstrations". In: *Proc. Robotics: Science and Systems (RSS)*. 2020.

[83] Manuel Wüthrich et al. "TriFinger: An Open-Source Robot for Learning Dexterity". In: *Conf. on Robot Learning (CoRL)*. 2020.

[84] Wilson Yan et al. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *Conf. on Robot Learning (CoRL)*. 2020.

[85] Brian Yang et al. "REPLAB: A Reproducible Low-Cost Arm Benchmark Platform for Robotic Learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019.

[86] Takuma Yoneda et al. "Grasp and Motion Planning for Dexterous Manipulation for the Real Robot Challenge". In: *arXiv preprint arXiv:2101.02842* (2021).

[87] Kevin Zakka et al. "Form2Fit: Learning Shape Priors for Generalizable Assembly from Disassembly". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020.

[88] Andy Zeng et al. "Transporter Networks: Rearranging the Visual World for Robotic Manipulation". In: *Conf. on Robot Learning (CoRL)*. 2020.