Bimanual Cloth Manipulation using Flow

Sujay Bajracharya

CMU-RI-TR-21-27

August 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
David Held, *chair*
Oliver Kroemer
Arpit Agarwal

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics*

Abstract

In this work, we address the problem of goal-directed cloth manipulation, a challenging task due to the deformability of cloth. Our insight is that optical flow, a technique normally used for motion estimation in video, can also provide an effective representation for corresponding cloth poses across observation and goal images. We introduce FabricFlowNet (FFN), a cloth manipulation policy that leverages flow as both an input and as an action representation to improve performance for folding tasks. FabricFlowNet also allows elegantly switching between dual-arm and single-arm actions based on the desired goal.

We show that FabricFlowNet outperforms state-of-the-art model-free and model-based cloth manipulation policies. We also present real-world experiments on a bimanual system, demonstrating effective sim-to-real transfer. In addition, we show that our method generalizes when trained on a single square cloth to other cloth shapes, such as t-shirts and rectangular cloths.

Acknowledgments

First, I would like to thank my advisor, Professor David Held, for giving me this opportunity and so much of his time. I have grown as a researcher thanks to his guidance and insight. Next, I would like to express my gratitude to everyone in RPAD, especially Thomas Weng and Wenxuan Zhou for being my mentors. I am grateful to everyone in our lab for their willingness to help and collaborate. I would also like to particularly thank Yufei Wang for contributing to parts of this project. Next, I thank my thesis committee: Oliver Kroemer and Arpit Agarwal, for offering their time and feedback. Finally, A special thanks to my parents for always supporting and encouraging me. This work was made possible by all of these people and I am forever grateful.

Contents

When this thesis is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Cloth manipulation has a wide range of applications in domestic and industrial settings. However, it has posed a challenge for robot manipulation: compared to rigid objects, fabrics have a higher-dimensional configuration space, can be only partially observable due to self-occlusions in crumpled configurations, and do not transform rigidly when manipulated. Early approaches for cloth manipulation relied on scripted actions; these policies are typically slow and do not generalize to arbitrary cloth goal configurations [3, 13, 28].

Recently, learning-based approaches have been explored for cloth manipulation [18, 30, 35, 36, 46], including model-free reinforcement learning to obtain a policy [22, 43]. For a cloth manipulation policy to be general to many different objectives, it must receive a representation of the current folding objective. A standard approach for representing a goal-conditioned policy is to input an image of the current cloth configuration together with an image of the goal [22, 35].

We will show a number of downsides to such an approach when applied to cloth manipulation. First, the policy must learn to reason about the relationship between the current observation and the goal, while also reasoning about the action needed to obtain that goal. These are both difficult learning problems; requiring the network to reason about them jointly exacerbates the difficulty. Additionally, previous work has used reinforcement learning (RL) to try to learn such a policy [22, 43]; however, a reward function can be a fairly weak supervisory signal, which makes it difficult to learn a complex cloth manipulation policy. Finally, while many desirable folding actions are more easily and accurately manipulated with bimanual actions, previous learning-based methods for goal-conditioned cloth manipulation have been restricted to single-arm policies.

In this thesis, we introduce FabricFlowNet (FFN), a goal-conditioned policy for bimanual cloth manipulation that uses optical flow to improve policy performance (see Fig. 1.1). Optical flow has typically been used for video-related tasks such as object tracking and estimating camera motion. We demonstrate that flow can also be used in the context of policy learning for cloth manipulation; we use an optical flow-type network to estimate the relationship between the current observation and a sub-goal. We use flow in two ways: first, as an input representation to our policy; second, after estimating the pick points for a pick-and-place policy, we query the flow image to determine the place actions. Our method is learned entirely with supervised learning, purely from random actions without any expert demonstrations during training and without reinforcement learning.

Our learned policy can perform bimanual manipulation and switches easily between dual and single-arm actions, depending on what is most suitable for the desired goal. Our approach significantly outperforms our best efforts to extend recent single-arm cloth manipulation approaches to bimanual manipulation tasks [18, 22]. We present

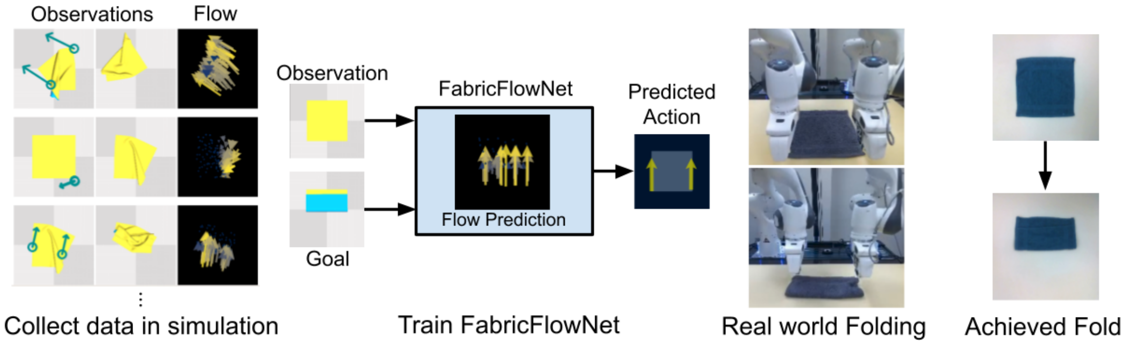


Figure 1.1: FabricFlowNet (FFN) overview. We collect a dataset of random actions and ground truth flow to train FFN. FFN learns to predict flow and uses it as both an input and action representation in a manipulation policy. FFN successfully performs single and dual-arm folding in the real world.

experiments evaluating our method's cloth manipulation performance on a dual-arm robot system and in simulation. FabricFlowNet outperforms state-of-the-art model-based and model-free baselines, and we provide extensive ablation experiments to demonstrate the importance of each component of our method to the achieved performance. Our method also generalizes with no additional training to other cloth shapes, textures, and colors. Our contributions include:

- A novel flow-based approach for learning goal-conditioned cloth manipulation policies that can perform dual-arm and single-arm actions
- A test suite for benchmarking goal-conditioned cloth manipulation algorithms encompassing and expanding on goals used in previous literature [15, 22]
- Experiments showing that FabricFlowNet significantly outperforms baselines and ablations and generalizes to other cloth colors and shapes, even without training on such variations.

Chapter 2

Related Work

There is a large volume of prior work in bimanual manipulation, state representation, policy learning for cloth manipulation, and the use of optical flow for policy learning that is related to this work.

2.1 Bimanual Manipulation

A large body of research exists on dual-arm, or bimanual, manipulation [38]. Dual-arm systems allow for more complex behaviors than single-arm systems at the cost of greater planning complexity [14, 37], leading to research on closed kinematic chain planning [4, 40], composable skill learning [7, 44], and rewarding synergistic behavior [8]. Prior work has also explored bimanual cloth manipulation [33]. Cloth manipulation is a highly underactuated task, and bimanual manipulation enables controlling multiple cloth points [5]. A common approach for cloth flattening is to lift a cloth with one arm and regrasp it with the other arm until it reaches the flattened configuration [3, 10, 13, 24, 28]. Previous work in this direction uses hard-coded policies [3, 13, 28], whereas we learn to achieve arbitrary folded configurations. Tanaka *et al.* [41] learn bimanual actions for goal-conditioned folding, using a voxel-based dynamics model to predict how actions will change the cloth state. We compare to a state-of-the-art method for model-based cloth manipulation [18] and show significantly improved performance. Dynamic bimanual manipulation has also been explored in simulation from ground-truth keypoints [20] and for unfolding cloth in the real world [16]; we perform real-world bimanual folding using depth image observations.

2.2 State Representations for Cloth Manipulation

Prior works have proposed various representations for cloth manipulation, such as parametrized shape models [29] or binary occupancy features [23]. Recent approaches have proposed to learn representations for cloth, such as using contrastive learning to learn a latent vector dynamics model [46]. Other approaches use contrastive learning to learn pixel-wise latent embeddings for cloth [6, 15, 39] which can be used to learn to imitate an expert demonstration [15]; in contrast, our approach doesn't require expert actions, just sub-goal states provided at test-time to define the task. Semantic segmentation has also been used to identify specific regions of cloth for grasping hems and bed making [32, 34]. Goal-conditioned transporter networks are another approach for deformable manipulation [35], where feature image crops are "transported" over a goal image, and place actions are executed where the features crop aligns best with goal image features. We compare our approach to state-of-the-art approaches for cloth manipulation and show significantly improved performance, due to our use of a flow-based representation.

2.3 Policy Learning for Cloth Manipulation.

Various approaches have been applied to policy learning for cloth smoothing, such as reinforcement learning [43] and imitation learning [36]. These approaches only use a single-arm for smoothing, whereas we perform dual-arm goal-conditioned cloth folding. Prior methods for learning goal-conditioned policies have used self-supervised learning to learn an inverse dynamics model for rope [30, 31]. For cloth manipulation, Lee *et al.* [22] learns a model-free value function, whereas other papers learn a cloth dynamics model in pixel-space [18], over a graph of keypoints [27], or using a cloth simulator [25]. We compare our approach to state-of-the-art approaches for cloth manipulation and show significantly improved performance.

2.4 Optical Flow for Policy Learning.

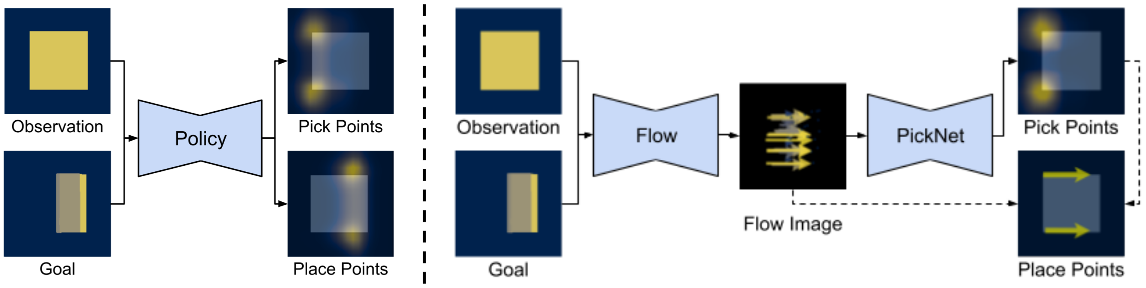
Optical flow is the task of estimating per-pixel correspondences between two images, typically for video-related tasks such as object tracking and motion estimation. State-of-the-art approaches use convolution neural networks (CNN) to estimate flow [12, 19, 42]. Optical flow between successive observations has previously been used as an input representation to capture object motion for peg insertion [11] or dynamic tasks [1]. Within the domain of cloth manipulation, Yamazaki *et al.* [45] similarly use optical flow on successive observations to identify failed actions. We use flow not to represent motion between successive images, but to correspond the cloth pose between observation and goal images, and to determine the placing action for folding. Argus *et al.* [2] use flow in a visual servoing task to compute residual transformations between images from a demonstration trajectory and observed images. In contrast, we learn a policy with flow to determine what cloth folding actions to take, not how to servo to a desired pose.

Chapter 3

Learning a Goal-Conditioned Policy for Bimanual Cloth Manipulation

3.1 Problem Formulation

Our objective is to enable a robot to perform cloth folding manipulation tasks. Let each task be defined by a sequence of sub-goal observations $\mathcal{G} : \{x_1^g, x_2^g, \ldots, x_N^g\}$, each of which can be achieved by a single (possibly bimanual) pick-and-place action from the previous sub-goal. We require sub-goals, rather than a single goal observation, because cloth can be highly self-occluded in a single goal observation (*e.g.,* an observation of a folded cloth) and fail to describe the full goal state. For our folding task, it is important that the intermediate folds are performed correctly, even if they become occluded by future folds. Defining a task using a sequence of sub-goals is



(a) Naive system

(b) FabricFlowNet (FFN)

Figure 3.1: (a) A naive approach to goal-conditioned policy learning is to input observation and goal images directly to the policy and predict the action. (b) FabricFlowNet separates representation learning from policy learning; it first estimates the correspondence between the observation and goal as a flow image. The flow is then used as the input to PickNet for pick point prediction, and as a way to compute place points without requiring additional learning.

found in other recent work [31]. Similar to prior work [30, 31], even if the sub-goals are obtained from an expert demonstration, we nonetheless do not assume access to the expert actions; this is a realistic assumption if the sub-goals are obtained from visual observations of a human demonstrator.

We assume that the agent does not have access to the sub-goal sequence $\mathcal{G}$ during training that it must execute during inference time. Thus, the agent must learn a general goal-conditioned policy $a_t = \pi(x_t, \mathcal{G})$, where x_t is the current observation of the cloth and $a_t \in \mathcal{A}$ is the action selected by the policy. In our approach, we input each sub-goal x_i^g sequentially to our policy: $a_t = \pi(x_t, x_i^g)$. A goal recognizer [31] can also be used to decide which sub-goal observation to input at each timestep. For convenience, we will interchangeably refer to x_i^g as a goal or sub-goal.

3.2 Overview

A common approach for a goal-conditioned policy is to input the current observation x_t and the goal observation x_i^g into to a neural network representation of a policy [22, 31] or a Q-function [22, 43]. However, the network must reason simultaneously about the relationship between the observation and the goal, as well as the correct action to achieve that goal. Our first insight is that we can improve performance by separating these two components; for our task, this refers to the correspondence for parts of the deformable cloth between the observation and goal, and use this to reason over pick and place action required to achieve the goal. Specifically, we represent this relationship using a "flow image" f, which indicates the correspondence between the current observation x_t and sub-goal x_i^g. Thus we propose using the flow image f as an improved input representation of the policy, rather than directly inputting the observation x_t and goal observation x_i^g.

Our second insight is that we can also use flow in the output representation of the policy. We use a pick and place action space; prior methods that learn pick and place policies for deformable object manipulation predict place points using the policy network, either explicitly [30, 31, 35, 36, 43, 46] or implicitly by transforming the inputs to a Q-function [22]. Instead, we simplify the problem by leveraging flow: our policy network only learns to predict the *pick* points. For the place point, we query the flow image f for the flow vector starting at the predicted pick location, and use the endpoint of that vector as the place point.

We demonstrate that using flow in the two ways described above for our policy achieves significantly improved performance compared to prior work. Furthermore, our approach extends naturally to dual-arm manipulation, allowing us to easily transition between single and dual-arm actions.

A schematic overview of our system can be found in Fig. 3.1b. We first compute the flow f between the current observation x_t and goal x_i^g. Next, we input the flow f to a policy network (PickNet), which outputs pick points p_i. We then query the flow image $f(p_i)$ to determine the place points for each robot arm. Further details of our approach are described below.

3.3 Estimating Flow between Observation and Goal Images

We learn flow to use it as an input representation to our pick prediction network, and as an action representation for computing place points. Given an observed depth image x_t and desired goal depth image x_i^g, we estimate the flow $f = (f^1, f^2)$, mapping each pixel (u, v) in x_t to its corresponding coordinates $(u', v') = (u + f^1(u), v + f^2(v))$ in x_i^g. This task formulation differs from standard optical flow tasks as the input image pair (x_t, x_i^g) are not consecutive images drawn from video frames. Pixel displacements from sequences of video frames are often small relative to the image size; in our case, the displacement can be large if there is a significant change in cloth pose from x_t to x_i^g.

To capture the complex correspondences between x_t and x_i^g, we train a convolutional neural network to estimate the flow image f. The training loss we use to supervise the network is endpoint error (EPE), the standard error for optical flow estimation. EPE is the Euclidean distance between the predicted flow vectors f and the ground truth f^*, averaged over all pixels: $\mathcal{L}_{\text{EPE}} = \frac{1}{N}\sum_{i=1}^{N} \|f^* - f\|_2$. We

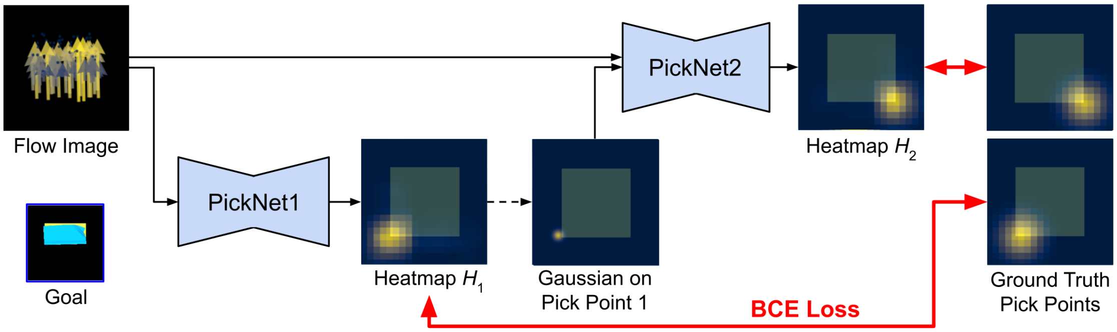


Figure 3.2: PickNet architecture. We utilize a two-network architecture for bimanual manipulation, where the second pick point is conditioned on the prediction of the first pick point.

use a cloth simulation to collect training examples with ground truth flow. The simulator provides the ground-truth correspondence between the particles of the cloth in different poses. The simulation cloth particles are not as dense as the depth image pixels; as a result, we only have ground-truth flow supervision for a sparse subset of the pixels that align with the cloth particles. Thus, we mask the loss to only supervise the flow for the pixels that align with the location of the cloth particles. We train the flow network using data collected from random actions. See Sec. 3.6 for more details on the simulator, data collection, and network training.

3.4 Learning to Predict Pick Points

Our bimanual action space $\mathcal{A}$ consists of actions $a = (p_1, p_2, q_1, q_2)$, where p and q are the pick and place points respectively, paired according to the subscripts. We train a neural network called PickNet to estimate the pick points p_1, p_2. Crucially, the input to PickNet is a flow image f, estimated between the current depth image x_t and the desired goal depth image x_i^g, as described in the previous section. The flow image indicates, for each pixel (u, v) in the current observation, the location

$f(u, v)$ that the pixel has moved to in the goal observation. Previous methods for goal-conditioned policy learning typically input the current image and goal image directly to the policy network [22, 35], requiring the policy to reason about both the observation-goal relationship and how to achieve the goal. In contrast, our flow network (Sec. 3.3 above) reasons about the observation-goal relationship, so that the policy network (PickNet) only needs to reason about the action, specifically the two pick points (p_1, p_2); computing the place points is described in Sec. 3.5.

For dual-arm actions, the pick points must be estimated conditionally, as the location of pick point p_1 on the cloth influences the optimal location of pick point p_2, and vice versa. To decouple this conditional estimation problem, we propose a two-network architecture, PickNet1 and PickNet2, to estimate the pick points (see Fig. 3.2). This architecture was inspired by Wu *et al.* [43], which used two networks for pick-conditioned placing; we instead use two networks to condition dual-arm picking. PickNet1 is a fully convolutional network that receives flow image f as input and outputs a single heatmap H_1 estimating the optimal pick points for arm 1. We compute the first pick point as $p_1 = \arg\max_p H_1(p)$. The second network, PickNet2, predicts the second arm's pick point p_2 conditioned on p_1; PickNet2 takes as input both the flow image f and an additional image with a 2D Gaussian centered on p_1, and is otherwise identical to PickNet1. PickNet2 outputs heatmap H_2, from which we compute the second pick point: $p_2 = \arg\max_p H_2(p)$. The two-network architecture decouples the conditionally dependent pick point predictions and does not require us to resort to heuristics to extract two pick points from a single heatmap. We refer to PickNet1 and PickNet2 together as "PickNet."

To train PickNet, we collect a dataset of random actions (see Sec. 3.6 for details) and record the current observation x_t, the bimanual action $a = (p_1, p_2, q_1, q_2)$, and the next observation x_{t+1}. We also estimate the flow f from x_t to x_{t+1}, learned in Sec. 3.3. We create ground truth pick heatmaps H_i^* for arm i using the recorded random action a, by placing a 2D Gaussian $\mathcal{N}(p_i, \sigma)$ on each ground truth pick location p_i. We then supervise PickNet using the binary cross-entropy (BCE) loss between predicted heatmaps H_1, H_2 and ground truth heatmaps H_1^*, H_2^*. However, there is ambiguity about which pick point should be output by PickNet1 and which should be output by PickNet2. To allow for flexibility, we compute the loss for both possible correspondences and use the minimum:

$$\begin{aligned} l_{\text{BCE}}(H_i, H_j, H_i^*, H_j^*) &= \text{BCE}(H_i, H_i^*) + \text{BCE}(H_j, H_j^*) \\ \mathcal{L}_{\text{Pick}} &= \min[l_{\text{BCE}}(H_1, H_2, H_1^*, H_2^*), l_{\text{BCE}}(H_2, H_1, H_1^*, H_2^*)] \end{aligned} \tag{3.1}$$

At inference time, PickNet outputs the pick points p_1, p_2, computed from the argmax of H_1, H_2 respectively, as described above.

3.5 Estimating the Place Points from Flow

After estimating the pick points p_1, p_2 from flow, the remaining step to predict a bimanual pick and place action $a = (p_1, p_2, q_1, q_2)$ is to estimate the place points q_1, q_2. A straightforward approach would be to train the network to predict place points q_1, q_2, similar to the pick points p_1, p_2 as described above. Instead, our approach uses the flow image to find the place points, so that the place points do not have to be learned separately.

Our approach makes the assumption that, to achieve a desired subgoal configuration, the point picked on the cloth should be moved to its corresponding position in the goal image (which is estimated by the flow). This is a simplifying assumption, since it is possible that the picked point will shift slightly after it is released by the gripper; our method does not take into account such small movements. Using this assumption, to compute the place points q_1, q_2, we query the flow f at each pick point p_1, p_2 to estimate the delta between the pick point location in the observation image and the corresponding location of the pick points in the goal image. We use these predicted correspondences as the place points: $q_i = f(p_i) + p_i$, for each arm i.

It is possible for the action predictions estimated by flow to produce overlapping or near-overlapping pick and place points, indicating that arm 1 and arm 2 should perform identical actions. We observe this behavior from PickNet when the goal is best achieved with a single-arm action, rather than a bimanual one. Therefore, to switch between executing a single-arm or bimanual action, we compute the L2 pixel distance between pick points $d_{\text{pick}} = \|p_1 - p_2\|_2$ and place points $d_{\text{place}} = \|q_1 - q_2\|_2$. We use a single-arm action when either distance is smaller than a threshold α, which we set to 30 for all experiments.

3.6 Implementation Details

3.6.1 Simulator

We use SoftGym [26], an environment for cloth manipulation built on the particle-based simulator Nvidia Flex, to collect training datasets. The simulator models cloth as particles connected by springs. We use pickers that simulate a grasping action by binding to the nearest cloth particle within a threshold to execute pick and place actions in SoftGym. We demonstrate that we are able to train our method in SoftGym and then transfer the policy to the real world in section 4.4.

3.6.2 Data Collection

We collect data in SoftGym by taking random pick and place actions on a simulated 30 cm square cloth. A simulated camera is placed at a height of 65 cm above the cloth. The random actions are biased to pick corners of the cloth mask (detected using Harris corner detection [17]) 45% of the time, and "true" corners of the square cloth 45% of the time. If the true corners are occluded then Harris corners are used instead. For the remaining 10%, the pick actions are uniformly sampled over the visible cloth mask. After the pickers grasp the cloth, they lift to a fixed height of 7.5 cm.

We constrain the place points of the action so that both place points are offset in the same direction and distance from their respective pick points. The direction is orthogonal to the line segment connecting the two pick points, and points towards the center of the image so the cloth does not move out of the frame (similar to Lee *et al.* [22]). The distance between the pick point and the place point along this direction is uniformly sampled between $[25, 100]$ px. The distance is truncated if it exceeds a margin of 20 px from the image edge, again to prevent moving the cloth out of the frame. While these heuristics may seem to overly constrain the data we collect, we observe that our data still contains highly diverse cloth configurations, as shown in Fig. 3.3.

For each sample, we save the initial depth observation image, the dual-arm pick and place pixel locations of the action, the next depth observation resulting from the executed action, and the cloth particle positions of both observations (See Fig. 3.3). The simulated camera for captures top down depth observations from above the support surface. We mask the depth observations to only include the cloth by setting

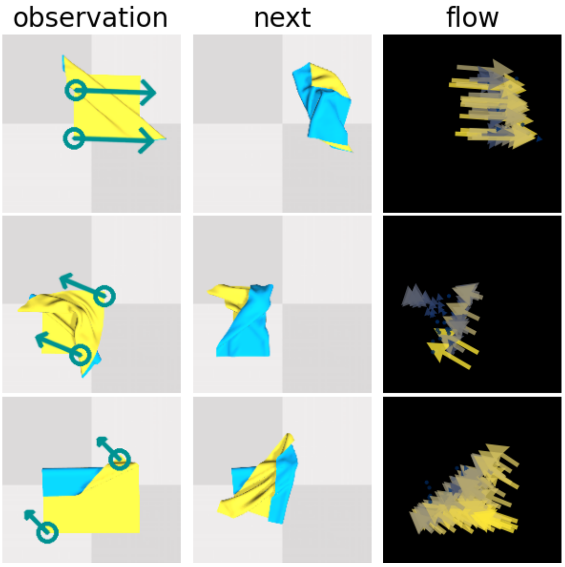


Figure 3.3: Training data for FFN. The figure shows RGB images in simulation for visualizing the example data but FFN utilizes only depth images to compute the flow.

all background pixels to zero. The dataset for training both the flow and pick networks consists of 20k samples from 10k episodes, where each episode consists of two dual-arm pick and place actions.

3.6.3 Flow Network Training.

We use FlowNet [19] as our flow network architecture. The input to FlowNet is the initial and next depth image from a sample in our dataset, concatenated channel-wise. The ground truth flow for supervising FlowNet comes from the cloth particles used by the simulator to model the cloth's dynamics: we collect the cloth particle positions for each observation in our dataset and correspond them across observations to get flow vectors (See Fig. 3.3). The ground truth flow is sparse because the cloth particles are sparse, so we train FlowNet using a masked loss that only includes pixels with corresponding ground truth flow. Similar to Lee *et al.* [22], we apply spatial augmentation of uniform random translation (up to 5 px) and rotation (up to 5 degrees) to augment the training data. We train the network using the dataset of 20k random actions described above. We train for approximately 600 epochs with the Adam [21] optimizer, learning rate 1e-4, weight decay 1e-4, and batch size 8.

3.6.4 PickNet Network Training.

PickNet1 and PickNet2 are fully-convolutional network architectures based on Lee *et al.* [22], with 4 convolutional layers in the encoder, each with 32 filters of size 5. The first three layers of the encoder have stride 2 and the last one has stride 1. The decoder consists of 2 interleaved convolutional layers and bilinear upsampling layers.

The input to the PickNet1 is a 200 × 200 flow image. PickNet2 receives the first pick point location (the argmax of the Picknet1 output, as described in the main text) as an additional input, represented as a 2D Gaussian $\mathcal{N}(p_1, \sigma)$ (where $\sigma = 5$). Similar to Nair *et al.* [30], the output of both networks is a 20 x 20 spatial grid. If the pick points predicted by PickNet are not on the cloth mask, we project them to the closest pixel on the mask using an inverse distance transform. In practice, we find that the predictions are usually either on the cloth mask or very close to the mask. To train PickNet1 and PickNet2, we use the same dataset of 20k random actions described above. We train for 300k steps with the Adam [21] optimizer, learning rate 1e-4, and batch size 10.

Chapter 4

Results

4.1 Folding goals

We evaluate our method on two sets of folding goals: 40 *one-step* goals that can be achieved with a single fold action (see Fig. 4.1a), and 6 *multi-step* goals that require multiple folding actions (see Fig. 4.2a). The multi-step goals each consist of a sequence of sub-goal images, with the next sub-goal presented after each action. This protocol follows from our problem formulation in section 3.1, and is similar to the protocol in Nair *et al.* [30]. Our goals include test goals from Ganapathi *et al.* [15] and Lee *et al.* [22] that are achievable with one arm, as well as additional goals more suitable for two-arm actions.

4.2 Baselines

We compare our method to Fabric-VSF [18], which learns a visual dynamics model and uses CEM to plan using the model. We only use Fabric-VSF with RGB-D input, as depth-only input performs poorly for folding tasks [18]. Note that FabricFlowNet only uses depth and does not rely on RGB, which enables our method to transfer easily to the real world without extensive domain randomization. We compare against single-arm Fabric-VSF as well as a dual-arm variant.

We also compare to Lee *et al.* [22], a model-free approach. We extend the original single-arm method to a dual-arm variant and compare against both. For both our method and the baselines, we only allow each method to perform one pick-and-place action for each subgoal (e.g. one pick and place action for each of the one-step goals).

4.2.1 Fabric-VSF [18] Implementation Details

The original Fabric-VSF [18] paper uses single arm actions and a top-down close camera view such that the cloth covers the whole image. To match the camera view, we set the camera height to be 45 cm above the table in our case. The training dataset consists of 7115 trajectories, each with 15 random pick-and-place actions, totaling 106725 data points. Note that this dataset is 5x larger than the 20k samples we train FFN on. The action size is bounded to roughly half the cloth width. During training, Fabric-VSF takes as input 3 context frames and predicts the next 7 target frames.

We trained 8 variants of Fabric-VSF. Each variant differs in the following aspects: 1) whether it uses single arm or dual arms; 2) during data collection, whether the pick-and-place actions are randomly sampled, or use the corner biasing sampling strategy as described in Sec. 3.6, and 3) whether it uses the original small action size ("Small Action", bounded to half of the cloth width) or a larger action size ("Large Action", bounded to the full cloth width). Other than these three changes, we set all other parameters to be the same as in the original paper. Therefore, the variant with single arm actions, no corner biasing during data collection, and small action size is exactly how Fabric-VSF is trained in the original paper.

After the training, we plan with cross-entropy method (CEM) to find actions for achieving a given goal image. We use the exact same CEM parameters as in the original paper, *i.e.,* we run CEM for 10 iterations, each with a population size of 2000 and elite size of 400.

4.2.2 Lee *et al.* [22] Implementation Details

Lee *et al.* [22] learns a fabric folding policy for a discrete action space using a fully convolutional state-action value function, or Q-network. Observation and goal images are stacked channel-wise, then duplicated and transformed to form a batch of m image rotations and n scales to represent different pick and place directions and action lengths. The whole batch is input to the Q-network to compute the Q-value of executing an action for each rotation and scale at every point on the image. The action corresponding to the max Q-value from the outputs is executed. The discrete action space of m rotations and n action lengths for Lee *et al.* [22] enables efficient policy learning, but greatly limits the actions of the learned policy compared to FFN.

We extend Lee *et al.* [22] from a single-arm approach to a dual-arm one. To represent two pickers instead of one, we input two pairs of observation and goal images to the Q-network. When rotating and scaling the images to represent different actions, the images are constrained to have the same rotation, but are allowed to be scaled differently. In other words, the dual-arm actions are constrained to execute pick and place actions in the same direction, but can have different pick and place lengths. The Q-network outputs a pair (one for each arm) of Q-value heatmaps for every action in the discrete action space (*i.e.,* every rotation and scale). The max Q-value in each of the two heatmaps is averaged, and the heatmap pair with the highest averaged Q-value is selected from the set of all discrete rotations and scales. The picker action corresponding to the argmax of each heatmap is executed.

We train each Lee *et al.* variant below using hyperparameters similar to the original paper [22], training for 25k steps with learning rate 1e-4, batch size 10, and evaluating performance on test goals every 500 steps to find the best performing step.

4.3 Simulation Experiments

4.3.1 Experiment Setup.

We evaluate our method and compare to state-of-the-art baselines in the SoftGym [26] simulator; real-world evaluations are below in Sec. 4.4. In simulation, we have access to the ground truth state of the cloth particles, which is not available in the real world. Our error metric is the average particle position error between the achieved and goal cloth configuration. We report the performance for each method on the full set of goals, including the one-step and multi-step goals (see Fig. 4.1a and Fig. 4.2a).

4.3.2 Results

Table 4.1 contains our simulation results for all methods. We report average particle distance error (in mm) for one-step goals only, multi-step goals only, and over both one-step and multi-step goals. Our results show that our method, FabricFlowNet, outperforms the original Fabric-VSF baseline and both the single-arm and dual-arm versions of Lee *et al.* . Lee *et al.* achieves better performance than Fabric-VSF with original paper settings. More information and additional results for variants of baselines can be found in sections 4.3.3 and 4.3.4.

Table 4.1: Average Particle Distance Error (mm) on Cloth Folding in Simulation

Method	One Step (40)	Multi Step (6)	All (46)
Fabric-VSF orig. [18]	24.958	51.128	38.043
Lee *et al.*, 1-Arm [22]	20.509	27.846	21.802
Lee *et al.*, 2-Arm	39.515	51.974	41.140
FabricFlowNet (Ours)	**7.211**	**22.926**	**9.261**

4.3.3 Fabric-VSF Variants

Additional results for the different Fabric-VSF variants are summarized in Table 4.2. We note that the variant using single arm actions, corner biasing for data collection, and large action size performs the best out of all variants. This variant outperforms FFN on overall error and one-step error, but performs slightly worse than FFN on multi-step error (See Fig. 4.1c and Fig. 4.2c for qualitative results). However, we note that Fabric-VSF was trained on 5x more data than FFN. Additionally, Fabric-VSF takes much longer to run at inference time, requiring ∼7 minutes of CEM iterations to compute a single action compared to ∼0.007 seconds for a forward pass through FFN. 7 minutes of CEM planning time is impractical for real-world folding. We also demonstrate in the following section that FFN generalizes to other cloth shapes better than Fabric-VSF.

Analyzing the performance between different Fabric-VSF variants, we note that adding corner biasing for data collection improves performance in most cases. For single arm actions, using large actions instead of small actions always leads to better performance; however, this is not true for the dual arm variants. Interestingly, we find that using dual arms tends to result in worse performance compared with using a single arm. The reason for this could be that during CEM planning, dual-arm variants double the action dimension, which increases complexity for CEM and makes it difficult to find optimal actions.

Table 4.2: Avg. Particle Distance Error for Fabric-VSF Variants

Baseline	1-Step (40)	Multi Step (6)	All (46)	Inf. Time
Fabric-VSF, 1-Arm, No CB, Sm. Action	12.922	46.051	17.243	∼420s
Fabric-VSF, 1-Arm, No CB, Lg. Action	8.969	41.125	13.163	∼420s
Fabric-VSF, 1-Arm, CB, Sm. Action	14.091	38.676	17.298	∼420s
Fabric-VSF, 1-Arm, CB, Lg. Action	**5.981**	23.713	**8.294**	∼420s
Fabric-VSF, 2-Arm, No CB, Sm. Action	24.595	50.264	27.943	∼420s
Fabric-VSF, 2-Arm, No CB, Lg. Action	81.853	116.275	86.343	∼420s
Fabric-VSF, 2-Arm, CB, Sm. Action	16.205	36.421	18.842	∼420s
Fabric-VSF, 2-Arm, CB, Lg. Action	10.514	32.354	13.363	∼420s
FFN (Ours)	7.211	**22.926**	9.261	∼**0.007s**

CB: Corner Bias Sm. Action: Small Action Lg. Action: Large Action Inf. Time: Inference Time

4.3.4 Lee *et al.* Variants

We trained variants of Lee *et al.* to compare single-arm vs. dual-arm performance, amount of training data (300 samples as in the original paper vs. 8k samples), and whether the original close images of the cloth ("Low Cam") performed better than images of the cloth from further away ("High Cam"). We also provide results for two variants of FFN trained on the same amount of data, one where actions are sampled from the discrete action space (*i.e.,* discretized action angles and lengths) in Lee *et al.* [22] ("Discrete Actions"), and the other where actions are sampled using our continuous action space described in Sec. 3.6 ("Cont. Actions"). Lee *et al.* [22] is an inherently discrete approach and cannot be trained to output continuous actions, nor can it be trained on data with actions outside of its discrete action space.

Table 4.3 shows results for Lee *et al.* variants and FFN trained on 300 training examples, as in the original paper. Note that the values in the first row differ slightly from those reported in the main body of the paper; this discrepancy was due to a bug in the evaluation code and will be corrected in an updated version of the paper. High Cam performs slightly better than Low Cam, and single-arm Lee *et al.* performs better than the dual-arm variant. FFN trained on 300 examples performs similarly to Lee *et al.*, but increasing the amount of training data for both methods yields better performance by FFN. FFN trained on data sampled with our continuous action space performs better than training on data sampled with the discrete action space. Lee *et al.* [22] has a discrete action space and can only be trained with data generated using discrete actions.

Table 4.3: Avg. Particle Distance Error for Lee *et al.* and FFN on 300 Training Examples

Baseline	One Step (40)	Multi Step (6)	All (46)
Lee *et al.*, 1-Arm, 300 Discrete Actions, Low Cam	20.595	27.642	21.514
Lee *et al.*, 1-Arm, 300 Discrete Actions, High Cam	19.088	22.628	19.577
Lee *et al.*, 2-Arm, 300 Discrete Actions, High Cam	39.515	51.974	41.140
FFN, 2-Arm, 300 Discrete Actions, High Cam	19.610	44.971	22.918
FFN, 2-Arm, 300 Cont. Actions, High Cam	14.821	34.833	17.431

Table 4.4 provides results for Lee *et al.* variants and FFN trained on ~8k training

examples. When FFN and Lee *et al.* are both trained on 8k examples, FFN outperforms all Lee *et al.* variants. FFN performs better when trained on continuous action data than with discrete action data, demonstrating that our continuous action data sampling approach performs better than the discrete approach used by Lee *et al.* FFN trained on 8k examples and continuous action data approaches the performance of our best FFN method trained on 20k examples.

Table 4.4: Avg. Particle Distance Error for Lee *et al.* and FFN on 8k Training Examples

Baseline	One Step (40)	Multi Step (6)	All (46)
Lee *et al.*, 1-Arm, 8k Discrete Actions, Low Cam	18.224	27.907	19.487
Lee *et al.*, 2-Arm, 8k Discrete Actions, High Cam	33.839	51.426	36.133
FFN, 2-Arm, 8k Discrete Actions, High Cam	14.424	25.501	15.869
FFN, 2-Arm, 8k Cont. Actions, High Cam	7.650	24.663	9.870
FFN, 2-Arm, 20k Cont. Actions, High Cam (Ours)	7.211	22.926	9.261

4.3.5 Ablations

We run series of ablations to evaluate the importance of the components of our system; results can be found in Table 4.5. Our ablations are designed to answer the following questions:

What is the benefit of using flow as input? We modify PickNet to receive depth images of the observation and goal as input to the network ("DepthIn"), as is commonly done in previous work on goal-conditioned RL [22, 35]. In this ablation, the PickNet needs to reason about the relationship between the observation and the goal as well as about the action. In contrast, in our method the flow network separately reasons about the relationship between the observation and the goal; the PickNet receives the flow image as input and thus only needs to reason about the action. This separation leads to a 12.5% improvement.

What is the benefit of using flow to choose the place point? In this ablation, we train a network to predict the place points directly ("PredictPlace"). This is in contrast to our approach where we use the flow field, evaluated at the pick point $f(p_i)$, to compute the place point q_i for arm i. Our approach leads to improved performance (28.1% improvement), showing the benefit to using flow as an action representation.

No flow: We combine the above two ablations and remove flow entirely, ("NoFlow"; ours has 46.6% improvement). The above ablations all indicate the strong benefit of using flow as both an input and action representation for cloth manipulation.

What is the benefit of biasing the data collection to grasping corners? For our method we utilize prior knowledge about cloth folding tasks and bias the training data to pick at corners of the cloth. In this ablation, we choose pick points randomly with no bias ("NoCornerBias", ours has 28.6% better performance).

Different architecture: We also compare our architecture for PickNet (See Sec. 3.4) to an alternate, simpler architecture that takes as input the flow image I_f and outputs two heatmaps corresponding to each of the pick points ("SinglePickNet"; ours has 3.8% better performance).

Does the loss formulation in Eq. 3.1 improve performance? We compare our method to an ablation where the first ground-truth heatmap is used to supervise PickNet1 and similarly for the second, i.e. $\mathcal{L}_{\text{Pick}} = l_{\text{BCE}}(H_1, H_2, H_1^*, H_2^*)$. ("NoMinLoss"; ours has 1.5% better performance).

Table 4.5: Avg. Particle Distance Error for Ablations

Ablation	One Step (40)	Multi Step (6)	All (46)
DepthIn	8.837	22.295	10.592
PredictPlace	10.949	25.764	12.881
NoFlow	15.873	27.181	17.348
NoCornerBias	11.113	25.365	12.972
SinglePickNet	7.402	24.451	9.627
NoMinLoss	7.348	23.067	9.400
FFN (Ours)	**7.211**	**22.926**	**9.261**

4.3.6 Generalization to different shape cloth

We evaluate FFN on generalization to different shaped cloths, namely a rectangular cloth and t-shirt. For rectangular cloth we test on 5 folding goals including 3 one step goals and 2 multi step goals (see Fig. 4.3a). Similarly, for t-shirt we report performance on 2 one-step goals and 1 multi-step goal (see Fig. 4.4a). We also compare to the best Fabric-VSF variant on generalization to a rectangular cloth. FFN generalizes well to new shapes, as shown in Table 4.6. Fabric-VSF, on the other hand, generalizes poorly, likely because it relies on planning with a learned visual dynamics model. FFN outperforms Fabric-VSF by a large margin. Fig. 4.3 provides a qualitative comparison. Fig. 4.4b shows qualitative results for generalization to t-shirt. We also provide generalization results in the real world in the following section 4.4.

Table 4.6: Avg. Particle Distance error (mm) for different cloth shapes in simulation

Method	Rectangle (n=5)	T-shirt (n=3)
Fabric-VSF, 1-Arm, Corner Bias, Lg. Action	30.051	-
FFN (Ours)	11.196	25.067

4.4 Real World Experiments

We evaluate FabricFlowNet in the real world and demonstrate that our approach successfully manipulates cloth on a real robot system.

4.4.1 Experiment Setup.

Our robot system consists of two 7-DOF Franka Emika Panda arms and a single wrist-mounted Intel RealSense D435 sensor (See Fig. 1.1). We plan pick and place trajectories using MoveIt! [9]. We evaluate on real-world images a 30x30 cm cloth, using 12 single-step and 6 multi-step goals (see Fig. 4.5) that form a representative subset of our simulation test goals.

To transfer our method from simulation to the real world, we align the depth between real and simulated images by subtracting the difference between the average depth of the real support surface (i.e. the table) and the simulated surface. We use color thresholding to obtain a cloth mask, and set non-cloth pixel values to 0. To account for variations in initial cloth pose, we train our networks with random translations (up to 5 pixels) and rotations (up to 5 degrees) of the input images. We found that these simple techniques were sufficient to transfer the networks trained entirely in simulation to the real world, since the simulated depth image matches reasonably well to the real depth image.

4.4.2 Results

Fig. 4.5 provides qualitative real world results, showing that we successfully achieve many of the goals. Our website contains videos of these trials (see link in the appendix). Our method transfers easily to the real world because we use only depth images as input, which appear similar in both simulation and the real world, unlike RGB images.

We compare FabricFlowNet to the NoFlow ablation from Sec. 4.3.5. Both methods were trained with the same sim-to-real techniques described in the previous section. While we do not have access to the true cloth position error in the real world, the Intersection-over-Union (IoU) metric on the achieved cloth masks serves as a reasonable proxy metric [22]. Table 4.7 provides mean IOU (mIOU) performance for NoFlow and FFN on real cloth goals. The NoFlow ablation performs considerably worse compared to FFN on real cloth folding. Qualitative results and the complete set of real square cloth goals are in Fig. 4.5; the complete set of real rectangle and T-shirt goals are in the main text. FFN achieves 0.857 mean IoU over the real test goals, whereas NoFlow only achieves 0.523.

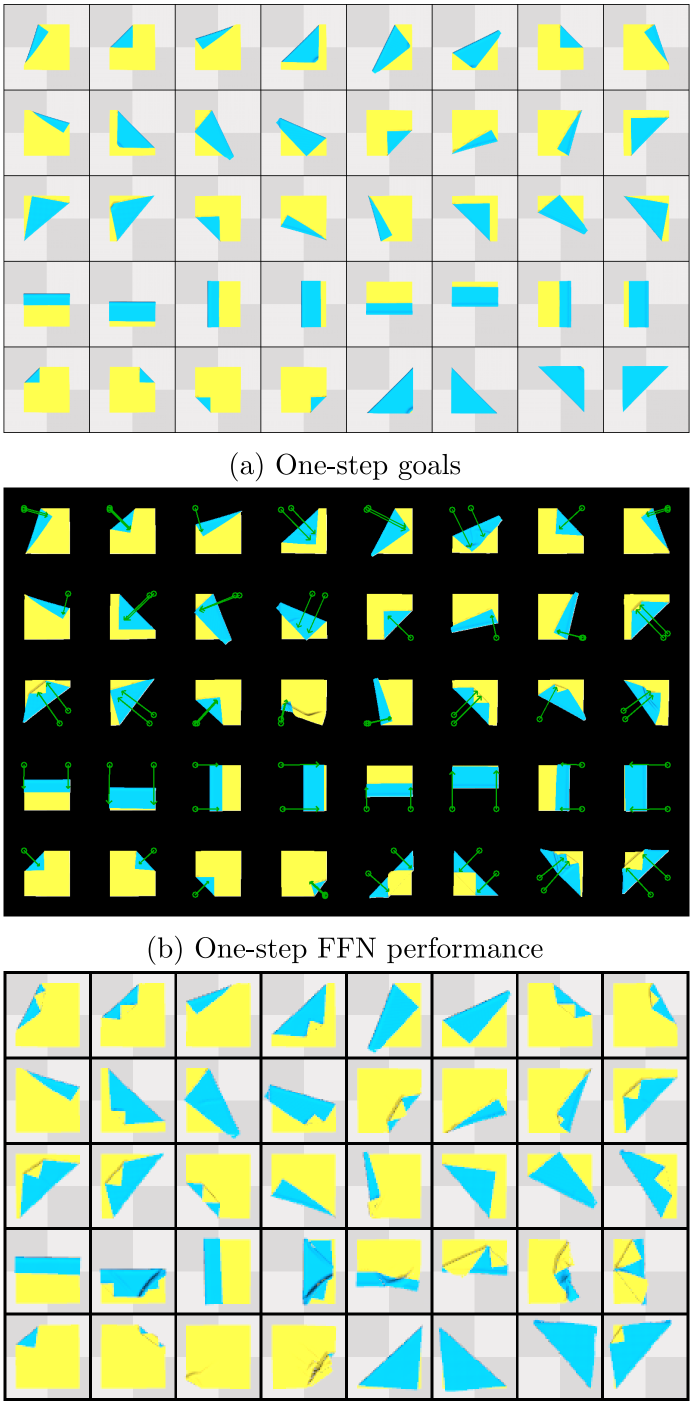
4.4.3 Generalization.

In addition to evaluating the folding policy on square cloth for various goal configurations, we also test the generalization of our method to other shapes of cloth. We evaluate the performance of FFN trained only on a square cloth on folding goals for a rectangular cloth as well as a t-shirt. These fabrics are also thinner than the square blue towel used in the real world experiments above. Fig. 4.6 shows that FFN trained on a square yellow cloth in simulation is able to generalize to other cloth shapes, textures, and colors (FFN only receives depth images as input). Table 4.7 also provides the mIoU for rectangle and t-shirt.

Table 4.7: mIoU for Folding Square Towel, Rectangular Cloth, and T-shirt

Method	One-Step Sq. (12)	Multi-Step Sq. (5)	All Sq. (17)	Rect. (4)	T-shirt (5)
NoFlow	0.561	0.447	0.523	-	-
FFN (Ours)	0.897	0.775	0.857	0.891	0.816

Square cloth results are averaged over 3 runs. For the rectangular cloth and T-shirt, the results are reported for a single run each.

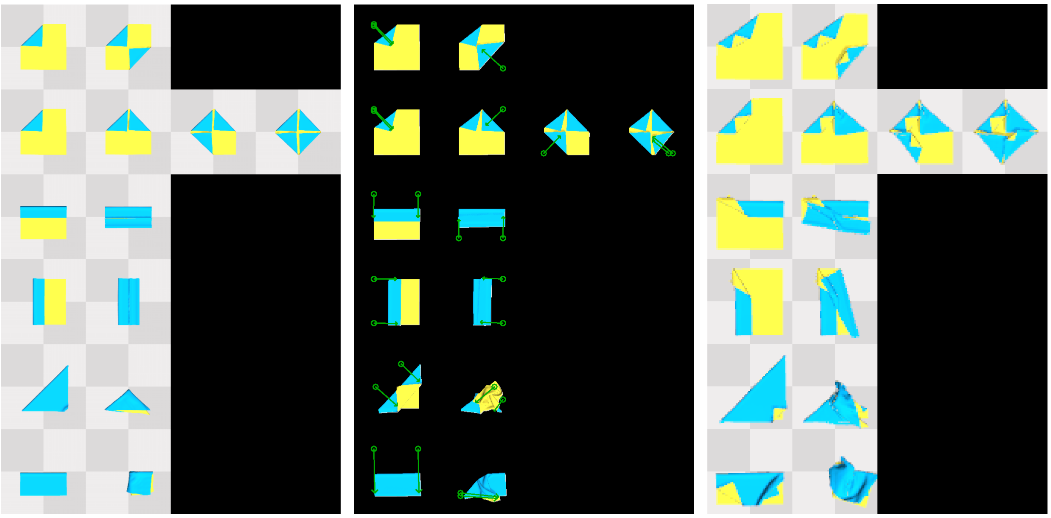


(a) One-step goals

(b) One-step FFN performance

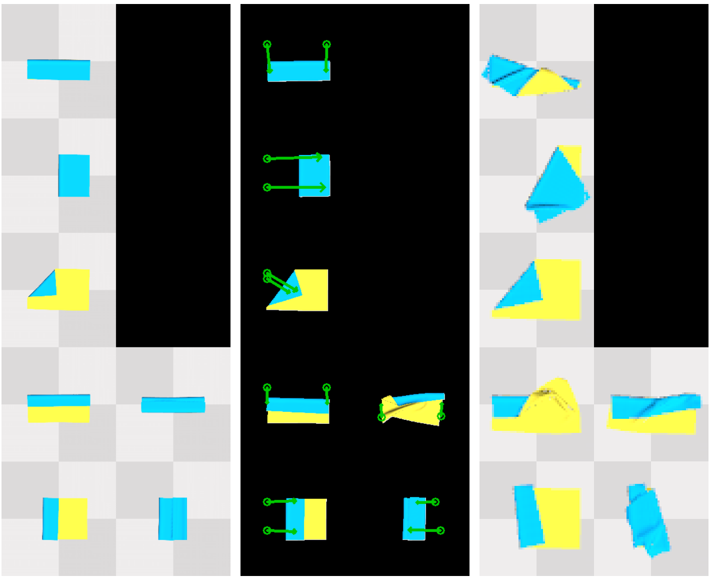
(c) One-step Fabric-VSF performance

Figure 4.1: a) Set of one step goals we evaluate on. b) Configurations achieved by FFN for each of the one-step goals. Arrows indicate the executed action. c) Cloth configurations achieved by Fabric-VSF for the one-step goals.



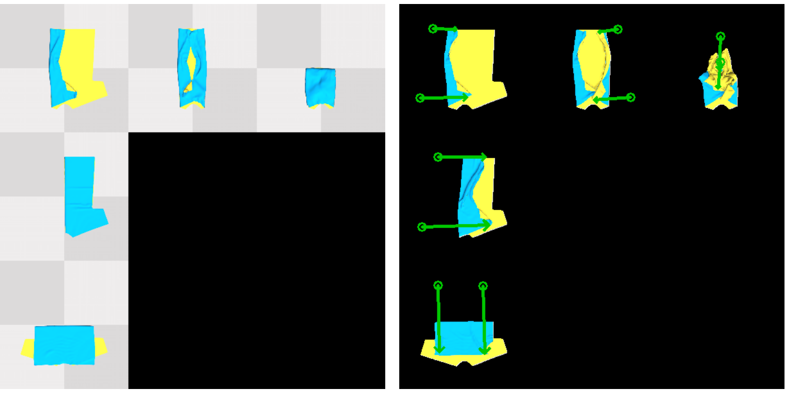
(a) Multi-step goals (b) Multi-step FFN performance (c) Multi-step VSF performance

Figure 4.2: a) Multi-step goals, where each row contains all sub-goals for a given multi-step goal. b) Cloth configurations achieved by FFN for the multi-step goals. c) Cloth configurations achieved by Fabric-VSF for the multi-step goals.



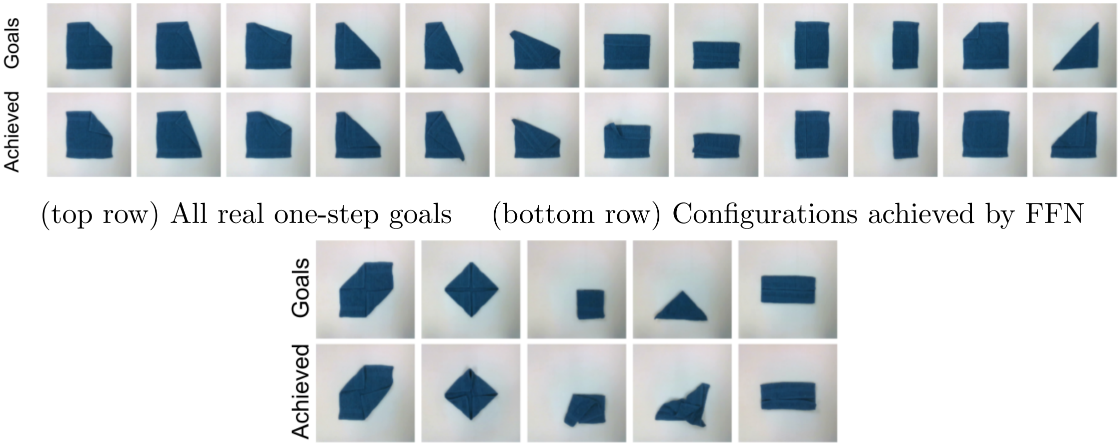
(a) Rect. cloth goals (b) FFN achieved (c) Fabric-VSF achieved

Figure 4.3: Qualitative performance of FFN vs. Fabric-VSF on rectangular cloth. Fabric-VSF uses a lower camera height than FFN (45 cm vs. 65 cm), thus the cloth looks slightly larger in Fabric-VSF images than those of FFN.



(a) tshirt goals (b) qualitative results for tshirt

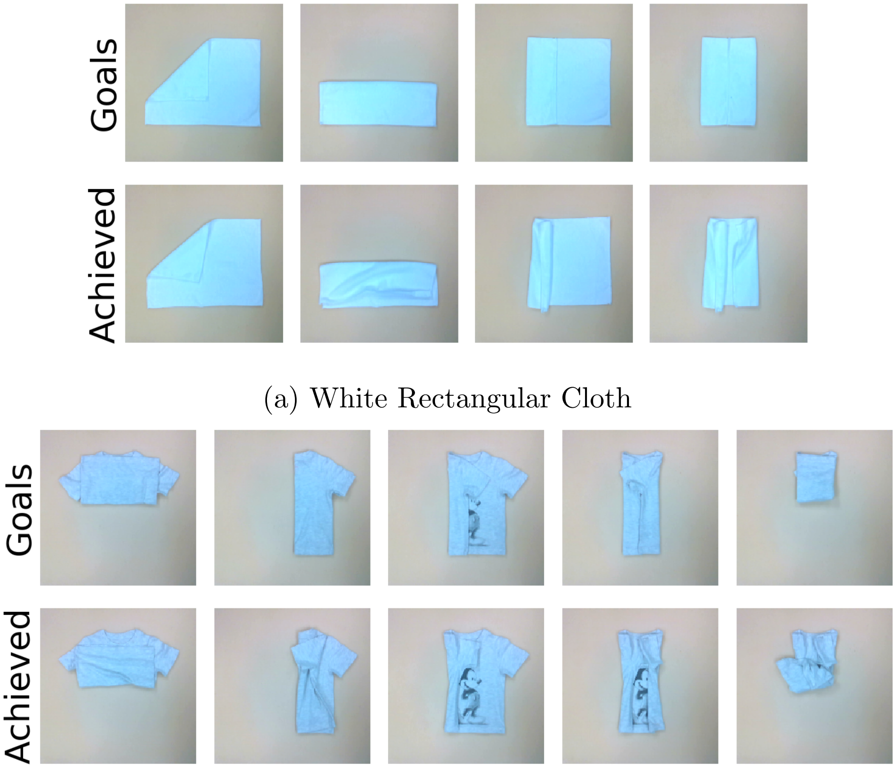
Figure 4.4: Qualitative results for generalization to different shape cloth. The method is trained on square towel data and tested on different cloth goals



(top row) All real one-step goals (bottom row) Configurations achieved by FFN

(top row) All real multi-step goals
(bottom row) Configurations achieved by FFN

Figure 4.5: Qualitative results for FFN on real world experiments. FFN only takes depth images as input, allowing it to easily transfer to cloth of different colors.



(a) White Rectangular Cloth

(b) Grey T-shirt

Figure 4.6: Generalization to new cloth shapes for FFN trained only on a square cloth in simulation. FFN achieves single and multi-step goals for rectangular fabric and a T-shirt.

Chapter 5

Conclusions and Future Work

In this work we present FabricFlowNet, a method which utilizes flow to learn goal-conditioned fabric folding. We leverage flow to represent the correspondence between observations and goals as well as for an action representation. We demonstrate our method for single and dual arm fabric folding. The method is trained entirely using random data in simulation. Our results show that separating the correspondence learning and the policy learning can improve performance on an extensive suite of folding goals in simulated and real environments. Our experiments also demonstrate generalization to different fabric shapes, textures, and colors.

FabricFlowNet can be extended in multiple ways. Currently we only allow a single action per subgoal. However, if the policy makes a mistake, it will cause a compounding effect and lead to poor performance for subsequent subgoals. Instead, we can easily modify FFN so that we take iterative corrective actions on each subgoal. The flow prediction could also act as a goal recognizer; if the flow between the current observation and the goal is sufficiently small, then the goal has been successfully achieved. FFN can be further improved by extending it to deal with mismatch between the observation and goals. In order to handle different size or shape cloth or intracategory variations we could map the observation to some canonical form to find correspondences. If we have a set of demonstrations/goals for different types of cloths we need to figure out dynamically which is closest to the currently observed cloth. Another area to explore is using scene flow instead of 2D optical flow. Similarly, we can apply the general idea of FabricFlowNet to other tasks than cloth folding, such as manipulation of deformable cloth bags. This involves manipulating multiple objects which may or may not be deformable and moving them using a deformable container like a bag. These are all potential directions we can explore as future work.

Appendix A

Additional Details

A.1 Ablation implementation details

A.1.1 DepthIn

The architecture for this ablation is identical to our main method, except that it takes depth images instead of flow images as input. We use a conditioned architecture with two PickNets; PickNet1 receives the observation and goal depth images as input both of size 200 × 200. The place point is computed by querying the flow image similar to our main method.

A.1.2 PredictPlace

We predict the place points similarly to the pick points by using an additional place network. The place network architecture is identical to PickNet. The input is a flow image and the output is the place point predictions.

A.1.3 NoFlow

This ablation is a combination of DepthIn and PredictPlace, where PickNet and PlaceNet both take observation and goal depth images as input.

A.1.4 NoCornerBias

This ablation is the same as our main method except for the training dataset. We use a dataset that does not bias the data to pick corners (See Sec. 3.6). Instead, the pick actions are always uniformly sampled over the visible cloth mask. We still constrain the folding actions for both arms to be in the same direction and distance from their respective pick points and point towards the center of the frame.

A.1.5 SinglePickNet

The architecture of PickNet is modified so that we only have one PickNet for both arms instead of the conditioned architecture used in our main method. The PickNet takes as input the flow image and outputs two heatmaps corresponding to the two pick points.

A.1.6 NoMinLoss

The loss in Eq. 1 is replaced with the following:

$$\mathcal{L}_{\text{NoMin}} = \text{BCE}(H_1, H_1^*) + \text{BCE}(H_2, H_2^*) \tag{A.1}$$

Appendix B

Code and Demos

B.1 FabricFlowNet Github Repository

https://github.com/sujaymanb/FabricFlowNet

B.2 FabricFlowNet Website

https://sites.google.com/view/fabricflownet/home

Bibliography

[1] Artemij Amiranashvili, Alexey Dosovitskiy, Vladlen Koltun, and Thomas Brox. Motion perception in reinforcement learning with dynamic objects. In *Conference on Robot Learning*, pages 156–168. PMLR, 2018. 2.4

[2] Max Argus, Lukas Hermann, Jon Long, and Thomas Brox. Flowcontrol: Optical flow based visual servoing. *arXiv preprint arXiv:2007.00291*, 2020. 2.4

[3] Christian Bersch, Benjamin Pitzer, and Sören Kammel. Bimanual robotic cloth manipulation for laundry folding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1413–1419, 2011. doi: 10.1109/IROS.2011.6095109. 1, 2.1

[4] Ricard Bordalba, Lluís Ros, and Josep M Porta. A randomized kinodynamic planner for closed-chain robotic systems. *IEEE Transactions on Robotics*, 2020. 2.1

[5] Júlia Borràs, Guillem Alenyà, and Carme Torras. A grasping-centered analysis for cloth manipulation. *IEEE Transactions on Robotics*, 36(3):924–936, 2020. 2.1

[6] Cheng Chi and Shuran Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. *arXiv preprint arXiv:2104.05177*, 2021. 2.2

[7] Rohan Chitnis, Shubham Tulsiani, Saurabh Gupta, and Abhinav Gupta. Efficient bimanual manipulation using learned task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1149–1155. IEEE, 2020. 2.1

[8] Rohan Chitnis, Shubham Tulsiani, Saurabh Gupta, and Abhinav Gupta. Intrinsic motivation for encouraging synergistic behavior. *International Conference on Learning Representations*, 2020. 2.1

[9] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012. 4.4.1

[10] Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James F. O'Brien, and Pieter Abbeel. Bringing clothing into desired configurations with limited perception. In *2011 IEEE International Conference on Robotics and Automation*, pages 3893–3900, 2011. doi: 10.1109/ICRA.2011.5980327. 2.1

[11] Siyuan Dong, Devesh K Jha, Diego Romeres, Sangwoon Kim, Daniel Nikovski, and Alberto Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. *arXiv preprint arXiv:2104.01167*, 2021. 2.4

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 2.4

[13] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimír Petrík, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE*

Transactions on Robotics, 32(6):1461–1478, 2016. doi: 10.1109/TRO.2016.2602376. 1, 2.1

[14] Aaron Edsinger and Charles C Kemp. Two arms are better than one: A behavior based control system for assistive bimanual manipulation. In *Recent progress in robotics: Viable robotic service to human*, pages 345–355. Springer, 2007. 2.1

[15] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph E Gonzalez, Nawid Jamali, et al. Learning dense visual correspondences in simulation to smooth and fold real fabrics. *arXiv preprint arXiv:2003.12698*, 2020. 1, 2.2, 4.1

[16] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *arXiv preprint arXiv:2105.03655*, 2021. 2.1

[17] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 3.6.2

[18] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.034. (document), 1, 2.1, 2.3, 4.2, 4.2.1, 4.1

[19] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 2.4, 3.6.3

[20] Rishabh Jangir, Guillem Alenyà, and Carme Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on*

Robotics and Automation (ICRA), pages 4630–4636. IEEE, 2020. 2.1

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3.6.3, 3.6.4

[22] Robert Lee, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. Learning arbitrary-goal fabric folding with one hour of real robot experience. *Conference on Robot Learning*, 2020. (document), 1, 2.3, 3.2, 3.4, 3.6.2, 3.6.3, 3.6.4, 4.1, 4.2, 4.2.2, 4.1, 4.3.4, 4.3.5, 4.4.2

[23] Yinxiao Li, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K. Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1046–1052, 2014. doi: 10.1109/IROS.2014.6942687. 2.2

[24] Yinxiao Li, Danfei Xu, Yonghao Yue, Yan Wang, Shih-Fu Chang, Eitan Grinspun, and Peter K. Allen. Regrasping and unfolding of garments using predictive thin shell modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 2.1

[25] Yinxiao Li, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter K. Allen. Folding deformable objects using predictive simulation and trajectory optimization. In *Proceedings of the 27th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 2.3

[26] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Proceedings of (CoRL) Conference on Robot Learning*, November 2020. 3.6.1, 4.3.1

[27] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, 2021. 2.3

[28] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315, 2010. doi: 10.1109/ROBOT.2010.5509439. 1, 2.1

[29] S. Miller, J. V. D. Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31:249 – 267, 2012. 2.2

[30] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017. 1, 2.3, 3.1, 3.2, 3.6.4, 4.1

[31] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 2.3, 3.1, 3.2

[32] Jianing Qian, Thomas Weng, Brian Okorn, and L. Zhang. Cloth region segmentation for robust grasp selection. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9553–9560, 2020. 2.2

[33] J. Sanchez, J. Corrales, B. Bouzgarrou, and Y. Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37:688 – 716, 2018. 2.1

[34] Daniel Seita, Nawid Jamali, Michael Laskey, Ajay Kumar Tanwani, Ron Berenstein, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep

Transfer Learning of Pick Points on Fabric for Robot Bed-Making. In *International Symposium on Robotics Research (ISRR)*, 2019. 2.2

[35] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. *arXiv preprint arXiv:2012.03385*, 2020. 1, 2.2, 3.2, 3.4, 4.3.5

[36] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 1, 2.3, 3.2

[37] Ruhizan Liza Ahmad Shauri and Kenzo Nonami. Assembly manipulation of small objects by dual-arm manipulator. *Assembly Automation*, 2011. 2.1

[38] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V. Dimarogonas, and Danica Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous Systems*, 60(10):1340–1353, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.07.005. 2.1

[39] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020. 2.2

[40] Raúl Suárez, Jan Rosell, and Néstor García. Using synergies in dual-arm

manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5655–5661, 2015. doi: 10.1109/ICRA.2015.7139991. 2.1

[41] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. Emd net: An encode–manipulate–decode network for cloth manipulation. *IEEE Robotics and Automation Letters*, 3(3):1771–1778, 2018. doi: 10.1109/LRA.2018.2800122. 2.1

[42] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 2.4

[43] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.065. 1, 2.3, 3.2, 3.4

[44] Fan Xie, Alexander Chowdhury, M. Clara De Paolis Kaluza, Linfeng Zhao, Lawson Wong, and Rose Yu. Deep imitation learning for bimanual robotic manipulation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2327–2337. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/18a010d2a9813e91907ce88cd9143fdf-Paper.pdf`. 2.1

[45] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, Kei Okada, and Masayuki Inaba. Bottom dressing by a dual-arm robot using a clothing state estimation based on dynamic shape changes. *International Journal of Advanced Robotic Systems*, 13(1):5, 2016. doi: 10.5772/61930. 2.4

[46] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning

predictive representations for deformable objects using contrastive estimation. *Robotics: Science and Systems*, 2020. 1, 2.2, 3.2