FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy

Anonymous Author(s)

Abstract: 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. FabricFlowNet also elegantly switches between dual-arm and single-arm actions based on the desired goal. We show that FabricFlowNet significantly 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. Finally, we show that our method generalizes when trained on a single square cloth to other cloth shapes, such as T-shirts and rectangular cloths. Video and other supplementary materials are available at: https://sites.google.com/view/fabricflownet.

Keywords: deformable object manipulation, optical flow, bimanual manipulation

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 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 [1, 2, 3].

Recently, learning-based approaches have been explored for cloth manipulation [4, 5, 6, 7, 8], including model-free reinforcement learning to obtain a policy [9, 10]. 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 [9, 8].

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 [9, 10]; however, a reward function is 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 paper, we introduce FabricFlowNet (FFN), a goal-conditioned policy for bimanual cloth manipulation that uses optical flow to improve policy performance (see Fig. 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

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

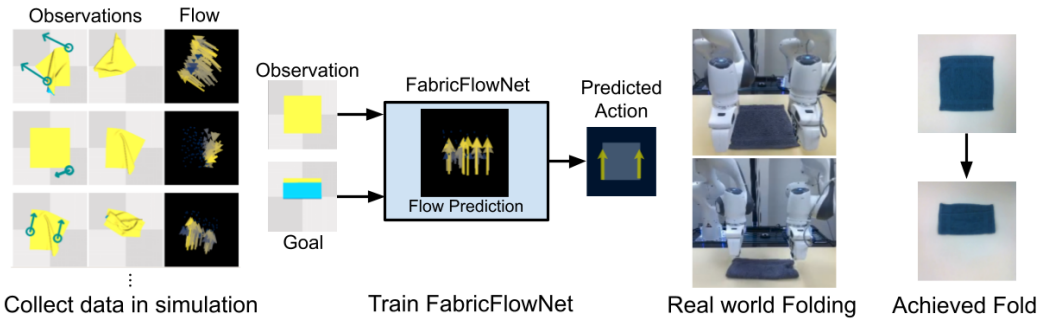


Figure 1: FabricFlowNet (FFN) overview. We collect a dataset of random actions and ground truth flow to train FFN. FFN learns to predict flow and uses it as both an input and action representation in a manipulation policy. FFN successfully performs single and dual-arm folding in the real world.

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 [4, 9]. We present 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 [11, 9]
- Experiments showing that FabricFlowNet significantly outperforms baselines and ablations and generalizes to other cloth colors and shapes, even without training on such variations.

2 Related Work

Bimanual Manipulation. A large body of research exists on dual-arm, or bimanual, manipulation [12]. Dual-arm systems allow for more complex behaviors than single-arm systems at the cost of greater planning complexity [13, 14], leading to research on closed kinematic chain planning [15, 16], composable skill learning [17, 18], and rewarding synergistic behavior [19]. Prior work has also explored bimanual cloth manipulation [20]. Cloth manipulation is a highly underactuated task, and bimanual manipulation enables controlling multiple cloth points [21]. 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 [22, 23, 1, 2, 3]. Previous work in this direction uses hard-coded policies [1, 2, 3], whereas we learn to achieve arbitrary folded configurations. Tanaka *et al.* [24] 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 [4] and show significantly improved performance. Dynamic bimanual manipulation has also been explored in simulation from ground-truth keypoints [25] and for unfolding cloth in the real world [26]; we perform real-world bimanual folding using depth image observations.

State Representations for Cloth Manipulation. Prior works have proposed various representations for cloth manipulation, such as parametrized shape models [27] or binary occupancy features [28]. Recent approaches have proposed to learn representations for cloth, such as using contrastive learning to learn a latent vector dynamics model [6]. Other approaches use contrastive learning to learn pixel-wise latent embeddings for cloth [29, 11, 30] which can be used to learn to imitate an expert

demonstration [11]; 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 [31, 32]. Goal-conditioned transporter networks are another approach for deformable manipulation [8], 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.

Policy Learning for Cloth Manipulation. Various approaches have been applied to policy learning for cloth smoothing, such as reinforcement learning [10] and imitation learning [5]. 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 [7, 33]. For cloth manipulation, Lee *et al.* [9] learns a model-free value function, whereas other papers learn a cloth dynamics model in pixel-space [4], over a graph of keypoints [34], or using a cloth simulator [35]. We compare our approach to state-of-the-art approaches for cloth manipulation and show significantly improved performance.

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 [36, 37, 38]. Optical flow between successive observations has previously been used as an input representation to capture object motion for peg insertion [39] or dynamic tasks [40]. Within the domain of cloth manipulation, Yamazaki *et al.* [41] 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.* [42] 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.

3 Learning a Goal-Conditioned Policy for Bimanual Cloth Manipulation

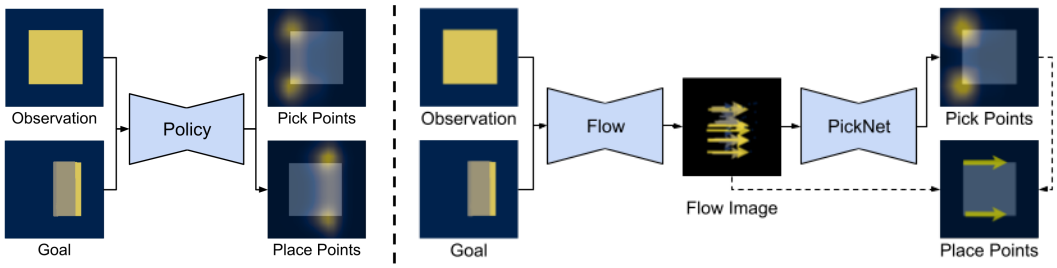
3.1 Problem Formulation

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, because a folded cloth can be highly self-occluded such that a single goal observation fails to describe the full goal state. Defining a task using a sequence of sub-goals is found in other recent work [33]. Similar to prior work [33, 7], 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. 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 [33] 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 directly into to a neural network representation of a policy [33, 9] or a Q-function [10, 9]. 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: we will learn to reason about the relationship between the observation and the goal, and separately use this relationship to reason over actions. 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



(a) Naive system (b) FabricFlowNet (FFN)

Figure 2: (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.

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 [5, 8, 6, 7, 33, 10] or implicitly by transforming the inputs to a Q-function [9]. 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. 2b. 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.

To capture the complex correspondences between x_t and x_i^g, we train a convolutional neural network to estimate the flow image f (see Appendix for details). 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 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 Pick-

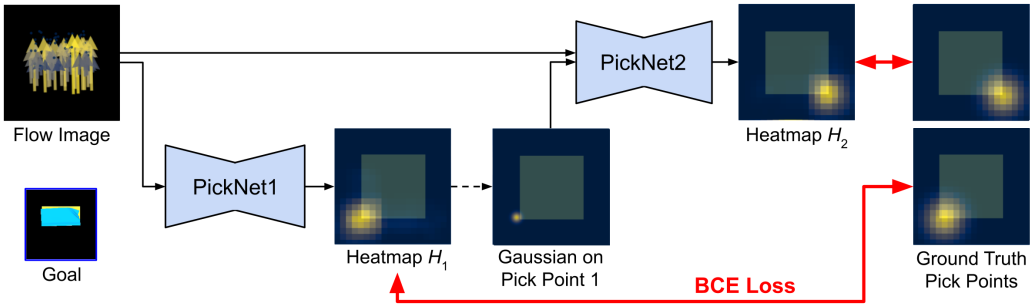


Figure 3: PickNet architecture. We utilize a two-network architecture for bimanual manipulation, where the second pick point is conditioned on the prediction of the first pick point.

Net 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 [9, 8], 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). This architecture was inspired by Wu *et al.* [10], 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, it might be unclear to the network 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:

$$\begin{aligned} l_{\text{BCE}}(H_i, H_j, H_i^*, H_j^*) &= \text{BCE}(H_i, H_i^*) + \text{BCE}(H_j, H_j^*) \\ \mathcal{L}_{\text{Pick}} &= \min[l_{\text{BCE}}(H_1, H_2, H_1^*, H_2^*), l_{\text{BCE}}(H_2, H_1, H_1^*, H_2^*)] \end{aligned} \tag{1}$$

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 (see Fig. 2a). 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

We use SoftGym [43], 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 collect data by taking random actions, biased towards grasping corners of the cloth. We demonstrate that we are able to train our method in SoftGym and then transfer the policy to the real world. Details on the data collection, as well as the network architecture and training details, can be found in the Appendix.

4 Experiments

4.1 Simulation Experiments

Experiment Setup. We evaluate our method and compare to state-of-the-art baselines in the SoftGym [43] simulator; real-world evaluations are below in Sec. 4.2. Our error metric is the average particle position error between the achieved and goal cloth configuration. We evaluate on two sets of goals: 40 *one-step* goals that can be achieved with a single fold action, and 6 *multi-step* goals that require multiple folding actions. 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 Sec. 3.1, and is similar to the protocol in Nair *et al.* [7]. Our goals include test goals from Ganapathi *et al.* [11] and Lee *et al.* [9] that are achievable with one arm, as well as additional goals more suitable for two-arm actions (see Appendix Fig. S1 for the full set of goals).

We compare our method to Fabric-VSF [4], 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 [4]. 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 also compare to Lee *et al.* [9], a model-free approach. We extend the 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). Additional implementation details for the baselines can be found in the Appendix.

4.1.1 Results

Table 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 Fabric-VSF and both the single-arm and dual-arm versions of Lee *et al.* . Lee *et al.* achieves better performance than Fabric-VSF. Qualitative

results for these simulated experiments, as well as results from other versions of these baselines, can be found in the Appendix.

Table 1: Average Particle Distance Error (mm) on Cloth Folding in Simulation

Method	One Step (40)	Multi Step (6)	All (46)
Fabric-VSF [4]	37.469	63.803	41.426
Lee *et al.*, 1-Arm [9]	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**

4.1.2 Ablations

We run series of ablations to evaluate the importance of the components of our system; results can be found in Table 2. 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 [9, 8]. 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"); see Appendix for details. 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. 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 2: Ablation Experiments

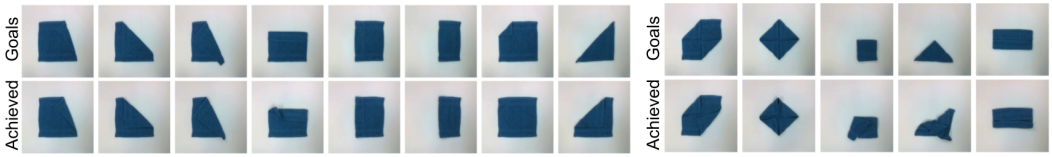
Ablation	DepthIn	PredictPlace	NoFlow	NoCornerBias	SinglePickNet	NoMinLoss
Error	10.592	12.881	17.348	12.972	9.627	9.400

4.2 Real World Experiments

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

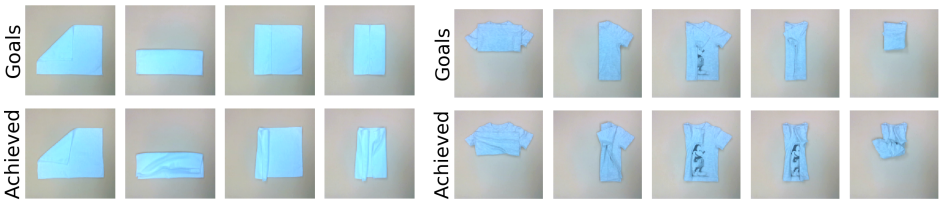
Experiment Setup. Our robot system consists of two 7-DOF Franka Emika Panda arms and a single wrist-mounted Intel RealSense D435 sensor (See Fig. 1). We plan pick and place trajectories using MoveIt! [44]. We evaluate on real-world images a 30x30 cm cloth, using 12 single-step and 6 multi-step goals (see Fig. 4 for examples) 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



(a) One-step Goals (b) Multi-step Goals

Figure 4: 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 5: 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.

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.

4.2.1 Results

Fig. 4 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 abstract). 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. 4.1.2. 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 [9]. FFN achieves 0.857 mean IoU over the real test goals, whereas NoFlow only achieves 0.523. See the Appendix for additional details.

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. 5 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).

5 Conclusion

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.

References

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

[2] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrík, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478, 2016. doi:10.1109/TRO.2016.2602376.

[3] C. Bersch, B. Pitzer, and S. 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.

[4] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi:10.15607/RSS.2020.XVI.034.

[5] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. Canny, and K. Goldberg. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[6] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto. Learning predictive representations for deformable objects using contrastive estimation. *Robotics: Science and Systems*, 2020.

[7] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. 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.

[8] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. *arXiv preprint arXiv:2012.03385*, 2020.

[9] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner. Learning arbitrary-goal fabric folding with one hour of real robot experience. *Conference on Robot Learning*, 2020.

[10] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. 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.

[11] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, et al. Learning dense visual correspondences in simulation to smooth and fold real fabrics. *arXiv preprint arXiv:2003.12698*, 2020.

[12] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. 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.

[13] A. Edsinger and C. 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.

[14] R. L. A. Shauri and K. Nonami. Assembly manipulation of small objects by dual-arm manipulator. *Assembly Automation*, 2011.

[15] R. Suárez, J. Rosell, and N. 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.

[16] R. Bordalba, L. Ros, and J. M. Porta. A randomized kinodynamic planner for closed-chain robotic systems. *IEEE Transactions on Robotics*, 2020.

[17] F. Xie, A. Chowdhury, M. C. De Paolis Kaluza, L. Zhao, L. Wong, and R. 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.

[18] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Efficient bimanual manipulation using learned task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1149–1155. IEEE, 2020.

[19] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Intrinsic motivation for encouraging synergistic behavior. *International Conference on Learning Representations*, 2020.

[20] 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.

[21] J. Borràs, G. Alenyà, and C. Torras. A grasping-centered analysis for cloth manipulation. *IEEE Transactions on Robotics*, 36(3):924–936, 2020.

[22] Y. Li, D. Xu, Y. Yue, Y. Wang, S.-F. Chang, E. Grinspun, and P. 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.

[23] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. 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.

[24] D. Tanaka, S. Arnold, and K. 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.

[25] R. Jangir, G. Alenyà, and C. Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636. IEEE, 2020.

[26] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *arXiv preprint arXiv:2105.03655*, 2021.

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

[28] Y. Li, Y. Wang, M. Case, S.-F. Chang, and P. 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.

[29] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

[30] C. Chi and S. Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. *arXiv preprint arXiv:2104.05177*, 2021.

[31] J. Qian, T. Weng, B. 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.

[32] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg. Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making. In *International Symposium on Robotics Research (ISRR)*, 2019.

[33] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[34] X. Ma, D. Hsu, and W. S. Lee. Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, 2021.

[35] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. 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.

[36] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[37] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. 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.

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

[39] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. *arXiv preprint arXiv:2104.01167*, 2021.

[40] A. Amiranashvili, A. Dosovitskiy, V. Koltun, and T. Brox. Motion perception in reinforcement learning with dynamic objects. In *Conference on Robot Learning*, pages 156–168. PMLR, 2018.

[41] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. 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.

[42] M. Argus, L. Hermann, J. Long, and T. Brox. Flowcontrol: Optical flow based visual servoing. *arXiv preprint arXiv:2007.00291*, 2020.

[43] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Proceedings of (CoRL) Conference on Robot Learning*, November 2020.

[44] S. Chitta, I. Sucan, and S. Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.