VisuoSpatial Foresight for Physical Sequential Fabric Manipulation

Ryan Hoque* · **Daniel Seita*** · **Ashwin Balakrishna** · **Aditya Ganapathi** · **Ajay Kumar Tanwani** · **Nawid Jamali** · **Katsu Yamane** · **Soshi Iba** · **Ken Goldberg**

Abstract Robotic fabric manipulation has applications in home robotics, textiles, senior care and surgery. Existing fabric manipulation techniques, however, are designed for specific tasks, making it difficult to generalize across different but related tasks. We build upon the Visual Foresight framework to learn fabric dynamics that can be efficiently reused to accomplish different sequential fabric manipulation tasks with a single goal-conditioned policy. We extend our earlier work on VisuoSpatial Foresight (VSF), which learns visual dynamics on domain randomized RGB images and depth maps simultaneously and completely in simulation. In this earlier work, we evaluated VSF on multi-step fabric smoothing and folding tasks against 5 baseline methods in simulation and on the da Vinci Research Kit (dVRK) surgical robot without any demonstrations at train or test time. A key finding was that depth sensing significantly improves performance: RGBD data yields an **80**% improvement in fabric folding success rate in simulation over pure RGB data. In this work, we vary 4 components of VSF, including data generation, visual dynamics model, cost function, and optimization procedure. Results suggest that training visual dynamics models using longer, corner-based actions can improve the efficiency of fabric folding by 76% and enable a physical sequential fabric folding task that VSF could not previously perform with 90% reliability. Code, data, videos, and supplementary material are available at `https://sites.google.com/view/fabric-vsf/`.

* Equal contribution

Ryan Hoque
ryanhoque@berkeley.edu

Daniel Seita
seita@berkeley.edu

1 Introduction

Advances in robotic manipulation of deformable objects has lagged behind work on rigid objects due to the far more complex dynamics and configuration space. Fabric manipulation in particular has applications ranging from senior care [24], sewing [64], ironing [42], bed-making [66] and laundry folding [49, 41, 87, 69] to manufacturing upholstery [78] and handling surgical gauze [73]. However, prior work in fabric manipulation has generally focused on designing policies that are only applicable to a *specific* task via manual design [49, 41, 87, 69] or policy learning [67, 84].

The difficulty in developing accurate analytical models of highly deformable objects such as fabric motivates using data-driven strategies to estimate models, which can then be used for general purpose planning. While there has been prior work in system identification for robotic manipulation [37, 27, 5, 60, 8, 9], many of these techniques depend on reliable state estimation from observations, which is especially challenging for deformable objects. One recent alternative to system identification is visual foresight [17, 20], which uses a large amount of self-supervised interactions to learn a visual dynamics model directly from raw image observations and has shown the ability to generalize to a wide variety of conditions [12]. This learned model can then be used for planning to perform different tasks at test time. The technique has been successfully applied to learning the dynamics of complex tasks such as pushing rigid objects [20] and basic fabric folding [17]. However, two limitations of prior work in visual foresight are 1) the data requirement for learning accurate visual dynamics models is often very high, requiring several days of continuous data collection on real robots [12, 17], and 2) experiments consider only relatively short horizon tasks with a wide range of valid goal images [17].

This paper is an extended version of our prior work, Hoque et al. [29], which presented VisuoSpatial Foresight (VSF) by integrating RGB and depth sensing to learn and plan over dynamics models in simulation using only random interaction data for training and domain randomization techniques for sim-to-real transfer. That paper applied VSF to smoothing and folding tasks (see Figure 1 for example rollouts). In this work, we explore modifications to all major stages of VisuoSpatial Foresight: the data generation, visual dynamics, cost function, and optimization procedure. Specifically, we make the following extensions:

1. A new dataset of 9,932 episodes for learning visual dynamics models for fabric manipulation with a corner selection bias and increased range of motion.
2. New simulation experiments evaluating the tradeoffs between different datasets, learned dynamics models, cost functions, and optimization procedures on system performance.
3. New physical experiments demonstrating 90% reliability on fabric folding with a da Vinci surgical robot, a task the robot was unable to perform successfully in the prior work [29].

Results suggest that the most beneficial extension is the new dataset containing actions that have longer pull distances and bias towards picking at corners. This leads to larger changes in the fabric configuration in regions more broadly relevant for ma-
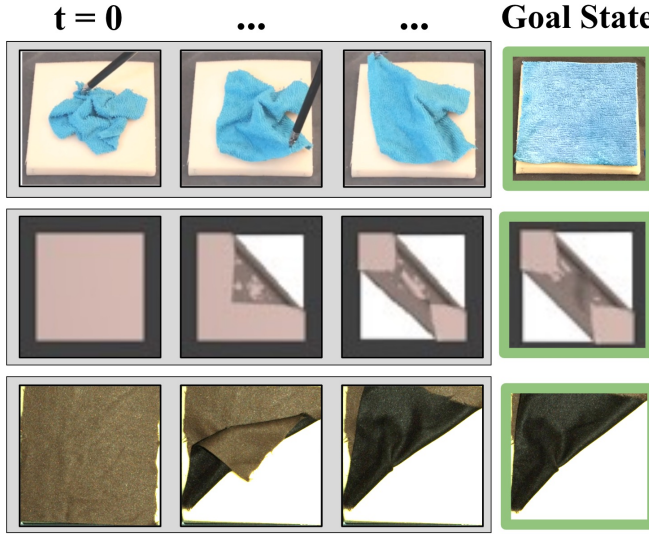
Fig. 1: Using VSF trained on simulated RGBD data, we learn a goal-conditioned fabric manipulation policy without any task demonstrations. **Top:** Subsampled frames from a smoothing episode on physical fabric with the da Vinci surgical robot. **Middle:** Subsampled frames from a double folding episode in simulation with a new dataset, new optimizer, and new cost function. **Bottom:** Subsampled frames from a new folding episode on physical fabric.

nipulation, and the learned dynamics models enable more reliable and efficient fabric folding on the physical robotic system.

2 Related Work

2.1 Geometric Approaches for Robotic Fabric Manipulation

Manipulating fabric is a long-standing challenge in robotics. In particular, prior work has focused on fabric smoothing, as it helps standardize the configuration of the fabric for subsequent tasks such as folding [6, 63]. One popular approach in these works is to first hang fabric in the air and allow gravity to "vertically smooth" it [56, 35, 36, 15]. Maitin-Shepard et al. [46], use this approach to achieve a 100% success rate in single-towel folding over 50 trials. For tasks involving larger fabrics like blankets [66], or those utilizing single-armed robots with a limited range of motion, such vertical smoothing may be infeasible. An alternative approach is to perform fabric smoothing on a flat surface using sequential planar actions as in [71, 72, 83], but these works assume initial fabric configurations closer to fully smoothed than those considered in this work. Similar work addresses both fabric smoothing and folding, such as by Balaguer and Carpin [2] and Jia et al. [31, 32]. These works assume the robot is initialized with the fabric already grasped, while we initialize the robot's end-effector away from the fabric.

2.2 Learning Fabric Manipulation in Simulation and in Real

There has been recent interest in learning sequential fabric manipulation policies with fabric simulators. For example, Seita et al. [67] and Wu et al. [84] learn fabric smoothing in simulation, the former using DAgger [61] and the latter using model-free reinforcement learning (MFRL). Similarly, Matas et al. [48] and Jangir et al. [30] learn policies for folding fabrics using MFRL augmented with task-specific demonstrations. All of these works obtain large training datasets from fabric simulators; examples of simulators with support for fabric include ARCSim [54], MuJoCo [77], PyBullet [11], Blender [10], and NVIDIA FLeX [44]. While these algorithms achieve impressive results, they are designed or trained for specific fabric manipulation tasks such as folding or smoothing, and do not reuse learned structure to generalize to a wide range of tasks.

Ganapathi et al. [23, 22] generalize to multiple tasks by learning fabric correspondences in simulation, but require a task demonstration at test time. Other recent work such as [68], [44] and [19] also aim to learn generalizable fabric manipulation policies in simulation but focus more on rearrangement, transportation and dressing tasks respectively rather than complex folding tasks. Lee et al. [40] successfully learn an arbitrary goal-conditioned fabric folding policy in a model-free manner that is able to achieve unseen fabric goal configurations at test time, but the policy is learned entirely on a real robot which may cause wear-and-tear on the physical system.

2.2.1 Model-Based Fabric Manipulation

Combining model-predictive control (MPC) with learned dynamics is a popular approach for robotics control that has shown success in learning robust closed-loop policies even with substantial dynamical uncertainty [3, 18, 70, 75, 74]. However, many of these prior works require knowledge or estimation of underlying system state, which can often be inaccurate, especially for highly deformable objects. As an alternative to estimating the system state, Finn and Levine [20] and Ebert et al. [17] introduce *visual foresight*, and demonstrate how MPC can plan over learned video prediction models to accomplish a variety of robotic tasks, including deformable object manipulation such as folding pants. However, the trajectories shown in Ebert et al. [17] are limited to a single pick and pull, while we focus on longer horizon sequential tasks that are enabled by a pick-and-pull action space rather than direct end effector control. Furthermore, the fabric manipulation tasks reported have a wide range of valid goal configurations, such as covering a utensil with a towel or moving a pant leg upwards. In contrast, we focus on achieving precise goal configurations via multi-step interaction with the fabric. Yan et al. [86] also take a model-based approach to fabric manipulation, and primarily focus on the fabric smoothing task using latent dynamics models. Lippi et al. [45] generate action plans in a low-dimensional latent space and evaluate on a single T-shirt folding task.

This paper is a direct extension of Hoque et al. [29], which learns a fabric dynamics model from RGB and depth images entirely in simulation and performs model-based planning over the learned dynamics model to achieve fabric smoothing and limited fabric folding tasks. This paper modifies and ablates various components of

the pipeline in [29] including the data generation process, video prediction model, cost function, and action sampling method to improve the folding performance in both simulation and on a physical robotic system.

2.2.2 *Planning with Visual Dynamics Models*

Prior work on visual foresight [20, 17, 12] generally collects data for training visual dynamics models in the real world, which is impractical and unsafe for robots such as the da Vinci surgical robot due to the sheer volume of data required for the technique (on the order of 100,000 to 1 million actions, often requiring several days of physical interaction [12]). One recent exception is the work of Nair et al. [53], which trains visual dynamics models in simulation for Tetris block matching. Finally, prior work in visual foresight learns visual dynamics models with RGB images, but we find that training with RBGD images improves performance, as depth data can provide valuable geometric information for fabric manipulation tasks involving multiple layers.

3 Problem Statement

We consider learning goal-conditioned fabric manipulation policies that enable planning to specific fabric configurations given a goal image of the fabric in the desired configuration. The fabric lies on top of a flat background plane. We assume that the fabric shape is square and that the sides of the fabric are colored differently, where each side is monochromatic. In this paper we test on three goals: a smooth configuration, a triangular single-folded configuration, and a double-folded configuration with three layers stacked in the center of the image, as shown in Figure 1.

We define the fabric configuration at time t as $\boldsymbol{\xi}_t$, represented via a mass-spring system with an $N \times N$ grid of point masses subject to gravity and Hookean spring forces. Due to the difficulties of state estimation for highly deformable objects, we consider overhead RGBD observations $\mathbf{o}_t \in \mathbb{R}^{56 \times 56 \times 4}$, which consist of three-channel RGB and single-channel depth images.

Each task is specified with a goal image observation $\mathbf{o}^{(g)} \in \mathbb{R}^{56 \times 56 \times 4}$ representing the goal g which indicates the appearance of the world the robot must achieve after interacting with the fabric within some finite time horizon T. Thus, we only consider tasks which can be defined with an image of the goal configuration of the fabric. We further assume that the tasks can be achieved with a sequence of pick-and-place actions with a single robot arm, which involve grasping a specific point on the top layer of the fabric and pulling it in a particular direction. The above assumptions hold for a variety of common manipulation tasks such as folding and smoothing. We consider four-dimensional actions,

$$\mathbf{a}_t = \langle x_t, y_t, \Delta x_t, \Delta y_t \rangle. \tag{1}$$

Each action $\mathbf{a}_t$ at time t involves grasping the top layer of the fabric at coordinate (x_t, y_t) with respect to an underlying background plane, lifting, translating by

$(\Delta x_t, \Delta y_t)$ while keeping height fixed, and then releasing and letting the fabric settle. When appropriate, we omit the time subscript t for brevity.

The objective is to learn a goal-conditioned policy which minimizes some goal-conditioned cost function $c_g(\tau)$ defined on realized interaction episodes with goal g and episode $\tau = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$, consisting of a sequence of image observations of the fabric.

4 VisuoSpatial Foresight

We build on the visual foresight framework introduced by Finn and Levine [20] to learn goal-conditioned fabric manipulation policies. In visual foresight, a video prediction model (also called a visual dynamics model) is trained on random interaction data of the robot in the environment. This model is trained to generate a sequence of predicted images (i.e., frames) that would result from executing a sequence of proposed actions in the environment given a history of observed images. Then, MPC is used to plan over this visual dynamics model with some cost function evaluating the discrepancy between predicted images and a desired goal image.

In Hoque et al. [29], we present VisuoSpatial Foresight, where 1) a visual dynamics model is trained on RGBD images instead of RGB images as in [20], and 2) visual dynamics are learned entirely in simulation. We find that these choices improve performance on complex fabric manipulation tasks in simulation and real, accelerate data collection, and limit wear-and-tear on a physical robot. In this work, we extend Hoque et al. [29] and explore the tradeoffs involved in each of several different design decisions for each core aspect of VisuoSpatial Foresight. As elaborated later in Section 6, we use the term VSF-1.0 to refer to the specific settings used in [29]. In this section, we review learning VisuoSpatial dynamics models (Section 4.1), model-based planning over the learned dynamics model (Section 4.2), and specifying planning costs (Section 4.3). Each subsection discusses the methodology from our prior work [29] and then new alternative techniques that we explore in this paper.

4.1 Learning VisuoSpatial Dynamics

To represent fabric dynamics, we train deep recurrent convolutional networks [25] to predict a sequence of RGBD output images conditioned on a sequence of RGBD context images and a sequence of actions. As noted in Babaeizadeh et al. [1], video prediction is inherently stochastic due to incomplete information provided from context images. For example, a pick-and-pull action applied on fabric will have different effects based on unknown stiffness and friction parameters. Therefore, we leverage two widely-used recurrent stochastic video prediction models: Stochastic Variational Video Prediction (SV2P) from Babaeizadeh et al. [1], which we used in [29], and Stochastic Video Generation (SVG) from Denton and Fergus [13], a more recent model which we evaluate in this work. We describe details in Sections 4.1.1 and 4.1.2, respectively.

4.1.1 Stochastic Variational Video Prediction

SV2P is an action-conditioned video prediction model which can predict future images conditioned on a sequence of prior images and proposed actions. In [29], we trained SV2P in a self-supervised manner on thousands of episodes of random interaction with the fabric from the simulation environment in Seita et al. [67], where an episode consists of a contiguous trajectory of observations and pick-and-pull actions (Equation 1).

Precisely, SV2P trains a generative model to predict a sequence of H output images conditioned on a context vector of m images and a sequence of actions starting from the most recent context image. To handle stochasticity, SV2P uses latent variables to capture different modes in the distribution of predicted images, thus making predictions conditioned on a vector of latent variables $\mathbf{z}_{t+m:t+m+H-1}$, each sampled from a Gaussian prior distribution at inference time. For SV2P, the prior $p(\mathbf{z})$ is fixed at each time step t, resulting in the generative model parameterized by θ:

$$
\begin{aligned}
p_\theta&(\hat{\mathbf{o}}_{t+m:t+m+H-1}|\hat{\mathbf{a}}_{t+m-1:t+m+H-2}, \mathbf{o}_{t:t+m-1}) \\
&= p(\mathbf{z}_{t+m}) \prod_{t'=t+m}^{t+m+H-1} p_\theta(\hat{\mathbf{o}}_{t'}|\mathbf{o}_{t:t+m-1}, \hat{\mathbf{o}}_{t+m:t'-1}, \mathbf{z}_{t'}, \hat{\mathbf{a}}_{t'-1}).
\end{aligned}
\tag{2}
$$

Here $\mathbf{o}_{t:t+m-1}$ are image observations from time t up to and including $t + m - 1$, $\hat{\mathbf{a}}_{t+m-1:t+m+H-2}$ is a candidate action sequence at timestep $t + m - 1$, and $\hat{\mathbf{o}}_{t+m:t+m+H-1}$ is the sequence of predicted images. Since the generative model is trained in a recurrent fashion, it can be used to sample an H-length sequence of predicted images $\hat{\mathbf{o}}_{t+m:t+m+H-1}$ for any $m > 0, H > 0$ conditioned on a current sequence of image observations $\mathbf{o}_{t:t+m-1}$ and an H-length sequence of proposed actions taken from $\mathbf{o}_{t+m-1}$, given by $\hat{\mathbf{a}}_{t+m-1:t+m+H-2}$. For more details on model architectures and training procedures for SV2P, we refer the reader to Babaeizadeh et al. [1]. We build upon author-provided open-source code for SV2P from [79].

4.1.2 Stochastic Video Generation

As an alternative to SV2P, we test with SVG [13], which has been found to predict sharper images on standardized benchmarks such as Stochastic Moving MNIST [13] and the BAIR Robot Pushing dataset [16]. SVG, however, does not support action conditioning, which is critical for model-based planning. We add support for action conditioning to SVG, as described below, with a similar approach as in prior work from Nair and Finn [52]. During training, similarly to SV2P, SVG samples latent variables $\mathbf{z}_t$ from a Gaussian posterior distribution; however, while SV2P samples $\mathbf{z}_t$ from the *same* distribution for each t, SVG samples from a *different*, time-dependent posterior distribution $q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t})$ with parameters ϕ (where the dependence on t makes it time-dependent). At inference time, SVG samples $\mathbf{z}_t$ from a Gaussian prior distribution, similarly to SV2P. Unlike SV2P's approach of sampling from a fixed prior $p(\mathbf{z}_t)$ for each t (see Equation 2), SVG uses a more flexible, time-varying prior distribution $p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t-1})$ with parameters ψ learned during training. Denton and
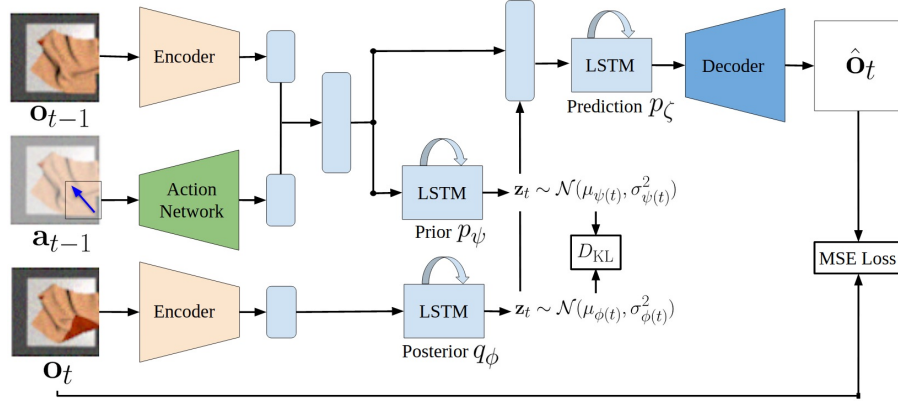
Fig. 2: Flow of data through the proposed action-conditioned SVG architecture during training, described in Section 4.1.2. Given the most recent context image $\mathbf{o}_{t-1}$ and the action $\mathbf{a}_{t-1}$ taken at that time (visualized with the overlaid arrow), the model is trained to predict the next image $\mathbf{o}_t$ via a Mean Square Error loss while simultaneously minimizing a KL divergence loss between the prior p_ψ and posterior q_ϕ Gaussian distributions. The encoder and decoder are convolutional networks with architectures similar to DCGAN. The prior p_ψ, posterior q_ϕ, and prediction p_ζ networks use LSTMs. Each action $\mathbf{a}_{t-1} \in \mathbb{R}^4$ is passed through a small learned network, whose output is concatenated with the encoder output of $\mathbf{o}_{t-1}$. During *training*, the latent variable $\mathbf{z}_t$ is sampled from the output of the posterior distribution, but during *inference* time, the posterior is removed and $\mathbf{z}_t$ is sampled from the prior.

Fergus [13] argue that these more flexible distributions lead to better video prediction quality. We refer the reader to [13] for further details.

Figure 2 shows the flow of image and action data through the proposed action-conditioned SVG variant used in this work. We use DCGAN-style [59] encoders and decoders to handle the image embedding, along with generic LSTMs [28] for the prior p_ψ, posterior q_ϕ, and frame predictor p_ζ components of SVG. To handle action conditioning, at each time step we feed actions (see Equation 1) through a small fully connected network with two layers, and then we concatenate the result with the embedded image from the encoder. The dimension of each embedded image is 128 and the output dimension of the action network is 32, resulting in a concatenated 160-dimensional vector, which is then propagated through the prior and prediction LSTMs. The action-conditioned SVG has a total of 12.6 million parameters, as compared to about 7.9 million parameters for SV2P.

4.2 Model-Based Planning with Model Predictive Control

The goal of the planning stage is to determine which action the robot should take at each time step t. At each step, VisuoSpatial Foresight minimizes a goal-conditioned planning cost function $c_g(\hat{\mathbf{o}}_{t+1:t+H})$ with goal g, which is a target image $\mathbf{o}^{(g)}$. The cost is evaluated over the H-length sequence of predicted images $\hat{\mathbf{o}}_{t+1:t+H}$ sam-

pled from the visuospatial dynamics model (see Section 4.1) conditioned on the current observation $\mathbf{o}_t$ and some proposed action sequence $\hat{\mathbf{a}}_{t:t+H-1}$. As in prior Visual Foresight work [12, 17, 20], we utilize MPC to plan action sequences to minimize $c_g(\hat{\mathbf{o}}_{t+1:t+H})$ over a receding H-step horizon at each time t. See Figure 3 for intuition on the planning phase in the context of fabric manipulation using VSF-1.0 from prior work [29].

There are a number of sampling-based, gradient-free methods to optimize the MPC objective. In our prior work [29], we used the Cross Entropy Method (CEM) [62], which we review in Section 4.2.1. In this work, we additionally test with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [26] (see Section 4.2.2), which allows for more rapid updates in the action sampling distribution. While the Model-Predictive Path Integral (MPPI) has also been shown to be successful in recent work [50], we find that its temporal smoothing is not well-suited for our action space, in which we specify large pick-and-pull actions that are not expected to be similar to prior actions in a given trajectory.

4.2.1 Cross Entropy Method (CEM)

In a high-dimensional space, sampling actions uniformly at random is unlikely to yield a high-quality solution to an optimization problem. To mitigate this issue, we use CEM, which samples from a multivariate Gaussian distribution and iteratively re-fits the Gaussian to the best performing samples. Specifically, for each iterations, CEM:

1. Samples N action sequences $\{\hat{\mathbf{a}}^{(1)}, \hat{\mathbf{a}}^{(2)}, \ldots, \hat{\mathbf{a}}^{(N)}\}$ from some $\mathcal{N}(\mu_i, \Sigma_i)$
2. Finds the M best sequences $\{\hat{\mathbf{a}}^{(1)}, \hat{\mathbf{a}}^{(2)}, \ldots, \hat{\mathbf{a}}^{(M)}\}$ according to cost $c_g(\cdot)$
3. Assigns $\mu_{i+1} = \text{mean}(\{\hat{\mathbf{a}}^{(1)}, \hat{\mathbf{a}}^{(2)}, \ldots, \hat{\mathbf{a}}^{(M)}\})$
4. Assigns $\Sigma_{i+1} = \text{var}(\{\hat{\mathbf{a}}^{(1)}, \hat{\mathbf{a}}^{(2)}, \ldots, \hat{\mathbf{a}}^{(M)}\})$

where "mean$(\cdot)$" and "var$(\cdot)$" denote the sample mean and covariance. However, CEM still scales poorly with dimensionality and can struggle with multimodal optimization landscapes due to its Gaussian structure.

4.2.2 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Like CEM, CMA-ES is an evolutionary strategy based on sampling actions from an iteratively re-fitted multivariate Gaussian distribution. However, in CEM, the standard deviations of the sampling distribution in consecutive iterations are highly correlated, making it difficult to rapidly adjust the variability of the sampling distribution. CMA-ES mitigates this issue by fitting a full covariance matrix to the elite samples and using it to tune the variance of the sampling distribution on each iteration in a more fine-grained manner. CMA-ES also updates the mean to a weighted average over the elites rather than a simple average. Additionally, while CEM updates a *large* population over a *small* number of iterations, we run CMA-ES with a *small* population over a *large* number of iterations, resulting in a more dynamic search of the optimization landscape less prone to averaging over multiple modes. We refer the reader to [26] for further details and to Appendix 10.3 for exact parameters used for both CEM and CMA-ES.
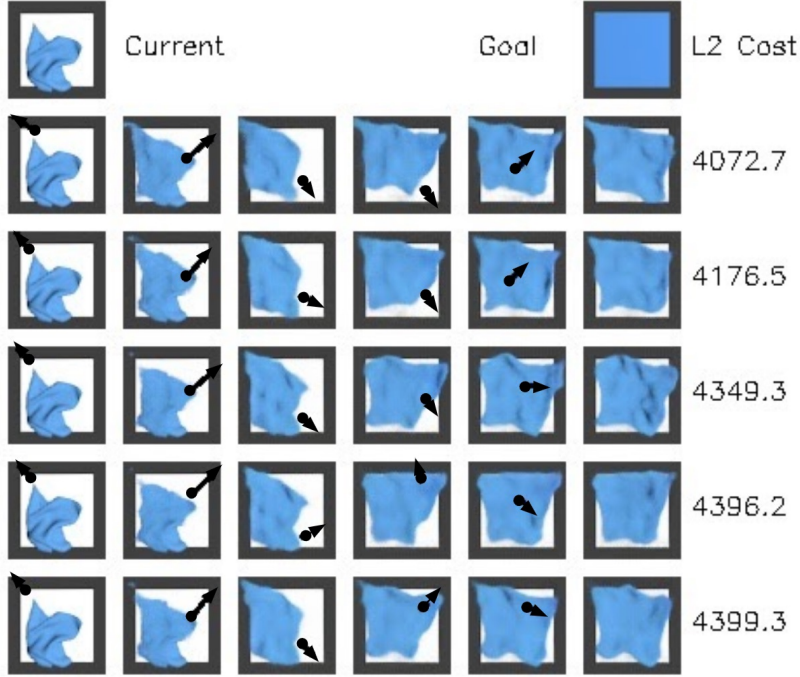
Fig. 3: Real plans generated by VSF-1.0 at test time for the smoothing task using Fabric-Random (see Section 5.2) with predictions from SV2P. We generate action sequences with the Cross Entropy Method (CEM) to approximately minimize the cost function, which evaluates L2 distance between the final image in the predicted trajectory and a provided goal image (see Section 4.3.1). Here we show the five CEM trajectories with the lowest cost, where the image sequences in each row first show the context image, followed by five outputs from the video prediction model. The black arrows are the pick-and-pull actions projected onto the images.

4.3 Planning Costs

The remaining ingredient for model-based planning (Section 4.2) is the cost function definition. A number of choices exist for the cost function $c_g(\cdot)$ used for model-based planning based on goal classifiers, optical flow, and learned distance measures between images [17, 85]. In our prior work [29], we utilized a simple cost function based on the Euclidean pixel distance between images (Section 4.3.1). In this work, we additionally test with a learned cost function which encodes structure about the underlying mesh of the fabric (Section 4.3.2).

4.3.1 Pixel L2 Cost

A simple cost function is the Euclidean (L2) pixel distance between the final predicted RGBD image at timestep t and the goal image $\mathbf{o}^{(g)}$ across all 4 channels. Precisely,

the planning cost is defined as follows:

$$c_g(\hat{\mathbf{o}}_{t+1:t+H}) = \|\mathbf{o}^{(g)} - \hat{\mathbf{o}}_{t+H}\|_2. \tag{3}$$

Figure 3 shows example plans generated by the system for smoothing and indicates the L2 cost between the final predicted image and the goal image in the rightmost column. We use "Pixel L2" to refer to this cost function in this paper.

4.3.2 Learned Vertex L2 Cost

While the Pixel L2 distance cost function is easy to implement and may be sufficient for simple goal images such as fully smooth fabric, it can fail to capture nuances and may focus on irrelevant artifacts in more complex goal images. To this end, we employ a data-driven approach to estimate the cost between two images. When collecting data used to train the visuospatial dynamics model, we can access and store the underlying state of the fabric due to the simulation environment. Therefore, we utilize the same data to train a cost function which estimates the difference in the underlying fabric state based on two images of the fabric in different configurations. Precisely, we annotate pairs of fabric images with the total Euclidean distance between the 3D meshes that constitute the fabric (see Section 5.1) in each image. We then train a convolutional neural network $f_{\mathrm{mesh}}(\cdot, \cdot)$ to predict the (normalized) mesh distance from images by minimizing the Mean Squared Error (MSE) loss on the dataset. The revised planning cost function takes a forward pass through this trained network:

$$c_g(\hat{\mathbf{o}}_{t+1:t+H}) = f_{\mathrm{mesh}}(\mathbf{o}^{(g)}, \hat{\mathbf{o}}_{t+H}), \tag{4}$$

and since VisuoSpatial Foresight data is task-agnostic, as described in Section 5.2, we use the same f_{mesh} network for all three major tasks considered in this work: smoothing, single folding, and double folding. We use "Vertex L2" to refer to this cost function. See Appendix 10.3 for details on the supplemental dataset and network architecture used for f_{mesh}.

5 Practical Implementation Details

In this section, we provide additional details to practically instantiate VisuoSpatial Foresight for goal-conditioned fabric manipulation. We discuss the fabric simulator used for data collection (Section 5.1), how data is collected in this simulator (Section 5.2), and how to train the visuospatial dynamics models (Section 5.3).

5.1 Fabric Simulator

VisuoSpatial Foresight requires a large amount of training data to predict full-resolution RGBD images. Since getting real data is cumbersome and imprecise, we use a fabric simulator to generate data quickly and efficiently. The fabric and robot simulator used in [29] and this work is built on top of the simulator in Seita et al. [67], which was

shown to be sufficiently accurate for imitation learning and sim-to-real transfer of fabric smoothing policies. We briefly review details of the simulator that are shared across both [29] and this work, while highlighting differences in Section 5.2.

The fabric is represented as a mass-spring system with a 25×25 grid of point masses [58] with springs connecting each point to its neighbors. Verlet integration [81] is used to update point mass positions using finite difference approximations, and self-collision is implemented by adding a repulsive force between points that are too close [4]. Damping is also applied to simulate friction. See Appendix 9 for further discussion on alternative fabric simulators and the simulation used in this work.

We use the open-source software Blender [10] to render (top-down) image observations $\mathbf{o}_t$ of the fabric. To facilitate sim-to-real transfer, we leverage domain randomization [76] of the fabric color, background plane shading, image brightness, and camera pose. We make a few changes to the observations relative to [67]. First, we use four-channel images: three for RGB and one for depth. Second, we reduce the size of observations to 56×56 from 100×100 to make it more computationally tractable to train visuospatial dynamics models. Finally, to enable transfer of policies from simulation to the real-world, we adjust the domain randomization techniques so that color, brightness, and positional hyperparameters are fixed *per episode* to ensure that the video prediction model learns to only focus on predicting changes in the fabric configuration, rather than changes due to domain randomization. See Appendix 11 for more details on the domain randomization parameters.

5.2 Data Generation

For generating training data for VisuoSpatial Foresight, episode starting states are sampled from four difficulty tiers with equal probability, where each tier differs in the initial amount of fabric coverage on the underlying plane supporting it. Tiers 1 through 3 are the same as those in [67].

- **Tier 0: Full Coverage.** $100. \pm 0.$ initial coverage, i.e., fully smooth.
- **Tier 1: High Coverage.** $78.3 \pm 6.9\%$ initial coverage, all corners visible. Generated by two short random actions.
- **Tier 2: Medium Coverage.** $57.6 \pm 6.1\%$ initial coverage, one corner occluded. Generated by a vertical drop followed by two actions to hide a corner.
- **Tier 3: Low Coverage.** $41.1 \pm 3.4\%$ initial coverage, 1-2 corners occluded. Generated by executing one action very high in the air and dropping.

5.2.1 Fabric-Random Data

In the prior work (Hoque et al. [29]), we collected a dataset consisting of 7,003 episodes of length 15 each, for a total of 105,045 $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$ transitions. Actions $(x, y, \Delta x, \Delta y)$ are sampled uniformly from [0,0,0,0] to [1,1,0.4,0.4], allowing a maximum pull of 40% of the plane width. Fabric color is randomized in a range around blue, and the underside of the fabric is darker by a fixed RGB delta. Henceforth, we refer to this data as Fabric-Random. Figures 3 and 6 provide examples of images from Fabric-Random without domain randomization.

All data is generated using the following policy: execute a randomly sampled action, but resample if the grasp point (x, y) is not within the bounding box of the 2D projection of the fabric, and truncate Δx and/or Δy at the edge of the plane if $(x + \Delta x, y + \Delta y)$ is out of bounds.

5.2.2 New Fabric-CornerBias Data

While we showed promising smoothing and folding simulation results using Fabric-Random in [29], we were unable to get a physical robot to successfully fold fabric. In addition, the pull vector action magnitudes for VSF-1.0 were relatively small compared to an imitation learning baseline from [67]. This meant VSF-1.0 was inefficient and took several more actions than necessary to complete smoothing or folding tasks. To address these issues, we propose and evaluate Fabric-CornerBias, a new fabric dataset with several notable differences over Fabric-Random.

Fabric-CornerBias consists of 9,932 length-10 episodes for a total of 99,320 $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$ data transitions, meaning that the data is about the same size as Fabric-Random. For visual clarity, the fabric color in the new data is centered around brown (as opposed to blue in Fabric-Random). During data generation, actions $(x, y, \Delta x, \Delta y)$ are sampled from [0,0,0,0] to [1,1,0.6,0.6], allowing a maximum pull of 60% of the plane width. While this increased range of motion may make subsequent video prediction more challenging, since longer pull vectors tend to result in larger relative pixel changes in future images, we hypothesize that including longer pull vectors in the training data will result in more accurate image predictions when considering such actions during MPC planning (see Section 4.2).

Many fabric manipulation tasks, including the smoothing and folding tasks we consider in this work, may be best approached by picking at fabric corners, as suggested by results in [44, 66, 84]. Therefore, we set 30% of all pick points to be the (x, y) coordinates of a randomly chosen corner, to which we have ground-truth access in the simulator. This "corner bias" is not present in Fabric-Random, which may have led VSF-1.0 to produce relatively less accurate future image predictions conditioned on actions that pick at corners. Due to this extra feature, we name the data "Fabric-CornerBias."

Dataset curation is an interesting topic in its own right. In general, the dataset should include states in regions that are relevant for the downstream tasks for reliable video prediction. It is difficult to reach states that require a precise sequence of actions with a purely random policy. In this case, the corner bias can help provide data broadly relevant for many smoothing and folding tasks. More complex tasks such as twisting or rolling fabric would require more careful dataset engineering.

Finally, to provide training data for the learned cost function (Section 4.3.2) we also collect the ground truth (x, y, z) coordinates of all 625 point masses for all time steps in all collected episodes. From Fabric-CornerBias we create a dataset of 99,320 RGBD image pairs annotated with ground truth mesh distance to train the learned cost function as described in Section 4.3.2.

5.3 Model Training

When training a visuospatial dynamics model (either SV2P or SVG) on Fabric-Random, as in [29] and following the notation from Equation 2, we set the number of context frames to $m = 3$ and number of output frames to $H = 7$, so that the model learns to predict 7 frames of an episode from the preceding 3 frames. On Fabric-CornerBias, since the number of actions per episode in the training data is 10 (instead of 15), we set the number of context frames to $m = 2$ and the number of output frames to $H = 5$ to allow for sampling at multiple time ranges within one episode. At test time, both models utilize only one context frame $m = 1$ and a planning horizon of $H = 5$ output frames. This yields the generative model $p_\theta(\hat{\mathbf{o}}_{t+1:t+5} | \hat{\mathbf{a}}_{t:t+4}, \mathbf{o}_t)$, as discussed in Section 4.1.

6 Simulation Experiments

In this section, we report experimental results on the fabric simulation environment. In Section 6.1, we qualitatively and quantitatively analyze the performance of visuospatial dynamics models on predicting images in held-out test episodes, for all combinations of datasets (Fabric-Random and Fabric-CornerBias) and models (SV2P and SVG). Section 6.2 presents results from our prior work [29] using VSF-1.0 settings: Fabric-Random data, SV2P, CEM, and Pixel L2 cost. We then introduce new results in Section 6.3 to test whether changing any set of parameter settings from those in Section 6.2 lead to better performance in downstream fabric manipulation tasks.

6.1 VisuoSpatial Dynamics Prediction Quality

An advantage of training visual dynamics models, as done in visual foresight methods, is that it enables inspection of models to see if predictions are accurate. We perform qualitative and quantitative analysis of action-conditioned video prediction model quality. For both Fabric-Random and Fabric-CornerBias data, we generate 400 episodes using the same data-generating procedure from Section 5.2, but with different random seeds to ensure that the test set contains novel images. We train separate SV2P and SVG models for both Fabric-Random and Fabric-CornerBias and evaluate each of these four models on the appropriate test set.

The models are trained to directly generate RGBD predictions, and we separate the color and depth components for qualitative analysis. Figures 4 and 5 show the ground truth as well as the predicted color and depth image sequences from SV2P and SVG, applied on examples of test-set episodes from Fabric-CornerBias. A prominent distinction between SV2P and SVG is that the former tends to produce blurrier images when predicting 4 or 5 images in the future as compared to SVG. However, SVG may be more susceptible to producing disjoint segments of the fabric. We hypothesize that this is because SV2P relies on an architecture which constrains the flow of predicted pixels [21] while SVG does not.

For a more quantitative measure of prediction quality, we calculate the average Structural SIMilarity (SSIM) index [82] over corresponding predicted and ground
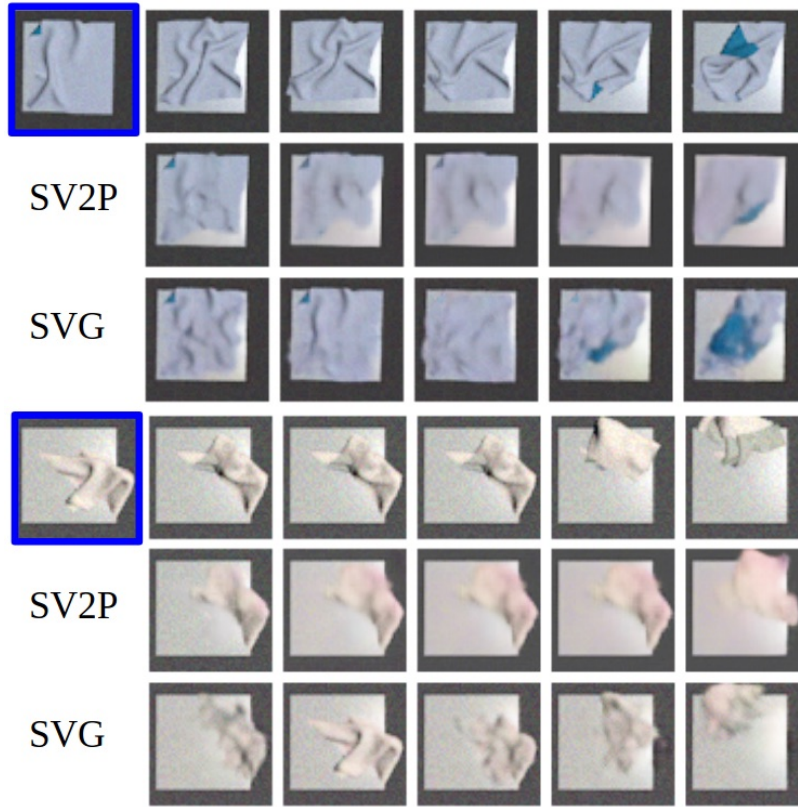
Fig. 4: Two comparisons between ground truth and predicted color images in simulation from SV2P and SVG models on held-out, domain-randomized test data from Fabric-CornerBias. SV2P and SVG are provided a single context ground truth image (indicated with the blue border) and a sequence of 5 actions. For each example, the first row has the ground truth image sequence, the second shows SV2P predictions, and the third shows SVG predictions. While quality gets blurrier across time, the predicted images may be sufficiently accurate for planning.

truth image pairs for the tested models. The SSIM is a scalar quantity between -1 and 1, where higher values correspond to greater image similarity. SSIM is commonly reported in prior video prediction research [1, 13, 21, 38, 39] for quantitatively benchmarking model quality. Tables 1 and 2 report the performance of the two models on each of the two datasets. We report the average SSIM across predicted images as a function of the time horizon. As expected, SSIM decreases with a longer time horizon, due to the difficulty in long-horizon frame prediction. The results also suggest that SV2P tends to produce more accurate predictions for a shorter time horizon, typically the first 1-2 future images, while SVG may be more accurate for longer horizon predictions (4-5 images in the future).
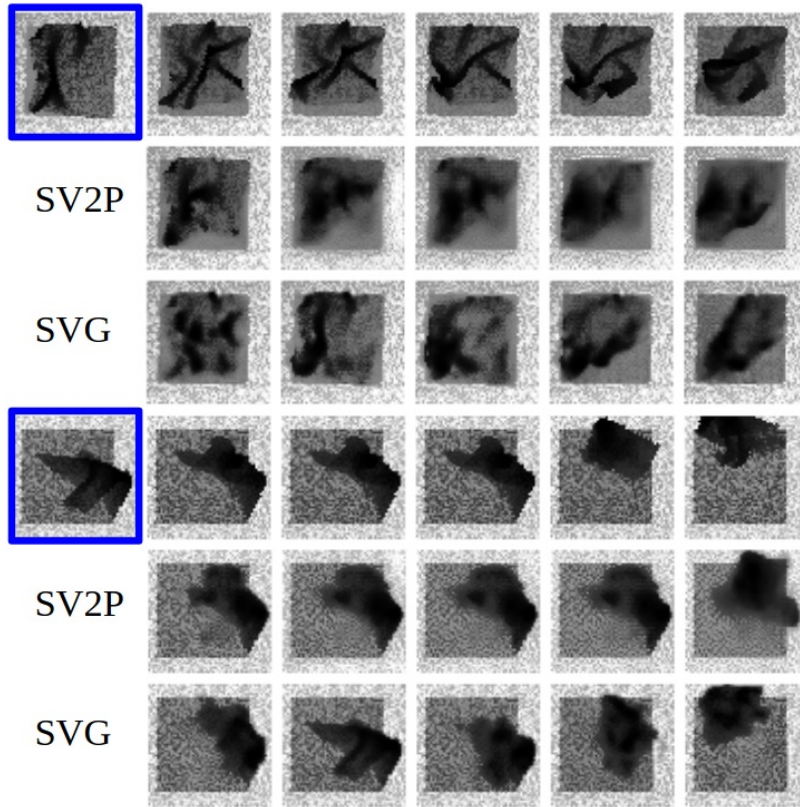
Fig. 5: Depth components of the examples in Figure 4, showing a similar trend. Depth values are scaled into $[0, 255]$ to make images readable.

Overall, the qualitative inspections and quantitative SSIM metrics suggest that using SV2P or SVG as the learned dynamics model may generate sufficiently accurate action-conditioned predictions for multiple images.

6.2 Prior Results from Smoothing and Folding in Simulation

All results presented in this section are from our prior work [29]. We report the performance of VSF with the Fabric-Random dataset, SV2P model, CEM optimizer, and Pixel L2 cost function. We refer to these particular choices of the data, model, optimizer, and cost function of VSF as VSF-1.0, to distinguish these settings from different ablations we test in new experiments in Section 6.3. We first evaluate VSF-1.0 on the smoothing task: maximizing fabric coverage, defined as the percentage of an underlying plane covered by the fabric. The plane is the same area as the fully smoothed fabric. We evaluate smoothing on three tiers of difficulty as reviewed in Section 5.2 (i.e., tiers 1, 2, and 3). Following our prior work [29], episodes can ter-

Table 1: SSIM measurements for Fabric-Random, over ground truth versus predicted images from SV2P and SVG models. Conditioned on one image and five actions starting from that image, the models must predict the next five images. We separate SSIM measurements for color (C) and depth (D) images and by time horizon (i.e. 1-5 time steps into the future). Results suggest that SV2P is more effective at predicting the first 1-2 images, but SVG may produce more accurate predictions beyond that.

(Data) Model	1	2	3	4	5
(C) SV2P	**0.822**	**0.710**	0.638	0.611	0.598
(C) SVG	0.755	0.682	**0.648**	**0.633**	**0.624**
(D) SV2P	**0.790**	**0.631**	0.527	0.470	0.433
(D) SVG	0.648	0.574	**0.540**	**0.523**	**0.511**

Table 2: SSIM measurements for Fabric-CornerBias, over ground truth versus predicted images from SV2P and SVG models. The table is formatted in a similar manner to Table 1 and shows a similar trend.

(Data) Model	1	2	3	4	5
(C) SV2P	**0.774**	**0.706**	0.639	0.616	0.605
(C) SVG	0.741	0.667	**0.642**	**0.631**	**0.625**
(D) SV2P	**0.758**	**0.657**	**0.577**	0.529	0.493
(D) SVG	0.679	0.602	**0.577**	**0.563**	**0.554**

minate earlier if a threshold of 92% coverage is triggered, or if any fabric point falls sufficiently outside of the fabric plane.

To see how VisuoSpatial Foresight performs against existing smoothing techniques, for each difficulty tier, we execute 200 episodes of VSF-1.0 and 200 episodes of each baseline policy discussed in Section 6.2.1. Note that VSF-1.0 does *not* explicitly optimize for coverage and only optimizes the Pixel L2 cost function from Equation 3, which measures Euclidean distance to a target image. In this case, we provide VSF-1.0 with a goal image of a fully smooth fabric. See Figure 6 for an example smoothing episode.

6.2.1 Baseline Methods

For fabric smoothing in simulation, we compare VSF-1.0 with the following 5 baselines as in Hoque et al. [29]. Further details about the implementation and training of VSF-1.0 and the last two baselines listed here are in Appendix 10.

(1) Random Randomly sample the pick point and pull direction.

(2) Highest Using ground truth state information, pick the fabric vertex with the maximum z-coordinate and set the pull direction to point to where the vertex would be if the fabric were perfectly smooth. This is straightforward to implement with depth sensing and was shown to work reasonably well for smoothing in [66].
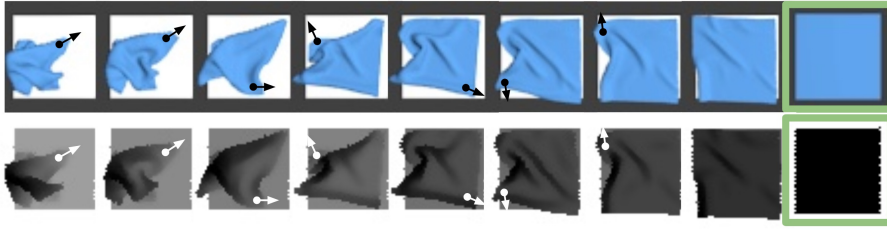
Fig. 6: A simulated episode executed by the VSF-1.0 policy on a Tier 3 starting state, given a smooth goal image (shown in the far right). The first row shows RGB images and the second shows the corresponding depth maps. The images are from the distribution specified in the Fabric-Random data and do not have domain randomization. In this example, the policy is able to successfully cross the coverage threshold of 92% after executing 7 actions. Actions are visualized with the overlaid arrows.

(3) Wrinkle As in Sun et al. [72], find the largest wrinkle and then pull perpendicular to it at the edge of the fabric to smooth it out. We use the ground truth state information in the implementation of this algorithm (as done in [67]) rather than image observations.

(4) Imitation Learning (IL) As in Seita et al. [67], train an imitation learning agent using DAgger [61] with a simulated corner-pulling demonstrator that picks and pulls at the fabric corner furthest from its target. DAgger can be considered as an oracle with "privileged" information as in Chen et al. [7] because during training, it queries a demonstrator which uses ground truth state information. For a fair comparison, we run DAgger so that it consumes roughly the same number of data points (we used 110,000) as VisuoSpatial Foresight during training, and we give the policy access to four-channel RGBD images. We emphasize that this is a distinct dataset from the one used for VSF-1.0 or any other VisuoSpatial Foresight variant in this subsection (Fabric-Random), which uses no demonstrations during data generation.

(5) Model-Free RL We run DDPG [43] and extend it to use demonstrations and a pre-training phase as suggested in Vecerik et al. [80]. We also use the Q-filter from Nair et al. [51]. We train with a similar number of data points as in IL and VisuoSpatial Foresight for a reasonable comparison. We design a reward function for the smoothing task that, at each time step, provides reward equal to the change in coverage between two consecutive states. Inspired by OpenAI et al. [55], we provide a $+5$ bonus for triggering a coverage success, and -5 penalty for pulling the fabric out of bounds.

6.2.2 Smoothing and Folding Results with VSF-1.0

Results in Table 3 indicate that VSF-1.0 significantly outperforms the analytic and model-free reinforcement learning baselines for fabric smoothing in simulation. It has similar performance to the IL agent, a "smoothing specialist" that rivals the performance of the corner pulling demonstrator used in training (see Appendix 10). See

Table 3: Simulated smoothing experimental results for VSF-1.0 and the baselines in Section 6.2. We report final coverage and number of actions per episode, averaged over 200 simulated episodes per tier, and use the same random seeds for a fair comparison. VSF-1.0 performs well even for difficult starting states. It attains similar final coverage as the Imitation Learning (IL) agent from [67] and outperforms the other baselines. The VSF-1.0 and IL agents were trained on equal amounts of domain-randomized RGBD data, but the IL agent has a demonstrator for every training state, whereas VSF-1.0 is trained with data collected from a random policy.

Tier	Method	Coverage	Actions
1	Random	25.0 ± 14.6	2.4 ± 2.2
1	Highest	66.2 ± 25.1	8.2 ± 3.2
1	Wrinkle	$91.3 \pm\ \ 7.1$	5.4 ± 3.7
1	DDPG and Demos	87.1 ± 10.7	8.7 ± 6.1
1	Imitation Learning	$94.3 \pm\ \ 2.3$	3.3 ± 3.1
1	VSF-1.0	$92.5 \pm\ \ 2.5$	8.3 ± 4.7
2	Random	22.3 ± 12.7	3.0 ± 2.5
2	Highest	57.3 ± 13.0	10.0 ± 0.3
2	Wrinkle	87.0 ± 10.8	7.6 ± 2.8
2	DDPG and Demos	82.0 ± 14.7	9.5 ± 5.8
2	Imitation Learning	$92.8 \pm\ \ 7.0$	5.7 ± 4.0
2	VSF-1.0	$90.3 \pm\ \ 3.8$	12.1 ± 3.4
3	Random	20.6 ± 12.3	3.8 ± 2.8
3	Highest	36.3 ± 16.3	7.9 ± 3.2
3	Wrinkle	73.6 ± 19.0	8.9 ± 2.0
3	DDPG and Demos	67.9 ± 15.6	12.9 ± 3.9
3	Imitation Learning	88.6 ± 11.5	10.1 ± 3.9
3	VSF-1.0	$89.3 \pm\ \ 5.9$	13.1 ± 2.9

Figure 6 for an example Tier 3 VSF-1.0 episode. Furthermore, we find that coverage values are statistically significant compared to all baselines other than IL, and that performance is not notably impacted by the use of domain randomization. Results from the Mann-Whitney U test [47] and a domain randomization ablation study are reported in Appendix 11. VSF-1.0, however, requires more actions than DAgger, especially on Tier 1, with 8.3 actions per episode compared to 3.3 per episode. Attempting to mitigate this by increasing the variance of CEM results in poor performance, as actions are sampled outside the truncated action distribution used to generate data. Indeed, this is one of the motivations for the design of Fabric-CornerBias, as described in Section 5.2.2.

We proceed to study the effect of the input modality (i.e. RGB, D, and RGBD) in VSF-1.0. See Figure 7 for a histogram of coverage values obtained on the simulated smoothing task. Here RGBD performs the best but only slightly outperforms RGB, which is perhaps unsurprising due to the relatively low depth variation in the smoothing task. We also vary the input modality in a fabric *folding* task. For folding, we use the same video prediction model, trained only with random interaction
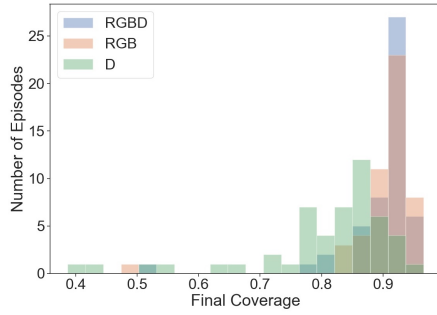
Fig. 7: Final coverage values on 50 VSF-1.0 simulated smoothing episodes from Tier 3 starting states. We fix the random seed so that each input modality (RGB, D and RGBD) begins with the same starting states.

Table 4: Simulated single folding (1-Fold) results. We run VSF-1.0 with the goal image in Figure 8 for 20 episodes when L2 is taken on the depth, RGB, and RGBD channels. The results suggest that adding depth allows us to significantly outperform RGB-only Visual Foresight on this task.

Cost Function	Successes	Failures	% Success
L2 Depth	0	20	0%
L2 RGB	10	10	50%
L2 RGBD	**18**	**2**	**90%**

data, and keep planning parameters the same besides the initial CEM variance (see Appendix 10.3). We change the goal image to the triangular, folded shape shown in Figure 8 and change the initial state to a smooth state (which can be interpreted as the result of smoothing). The two sides of the fabric are shaded differently, with the darker shade on the bottom layer. Due to the action space bounds (Section 5.2), getting to this goal state directly is not possible in less than two actions and requires a precise sequence of pick-and-pull actions.

We visually inspect the final states in each episode, and classify them as successes or failures, as done in other work on fabric folding [40]. For RGBD images, this decision boundary empirically corresponds to an L2 threshold of about 8000; see Figure 8 for a typical success case. In Table 4 we compare performance of L2 cost taken over RGB, depth, and RGBD channels. RGBD significantly outperforms the other modes, which correspond to Visual Foresight and "Spatial Foresight" (depth only) respectively, suggesting the usefulness of augmenting Visual Foresight with depth maps.

6.3 Simulation Results from Variations of VSF Settings

This section contains newer results not in prior work [29]. Here, we study the choice of dataset, visual dynamics model, optimization method, and planning cost function on performance in simulation. We evaluate 20 trials of smoothing, folding ("1-Fold"),
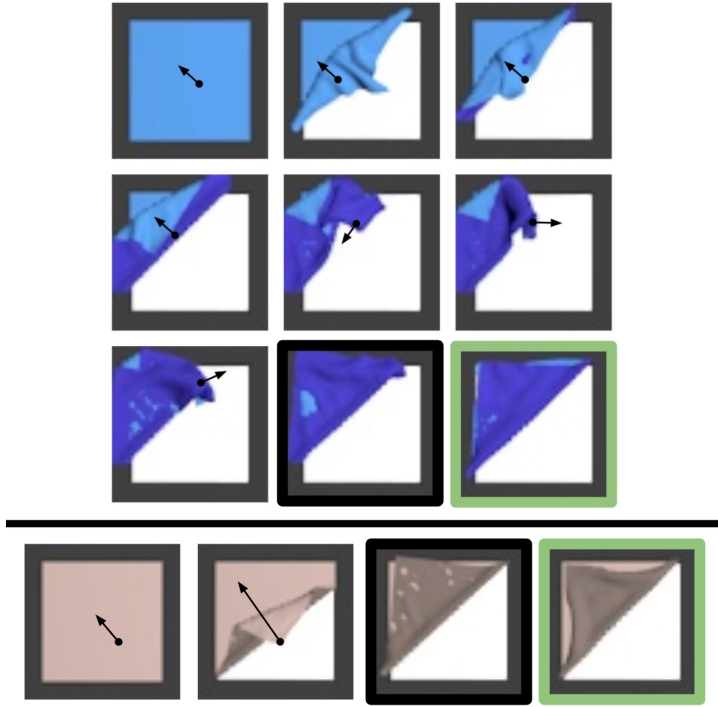
Fig. 8: RGB observations of successful folding episodes in simulation. **Top:** A rollout using VSF-1.0. The goal image is boxed in green, while the final frame in the episode is boxed in black. Here it takes 7 actions (left-to-right, top-to-bottom) from smooth to approximately folded. **Bottom:** A rollout using VisuoSpatial Foresight trained on the new dataset Fabric-CornerBias (Section 6.3). Here it only takes 2 actions and results in a higher quality fold. There are several areas of the fabric simulator which have overlapping layers due to the difficulty of accurately modeling fabric-fabric collisions in simulation, which explain the light patches in the figure.

and double folding ("2-Fold"), on each of 12 possible settings. See Figure 1 for examples of these goals. Note that "1-Fold" refers to the structure of the *goal image*, not the minimum number of actions it requires to reach (which is 2 with the current action space). Recall that VSF-1.0 in prior work [29] and the previous section represents 1 of these 12 settings (namely, Fabric-Random, SV2P, CEM, and Pixel L2) for VisuoSpatial Foresight. Note that we consider 12 settings instead of 16 because we did not record the fabric mesh state when generating Fabric-Random, which makes combinations of Fabric-Random and the learned cost function from Equation 4 impossible. See Table 5 for quantitative results, which contains all $3 \times 12 = 36$ combinations of choices for VSF. We find that no single combination achieves the best performance on all three tasks, suggesting tradeoffs in the selection of each component of VSF.

Table 5: Success rate (or mean coverage for smoothing) and number of actions (among successful episodes, or "N/A" if no successful episodes) of smoothing ("Smooth"), single folding ("1-Fold"), and double folding ("2-Fold") on all possible settings of dataset, visual dynamics model, optimization method, and planning cost (see Section 4). We run 20 trials for each row. All smoothing results are from Tier 3 starting states.

	Dataset	Model	Optimizer	Cost	Task	Success	# Actions
1	Fabric-Random	SV2P	CEM	Pixel L2	Smooth	86.5	10.7 ± 4.1
2	Fabric-Random	SV2P	CMA-ES	Pixel L2	Smooth	50.0	5.0 ± 4.0
3	Fabric-Random	SVG	CEM	Pixel L2	Smooth	71.3	10.8 ± 3.9
4	Fabric-Random	SVG	CMA-ES	Pixel L2	Smooth	41.0	5.2 ± 4.1
5	Fabric-CornerBias	SV2P	CEM	Pixel L2	Smooth	84.4	12.4 ± 3.9
6	Fabric-CornerBias	SV2P	CEM	Vertex L2	Smooth	**88.0**	10.9 ± 3.9
7	Fabric-CornerBias	SV2P	CMA-ES	Pixel L2	Smooth	44.2	6.4 ± 4.7
8	Fabric-CornerBias	SV2P	CMA-ES	Vertex L2	Smooth	48.3	6.8 ± 5.6
9	Fabric-CornerBias	SVG	CEM	Pixel L2	Smooth	67.0	12.3 ± 3.6
10	Fabric-CornerBias	SVG	CEM	Vertex L2	Smooth	72.8	10.1 ± 4.0
11	Fabric-CornerBias	SVG	CMA-ES	Pixel L2	Smooth	39.3	6.8 ± 4.7
12	Fabric-CornerBias	SVG	CMA-ES	Vertex L2	Smooth	44.3	6.7 ± 4.8
13	Fabric-Random	SV2P	CEM	Pixel L2	1-Fold	90	8.3 ± 1.2
14	Fabric-Random	SV2P	CMA-ES	Pixel L2	1-Fold	5	6.0 ± 0.0
15	Fabric-Random	SVG	CEM	Pixel L2	1-Fold	0	N/A
16	Fabric-Random	SVG	CMA-ES	Pixel L2	1-Fold	0	N/A
17	Fabric-CornerBias	SV2P	CEM	Pixel L2	1-Fold	**95**	2.0 ± 0.0
18	Fabric-CornerBias	SV2P	CEM	Vertex L2	1-Fold	90	2.1 ± 0.2
19	Fabric-CornerBias	SV2P	CMA-ES	Pixel L2	1-Fold	15	1.3 ± 0.5
20	Fabric-CornerBias	SV2P	CMA-ES	Vertex L2	1-Fold	10	3.0 ± 2.0
21	Fabric-CornerBias	SVG	CEM	Pixel L2	1-Fold	10	8.5 ± 2.1
22	Fabric-CornerBias	SVG	CEM	Vertex L2	1-Fold	10	2.5 ± 0.7
23	Fabric-CornerBias	SVG	CMA-ES	Pixel L2	1-Fold	0	N/A
24	Fabric-CornerBias	SVG	CMA-ES	Vertex L2	1-Fold	0	N/A
25	Fabric-Random	SV2P	CEM	Pixel L2	2-Fold	30	5.2 ± 1.7
26	Fabric-Random	SV2P	CMA-ES	Pixel L2	2-Fold	30	3.3 ± 0.9
27	Fabric-Random	SVG	CEM	Pixel L2	2-Fold	0	N/A
28	Fabric-Random	SVG	CMA-ES	Pixel L2	2-Fold	0	N/A
29	Fabric-CornerBias	SV2P	CEM	Pixel L2	2-Fold	10	7.5 ± 2.5
30	Fabric-CornerBias	SV2P	CEM	Vertex L2	2-Fold	10	5.5 ± 0.5
31	Fabric-CornerBias	SV2P	CMA-ES	Pixel L2	2-Fold	15	3.3 ± 1.3
32	Fabric-CornerBias	SV2P	CMA-ES	Vertex L2	2-Fold	**40**	2.4 ± 0.5
33	Fabric-CornerBias	SVG	CEM	Pixel L2	2-Fold	0	N/A
34	Fabric-CornerBias	SVG	CEM	Vertex L2	2-Fold	5	8.0 ± 0.0
35	Fabric-CornerBias	SVG	CMA-ES	Pixel L2	2-Fold	0	N/A
36	Fabric-CornerBias	SVG	CMA-ES	Vertex L2	2-Fold	0	N/A

6.3.1 Dataset Comparison

Keeping all VSF-1.0 settings constant besides dataset choice indicates that Fabric-CornerBias has no noticeable impact on smoothing performance (Row 5), improves folding performance (Row 17), and hurts double folding performance (Row 29). However, the best setting for double folding performance includes Fabric-CornerBias (Row 32). The most dramatic improvement from Fabric-CornerBias is in the efficiency of the folding rollouts. In particular, with SV2P, CEM, and Pixel L2, switching the dataset alone decreases the mean number of actions from 8.3 to 2.0 and yields

higher quality rollouts (see Figure 8). This increase in efficiency suggests that physical fabric folding will be more feasible, as the sim-to-real dynamics mismatch is not able to compound over time (Section 7.2).

6.3.2 Visual Dynamics Model Comparison

In all smoothing and folding experiments, results suggest that planning using the action-conditioned SVG video prediction model from Figure 2 leads to lower quality results compared to SV2P. The best smoothing result with SVG (row 10, 72.8% coverage, with CEM and Vertex L2 on Fabric-CornerBias) lags behind the best SV2P smoothing result (row 6, 88.0% coverage, also with CEM and Vertex L2 on Fabric-CornerBias). We hypothesize that the lower performance metrics with SVG as compared to SV2P may be partially explained from the model architectures plus the amount of data available. The architecture of SV2P [1], based on [21], involves predicting transformations of pixels which are constrained to avoid moving too much in predicted future images, whereas SVG [13] does not apply a similar constraint when predicting images. This may cause it to predict images of highly disjoint fabric, which we qualitatively observe in the image predictions during MPC. While the constraints imposed on SV2P may cause it to be less expressive than SVG given sufficient data, the data size of Fabric-CornerBias, containing 99,320 data transitions (see Section 5.2) may not be large enough to show benefits for SVG.

6.3.3 Optimization Method Comparison

In all smoothing and folding experiments involving CMA-ES, performance is far below that of CEM. However, CMA-ES improves performance and efficiency in double folding, especially among the experiments involving Fabric-CornerBias. To better understand why this is the case, we inspect VSF plans for single folding in Figure 9 and double folding in Figure 10. Despite the poor performance on folding, we find that CMA-ES actually arrives at a lower cost solution, indicating that CMA-ES may be exploiting inaccuracies in the visual dynamics model. The CMA-ES solution generally involves larger action deltas that can cause resulting states to deviate from the predictions, which may be due to the smaller population size during optimization. However, for the double-folding experiments (Figure 10), CEM is unable to find a high quality solution, while CMA-ES is able to find one. We hypothesize that this behavior is due to averaging over a multimodal optimization landscape with a large population size. Due to the structure of the double folding goal image, the order in which the top right corner and bottom left corner are folded toward the center does not impact the quality of the solution. Since we run CMA-ES with many fewer samples per iteration, it is less likely to reach both optima at the same time.

6.3.4 Cost Function Comparison

In smoothing and folding experiments, changing the cost function to the learned vertex distance estimator does not have significant impact on performance in either direction, though Vertex L2 does slightly boost coverage for smoothing. This is perhaps
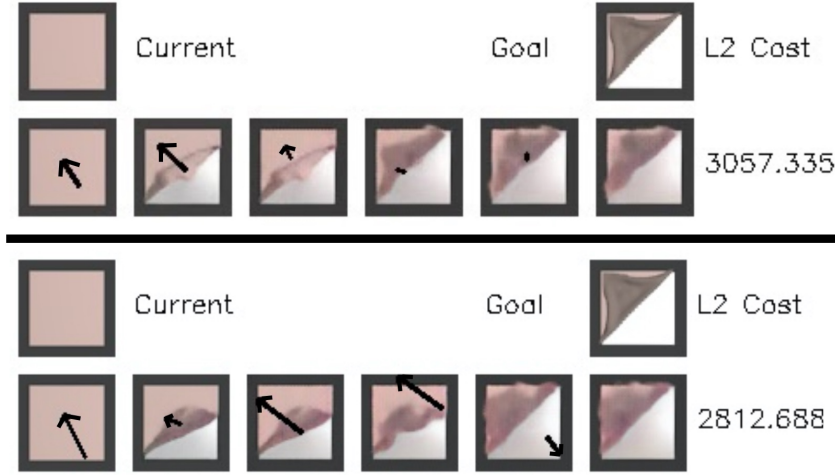
Fig. 9: **Top:** A VSF folding plan with Fabric-CornerBias, SV2P, Pixel L2 cost and the CEM optimizer. **Bottom:** A VSF folding plan with Fabric-CornerBias, SV2P, Pixel L2 cost and the CMA-ES optimizer. As in Figure 3, the five images after the current image are generated by the visual dynamics model. CMA-ES arrives at a lower cost solution but converges on more drastic actions that are more liable to result in states that deviate from predictions, and may cause the fabric to go out of bounds.

unsurprising, as Pixel L2 is likely sufficient for goal images with simple visual structure (i.e., a square or triangle with a single color). However, with the more complex double folding goal image, comparison of Rows 31 and 32 (where dataset is Fabric-CornerBias, visual dynamics model is SV2P, and optimizer is CMA-ES) indicates that the learned Vertex L2 significantly outperforms Pixel L2. In Figure 10 we see that minimizing the Vertex L2 cost appropriately guides CMA-ES to a trajectory with a final predicted image similar to the double folding goal image.

7 Physical Experiments

We evaluate VisuoSpatial Foresight on a physical da Vinci surgical robot [33]. We use the same experimental setup as in Seita et al. [67], including the calibration procedure to map pick points (x, y) into positions and orientations with respect to the robot's base frame. The sequential tasks we consider are challenging due to the robot's imprecision [65]. We use a Zivid One Plus camera mounted 0.9 meters above the workspace to capture RGBD images. We manipulate a 5" by 5" piece of fabric (blue for smoothing, brown for folding) and apply some damping to mitigate stiffness due to its small size. In Section 7.1, we report results from our prior work [29]. In Section 7.2, we show new results with physical fabric folding using a set of parameters which we refer to as "VSF-2.0."
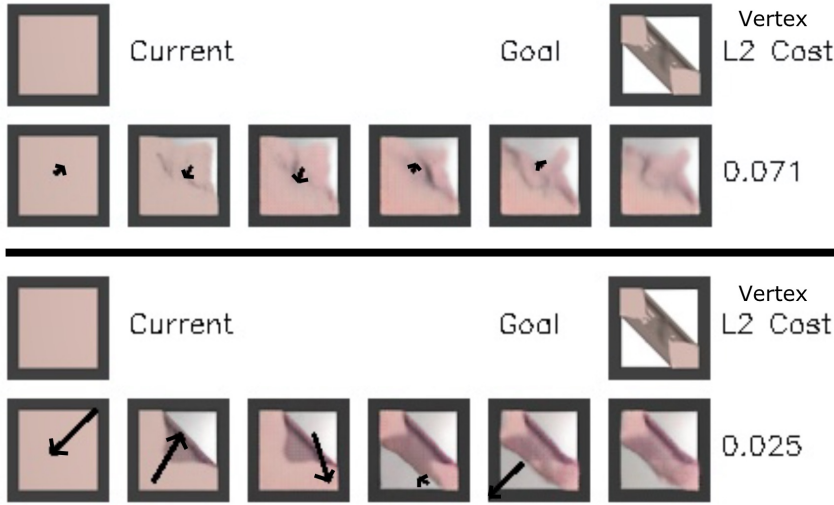
Fig. 10: **Top:** A VSF double folding plan with Fabric-CornerBias, SV2P, Vertex L2 cost and the CEM optimizer. **Bottom:** A VSF double folding plan with Fabric-CornerBias, SV2P, Vertex L2 cost and the CMA-ES optimizer. CMA-ES is able to find a much better plan for achieving the goal state.

Table 6: Physical smoothing robot experiment results for Imitation Learning (IL), i.e. DAgger, and VSF-1.0. For both methods, we choose the policy snapshot with highest performance in simulation, and each are applied on all tiers (T1, T2, T3). We show results across 10 episodes of IL per tier and 5 episodes of VSF-1.0 per tier, and show average starting and final coverage, maximum coverage at any point per episode, and the number of actions. Results suggest that VSF-1.0 attains final coverage comparable to or exceeding that of IL despite not being trained on demonstration data, though VSF-1.0 requires more actions per episode.

(Tier) Method	(1) Start	(2) Final	(3) Max	(4) Actions
(1) IL	74.2 ± 5	92.1 ± 6	92.9 ± 3	4.0 ± 3
(1) VSF-1.0	78.3 ± 6	$\mathbf{93.4} \pm \mathbf{2}$	93.4 ± 2	8.2 ± 4
(2) IL	58.2 ± 3	84.2 ± 18	86.8 ± 15	9.8 ± 5
(2) VSF-1.0	59.5 ± 3	$\mathbf{87.1} \pm \mathbf{9}$	90.0 ± 5	12.8 ± 3
(3) IL	43.3 ± 4	75.2 ± 18	79.1 ± 14	12.5 ± 4
(3) VSF-1.0	41.4 ± 3	$\mathbf{75.6} \pm \mathbf{15}$	76.9 ± 15	15.0 ± 0

7.1 Physical Fabric Smoothing

Results in this section are from our prior work [29]. We evaluate the Imitation Learning and VSF-1.0 policies from Section 6.2. We do not test with the model-free DDPG policy baseline, as it performed significantly worse than the other two methods. For IL, this is the final model trained with 110,000 actions based on a corner-pulling
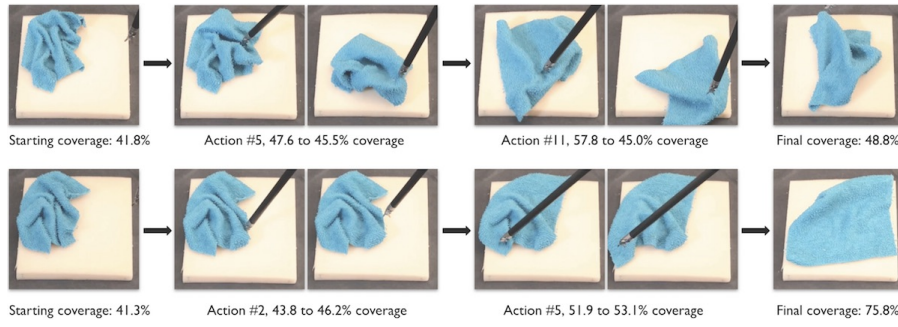
Fig. 11: A qualitative comparison of physical da Vinci episodes with an Imitation Learning policy (top row) and a VisuoSpatial Foresight policy (bottom row) from our prior work [29] using VSF-1.0 settings. The rows show screen captures taken from the third-person video view for recording episodes; these are not the input to VisuoSpatial Foresight. To facilitate comparisons among IL and VSF-1.0, we manually make the starting fabric state as similar as possible. Over the course of several actions, the IL policy sometimes takes actions that are highly counter-productive, such as the 5th and 11th actions above. Both pick points are reasonably chosen, but the large deltas cause the lower right fabric corner to get hidden. In contrast, VSF-1.0 takes shorter pulls on average, with representative examples shown above for the 2nd and 5th actions. At the end, the IL policy gets just 48.8% coverage (far below its usual performance), whereas VSF-1.0 gets 75.8%. See Table 6 for more results.

demonstrator with access to state information. This uses slightly more than the 105,045 actions used for training VSF-1.0. To match the simulation setup, we limit each episode to a maximum of 15 actions. For both methods, we initialize the fabric in highly rumpled states which mirror those from the simulated tiers. We run ten episodes per tier for IL and five episodes per tier for VSF-1.0, for 45 episodes in all. Within each tier, we attempt to make starting fabric states reasonably comparable among IL and VSF-1.0 episodes (see Figure 11). We present quantitative results in Table 6 that suggest that VSF-1.0 gets final coverage results comparable to that of IL, despite not being trained on any corner-pulling demonstration data. However, it sometimes requires more actions to complete an episode and takes significantly more time to plan an action (on the order of 20 more seconds per action), since the Cross Entropy Method requires thousands of forward passes through a deep neural network while IL requires only a single pass.

As an example, Figure 11 shows a time lapse of a subset of actions for one episode from IL and VSF-1.0. Both begin with a fabric of roughly the same shape to facilitate comparisons. On the fifth action, the IL policy has a pick point that is slightly north of the ideal spot. The pull direction to the lower right fabric plane corner is reasonable, but due to the pull length, combined with a slightly suboptimal pick point, the lower right fabric corner gets covered. This makes it harder for a policy trained from a corner-pulling demonstrator to get high coverage. In contrast, the VSF-1.0 policy takes actions of shorter magnitudes and does not fall into this trap.
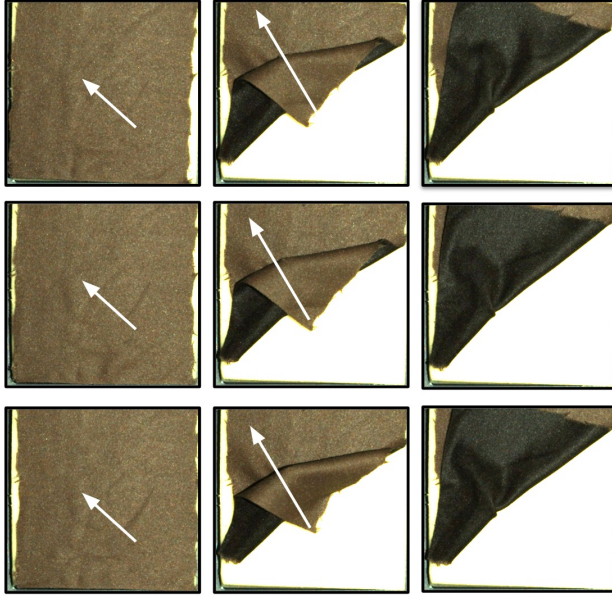
Fig. 12: We evaluate the VSF-2.0 policy on a fabric folding task on the physical surgical robotic system with Fabric-CornerBias, SV2P, CEM, and Pixel L2 cost. Each of the three rows depicts top-down RGB images of one of the 9 out of 10 successful 2-step rollouts with a square piece of brown fabric.

7.2 Physical Fabric Folding

We next evaluate VisuoSpatial Foresight on a fabric folding task, starting from a smooth state. In our prior work [29], we were unable to successfully perform folding on the physical system with VSF-1.0. When comparing the real fabric with simulated fabric, we found a gap between the physics of our simulator and that of the real fabric, as fabric dynamics are notoriously difficult to model [54]. Unlike smoothing, folding can require more nuanced actions such as reversing the surface normals of the fabric. Since VSF-1.0 took 8.3 actions on average to fold the fabric in simulation, in real episodes this dynamics gap compounded over time and was exacerbated by the imprecision of cable-driven robots like the dVRK [65]. In this work, we find that training visual dynamics on Fabric-CornerBias makes it possible for VisuoSpatial Foresight to successfully fold fabric in simulation with just 2 actions in 19 out of the 20 successful trials (Row 17 in Table 5), which may be short enough to prevent the dynamics gap from building to irrecoverable levels. We refer to this new set of VSF parameters using Fabric-CornerBias, SV2P, CEM, and Pixel L2 as "VSF-2.0."

To test this hypothesis, we perform a VSF-2.0 plan on the physical system in an open-loop fashion. Such an approach is viable only if it is possible to register the initial fabric state into simulation; in the fabric folding case, this is trivial, as the fabric starts fully smooth. To correct for near misses, the system moves the pick point to the nearest point on the fabric, which it computes by color masking the real RGB

observations. In 9 of 10 trials, the robot successfully folds using two actions, with the only failure case due to picking multiple layers of the fabric when intending to pick the top layer. See Figure 12 for 3 of these trials and the project website for videos.

8 Conclusion and Future Work

Our prior work [29] presented VisuoSpatial Foresight, which leverages a combination of RGB and depth information to learn goal conditioned fabric manipulation policies for a variety of sequential tasks. In [29], we train a video prediction model on purely random interaction data with fabric in simulation, and demonstrate that planning over this model with MPC results in a policy that achieves 90% success rate for fabric smoothing and folding tasks.

In this work, we investigate new alternatives to the four core aspects of VisuoSpatial Foresight: data generation, visuospatial dynamics model, cost function, and optimization procedure. To improve data, we introduce Fabric-CornerBias as a new dataset with longer action pull vector magnitudes and a bias towards picking at corners. We propose and test an action-conditioned version of SVG for modeling visuospatial dynamics. To optimize the MPC objective during planning, we test Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Finally, we train a learned cost function as an alternative to L2 pixel differences in images. Smoothing and folding results in simulation suggest that the new data, Fabric-CornerBias, is the most promising route to improving results. Using this new data for VisuoSpatial Foresight allows us to learn more accurate visual dynamics models because the dataset contains actions that are more broadly relevant for fabric manipulation. These actions are biased towards picking fabric corners and have magnitudes more reflective of the actions required to do fabric manipulation tasks such as smoothing and folding. The resulting improvement in efficiency led to successful fabric folding on the physical robotic system in 9 out of 10 trials, while in our prior work [29] we were unable to successfully fold fabric in physical trials.

In light of these results, future work will attempt to understand the effect of the distribution and magnitude of the dataset used to train visual dynamics models on VisuoSpatial Foresight. We plan to generate orders of magnitude more data, and will benchmark performance as a function of data size and other properties. In addition, we will test VisuoSpatial Foresight on different fabric shapes, and investigate ways to incorporate bilateral manipulation or human-in-the-loop policies.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Babaeizadeh M, Finn C, Erhan D, Campbell RH, Levine S (2018) Stochastic Variational Video Prediction. In: International Conference on Learning Representations (ICLR)
2. Balaguer B, Carpin S (2011) Combining Imitation and Reinforcement Learning to Fold Deformable Planar Objects. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
3. Balakrishna A, Thananjeyan B, Lee J, Zahed A, Li F, Gonzalez JE, Goldberg K (2019) On-Policy Robot Imitation Learning from a Converging Supervisor. In: Conference on Robot Learning (CoRL)
4. Baraff D, Witkin A (1998) Large Steps in Cloth Simulation. In: ACM SIG-GRAPH
5. Berkenkamp F, Schoellig AP, Krause A (2016) Safe Controller Optimization for Quadrotors with Gaussian Processes. In: IEEE International Conference on Robotics and Automation (ICRA)
6. Borras J, Alenya G, Torras C (2019) A Grasping-centered Analysis for Cloth Manipulation. arXiv preprint arXiv:190608202
7. Chen D, Zhou B, Koltun V, Krahenbuhl P (2019) Learning by Cheating. In: Conference on Robot Learning (CoRL)
8. Chiuso A, Pillonetto G (2019) System Identification: A Machine Learning Perspective. Annual Review of Control, Robotics, and Autonomous Systems
9. Chua K, Calandra R, McAllister R, Levine S (2018) Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models. In: Neural Information Processing Systems (NeurIPS)
10. Community BO (2018) Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, URL `http://www.blender.org`
11. Coumans E, Bai Y (2016–2019) Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`
12. Dasari S, Ebert F, Tian S, Nair S, Bucher B, Schmeckpeper K, Singh S, Levine S, Finn C (2019) RoboNet: Large-Scale Multi-Robot Learning. In: Conference on Robot Learning (CoRL)
13. Denton E, Fergus R (2018) Stochastic Video Generation with a Learned Prior. In: International Conference on Machine Learning (ICML)
14. Dhariwal P, Hesse C, Klimov O, Nichol A, Plappert M, Radford A, Schulman J, Sidor S, Wu Y, Zhokhov P (2017) OpenAI Baselines. `https://github.com/openai/baselines`
15. Doumanoglou A, Kargakos A, Kim TK, Malassiotis S (2014) Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and

Probabilistic Planning. In: IEEE International Conference on Robotics and Automation (ICRA)

16. Ebert F, Finn C, Lee AX, Levine S (2017) Self-Supervised Visual Planning with Temporal Skip Connections. In: Conference on Robot Learning (CoRL)

17. Ebert F, Finn C, Dasari S, Xie A, Lee A, Levine S (2018) Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. arXiv preprint arXiv:181200568

18. Erickson Z, Clever HM, Turk G, Liu CK, Kemp CC (2018) Deep Haptic Model Predictive Control for Robot-Assisted Dressing. In: IEEE International Conference on Robotics and Automation (ICRA)

19. Erickson Z, Gangaram V, Kapusta A, Liu CK, Kemp CC (2020) Assistive Gym: A Physics Simulation Framework for Assistive Robotics. In: IEEE International Conference on Robotics and Automation (ICRA)

20. Finn C, Levine S (2017) Deep Visual Foresight for Planning Robot Motion. In: IEEE International Conference on Robotics and Automation (ICRA)

21. Finn C, Goodfellow I, Levine S (2016) Unsupervised Learning for Physical Interaction through Video Prediction. In: Neural Information Processing Systems (NeurIPS)

22. Ganapathi A, Sundaresan P, Thananjeyan B, Balakrishna A, Seita D, Hoque R, Gonzalez JE, Goldberg K (2020) Mmgsd: Multi-modal gaussian shape descriptors for correspondence matching in 1d and 2d deformable objects. In: International Conference on Intelligent Robots and Systems (IROS) Workshop on Managing Deformation, IEEE

23. Ganapathi A, Sundaresan P, Thananjeyan B, Balakrishna A, Seita D, Grannen J, Hwang M, Hoque R, Gonzalez JE, Jamali N, Yamane K, Iba S, Goldberg K (2021) Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics. In: IEEE International Conference on Robotics and Automation (ICRA)

24. Gao Y, Chang HJ, Demiris Y (2016) Iterative Path Optimisation for Personalised Dressing Assistance using Vision and Force Information. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

25. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, `http://www.deeplearningbook.org`

26. Hansen N, Auger A (2011) CMA-ES: Evolution Strategies and Covariance Matrix Adaptation. Association for Computing Machinery, New York, NY, USA

27. Hewing L, Liniger A, Zeilinger M (2018) Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars. In: European Controls Conference (ECC)

28. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Computation 9

29. Hoque R, Seita D, Balakrishna A, Ganapathi A, Tanwani AK, Jamali N, Yamane K, Iba S, Goldberg K (2020) VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In: Robotics: Science and Systems (RSS)

30. Jangir R, Alenya G, Torras C (2020) Dynamic Cloth Manipulation with Deep Reinforcement Learning. In: IEEE International Conference on Robotics and Automation (ICRA)

31. Jia B, Hu Z, Pan J, Manocha D (2018) Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary. In: IEEE International Conference on Robotics and Automation (ICRA)

32. Jia B, Pan Z, Hu Z, Pan J, Manocha D (2019) Cloth Manipulation Using Random-Forest-Based Imitation Learning. In: IEEE International Conference on Robotics and Automation (ICRA)

33. Kazanzides P, Chen Z, Deguet A, Fischer G, Taylor R, DiMaio S (2014) An Open-Source Research Kit for the da Vinci Surgical System. In: IEEE International Conference on Robotics and Automation (ICRA)

34. Kingma DP, Ba J (2015) Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR)

35. Kita Y, Ueshiba T, Neo ES, Kita N (2009) A Method For Handling a Specific Part of Clothing by Dual Arms. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

36. Kita Y, Ueshiba T, Neo ES, Kita N (2009) Clothes State Recognition Using 3D Observed Data. In: IEEE International Conference on Robotics and Automation (ICRA)

37. Kocijan J, Murray-Smith R, Rasmussen C, Girard A (2004) Gaussian Process Model Based Predictive Control. In: American Control Conference (ACC)

38. Kumar M, Babaeizadeh M, Erhan D, Finn C, Levine S, Dinh L, Kingma D (2020) VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation. In: International Conference on Learning Representations (ICLR)

39. Lee AX, Zhang R, Ebert F, Abbeel P, Finn C, Levine S (2018) Stochastic Adversarial Video Prediction. arXiv preprint arXiv:180401523

40. Lee R, Ward D, Cosgun A, Dasagi V, Corke P, Leitner J (2020) Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience. In: Conference on Robot Learning (CoRL)

41. Li Y, Yue Y, Grinspun DXE, Allen PK (2015) Folding Deformable Objects using Predictive Simulation and Trajectory Optimization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

42. Li Y, Hu X, Xu D, Yue Y, Grinspun E, Allen PK (2016) Multi-Sensor Surface Analysis for Robotic Ironing. In: IEEE International Conference on Robotics and Automation (ICRA)

43. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2016) Continuous Control with Deep Reinforcement Learning. In: International Conference on Learning Representations (ICLR)

44. Lin X, Wang Y, Olkin J, Held D (2020) SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. In: Conference on Robot Learning (CoRL)

45. Lippi M, Poklukar P, Welle MC, Varava A, Yin H, Marino A, Kragic D (2020) Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

46. Maitin-Shepard J, Cusumano-Towner M, Lei J, Abbeel P (2010) Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding. In: IEEE International Conference on Robotics and Au-

tomation (ICRA)

47. Mann H, Whitney D (1947) On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other. Annals of Mathematical Statistics

48. Matas J, James S, Davison AJ (2018) Sim-to-Real Reinforcement Learning for Deformable Object Manipulation. Conference on Robot Learning (CoRL)

49. Miller S, van den Berg J, Fritz M, Darrell T, Goldberg K, Abbeel P (2012) A Geometric Approach to Robotic Laundry Folding. In: International Journal of Robotics Research (IJRR)

50. Nagabandi A, Kahn G, Fearing R, Levine S (2018) Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In: IEEE International Conference on Robotics and Automation (ICRA)

51. Nair A, McGrew B, Andrychowicz M, Zaremba W, Abbeel P (2018) Overcoming Exploration in Reinforcement Learning with Demonstrations. In: IEEE International Conference on Robotics and Automation (ICRA)

52. Nair S, Finn C (2020) Goal-Aware Prediction: Learning to Model What Matters. In: International Conference on Machine Learning (ICML)

53. Nair S, Babaeizadeh M, Finn C, Levine S, Kumar V (2020) Time Reversal as Self-Supervision. In: IEEE International Conference on Robotics and Automation (ICRA)

54. Narain R, Samii A, O'Brien JF (2012) Adaptive Anisotropic Remeshing for Cloth Simulation. In: ACM SIGGRAPH Asia

55. OpenAI, Andrychowicz M, Baker B, Chociej M, Jozefowicz R, McGrew B, Pachocki J, Petron A, Plappert M, Powell G, Ray A, Schneider J, Sidor S, Tobin J, Welinder P, Weng L, Zaremba W (2019) Learning Dexterous In-Hand Manipulation. In: International Journal of Robotics Research (IJRR)

56. Osawa F, Seki H, Kamiya Y (2007) Unfolding of Massive Laundry and Classification Types by Dual Manipulator. Journal of Advanced Computational Intelligence and Intelligent Informatics 11(5)

57. Pomerleau DA (1991) Efficient Training of Artificial Neural Networks for Autonomous Navigation. Neural Comput 3

58. Provot X (1995) Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In: Graphics Interface

59. Radford A, Metz L, Chintala S (2016) Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: International Conference on Learning Representations (ICLR)

60. Rosolia U, Borrelli F (2019) Learning how to Autonomously Race a Car: a Predictive Control Approach. In: IEEE Transactions on Control Systems Technology

61. Ross S, Gordon GJ, Bagnell JA (2011) A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS)

62. Rubinstein R (1999) The Cross-Entropy Method for Combinatorial and Continuous Optimization. Methodology And Computing In Applied Probability

63. Sanchez J, Corrales JA, Bouzgarrou BC, Mezouar Y (2018) Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey. In: International Journal of Robotics Research (IJRR)

64. Schrimpf J, Wetterwald LE (2012) Experiments Towards Automated Sewing With a Multi-Robot System. In: IEEE International Conference on Robotics and Automation (ICRA)

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

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

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

68. Seita D, Florence P, Tompson J, Coumans E, Sindhwani V, Goldberg K, Zeng A (2021) Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In: IEEE International Conference on Robotics and Automation (ICRA)

69. Shibata S, Yoshimi T, Mizukawa M, Ando Y (2012) A Trajectory Generation of Cloth Object Folding Motion Toward Realization of Housekeeping Robot. In: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)

70. Shin C, Ferguson PW, Pedram SA, Ma J, Dutson EP, Rosen J (2019) Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control. In: IEEE International Conference on Robotics and Automation (ICRA)

71. Sun L, Aragon-Camarasa G, Cockshott P, Rogers S, Siebert JP (2014) A Heuristic-Based Approach for Flattening Wrinkled Clothes. Towards Autonomous Robotic Systems TAROS 2013 Lecture Notes in Computer Science, vol 8069

72. Sun L, Aragon-Camarasa G, Rogers S, Siebert JP (2015) Accurate Garment Surface Analysis using an Active Stereo Robot Head with Application to Dual-Arm Flattening. In: IEEE International Conference on Robotics and Automation (ICRA)

73. Thananjeyan B, Garg A, Krishnan S, Chen C, Miller L, Goldberg K (2017) Multilateral Surgical Pattern Cutting in 2D Orthotropic Gauze with Deep Reinforcement Learning Policies for Tensioning. In: IEEE International Conference on Robotics and Automation (ICRA)

74. Thananjeyan* B, Balakrishna* A, Nair S, Luo M, Srinivasan K, Hwang M, Gonzalez JE, Ibarz J, Finn C, Goldberg K (2020) Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones. In: NeurIPS Robot Learning Workshop, NeurIPS

75. Thananjeyan B, Balakrishna A, Rosolia U, Li F, McAllister R, Gonzalez JE, Levine S, Borrelli F, Goldberg K (2020) Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks. In: IEEE Robotics and Automation Letters (RA-L)

76. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

77. Todorov E, Erez T, Tassa Y (2012) MuJoCo: A Physics Engine for Model-Based Control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

78. Torgerson E, Paul F (1987) Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing. In: IEEE International Conference on Robotics and Automation (ICRA)

79. Vaswani A, Bengio S, Brevdo E, Chollet F, Gomez AN, Gouws S, Jones L, Kaiser L, Kalchbrenner N, Parmar N, Sepassi R, Shazeer N, Uszkoreit J (2018) Tensor2tensor for neural machine translation. CoRR abs/1803.07416, URL http://arxiv.org/abs/1803.07416

80. Vecerik M, Hester T, Scholz J, Wang F, Pietquin O, Piot B, Heess N, Rothörl T, Lampe T, Riedmiller M (2017) Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. arXiv preprint arXiv:170708817

81. Verlet L (1967) Computer Experiments on Classical Fluids: I. Theormodynamical Properties of Lennard-Jones Molecules. Physics Review 159(98)

82. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. Trans Img Proc

83. Willimon B, Birchfield S, Walker I (2011) Model for Unfolding Laundry using Interactive Perception. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

84. Wu Y, Yan W, Kurutach T, Pinto L, Abbeel P (2020) Learning to Manipulate Deformable Objects without Demonstrations. In: Robotics: Science and Systems (RSS)

85. Xie A, Singh A, Levine S, Finn C (2018) Few-Shot Goal Inference for Visuomotor Learning and Planning. In: Conference on Robot Learning (CoRL)

86. Yan W, Vangipuram A, Abbeel P, Pinto L (2020) Learning Predictive Representations for Deformable Objects Using Contrastive Estimation. In: Conference on Robot Learning (CoRL)

87. Yang PC, Sasaki K, Suzuki K, Kase K, Sugano S, Ogata T (2017) Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning. In: IEEE Robotics and Automation Letters (RA-L)

We structure this Appendix as follows:

- Appendix 9 compares and contrasts various fabric simulators.
- Appendix 10 lists hyperparameters and provides details for training policies.
- Appendix 11 provides more details on the smoothing experiments.

9 Fabric Simulators

As in the prior paper [29], we use the fabric simulator originally developed in Seita et al. [67]. This simulator possesses an ideal balance between ease of code implementation, speed, and accuracy, and was able to lead to reasonable smoothing policies in prior work. We considered using simulators from ARCSim [54], MuJoCo [77], PyBullet [11], Blender [10], or NVIDIA FLeX [44], but did not use them for several reasons outlined below.

High-fidelity simulators, such as ARCSim, take too long to simulate to get sufficient data for training visual dynamics models. Furthermore, it is difficult to simulate rudimentary grasping behavior in ARCSim because it does not represent fabric as a fixed grid of vertices, which means grasping cannot be simulated by pinning vertices.

Blender includes a new fabric simulator, with substantial improvements after 2017 for more realistic shearing and tensioning. These changes, however, are only supported in Blender 2.8, not Blender 2.79, and we used 2.79 because Blender 2.8 does not allow background processes to run on headless servers, which prevented us from running mass data collection. Additionally, Blender does not allow the dynamic re-grasping of mesh vertices during simulation which makes long horizon cloth manipulation and data collection difficult.

MuJoCo is a widely utilized physics simulator for deep reinforcement learning benchmarks [77]. The first MuJoCo version providing full support for fabric manipulation was released in October 2018. Currently, the only work that integrates the fabric simulator with simulated robot grasps is from Wu et al. [84], which was developed concurrently with the prior work [29]. Upon investigating the open-source code, we found that MuJoCo's fabric simulator did not handle fabric self-collisions better than the simulator from [67], and hence did not pursue it further.

The PyBullet simulator code from Matas et al. [48] showed relatively successful fabric simulation, but it was difficult for us to adapt the author's code to the proposed work, which made significant changes to the off-the-shelf PyBullet code. PyBullet's fabric simulator was upgraded and tested for more fabric-related tasks in Seita et al. [68], but still suffers from self-collisions and fabric which tends to get crumpled.

In concurrent work, SoftGym [44] benchmarks deep reinforcement learning algorithms on deformable object manipulation tasks, including those with fabrics. SoftGym provides fabric simulation environments utilizing NVIDIA FLeX, which models deformable objects in a particle and position based dynamical system similar to the mass-spring system used in the fabric simulator from [29, 67] and also incorporates self-collision handling. SoftGym is concurrent work, and future work will investigate the feasibility of utilizing Flex. Additionally, we will compare the performance of the model-based policies presented in this work to the model-free policies evaluated in [44] on similar smoothing and folding tasks.
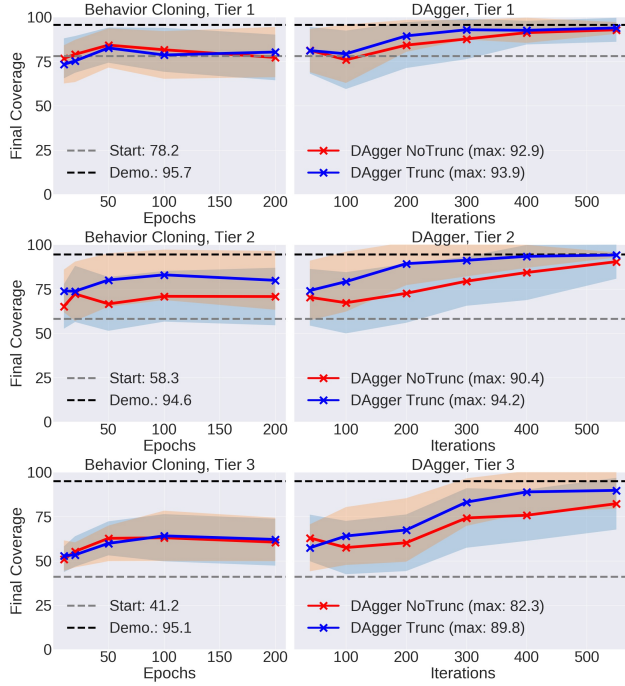
Fig. 13: Average coverage over 50 simulated test-time episodes at checkpoints (marked "X") during the behavior cloning and DAgger phases. For each setting of no action truncation and action truncation, we deploy a single DAgger policy deployed on all tiers. Using dashed lines, we annotate the average starting coverage and the corner pulling demonstrator's average final coverage.

10 Details of Learning-Based Methods

We describe implementation and training details of the three learning-based methods tested: imitation learning, model-free reinforcement learning, and model-based VisuoSpatial Foresight. The other baselines tested — random, highest point, and wrinkles — are borrowed unmodified from prior open-source code [67].

10.1 Imitation Learning Baseline: DAgger

This section contains details and results from our prior work [29]. DAgger [61] is implemented directly from the open source DAgger code in Seita et al. [67]. This was originally based on the open-source OpenAI baselines [14] library for parallel environment support to overcome the time bottleneck of fabric simulation.

We ran the corner pulling demonstrator for 2,000 trajectories, resulting in 6,697 image-action pairs $(\mathbf{o}_t, \mathbf{a}'_t)$, where the notation $\mathbf{a}'_t$ indicates the action is labeled and comes from the demonstrator. Each trajectory was randomly drawn from one of

the three tiers in the simulator with equal probability. We then perform a behavior cloning [57] "pre-training" period for 200 epochs over this offline data, which does not require environment interaction.

After behavior cloning, each DAgger iteration rolls out 20 parallel environments for 10 steps each (hence, 200 total new samples) which are labeled by the corner pulling policy, the same policy that created the offline data and uses underlying state information. These are inserted into a replay buffer of image-action samples, where all samples have actions labeled by the demonstrator. The replay buffer size is 50,000, but the original demonstrator data of size 6,697 is never removed from it. After environment stepping, we draw 240 minibatches of size 128 each for training and use Adam [34] for optimization. The process repeats with the agent rolling out its updated policy. We run DAgger for 110,000 steps across all environments (hence, 5,500 steps per parallel environment) to make the number of actions consumed to be roughly the same as the number of actions used to train the video prediction model. This is significantly more than the 50,000 DAgger training steps in prior work [67]. Table 7 contains additional hyperparameters.

The actor (i.e., policy) neural network for DAgger uses a design based on Seita et al. [67] and Matas et al. [48]. The input to the policy are RGBD images of size $(56 \times 56 \times 4)$, where the four channels are formed from stacking an RGB and a single-channel depth image. The policy processes the input through four convolutional layers that have 32 filters with size 3×3, and then uses four fully connected layers with 256 nodes each. The actor network has 0.8 million parameters.

The result from the actor policy is a 4D vector representing the action choice $\mathbf{a}_t \in \mathbb{R}^4$ at each time step t. The last layer is a hyperbolic tangent which makes each of the four components of $\mathbf{a}_t$ within $[-1, 1]$. During action truncation, we further limit the two components of $\mathbf{a}_t$ corresponding to the deltas into $[-0.4, 0.4]$.

A set of graphs representing learning progress for DAgger is shown in Figure 13, where for each marked snapshot, we roll it out in the environment for 50 episodes and measure final coverage. Results suggest the single DAgger policy, when trained with 110,000 total steps on RGBD images, performs well on all three tiers with performance nearly matching the 95-96% coverage of the demonstrator.

We trained two variants of DAgger, one with and one without the action truncation to $[-0.4, 0.4]$ for the two deltas Δx and Δy. The model trained on truncated actions outperforms the alternative setting, and it is also the setting used in VSF-1.0, hence we use it for physical robot experiments. We choose the final snapshot as it has the highest test-time performance, and we use it as the policy for simulated and real benchmarks in the main part of the paper.

10.2 Model-Free Reinforcement Learning Baseline: DDPG

This section contains details and results from our prior work [29]. To provide a second competitive baseline, we apply model-free reinforcement learning. Specifically, we use a variant of Deep Deterministic Policy Gradients (DDPG) [43] with several improvements as proposed in the research literature. Briefly, DDPG is a deep reinforcement learning algorithm which trains parameterized actor and critic models,

Table 7: DAgger hyperparameters.

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

Table 8: DDPG hyperparameters.

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

each of which are normally neural networks. The actor is the policy, and the critic is a value function.

First, as with DAgger, we use demonstrations [80] to improve the performance of the learned policy. We use the same demonstrator data of 6,697 samples from DAgger, except this time each sample is a tuple of $(\mathbf{o}_t, \mathbf{a}'_t, r_t, \mathbf{o}_{t+1})$, including a scalar reward r_t (to be described) and a successor state $\mathbf{o}_{t+1}$. This data is added to the replay buffer and never removed. We use a pre-training phase (of 200 epochs) to initialize the actor and critic. We also apply L_2 regularization for both the actor and critic networks. In addition, we use the Q-filter from Nair et al. [51] which may help the actor learn better actions than the demonstrator provides, perhaps for cases when naive corner pulling might not be ideal. For a fairer comparison, the actor network for DDPG uses the same architecture as the actor for DAgger. The critic has a similar architecture as the actor, with the only change that the action input $\mathbf{a}_t$ is inserted and concatenated with the features of the image $\mathbf{o}_t$ after the four convolutional layers, and
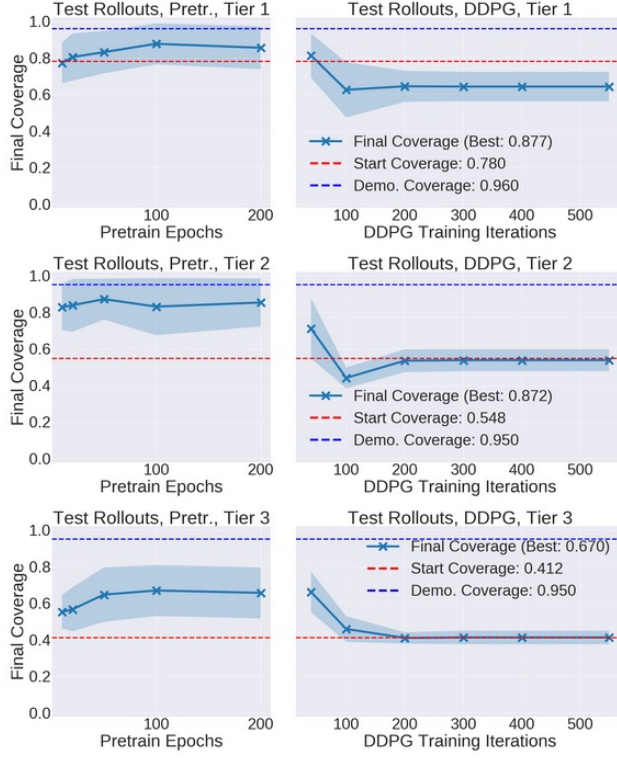
Fig. 14: Average coverage over 50 simulated test-time episodes at checkpoints (marked "X") during the pre-training DDPG phase, and the DDPG phase with agent exploration. This is presented in a similar manner as in Figure 13 for DAgger. Results suggest that DDPG has difficulty in training a policy that can achieve high coverage.

before the fully connected portion. As with the imitation learning baseline, the inputs are RGBD images of size $(56 \times 56 \times 4)$.

We design a dense reward to encourage the agent to achieve high coverage. At each time, the agent gets reward based on:

- A small negative living reward of -0.05
- A small negative reward of -0.05 for failing to grasp any point on the fabric (i.e., a wasted grasp attempt).
- A delta in coverage based on the change in coverage from the current state and the prior state.
- A +5 bonus for triggering 92% coverage.
- A -5 penalty for triggering an out-of-bounds condition, where the fabric significantly exceeds the boundaries of the underlying fabric plane.

We designed the reward function by informal tuning and borrowing ideas from the reward in OpenAI et al. [55], which used a delta in joint angles and a similar bonus for moving a block towards a target, or a penalty for dropping it. Intuitively,

Table 9: Visual MPC hyperparameters for CEM and CMA-ES. The notation $(\cdot) \times H$ indicates tiling the preceding array $H = 5$ times to fill the planning horizon.

Hyperparameter	Value
Number of CEM iterations	10
CEM population size	2000
CEM α	0.1
CEM planning horizon	5
CEM initial mean μ	$(0, 0, 0, 0) \times H$
CEM initial variance Σ	$(0.25, 0.25, 0.08, 0.08) \times H$
Number of CMA-ES iterations	250
CMA-ES population size	12
CMA-ES initial mean μ	$(0, 0, 0, 0) \times H$
CMA-ES initial variance Σ	$(0.25, 0.25, 0.25, 0.25) \times H$

an agent may learn to take a slightly counter-productive action which would decrease coverage (and thus the delta reward component is negative), but which may enable an easier subsequent action to trigger a high bonus. This reward design is only suited for smoothing. As with the imitation learning baseline, the model-free DDPG baseline is not designed for non-smoothing tasks.

Figure 14 suggests that the pre-training phase, where the actor and critic are trained on the demonstrator data, helps increase coverage. The DDPG portion of training, however, results in performance collapse to achieving no net coverage. Upon further inspection, this is because the actions collapsed to having no "deltas," so the robot reduces to picking up but then immediately releasing the fabric. Due to the weak performance of DDPG, we do not benchmark the policy on the physical robot.

10.3 VisuoSpatial Foresight

The main technique considered in this paper and our prior work [29] is VisuoSpatial Foresight (VSF), an extension of Visual Foresight [17]. It consists of a training phase followed by a planning phase. An overview of VisuoSpatial Foresight is provided in Section 4, and practical implementation details are in Section 5. For the planning phase described in Section 4.2, we tuned the hyperparameters in Table 9. The CEM variance reported is the diagonal covariance used for folding and double folding. We found that for smoothing, a lower CEM variance (0.25, 0.25, 0.04, 0.04) results in better performance, though it may encourage the policy towards taking shorter actions. For CMA-ES, we use the open source Python implementation PyCMA (`https://pypi.org/project/cma/`), changing only the number of iterations, initial mean, and initial variance from default parameters. CMA-ES and CEM take a similar amount of computation time.

As described in Section 4.3, we evaluate with a Pixel L2 and learned Vertex L2 cost function. For the Pixel L2 cost function (Equation 3), we remove the 7 pixels on each side of the image to get rid of the impact of the dark border, using only the inner 42×42 region of the 56×56 image. For the Vertex L2 cost function, as

described in Section 4.3.2, we generate a second dataset from the primary dataset collected. Each of the 9,932 episodes in Fabric-CornerBias can contribute up to $\binom{10}{2}$ image pairs to use in the second dataset, but we sample only 10 of these possible pairs from each episode to keep the dataset size modest (and the same size as the primary dataset). Specifically, we use the following 10 pairs, chosen for their variable gaps in temporal distance: $\{(\mathbf{o}_1, \mathbf{o}_2), (\mathbf{o}_1, \mathbf{o}_3), (\mathbf{o}_1, \mathbf{o}_5), (\mathbf{o}_1, \mathbf{o}_9), (\mathbf{o}_6, \mathbf{o}_8), (\mathbf{o}_6, \mathbf{o}_{10}),$ $(\mathbf{o}_6, \mathbf{o}_7), (\mathbf{o}_3, \mathbf{o}_4), (\mathbf{o}_3, \mathbf{o}_7), (\mathbf{o}_3, \mathbf{o}_9)\}$. During training, we flip the order of half of the data points to encourage the network to ignore the direction of time in its estimation of mesh distance. As mentioned in Section 4.3.2, we annotate all data points with the sum of the distances between corresponding points in the ground truth mesh states, i.e.

$$\sum_{i=0}^{625} ||p_1^{(i)} - p_2^{(i)}||_2^2$$

where $p_1^{(i)}$ is the (x, y, z) coordinates of the i-th point of the mesh shown in the first image and $p_2^{(i)}$ is the (x, y, z) coordinates of the i-th point of the mesh in the second image. We divide all labels by the maximum value for more stable training. Finally, for the network architecture, we use the same CNN as the actor in the DAgger baseline as described in Section 10.1. However, to accommodate the second image input, we pass both images through the same convolutional layers and concatenate the outputs to a 5184-dimensional vector before applying the fully connected layers. The resulting network has about 1.5 million parameters.

11 Supplementary Smoothing Results

11.1 Statistical Significance Tests

We run the Mann-Whitney U test [47] on the coverage and number of action results reported in Table 3 for VSF-1.0 against all baselines other than Imitation Learning, to which we wish to perform similarly. See Table 10 for computed p-values. We conclude that we can confidently reject the null hypothesis that the values are drawn from the same distribution for all metrics except Tier 2 coverage for Wrinkle and the Tier 1 and Tier 3 number of actions for DDPG ($p < 0.02$). Note that Tier 3 results are most informative, as it is the most difficult tier.

11.2 Domain Randomization Ablation

For Fabric-Random, we run 50 simulated smoothing episodes per tier with a policy trained *without* domain randomization and compare with the 200 episodes from Table 3. In the episodes without domain randomization, we keep fabric color, camera angle, background plane shading, and brightness constant at training and testing time. In the episodes with domain randomization, we randomize these parameters in the training data and test in the same setting as the experiments without domain randomization, which can be interpreted as a random initialization of the domain randomized parameters. In particular, we vary the following:

Table 10: Mann-Whitney Test p-values for coverage and number of actions of VSF-1.0 compared with Random, Highest, Wrinkle and DDPG baselines across all tiers of difficulty for smoothing.

Tier	Policy	Coverage p-value	Actions p-value
1	Random	0.0000	0.0000
1	Highest	0.0000	0.0002
1	Wrinkle	0.0040	0.0015
1	DDPG	0.0044	0.3670
2	Random	0.0000	0.0000
2	Highest	0.0000	0.0000
2	Wrinkle	0.2323	0.0091
2	DDPG	0.0000	0.0000
3	Random	0.0000	0.0000
3	Highest	0.0000	0.0000
3	Wrinkle	0.0030	0.0199
3	DDPG	0.0000	0.0674

- Fabric color RGB values uniformly between (0, 0, 128) and (115, 179, 255), centered around blue.
- Background plane color RGB values uniformly between (102, 102, 102) and (153, 153, 153).
- RGB gamma correction with gamma uniformly between 0.7 and 1.3.
- A fixed amount to subtract from the depth image between 40 and 50 to simulate changing the height of the depth camera.
- Camera position (x, y, z) as $(0.5 + \delta_1, 0.5 + \delta_2, 1.45 + \delta_3)$ meters, where each δ_i is sampled from $\mathcal{N}(0, 0.04)$.
- Camera rotation with Euler angles sampled from $\mathcal{N}(0, 90^{\circ})$.
- Random noise at each pixel uniformly between -15 and 15.

From the results in Table 11, we find that final coverage values are similar whether or not we use domain randomization on training data, suggesting our domain randomization techniques do not have an adverse effect on performance in simulation.

To analyze robustness of the policy to variation in the randomized parameters, we also evaluate the former two policies (trained with and without domain randomization) with randomization in the test environment on Tier 3 starting states. Specifically, we change the color of the fabric in fixed increments from its non-randomized setting (RGB (25, 89, 217)) until performance starts to deteriorate. In Table 12, we observe that the domain randomized policy maintains high coverage within the training range (RGB (0, 0, 128) to (115, 179, 255)) while the policy without domain randomization suffers as soon as the fabric color is slightly altered.

Table 11: Coverage and number of actions for simulated smoothing episodes from Fabric-Random, with and without domain randomization on training data, where the domain randomized results are from Table 3.

Tier	Domain Randomized?	Coverage	Actions
1	Yes	92.5 ± 2.5	8.3 ± 4.7
1	No	93.0 ± 3.0	6.9 ± 4.1
2	Yes	90.3 ± 3.8	12.1 ± 3.4
2	No	91.2 ± 9.2	8.7 ± 3.6
3	Yes	89.3 ± 5.9	13.1 ± 2.9
3	No	85.1 ± 12.8	9.9 ± 3.9

Table 12: Coverage and number of actions for Tier 3 simulated smoothing episodes with and without domain randomization on Fabric-Random training data, where we vary fabric color in fixed increments. (26, 89, 217) is the default blue color and (128, 191, 115) is slightly outside the domain randomization range. Values for the default setting are repeated from Table 11 and all other data points are averaged over 20 episodes.

RGB Values	DR?	Coverage	Actions
(26, 89, 217)	Yes	89.3 ± 5.9	13.1 ± 2.9
(51, 115, 191)	Yes	89.3 ± 10.3	11.7 ± 3.5
(77, 140, 166)	Yes	91.4 ± 3.1	11.7 ± 3.1
(102, 165, 140)	Yes	85.6 ± 10.1	13.2 ± 2.7
(128, 191, 115)	Yes	54.7 ± 6.5	10.3 ± 4.0
(26, 89, 217)	No	85.1 ± 12.8	9.9 ± 3.9
(51, 115, 191)	No	60.7 ± 13.6	7.4 ± 2.4