Disruption-Resistant Deformable Object Manipulation on Basis of Online Shape Estimation and Prediction-Driven Trajectory Correction

Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki

***Abstract*—We consider the problem of deformable object manipulation with variable goal states and mid-manipulation disruptions. We propose an approach that integrates online shape estimation, prediction of shape transitions, and mid-manipulation trajectory correction. All functionalities are implemented using two neural network architectures. We apply this approach to the problem of cloth folding, and perform evaluation experiments in simulation and on robot hardware. We demonstrate that the system can achieve good approximation of given goal states, even when the manipulation process is disrupted by cloth slipping or external interference.**

***Index Terms*—Deep learning in grasping and manipulation, model learning for control, neural and fuzzy control, dual arm manipulation, cloth manipulation.**

I. Introduction

ROBOTIC manipulation of cloth objects is complicated by the fact that these objects take on many different shapes. However, humans manipulate cloth with great dexterity. This raises a number of fundamental questions for the pursuit of robotic manipulation of deformable objects. How do humans mentally image the object's deformations? How do we decide our manipulation strategy? How do we conceptualize the manipulations we perform, and how do we mentally grasp the state of the object? These mechanisms remain ill-understood and therefor hard to replicate.

Our overarching goal here is to establish a framework for functionally approximating human cloth manipulation. The focus of the present work is on continuous shape monitoring and self-correction. We place particular emphasis on the following criteria: (1) An up-to-date representation of the object's shape must be maintained throughout the manipulation process. This requires estimation of a topological representation of the object on basis of discrete sensor data, which must be robust against ambiguity resulting from occlusion. (2) An up-to-date prediction of the manipulation's outcome must be maintained, and manipulation trajectories must be corrected on the spot when the necessity arises. In short, we must perform shape estimation, shape change prediction, and motion correction in a continuous online cycle.

Existing work has realized these functionalities in isolation [1], [2]. If we loosen the demand for online operation, work combining multiple of these elements can also be found [3]. However, to the best of our knowledge, realising the estimation-prediction-correction cycle in online fashion has remained a challenge. The current work proposes methods for realization of the constituent functions and their integration in an online cycle. Below we list the core features.

a) Shape estimation: We represent cloth shapes as mesh structures, and estimate the configuration of the mesh from point clouds obtained from a RGBD sensor. By assigning uncertainties to the individual vertices of the mesh, we make it possible to express ambiguity in our estimations. Processing time is $<$150ms on average.
b) Shape prediction: We predict cloth shape evolution as the robot performs a given manipulation. Processing time is $<$11ms for 100 frames on average.
c) Trajectory correction: When the predicted outcome for the present manipulation diverges from the goal, we revise the remainder of the manipulation trajectory in order to realign the expected outcome with the goal.

We believe this combination of functionalities allows us to capture more of the flexibility seen in human deformable object manipulation than has previously been achieved.

The paper is structured as follows. The next section discusses related work. Section III explains the global structure of our approach. Section IV describes shape estimation, Section V shape prediction, and Section VI manipulation generation. Section VII reports and discusses the results of our simulation and real-world experiments. Section VIII concludes the paper.

II. Related Work

A. Cloth Shape Representations

Automated cloth manipulation is an active field of research. Some approaches explicitly estimate cloth shapes. A common approach is to model the cloth as a polygon model. Miller *et al.* [4] match polygon models to comparatively complex topologies such as long-sleeved shirts. Stria *et al.* [5] similarly estimate clothing shapes, and perform folding. Twardon *et al.* [6] demonstrated tracking of garment openings (e.g., sleeve ends)

Manuscript received October 15, 2020; accepted February 5, 2021. Date of publication February 19, 2021; date of current version March 30, 2021. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and reviewers' comments. This work was supported by NEDO and JSPS KAKENHI. *(Corresponding author: Daisuke Tanaka.)*

Daisuke Tanaka is with the Department of Science and Technology, Graduate School of Medicine, Science and Technology, Shinshu University, Nagano 380-8553, Japan (e-mail: 19hs203j@shinshu-u.ac.jp).

Solvi Arnold and Kimitoshi Yamazaki are with the Department of Mechanical Systems Engineering, Shinshu University, Nagano 380-8553, Japan (e-mail: s_arnold@shinshu-u.ac.jp; kyamazaki@shinshu-u.ac.jp).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3060679, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3060679

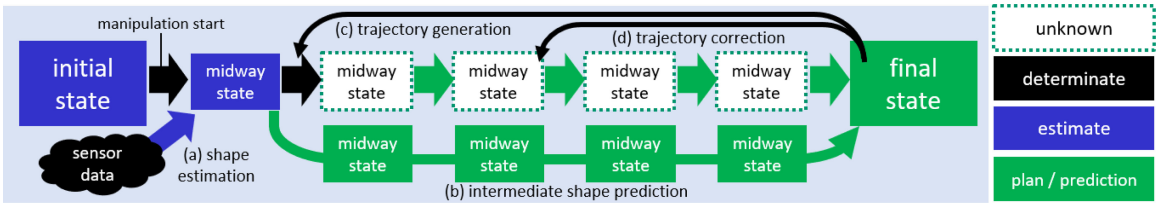

Fig. 1. Overview of system functions and their integration.

by means of ACBM. These methods can be applied online, but are insufficiently expressive to capture fine detail and complex shape configurations.

For finer shape representation, mesh models are an option. Kita *et al.* [7] and Li *et al.* [8] demonstrated accurate shape estimation for suspended garments using an active recognition strategy wherein cloth objects are lifted and manipulated for the purpose of shape estimation. Active strategies are effective for acquiring initial models of cloth objects, but unsuitable for estimating shape continuously during goal-directed manipulations. Willimon *et al.* [9] and Han *et al.* [1] also propose estimation routines for deformable objects, but the amount of deformation considered is less than many common cloth manipulation scenarios, including ours, require.

B. Motion Generation

Various strategies have been proposed for generating manipulation motions, with significant variation in the scope of the problem setting. Van den Berg *et al.* [10] target folding with given manipulation procedures. They solve the problem of determining whether the given procedure is possible with a given number of grippers, and translate it into gripper trajectories under a set of idealized assumptions about cloth dynamics. Execution is open-loop. Maitin-Shepard *et al.* [11] and Doumanoglou *et al.* [3] propose folding pipelines from unordered states to folded states with intermittent recognition, operating in fixed, flowchart-style manipulation procedures. Li *et al.* [2] used simulation to generate folding trajectories, obtaining high quality trajectories and online performance. Sun *et al.* [12] perform flattening using a geometric approach, generating high quality 2.5D representations of wrinkled cloth. Seita *et al.* [13] and Wu *et al.* [14] generate manipulations for smoothing a square cloth using reinforcement learning.

Limitations of the above approaches are lack of on-line trajectory correction during manipulation, and the assumption of fixed goal states. Petrík and Kyrki [15] realize fine feedback control with robustness to material variation for the constrained case of folding a strip of cloth in two, using a reinforcement learning approach with a low-dimensional state representation. Hu *et al.* [16] tightly interlink recognition and motion generation using machine learning techniques operating on raw sensor data. Yang *et al.* [17] achieved online cloth folding by generating motions directly from sensor images. However, the dynamics model and manipulations learned do not necessarily transfer well to different goals.

Work accommodating variable goals remains scarce. We have proposed a system for generating multi-step manipulation plans with variable goals [18], [19] using forward models of the cloth dynamics. Subsequently, Kawaharazuka *et al.* [20], Hoque *et al.* [21] and Yan *et al.* [22] proposed forward model-based approaches capable of accommodating variable goal states to varying extents.

Here, we focus on execution of individual manipulations with various goal states, combining intra-manipulation mesh estimation with forward model-based shape prediction of the course of the manipulation in order to realize on-line trajectory correction. Eventually we aim for integration with the mesh-based version of our multi-step planning system [23].

III. Functional Overview and Approach

A. Function Overview and Integration

We consider the task of manipulating a cloth object into a given goal configuration. Fig. 1 shows an overview of our approach. First, we estimate the initial shape of the object from sensor data. Then we initialize the manipulation for transforming the object from its current shape to the target shape (manipulation planning is treated in [23]). During manipulation, we continuously perform shape estimation (a) and shape prediction (b) on basis of the sensor data acquired over the course of the manipulation. By comparing the predictions with the goal shape, we monitor the progress of the manipulation process. When the difference between prediction and goal shape exceeds the admissibility threshold, function (d) revises the manipulation motion on-the-spot, and manipulation resumes.

The advantage of this mechanism is its ability to respond to unexpected situations during manipulation. For example, if the cloth slips over the surface of the work surface, or is pulled by an external force, the system can adjust the manipulation on-the-spot through estimation of the new cloth shape and prediction of the shape evolution for revised trajectories.

B. Approach

Among the functions in Fig. 1, (a) shape estimation and (b) shape prediction in particular require online performance, which is challenging. We approach this challenge as follows. We train a neural network to generate initial probabilistic shape estimates in milliseconds. We combine this estimate with prior knowledge of the object topology through a short energy minimization process ("refinement") to find a shape that is both realistic and consistent with the estimate. For shape prediction we use a second network, which learns the relation between hand motions and cloth shape change at fine temporal granularity. This lets us predict the evolution of the cloth shape, again with processing times on the order of milliseconds.

The next sections explain our methods for shape estimation (Section 4), and shape transition prediction and trajectory correction (Section V). Both assume a dataset of manipulation examples, containing manipulation trajectories and the corresponding cloth shape evolution at fine temporal granularity. In the present paper we generate this dataset in simulation. Dataset generation is detailed in Section VII.

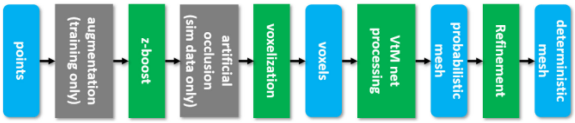

Fig. 2. Shape estimation pipeline. Processes in gray are specific to training and evaluation on simulation data.

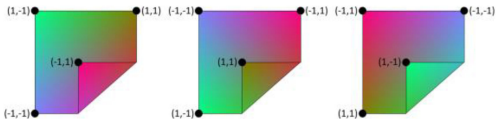

Fig. 3. Examples of equivalent mesh representations of the same shape. Geodesic coordinates for the corner vertices are shown. Texture added to visualize how the mesh is laid out in Cartesian space.

IV. Shape Estimation

Fig. 2 shows an overview of the shape estimation pipeline. We explain the constituent parts and processes below.

A. Voxel-To-Mesh Net

Our shape estimation approach employs a neural network (NN) that generates a probabilistic mesh estimate on basis of a voxel representation of the current cloth shape. We refer to this NN as the Voxel-to-Mesh net (VtM for short) below. We then apply an energy minimization procedure to derive a deterministic mesh representation from the probabilistic estimate and prior knowledge about the cloth object. We refer to this process as "refinement" below. The choice for NN-based shape estimation is motivated by three factors. (1) Speed: initial shape estimates are generated in less than 4ms on average. (2) Simplicity and generalizability: whereas geometric methods often require task-specific categorization of shape elements (e.g., wrinkle categories [12]), our NN-based estimation strategy is low on assumptions and in principle applicable to a broad variety of tasks and cloth topologies. (3) Occlusion handling: occlusion handling is a complex problem for geometric methods. Notable geometric approaches are limited to occlusion-free shapes (e.g., the 2.5D descriptions in [12]) or manipulate items into low-occlusion configurations for shape estimation (e.g., [24]), which can be inefficient. The NN approach can learn the rules governing positional uncertainty implicitly from data, quantifying uncertainty at fine granularity with minimal computational cost.

1) Architecture: The VtM net is a Multi-Layer Perceptron (MLP) architecture (we have experimented with 3D convolutional architectures as well, but achieved better results with the fully connected architecture). Input is a $32 \times 32 \times 32$ binary voxelization of a cloth shape. Output is a $32 \times 32 \times 6$ probabilistic mesh representation. Input and output volumes are flattened for network I/O. In between are four hidden layers of 4096 neurons each. Hidden layers use the hyperbolic tangent (tanh) activation function. Each $1 \times 1 \times 6$ subvolume of the output defines a 3D multivariate normal distribution with a diagonal covariance matrix. We denote the means as μ_x, μ_y, μ_z and the non-zero elements of the covariance matrix as σ_x, σ_y, σ_z. The activation function on the output layer differs for μ and σ values, as the latter should take positive values only. For μ-outputs we use the linear activation function and for σ-values we use $a_{out} = ELU(a_{in}) + 1.05$, where ELU is the Exponential Linear Unit activation function. This function ensures that σ-values are positive, and larger than 0.05 (this helps to stabilize training as σ-values growing too small can lead to incidental extreme loss values).

2) Input/Output Processing: We define the net's "workspace" as a volume of space running from (−1, −1, 0) to (1, 1, 1/3). For all network I/O, cloth shapes are scaled and translated to fit this space. Shapes are translated so that their centres are at (0, 0) in the XY plane, by projecting them onto the XY plane, finding the centre of the projection, and subtracting the centre coordinates from the coordinates of the points comprising the shape. Shapes are scaled such that a fully spread out, axis-aligned cloth runs from (−0.7, −0.7) to (0.7, 0.7). Quantities in the remainder of this section apply in this normalized format.

For input, we convert the input shape representation to a voxel representation, with the voxel volume spanning the workspace defined above. Before voxelization, we non-linearly boost z-coordinates as follows:

$$z_{boosted} = \tanh(3z) \tag{1}$$

The non-linearity in this transformation has the effect of emphasising depth differences close to the work surface ($z = 0$) and deemphasising depth differences further above the work surface. Regions of the cloth that are being lifted up generally present simple shapes, as they hang down under the effect of gravity. These parts can be interpreted well enough at crude z-axis resolution. Regions resting on the work surface or on underlying layers of cloth present more detail due to wrinkling and layering. After applying this transformation, we convert point data to voxel representation by setting all voxels containing at least one point to 1 and all other voxels to 0.

3) Occlusion: Measurement of the real cloth is subject to two types of occlusion: self-occlusion and occlusion by the hands and arms of the robot. We handle self-occlusion by artificially applying occlusion consistent with our hardware setup to simulation data during training. This occlusion applies to the voxel input, while target output (ground truth mesh) remains unoccluded. While we centre states for estimation, real-world occlusion occurs before this centring. To ensure that the net can handle the range of occlusions that occur in the real-world manipulation setup, we apply random offsets to the relative camera position used for calculating artificial occlusion during training.

4) VtM Net Training: Each possible shape has eight equivalent mesh representations (i.e., there are eight equivalent assignments of Cartesian coordinates to the mesh' geodesic coordinates). Fig. 3 illustrates mesh equivalence with some examples. Consequently, there are eight correct answers for each input. We account for this by defining the training loss for the VtM net as follows:

$$loss_{VtM} = MIN\left(\left\{NLL\left(\boldsymbol{s}^i, \hat{\boldsymbol{s}}\right) | i \in [0, \ldots, 7]\right\}\right) \tag{2}$$

Here $\boldsymbol{s}^i$ is the i^{th} mesh representation in an arbitrary ordering of the set of equivalent mesh representations of the ground truth, and $\hat{\boldsymbol{s}}$ is the probabilistic mesh estimate output by the net. *NLL* is shorthand for Negative Log-Likelihood and *MIN* selects the smallest value from a set of values.

Training employs various types of data augmentation. The simulation data we use for training is in mesh format, so during training batch generation we convert meshes to noisy point clouds by taking the set of vertices as points, duplicating each point to increase the point count, and adding Gaussian noise

$\gamma \sim N(0,\ 0.01)$ to each point independently. Augmentation with noise improves robustness and promotes generalization to real-world data. We apply random mirroring on the X axis and random rotation around the Z-axis. To improve robustness to slight variations in the relative position and angle of the camera in real-world experiments, we add tilt and Z-shift augmentations. The tilt augmentation tilts the state over the X and Y axis by angles (in degrees) drawn from $U(-5, 5)$, and Z-shift translates the state on the Z-axis by a distance draw from $U(0, 0.02)$, where *U(a, b)* denotes the continuous uniform distribution over the interval *[a, b]*. Tilt and Z-shift are applied to input but not to ground truth, so the net learns to remove them. We train the net using the SignSGD update rule [25].

The training and augmentation logic facilitates transfer to real cloth. Augmentation and noise on the input improve the net's robustness to noisy real input, while it is trained to output clean shapes from the distribution that generated the training set. Hence the net "interprets" real states into similar states from the training distribution, to some extent. This bias reduces the need to harden processes further down the pipeline.

B. Mesh Refinement

The VtM net produces probabilistic mesh estimate $\tilde{\boldsymbol{s}}^p$. We convert this estimate into a deterministic estimate $\tilde{\boldsymbol{s}}^d$ that is both plausible w. r. t. $\tilde{\boldsymbol{s}}^p$ and consistent with prior knowledge about the cloth, using a refinement procedure that incorporates the cloth's topology in the form of a spring model.

Refinement employs the following losses: Negative log-likelihood of $\tilde{\boldsymbol{s}}^d$ w.r.t. $\tilde{\boldsymbol{s}}^p$ (loss_{nll}), spring energy ($\text{loss}_{\text{spring}}$), and an upward bias ($\text{loss}_{\text{up}}$). For computing $\text{loss}_{\text{spring}}$, we define a set of springs between the vertices, following a spring pattern common to cloth simulation (see e.g., [26]). Let k be the distance between orthogonally neighbouring vertices. A vertex at indices (u, v) connects to neighbour vertices $(u, v \pm k)$ and $(u \pm k, v)$ ("stretch" springs), $(u \pm k, v \pm k)$ ("shear" springs), and $(u, v \pm 2k)$ and $(u \pm 2k, v)$ ("bend" springs), insofar these vertices exist. Spring energy loss is then calculated as follows.

$$loss_{spring} = \sum_{i=0}^{N_{springs-1}} (l_i - r_i)^2 \tag{3}$$

Here l_i is the current length of the spring, r_i is the resting length of the spring (i.e., its length in the cloth's fully spread-out default state), and $N_{springs}$ is the total number of springs. Consideration of self-collision would be desirable, but because of its high computational cost we omit it.

The third loss biases refinement against downward adjustment of vertex positions, because this can push vertices into the work surface. The upward bias ensures that wrinkles produced by optimization form in upward direction. Upward bias loss is computed as the mean over $max(0, \tilde{\boldsymbol{s}}^p_{\mu z} - \tilde{\boldsymbol{s}}^d_z)$, where $\tilde{\boldsymbol{s}}^p_{\mu z}$ and $\tilde{\boldsymbol{s}}^d_z$ are the μ_z and z components of $\tilde{\boldsymbol{s}}^p$ and $\tilde{\boldsymbol{s}}^d$ and subtraction and *max* operate element-wise. Spring loss is multiplied by 5000 to be on the same order of magnitude as loss_{nll}. Upward bias loss is multiplied by 1000. The refinement process starts by initialising the vertex positions. Recall that the purpose of the VtM net in our system is to continually track the cloth shape as the cloth is being manipulated. Consequently, consecutive inputs usually correspond to consecutive moments in time (frames), and represent similar shapes. We exploit this fact by initializing refinement with the refinement result obtained for the preceding frame, if a preceding frame exists. This tends to reduce refinement time cost, and can help to disambiguate the current frame in some cases. When no preceding frame exists, or its difference with the μ component of the present $\tilde{\boldsymbol{s}}^p$ exceeds a given threshold, we initialize with the μ component itself. We update vertex positions using gradient descent on the sign of the gradients of the compound loss, with an update rate of 0.001, running until the loss stabilizes or 300 iterations have passed.

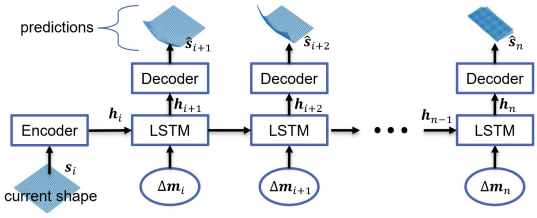

Fig. 4. Overview of network architecture.

V. Shape Transition Prediction

A. Mesh Representation

For prediction, we represent each vertex as a tuple of five values: its coordinates in 3D space, and two values g_r, g_l specifying the vertex's grasp state, taking value 1 when the vertex is grasped by the right and left hand, respectively, and 0 otherwise. Shape $\boldsymbol{s}_i$ denotes the list of vertices at frame i.

B. DNN-Based Shape Transition Prediction

Fig. 4 shows the global network architecture. The network consists of an encoder, an LSTM (Long Short-Term Memory [27]) module, and a decoder. The net predicts the sequence of shapes traversed as the robot's hands pass through the sequence of points $\boldsymbol{m}_{i:n}$ from state $\boldsymbol{s}_i$. Here $\boldsymbol{m}_i = (\boldsymbol{m}_i^j | j \in [0, hands))$, with $hands$ indicating the number of hands used in the manipulation. We define motion vectors describing the hand motions per frame as $\Delta\, \boldsymbol{m}_i = (\Delta \boldsymbol{m}_i^j | j \in [0, hands))$, where $\Delta\, \boldsymbol{m}_i^j = \boldsymbol{m}_{i+1}^j - \boldsymbol{m}_i^j$. Below we explain the role of each network module.

The encoder compresses a given mesh representation $\boldsymbol{s}_i$ into a low-dimensional latent encoding $\boldsymbol{h}_i$. The decoder does the opposite, recovering full state representation $\hat{\boldsymbol{s}}_i$ from latent encoding $\boldsymbol{h}_i$. Encoding into latent representation serves two purposes: reduction of computational cost, and acquisition of a representation format that facilitates prediction. By training the modules end-to-end, we obtain a latent encoding format optimized for prediction, while also allowing recovery of the full shape representation. The encoder consists of 3 convolutional layers with channels depths of 64, 128, 256, kernel sizes 3×3, 3×3, 5×5, and strides 2, 3, 1, followed by a dense layer with an output dimensionality of 256. All layers use the tanh activation function. Input is presented as a $32 \times 32 \times 5$ volume, with each $1 \times 1 \times 5$ subvolume representing one vertex. The decoder largely mirrors this architecture, using transposed convolution instead of convolution layers, and using linear activation on its output neurons. Its output is a $32 \times 32 \times 3$ volume specifying the predicted coordinates for each vertex.

The LSTM module predicts the shape evolution for a given trajectory. The net consists of 10 layers containing 256 LSTM units each, and has six input neurons. We initialize the internal

state (i.e., activation values) of all layers with latent encoding $\boldsymbol{h}_i$ from the encoder. We then iterate through $\Delta \boldsymbol{m}_{i:n}$, feeding its elements in order on the input neurons. The sequence of internal states of the last layer observed over the course of this process gives the latent encoding sequence $\boldsymbol{h}_{i+1:n}$ describing the shape evolution in latent form. By passing this sequence through the decoder, we obtain the shape sequence $\hat{\boldsymbol{s}}_{i+1:n}$.

C. Shape Transition Learning

The network is trained on our dataset of manipulation examples. For each example presented during training, we select an integer i, $0 \leq i < n$, where n is the length of the example in frames. We let the net process $\boldsymbol{s}_i$ and $\Delta \boldsymbol{m}_{i:n-1}$ to obtain prediction sequence $\hat{\boldsymbol{s}}_{i+1:n}$, and compute the training loss (MSE) over $\boldsymbol{s}_{i+1:n}$ and $\hat{\boldsymbol{s}}_{i+1:n}$. Using random starting points ensures that the shape transition can be predicted from any point in the manipulation. We apply rotational augmentation. We train the net using the Adam update rule [28] until loss converges.

VI. Trajectory Optimization

We perform trajectory optimization by minimising the difference between the predicted manipulation outcome and the goal shape. We denote the sequence of predicted shapes as $\hat{\boldsymbol{s}}_{i+1}, \ldots, \hat{\boldsymbol{s}}_n$ and define the following cost function:

$$\mathrm{L}\ (\boldsymbol{s}^*, \hat{\boldsymbol{s}}_n, \boldsymbol{m}_{i:n}) = \ \mathrm{MSE}\,(\boldsymbol{s}^*, \hat{\boldsymbol{s}}_n) + w \cdot len\,(\boldsymbol{m}_{i:n}) \tag{4}$$

$$len\ (\boldsymbol{m}_{i:n}) = \frac{1}{hands} \sum_{j\,=\,0}^{hands-1} \left\| \Delta m_i^j \right\| + \ldots + \left\| \Delta m_{n-1}^j \right\| \tag{5}$$

Where MSE denotes the mean squared error, $len(\boldsymbol{m})$ calculates the physical length of trajectory $\boldsymbol{m}$, and w is a weight parameter. Shapes are assignments of coordinate values to the vertices of the cloth mesh, so the MSE over two shapes can be calculated by computing the squared error over corresponding coordinate values and averaging them. We penalize trajectory length to avoid generating unnecessarily long trajectories. We can now formalize trajectory optimization as the following minimization problem:

$$\underset{\boldsymbol{m}_{i:n}}{\operatorname{argmin}}\, \mathrm{L}\,(\boldsymbol{s}^*, \hat{\boldsymbol{s}}_n, \boldsymbol{m}_{i:n}) \tag{6}$$

We optimize trajectories by obtaining gradients w. r. t. $\mathrm{L}(\boldsymbol{s}^*, \hat{\boldsymbol{s}}_n, \boldsymbol{m}_{i:n})$ for inputs $\Delta \boldsymbol{m}_{i:n-1}$, through back-propagation [29], and adjusting the inputs using the Adam update rule [28]. The procedure for trajectory optimization is as follows:

1) If $i = 0$, Initialize the trajectory points $\boldsymbol{m}_{i+1:n}$.
2) Predict sequence $\hat{\boldsymbol{s}}_{i+1:n}$ from $\boldsymbol{s}_i$ and $\Delta \boldsymbol{m}_{i:n-1}$.
3) Calculate the loss over predicted outcome $\hat{\boldsymbol{s}}_n$ and goal shape $\boldsymbol{s}^*$, and obtain gradients by back-propagation.
4) Update $\boldsymbol{m}_{i+1:n}$ along the gradients.

Steps 2 through 4 are repeated until the loss falls below a given threshold or a set number of loops has passed. Trajectory length n is derived from the trajectory used to initialize the optimization process by adding 40, and remains fixed during optimization. The additional frames are to allow the cloth shape to stabilize after being released by the grippers. During stabilisation, manipulation input is blank (all-zero), but prediction of shape development continues.

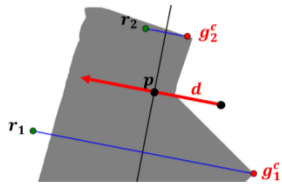

Fig. 5. Example calculation of trajectories (blue lines) and release points ($\boldsymbol{r}_1$, $\boldsymbol{r}_2$) from grasp points ($\boldsymbol{g}_1^C, \boldsymbol{g}_2^C$) and displacement vector $\boldsymbol{d}$. The gray shape in the background is the cloth silhouette (2D projection).

Given goal shape s^*, current shape $\boldsymbol{s}_i$ and trajectory points $\boldsymbol{m}_{i:n}$, optimization can be performed at any time in the manipulation process. We trigger the optimization process when the divergence between the predicted outcome $\hat{\boldsymbol{s}}_n$ and the goal state exceeds a given admissibility threshold.

VII. Experiments

A. Manipulation Format

We consider single- and dual-handed manipulations on a square cloth. For data generation and trajectory initialization, we represent manipulations as real-valued vectors of length six, defining clean arc trajectories. The format is as follows. The first four values define two grasp points, $\boldsymbol{g}_1^G$ and $\boldsymbol{g}_2^G$, by their geodesic coordinates (u, v) on the cloth, with the cloth surface running from $(-1, -1)$ to $(1, 1)$. The second grasp point can take a null value, indicating a single-handed manipulation. The last two values define a displacement vector $\boldsymbol{d}$ given in 2D Cartesian coordinates (x, y). Given a cloth state (mesh), this representation determines grasp point trajectories as follows. We map geodesic grasp points $\boldsymbol{g}_i^G$ to Cartesian grasp points $\boldsymbol{g}_i^C$ using the cloth mesh. For two-handed grasps, we then compute point $\boldsymbol{p}$:

$$\boldsymbol{p} = \frac{\boldsymbol{g}_1^C + \boldsymbol{g}_2^C}{2} + \frac{\boldsymbol{d}}{2} = \frac{\boldsymbol{g}_1^C + \boldsymbol{g}_2^C + \boldsymbol{d}}{2} \tag{7}$$

Let $\boldsymbol{m}$ be a line through $\boldsymbol{p}$ perpendicular to $\boldsymbol{d}$. The x and y coordinates for Cartesian release points $\boldsymbol{r_i}$ are found by mirroring $\boldsymbol{g}_i^C$ over line $\boldsymbol{m}$ on the XY-plane. Fig. 5 shows an example. For single-handed grasps, the x and y coordinate of the single release point $\boldsymbol{r}_1$ is given by $\boldsymbol{g}_1^C + \boldsymbol{d}$. The z coordinate for $\boldsymbol{r}_i$ is given by $\boldsymbol{g}_i^C.z + min(0.2,\ k/2)$, where k is the distance between $\boldsymbol{g}_i^C$ and $\boldsymbol{r}_i$ in the XY-plane. We find circle c centred at height $\boldsymbol{g}_i^C.z$, perpendicular to the XY plane, and passing through $\boldsymbol{g}_i^C$ and $\boldsymbol{r}_i$. The shortest segment of c connecting $\boldsymbol{g}_i^C$ and $\boldsymbol{r}_i$ defines the trajectory for point i.

B. Dataset Generation

We generate a dataset of 3691 manipulation sequences consisting of 3 manipulations each, using the ARCSim cloth simulator [30], [31]. Each sequence starts with the cloth laid out flat. Single- and dual-handed manipulations are generated randomly in a proportion of 1:2. Grasp points are selected randomly from the convex corners of the projection of the cloth shape onto the XY plane. Candidate points are found by means of corner detection [32]. Displacement vectors are generated randomly with a maximum length of 2.0 (measured in the normalized workspace of Section 4). The cloth mesh is stored at each frame of the simulation. The total number of frames in the dataset is

TABLE I
MESH ESTIMATION ACCURACY

Set	Type	Test mean (st.dev.) / median	Train mean (st.dev.) / median
Main set	μ	0.0534 (0.0444) / 0.0403	0.0537 (0.0435) / 0.0410
	R	0.0547 (0.0479) / 0.0388	0.0543 (0.0466) / 0.0394
First step	μ	0.0156 (0.0125) / 0.0127	0.0164 (0.0155) / 0.0131
	R	0.0135 (0.0100) / 0.0116	0.0141 (0.0136) / 0.0119
Tshirt	μ	0.0240 (0.0197) / 0.0203	n/a
	R	0.0218 (0.0182) / 0.0192	n/a
Sweater	μ	0.0234 (0.0186) / 0.0203	n/a
	R	0.0215 (0.0170) / 0.0191	n/a
Swimsuit	μ	0.0227 (0.0190) / 0.0191	n/a
	R	0.0205 (0.0169) / 0.0179	n/a

Error unit: length of the side of the cloth. μ: μ-component of network estimate. R: refined estimate.

TABLE II
ESTIMATION AND REFINEMENT TIME COST

Time cost	Main set mean (st.dev.) / median	First step mean (st.dev.) / median
Estimation	3.24 (1.6) / 3.18 ms	3.24 (2.0) / 3.19 ms
Refinement	142 (116) / 112 ms	110 (65) / 100 ms

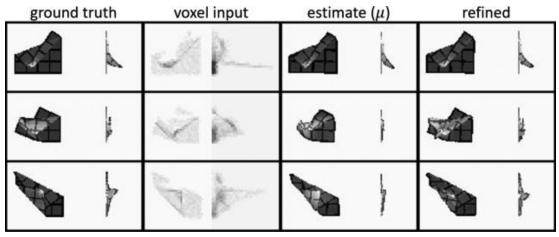

Fig. 6. Examples of shape estimation and refinement. Each panel shows top-down and lateral views. Note the occlusion in the voxel input in the first example, detail recovery by refinement in the second example, and recovery of height relief by refinement in the third example.

TABLE III
PREDICTION ACCURACY AND TIME COST

n_{frames}	Test error	Train Error	Time cost
50	0.030	0.029	6.4 ms
100	0.039	0.034	10.8 ms
200	0.059	0.058	17.9 ms

Error unit: length of the side of the cloth. Time unit: milliseconds

2276887. We use 3491 sequences as training data, and 100 each for validation and test sets.

C. Environment for Simulation Experiments

For evaluating trajectory correction in simulation, we set the simulation up to accept frame-by-frame input in the form of 3D movement vectors for the grasped points. We implement three evaluation scenarios.

1) Undisturbed manipulation. In this scenario, the cloth behaves exactly as in the dataset. This should allow accurate prediction throughout, so manipulation should be successful with minimal correction.
2) Manipulation with cloth slipping. This scenario reduces the friction between cloth and work surface. This causes the cloth to slide somewhat over the course of the manipulation, necessitating correction.
3) Manipulation with external interference. This scenario mimics a situation where an external force moves the work surface during manipulation (or, equivalently, pulls the cloth over the work surface). Movement is in the direction of the displacement vector and runs from the 10th frame to halfway the manipulation.

D. Mesh Estimation Results

We evaluate shape estimation on the shape data of 100 manipulation sequences (300 manipulations, 61660 frames) from the test set and 100 sequences (300 manipulations, 62226 frames) from the training set. For consecutive frames, we initialize refinement with the refinement result of the preceding frame, unless average vertex distance with the μ component of the estimate exceeds 0.1 times the cloth length.

Estimation accuracy is reported in Table I, time cost in Table II, and example results are shown in Fig. 6. Errors are measured as Euclidean distance between estimated and actual vertex positions, with the length of the side of the cloth as unit. Errors vary with the complexity of the cloth shape. Since we focus here on manipulations starting from the spread-out state, we report errors for the first manipulation in each sequence (marked as *first step* in Table I) in addition to the errors over the full set (marked *main set* in Table I). The distance between the μ component of the estimation and the target averages to about 1/20th of the cloth length for the full set, and slightly less than 1/60th for first manipulations. Refinement does not notably improve accuracy. This is expected: the net is trained to minimize error exclusively, whereas refinement also considers shape realism. For example, estimates often omit wrinkles, indicating them as regions of increased uncertainty instead. This reduces the surface of the mesh described by the μ component. Refinement will restore cloth surface by producing wrinkles in areas of increased uncertainty, moving vertices away from their μ values. Unless the wrinkles happen to align closely with the actual wrinkles, this increases the error.

All training data is generated with identical material properties. To assess how these properties affect accuracy, we generate three single-step test sets with different material properties. From the material definitions included with ARCSim, we selected the t-shirt, sweater, and swimsuit materials (for details about the material definitions we refer to [33]). Cloth topology and other settings were unchanged. Estimation accuracy is shown in Table I. We observe some deterioration, but errors remain within 1/40 of the cloth length. Domain randomization could likely further improve robustness to material variation.

E. Shape Prediction Results

We evaluate shape prediction for the full test set and 300 manipulations examples from the training set. For each example we perform prediction over horizons of 50, 100, and 200 frames, which are representative horizons for real manipulation scenarios. We select the starting state randomly from the range $[0, n_{example} - n_{horizon}]$ where $n_{example}$ is the full number of frames in the example and $n_{horizon}$ indicates how far into the future we are predicting (i.e., from starting frame i, we predict frame $i + n_{horizon}$). When $n_{example} < n_{horizon}$, we predict the full example starting at frame 0. Errors measure average distance between corresponding vertex pairs in ground truth and prediction, with the cloth length as unit. Table III shows accuracy and time cost.

TABLE IV
ERROR BETWEEN OUTCOME SHAPE AND GOAL SHAPE

Scenario	With correction	No correction
1. Undisturbed	0.022	0.026
2. Cloth slipping	0.048	0.059
3. External disruption	0.031	0.111

Unit: length of the side of the cloth.

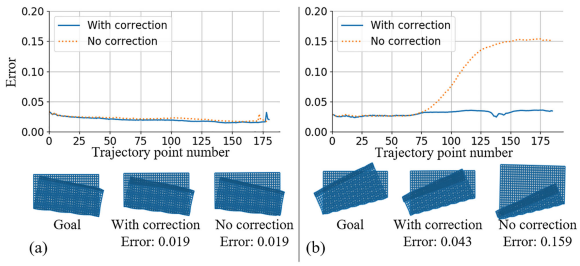

Fig. 7. Example of manipulation results. Examples of (a) scenario 1 and (b) scenario 3. The graph shows the error value between the predicted shape and the target shape at each trajectory point.

F. Trajectory Correction Results

We performed folding in simulation to evaluate trajectory correction. We set the initial shape s_0 and goal shape s^*, and initialize the trajectory points $m_{0:n}$ to evenly divide a round arc trajectory defined in the format described in 7A. To isolate trajectory correction we omit shape estimation in this experiment, using meshes from the simulation directly. We evaluate the three scenarios described in Section 7C. Each scenario is run for 50 manipulations from the test set (first-in-sequence manipulations). For comparison, we also run each manipulation without trajectory correction. Table IV shows accuracy averages. We observe that in each scenario, trajectory correction reduces the error between outcome and goal shape. Time cost for correction ranges from 1 to 20 seconds, and depends on the frame length of the remaining trajectory.

Fig. 7 shows example results. In Fig. 7a we see that correction has little influence on the result in scenario 1: close approximation of the goal is achieved with and without correction. The predicted error w. r. t. the goal shape remains near-constant over the course of the manipulation. This indicates that under conditions consistent with the training data, we can accurately predict the course of the manipulation.

Scenarios 2 and 3 diverge from the training conditions, resulting in large errors if no trajectory correction is performed, as seen in the results for scenario 3 shown in Fig. 7b. The error graph shows that the widening error with the goal shape is evident from the network's predictions during manipulation (dotted line). When trajectory correction is enabled, the predicted error is effectively suppressed (solid line). This results in a much better approximation of the goal state.

G. Hardware Experiments

We performed cloth folding on robot hardware, using HIRO (Kawada Robotics), a dual-armed robot with six degrees of freedom per arm. The cloth is observed through an RGBD sensor (Azure Kinect) placed opposite to the robot. We use square cloths of 24cm by 24cm. Fig. 8 shows our setup.

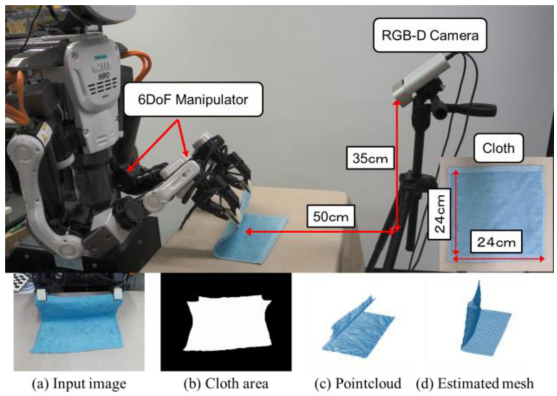

Fig. 8. Environment for folding with actual robot. The shape of the cloth during manipulation is estimated using information from an RGBD camera placed oppositve to the robot. Bottom row shows result of estimating the cloth shape during the manipulation shown in the top panel.

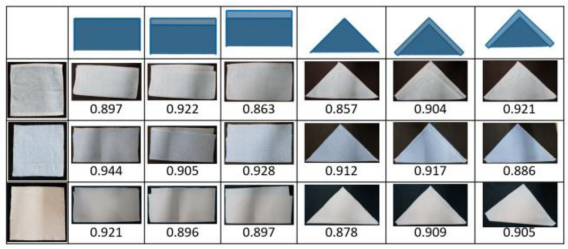

Fig. 9. Outcomes and IoU scores for real cloth folding with six goal shapes and three materials.

RGBD input is processed as follows. We isolate the cloth area using Grabcut [34] and contour detection, as shown in Fig. 8b. Then we obtain a 3D point cloud of the cloth by retrieving the depth values from the region of the depth image corresponding to the cloth area. We convert the point cloud to a voxel representation, and estimate the mesh using the VtM net and the refinement procedure, resulting in a shape estimate as shown in Fig. 8d. At the start of a manipulation, the experimenter instructs the robot to grasp the points indicated by the planned manipulation. Shape estimation and prediction are executed at intervals of 10 trajectory frames at a time, and we set trajectory correction to trigger when the MSE over the predicted outcome and the goal exceeds 0.003.

We perform two experiments. The first evaluates performance on six goal shapes for three cloths: a typical hand towel, a square of stretchy t-shirt fabric, and a square of thin but fairly stiff woven fabric. We evaluate each combination of goal and fabric once, for a total of 18 cases. Results and scores are shown in Fig. 9. Scores represent the IoU (intersection over union) taken over the XY projection of the goal shape and a mask image of the result shape taken from top-down view. We obtain an average IoU of 0.904 overall. The t-shirt fabric is very supple and buckles easily. In preliminary experimentation, this sometimes led to shape collapse that is hard to recover from.

The second experiment evaluates manipulation with external disruptions. We manually disrupt the manipulation by pulling on the tablecloth covering the work desk halfway into the manipulation, thereby displacing the cloth by a few centimeters. We use the hand towel for these trials. Results and IoU scores are shown in Fig. 10 for experiments with and without trajectory

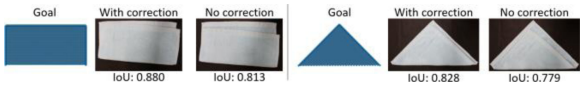

Fig. 10. Outcomes and IoU scores for real cloth folding with mid-manipulation disturbance, with and without trajectory correction.

correction. We observe that trajectory correction substantially improves outcomes, producing satisfactory approximations of the goal shapes.

Time cost for shape estimation and trajectory search did not differ significantly from the simulation experiments. Footage of hardware experiments can be found in the video accompanying this paper.

VIII. Conclusion & Future Work

We presented an approach for deformable object manipulation on basis of shape estimation, shape prediction, and trajectory correction. We quantitatively evaluated the individual parts in simulation, and tested the integrated system on a selection of test cases in simulation and on robot hardware. We find that the system can successfully produce a variety of goal shapes, even when disruptions occur during the manipulation.

While our experiments are presently limited to square cloth, the system should in principle be applicable to alternative topologies with limited modification. However, the assumption that the topology is known can be limiting. Handling items for which we do not have sufficient prior topological knowledge will require integration with garment recognition and shape parametrization routines, as seen in e.g., [10].

Future directions include more extensive evaluation of the integrated system. We also pursue integration with our previously proposed multi-step manipulation planning system [23]. In this integration the planning system would provide initial trajectories and intermediate goal shapes for execution by the system proposed here. Lastly, we aim to extend the system to more topologically complex objects, such as clothes, and explore domain randomization strategies to further improve generalization over materials.

References

[1] T. Han, X. Zhao, P. Sun, and J. Pan, "Robust shape estimation for 3D deformable object manipulation," 2018, *arXiv:1809.09802*.

[2] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *Proc. IEEE/RSJ Int'l. Conf. Intell. Robots Syst.*, 2015, pp. 6000–6006.

[3] A. Doumanoglou *et al.*, "Folding clothes autonomously: A complete pipeline," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1461–1478, Dec. 2016.

[4] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *Proc. IEEE Int'l. Conf. Robot. Automat.*, 2011, pp. 4861–4868.

[5] J. Stria *et al.*, "Garment perception and its folding using a dual-arm robot," in *Proc. IEEE/RSJ Int'l. Conf. Intell. Robots Syst.*, 2014, pp. 61–67.

[6] L. Twardon and H. Ritter, "Active boundary component models for robotic dressing assistance," in *Proc. IEEE/RSJ Int'l. Conf. Intell. Robots Syst.*, 2016, pp. 2811–2818.

[7] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Clothes handling based on recognition by strategic observation," in *Proc. 11th IEEE-RAS Int'l. Conf. Humanoid Robots*, 2011, pp. 53–58.

[8] Y. Li *et al.*, "Regrasping and unfolding of garments using predictive thin shell modelling," in *Proc. IEEE Int'l Conf. Robot. Automat.*, 2015, pp. 1382–1388.

[9] B. Willimon, S. Hickson, I. Walker, and S. Birchfield, "An energy minimization approach to 3D non-rigid deformable surface estimation using RGBD data," in *Proc. IEEE/RSJ Int'l. Conf. Intell. Robots Syst.*, 2012, pp. 2711–2717.

[10] J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel, "Gravity-based robotic cloth folding," *Algorithmic Foundations Robot. IX. Springer Tracts Adv. Robot.*, vol. 68, 2010, doi: 10.1007/978-3-642-17452-0_24.

[11] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *Proc. IEEE Int'l. Conf. Robot. Automat.*, 2010, pp. 2308–2315.

[12] L. Sun, G. Aragon-Camarasa, S. Rogers, and J. P. Siebert, "Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening," in *Proc. IEEE Int'l Conf. Robot. Automat.*, 2015, pp. 185–192.

[13] D. Seita *et al.*, "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor," 2019, *arXiv:1910.04854*.

[14] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," *Robot.: Sci. Syst.*, 2020.

[15] V. Petrik and V. Kyrki, "Feedback-based fabric strip folding," in *Proc. IEEE/RSJ Int'l Conf. Intell. Robots Syst.*, 2019, doi: 10.1109/IROS40897.2019.8967657.

[16] Z. Hu, P. Sun, and J. Pan, "Three-Dimensional deformable object manipulation using fast online gaussian process regression," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 979–986, Apr. 2018.

[17] P. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 397–403, Apr. 2017.

[18] D. Tanaka, S. Arnold, and K. Yamazaki, "EMD net: An encode–manipulate–decode network for cloth manipulation," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1771–1778, Jul. 2018.

[19] S. Arnold and K. Yamazaki, "Fast and flexible multi-step cloth manipulation planning using an encode-manipulate-decode network (EM∗D net)," *Front. Neurorobotics*, vol. 13, 2019.

[20] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, "Dynamic manipulation of flexible objects with torque sequence using a deep neural network," in *Proc. IEEE Int'l. Conf. Robot. Automat.*, 2019, pp. 2139–2145.

[21] R. Hoque *et al.*, "VisuoSpatial foresight for multi-step, multi-task fabric manipulation," *Robot.: Sci. Syst.*, 2020.

[22] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," 2020, *arXiv:2003.05436*.

[23] S. Arnold, D. Tanaka, and K. Yamazaki, "Cloth manipulation planning with mesh representations and incomplete domain knowledge," in *Proc. 38th Annu. Conf. Robot. Soc. Jpn.*, 2020.

[24] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Proc. IEEE Int'l. Conf. Robot. Automat.*, 2011, pp. 3893–3900, doi: 10.1109/ICRA.2011.5980327.

[25] J. Bernstein, Y. X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimization for non-convex problems," *arXiv:1802.04434*.

[26] K. J. Choi and H. S. Ko, "Stable but responsive cloth," *ACM Trans Graph*, vol. 21, no. 3, pp. 604–611, 2002.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int'l. Conf. Learn. Repres.*, 2015, *arXiv:1412.6980*.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[30] R. Narain, T. Pfaff, and J. F. O'Brien, "Folding and crumpling adaptive sheets," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 51:1–51:8, Jul. 2013.

[31] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 147:1–147:10, Nov. 2012.

[32] J. Shi and Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[33] H. Wang, J. F. O'Brien, and R. Ramamoorthi, "Data-driven elastic models for cloth: Modeling and measurement," *ACM Trans. Graph.*, vol. 30, no. 4 Article 71, 12 pages, Jul. 2011.

[34] C. Rother, V. Kolmogorov,and, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph. Proc. SIGGRAPH*, vol. 23, no. 3, pp. 309–314, 2004.