Cloth Manipulation with Estimation of Material Properties

○Solvi Arnold (Shinshu University)　Kimitoshi Yamazaki (Shinshu University)

1. Introduction

Planning manipulations on cloth objects is challenging due to such objects' large state spaces and complex dynamics. Manipulation is further complicated by material variation. Material properties cannot reliably be assessed from passive observation of an object, but can have significant impact on manipulation outcomes. Recent approaches have used trained models of the cloth dynamics for generating manipulations. If there is a mismatch in material properties between training data and test conditions, manipulation accuracy will deteriorate. This is hard to avoid when training on simulation data. Precise measurement of the material properties to match simulation to reality is challenging, and must be repeated for different fabrics. This is impractical and error-prone.

The alternative we explore here is to cover a *region* of the material property space, and add functionality for estimating material properties during manipulation. This strategy avoids the need to measure and prepare separate data for different fabrics, as well as the need to explicitly provide fabric information to the system at run-time.

Work addressing material variation in the context of cloth manipulation remains scarce, but notable examples are [1] and [2]. In [1], folding of a fabric strip of unknown stiffness is performed using a Reinforcement Learning approach. In [2], bending and stretching stiffness are modelled as parametric biases, and estimated during dynamic manipulation (spreading a sheet) with variable-stiffness joints. Our approach to material estimation is similar, but we focus on folding instead of spreading, model shape predictions and material properties probabilistically, expand the repertoire of estimated properties, and address variable goals. However, in contrast to [2], our work is presently limited to simulation experiments, manipulation in our test scenarios is near-static, and joint stiffness is not considered.

2. System

We consider the problem of automatically manipulating a cloth object into a goal shape set at run-time. We assume a dual-armed robot as manipulator, capable of grasping two points of the object, and moving them freely within the workspace. A manipulation consists of the robot lifting the grasp points, moving them along a 180° arc-like trajectory, and releasing them. In our experiments, we use the task of folding a T-shirt approximately in two as test case (see Figure 1 for examples). In the present paper we evaluate manipulation generation with material property knowledge of varying uncertainty, and property estimation during execution of a given manipulation trajectory. Experiments in this work are in simulation, so we have direct access to the mesh. Application to real cloth requires functionality for extracting mesh representations from observations. We addressed this challenge elsewhere [4][5]).

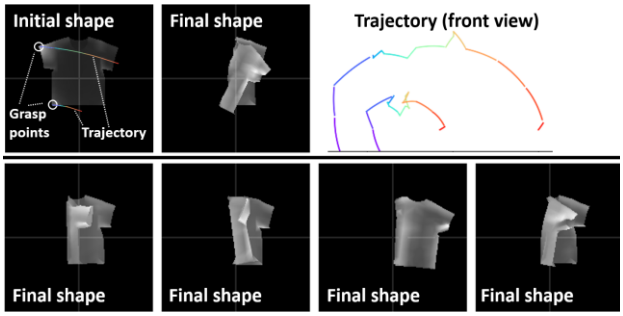

Fig. 1. Manipulation examples. Top: initial shape, final shape, and trajectory of one example. Bottom: final shapes of four different examples.

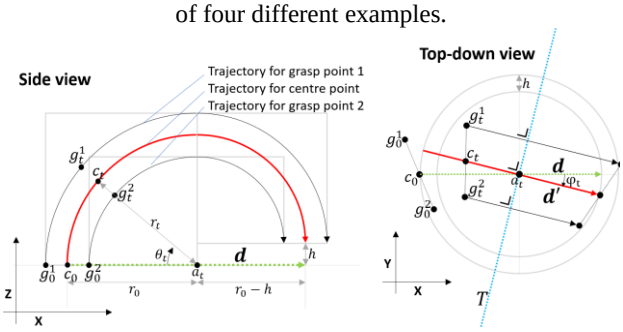

Fig. 2. Deriving the positions of grasp points (g_t^1, g_t^2) at time t from the high-level manipulation specification $(g_0^1, g_0^2, \boldsymbol{d})$ and control parameters $(\theta_t, r_t, \varphi_t, a_t)$. Drop height h is a fixed system parameter. Rotation around axis T (perpendicular to $\boldsymbol{d}'$) progresses the manipulation.

2.1. Manipulation Trajectory Representation

Manipulations lift, displace, lower, and release the grasped vertices of the object. Hence, trajectories should be globally arc-like, while allowing flexibility at fine granularity. The positions of the grasped points at time t are denoted (g_t^1, g_t^2), and derived from a set of variables: $(\theta_t, r_t, \varphi_t, a_t, g_0^1, g_0^2, \boldsymbol{d})$. Points g_0^1, g_0^2 are the original grasp positions, and $\boldsymbol{d}$ is a 2D vector indicating displacement of the centre point between the grasp points in the XY plane. These values are given and fixed per manipulation. They may be set by the user or generated by a higher-level planning process, as proposed elsewhere [5]. Manipulation is controlled using the control variables $\theta_t, r_t, \varphi_t, a_t$. Figure 2 illustrates how (g_t^1, g_t^2) are found. Initial values for r_0 and a_0 are derived from g_0^1, g_0^2 and $\boldsymbol{d}$, while θ_0 and φ_0 are initialised to zero.

A trajectory is described by a sequence of deltas for the control variables. By applying the deltas in order to the control variables and recalculating (g_t^1, g_t^2), we obtain the trajectory. We assume a control update interval of 12

simulation frames, and we refer to a run of 12 frames as a *segment*. For the frames within a segment, we repeat the same deltas. Trajectories themselves are regularly regenerated during manipulation to incorporate shape observations, so in each manipulation session there exists a sequence of trajectories. A manipulation trajectory of k segments, at generation round i in a manipulation session, is denoted as follows.

$$m_i = \langle \Delta\theta_{i,j}, \Delta r_{i,j}, \Delta\varphi_{i,j}, \Delta a^x_{i,j}, \Delta a^y_{i,j} | j \in [1,k] \rangle \quad (1)$$

Note that a denotes a point in the XY plane, and thus consists of two coordinates.

2.2. Material Property Representation

We let $P = \{friction, bend, stretch, thickness\}$ represent the set of material properties under consideration. We denote the actual value of $p \in P$ as q^p, and represent knowledge about property $p \in P$ at segment i as $\hat{q}^p_i = (c^p_i, w^p_i)$, where c^p_i and w^p_i define the centre and width of a truncated continuous uniform distribution on the $[0,1]$ interval as follows:

$$U_{[max(c^p_i - w^p_i, 0), min(c^p_i - w^p_i, 1)]} \quad (2)$$

Hence material knowledge is represented as the material property value lying within a specified range. Absence of knowledge can be expressed as $(c^p_i = 0.5, w^p_i = 1.0)$, whereas absolute certainty about a property's value can be expressed with $w^p_i = 0.0$.

2.3. Neural Network

The neural network consists of encoder, prediction, and decoder modules. We omit description of the architecture here, but note that it closely follows [4], expect that the layer count in the prediction module is reduced to 5.

The compound network takes as input 1) shape observations in mesh format, 2) manipulation trajectories, and 3) material property knowledge, and outputs shape prediction in probabilistic mesh format. For each vertex v in the cloth mesh, the net outputs tuple $\langle \mu^v_x, \mu^v_y, \mu^v_z, \sigma^v_x, \sigma^v_y, \sigma^v_z \rangle$ defining a multivariate normal distribution with means $\mu^v_x, \mu^v_y, \mu^v_z$ and a diagonal covariance matrix with values $\sigma^v_x, \sigma^v_y, \sigma^v_z$ on its diagonal.

3. Data Generation & Network Training

We generate a set of manipulation examples using the ARGUS cloth simulator [6]. The scenario we test here is folding a T-shirt approximately in two from left to right using a dual-armed manipulator. With the T-shirt laid out flat on a work surface, we set g^1_0, g^2_0 to grasp the T-shirt by the top corner of the left sleeve and the bottom left corner of the T-shirt body. Displacement vector $\boldsymbol{d}$ is set to point rightwards, and its length is tuned so that performing manipulation with fixed control input $m_t = (\Delta\theta_t, \Delta r_t, \Delta\varphi_t, \Delta a^x_t, \Delta a^y_t) = (1,0,0,0,0)$ results in the shirt being folded approximately in two. For each manipulation example, the control sequence m and material properties $q^p, p \in P$ are randomised and recorded. ARGUS uses high-dimensional specifications of bending/stretching stiffness. For simplicity, we define a basic anisotropic material, and let our bending/stretching stiffness parameters act as multipliers on this fabric specification. We denote the elements of each manipulation example as follows: s_i denotes to the i^{th} shape in a shape sequence (i.e. the shape at segment i), m_i denotes to the manipulation input at segment i, and q denotes the material properties for the example. Manipulation examples are shown in Figure 1.

We train the neural network end-to-end on batches of manipulation examples. Ranges for all material properties are normalised to the $[0,1]$ range. During batch generation, we randomly generate instances of material property knowledge $\hat{q}^p = (c^p, w^p)$ from the actual property values q^p as follows.

$$w^p = R(0,1)^2 \qquad c^p = q^p + \frac{R(-w^p, w^p)}{2} \quad (3)$$

Here $R(low, high)$ uniformly generates a random value in $[low, high]$. This results in a property knowledge representation that is true for the example, and varies in its level of uncertainty. Material input is constant over segments, so we omit the segment index. The starting segment for each example in a batch is randomised in order to learn to predict form any point in the trajectory. Training loss is the negative log-likelihood. Weights are updated using the SignSGD update rule [7], with automatic learning rate adjustment.

4. Trajectory Planning

A manipulation session starts with generation of the initial manipulation trajectory. We then execute the first segment, observe the resulting cloth shape, and regenerate the remainder of the trajectory. This process repeats every segment, until the release point of the (then-current) trajectory is reached, at which point the cloth is released and left to settle. During trajectory generation, we run multiple instances of the generation process in parallel. This produces a set of candidate solutions, from which we select the solution with the best residual loss. Below we explain the process in more detail.

Given shape s_i at the start of segment i, goal shape s^*, and material property information $\hat{q}_i = \langle (c^p_i, w^p_i) | p \in P \rangle$, we generate a k-step manipulation sequence m_i for producing s^* from s_i as follows. We assume to have some initial sequence m^{init}_i of length (segment count) $b \geq k$, which may be randomly initialised or initialised from a preceding step of the process. For the initial trajectory, we use the default trajectory $\langle m_{i,j} = (2,0,0,0,0) | j \in [1,b] \rangle$ as basis, and add Gaussian noise independently for each instance of the trajectory generation process. When regenerating the trajectory, we start with the preceding trajectory

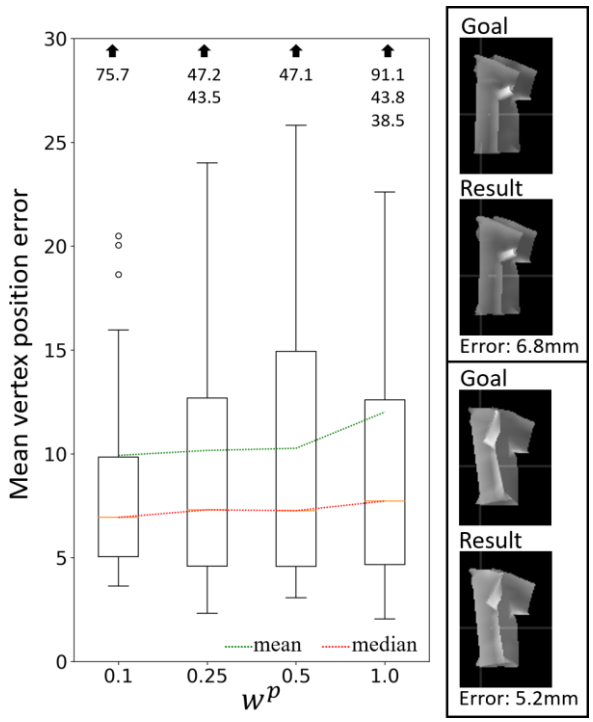

Fig. 3. Left: accuracy of trajectory planning for various levels of material uncertainty. Unit: millimetres. Error is measured as mean distance between goal position and obtained position over all vertices in the mesh. Numbers at top of graph indicate outliers. The same w^p value is used for all material properties. Right: example manipulation outcomes.

minus its first segment (which was executed), and again add Gaussian noise independently for each instance of the generation process. The number of segments in the manipulation sequence will be automatically adjusted within range $[1, b]$ by the generation process, so b should be set with some leeway. For the initial trajectory, we set $b = 25$. For subsequent generation rounds, we set b to the length of the preceding trajectory plus 5.

We set up the neural network with the number of passes through its prediction module set to b. We input s_i, $\hat{q}_i$, and m_i into the network, and perform forward propagation to obtain a sequence of probabilistic shape predictions $\hat{s}^m_{i,1:b} = \langle \hat{s}^m_{i,j} | j \in [1, \dots, b] \rangle$. We calculate the planning loss as follows.

$$L_{planning}(\hat{s}^m_{i,1:b}, s^*) = MSE(\hat{s}^m_{i,u}, s^*) + \beta \cdot L_{act}(m) \quad (4)$$

$$u = \begin{cases} \underset{k}{\operatorname{argmin}}[\theta_{i,k} \geq 180] & if \ \theta_{i,b} \geq 180 \\ q & else \end{cases} \quad (5)$$

Here MSE is the mean squared error, $\theta_{i,k}$ is the value of θ at segment k (i.e. the sum of all $\Delta\theta$ up to segment k), L_{act} is a loss promoting smooth, forward progressing trajectories, and β is a system parameter balancing the loss terms. MSE here ignores the $\sigma_{x,y,z}$ elements of the probabilistic shape prediction, using the $\mu_{x,y,z}$ variables as regular coordinates.

By backpropagating the planning loss through the network to the manipulation inputs, we obtain gradients for all elements of m_i. We update m_i on basis of these gradients, using the rProp- update rule [8]. We repeat this update procedure until m_i stabilises. As noted, multiple initialisations of m_i are optimised in parallel, and we select the solution that minimises the residual planning loss.

Note that the planning loss selects one shape $\hat{s}^m_{i,u}$ from $\hat{s}^m_{i,1:b}$ for the MSE calculation. The selected shape is the shape obtained right after the segment where the θ angle (which tracks progress along the trajectory arc) reaches 180°, i.e. the shape following release of the cloth. Where this point is reached depends on the control input, and can therefor change as the control input is updated. Segments after the cloth release point are meaningless, so we truncate them from the final solution. Hence, the number of segments in the returned solution is adjusted automatically.

We evaluate trajectory generation for various levels of knowledge about the cloth material, varying w^p from 0.1 to 1.0. For each case tested, c^p is set to a random value drawn uniformly from $[q^p - w^p/2, q^p + w^p/2]$. Goal shapes are sourced from the training data set. Figure 3 shows results. For low uncertainty ($w^p = 0.1$), the median error is 6.9mm. Although the difference is slight, mean and median error increase with w^p, indicating that material property knowledge contributes to trajectory generation accuracy. We also observe that higher uncertainty produces a larger spread of error values.

5. Material Estimation

After manipulating the cloth for $i \geq 1$ steps, we can estimate the material property information, using the manipulation history $m_{0:i-1} = \langle m_j | j \in [0, i) \rangle$ and the sequence of shape observations $s_{0:i}$ obtained over the course of that history. The procedure is similar to trajectory generation, but instead of fixing material property input and optimising the control inputs to maximise similarity between predicted outcome and goal shape, we fix the control input $m_{0:i-1}$, and optimise the material knowledge input $\hat{q}_i$ to maximise compatibility between the predicted shape sequence and the observed shape sequence.

If we have no preceding estimate of $\hat{q}_i$, as may often be the case when $i = 1$, we initialise $\hat{q}_i$ to the "no information" valuation, i.e. $\hat{q}_i = (c^p_i, w^p_i) = (0.5, 1.0), p \in P$. Otherwise, we initialise with the previous estimate. In both cases we apply Gaussian noise to the initialisation.

We set up the prediction network with the number of passes through the prediction module set to i. We run prediction with s_0 as state input, $m_{0:i-1}$ as control input, and current material estimate $\hat{q}_i$ as material information input. This yields a sequence of probabilistic shape predictions $\hat{s}^m_{1:i}$. We then calculate the material estimation loss as the negative log-likelihood of the observed sequence w.r.t. the predicted sequence:

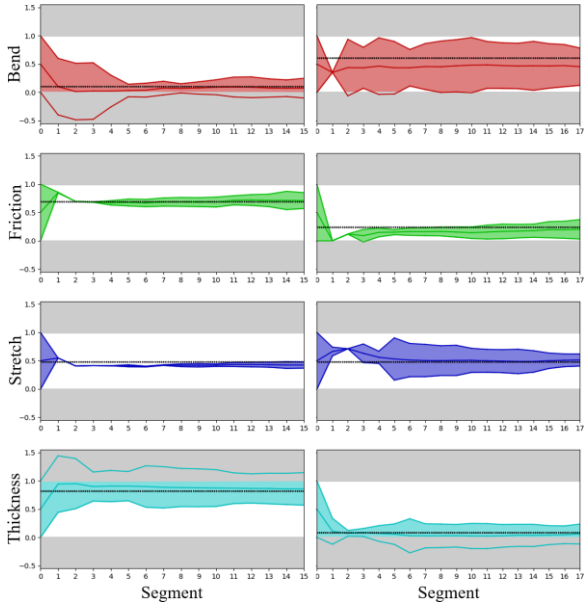

Fig. 4. Examples of iterative material estimation. Dotted black line: q^p. Shaded area: estimated region, (c^p and $c^p \pm w^p/2$ marked with coloured lines). All material properties normalised to the $[0,1]$ range.

Table 1: Material estimation results (mean, SD, median).

$N = 50$	**Bend**	**Friction**	**Stretch**	**Thickness**
$\|c^p - q^p\|$	.061 (.053) median: .044	.043 (.034) median: .034	.049 (.033) median: .044	.044 (.032) median: .037
w^p	.28 (.18) median: .27	.29 (0.17) median: .28	.28 (.17) median: .25	.25 (.16) median: .24
Failure*	8%	2%	2%	0%

* Failure criterion: $\|c^p - q^p\| > 0.1$ & $c^p \notin [c\hat{}p - w\hat{}p\ /\ 2,\ c\hat{}p + w\hat{}p\ /\ 2]$.

$$L_{estimate}(\hat{s}^m_{1:i}, s_{1:i}) = NLL(\hat{s}^m_{1:i}, s_{1:i}) \qquad (6)$$

By backpropagating the estimation loss through the network to the material inputs, we obtain gradients for all elements of $\hat{q}_i$. We update $\hat{q}_i$ on basis of these gradients, using the rProp- update rule [8]. We repeat this procedure until $\hat{q}_i$ stabilises. As before, multiple initialisations are optimised in parallel, and we finally select the candidate solution that minimises the residual estimation loss.

We evaluate estimation ability by replaying manipulation examples from the training data set, and running material estimation with the manipulation history and shape observation history after each manipulation segment. Table 1 reports quantitative results of estimation after the final segment, as mean, standard deviation, and median of $\|c^p - q^p\|$ and w^p, and failure rates. The value of c^p lies near q^p on average and the w^p is fairly well constrained. One problem we observe is that for cases where c^p closely approximates q^p, w^p can drop to zero (i.e. the system overshoots to zero when a small w^p would be appropriate). Figure 4 shows examples of estimation development over two cases. Note that not all manipulation trajectories allow identification of all material properties; a manipulation that is not affected much by a given property will not be informative with regard to that property. Probabilistic representation of material properties allows the system to properly express such absence of knowledge.

6. Conclusions & Future Work

We proposed a cloth manipulation system capable of generating manipulation trajectories and inferring material properties during manipulation. We showed that good accuracy is obtained for both functionalities. However, the impact of material knowledge in our test case is small, limiting potential trajectory planning accuracy gains from material inference. We plan to explore more complex folding scenarios in which material properties have a larger impact. A limitation of the results presented here is that trajectory generation and material estimation are evaluated separately. The next steps for this work will be to integrate these processes to operate in parallel, and integrate the system with robot hardware.

Acknowledgements

This work is partly supported by NEDO and JSPS KAKENHI Grant Number JP20H04262.

References

[1] V. Petrík, V. Kyrki: Feedback-based Fabric Strip Folding. IROS 2019.

[2] K. Kawaharazuka, A. Miki, M. Bando, K. Okada, M. Inaba: Dynamic Cloth Manipulation Considering Variable Stiffness and Material Change Using Deep Predictive Model With Parametric Bias. *Front. Neurorobot. 16:890695.*, 2022.

[3] D. Tanaka, S. Arnold, K. Yamazaki: Disruption-Resistant Deformable Object Manipulation on basis of Online Shape Estimation and Prediction-Driven Trajectory Correction. *IEEE RA-L* 6(2): 3809-3816, 2021.

[4] S. Arnold, K. Yamazaki: Cloth Manipulation Planning on Basis of Mesh Representations with Incomplete Domain Knowledge and Voxel-to-Mesh Estimation. *arXiv:2103.08137*, 2021.

[5] S. Arnold, K. Yamazaki: Fast and Flexible Multi-Step Cloth Manipulation Planning using an Encode-Manipulate-Decode Network (EM*D Net). *Front. in Neurorobot.*, vol. 13, 2019.

[6] G. Daviet, R. Narain, F. Bertails-Descoubes, M. Overby, G. Brown, L. Boissieux, J. Li: An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2018.

[7] J. Bernstein, Y. Wang, K. Azizzadenesheli, A. Anandkumar: signSGD: Compressed Optimisation for Non-Convex Problems. *arXiv:1802.04434*, 2018.

[8] C. Igel, M. Hüsken: Improving the Rprop Learning Algorithm. *Proceedings of the Second International Symposium on Neural Computation, NC'2000*, 2000.