Learning Simulatable Models of Cloth with Spatially-varying Constitutive Properties for Robotics

Guanxiong Chen[1], Shashwat Suri[1], Yuhao Wu[1], Ganidhu K. Abeysirigoonawardena[1],
Etienne Vouga[2], David I.W. Levin[3], Dinesh K. Pai[1],
[1] University of British Columbia, [2] University of Texas Austin,
[3] University of Toronto

Abstract

Real fabrics exhibit complex spatial variation from stitching, hemming, and other processes. Simulating these with finite element methods is computationally demanding, and suffers from membrane locking artifacts that make cloth artificially stiff. We introduce Mass-Spring Net, a learned, simulatable model that can model complex, spatially-varying fabric behavior from motion observations alone. Our approach accurately models spatially varying properties, is robust to membrane locking, and can potentially enable fast fabric manipulation in robotics. Compared to prior work our method achieves much-faster training time, resists membrane locking that exist in synthetic training data, and early results show that it maintains high accuracy and good generalization to novel scenarios.

1. Introduction

Accurate and fast simulation of cloth surrogates is crucial for robotics simulation and training [7, 11, 17, 19], particularly for fabric manipulation tasks [2, 4, 12, 14–16, 19, 20]. Oftentimes, these applications require real-time performance, while maintaining physical realism to enable effective robot learning and control. However, modeling and simulation of real-world cloth objects remains challenging for several reasons. First, real cloth objects often exhibit complex, spatially varying constitutive properties that significantly affect behavior. A simple example is shown in Fig. 1. Heat-shrinking, stitching, printing, and bonding introduce spatially varying material properties that are challenging to model using standard methods. Second, FEM cloth simulations can suffer from numerical artifacts such as membrane locking [13], which lead to unrealistic bending behavior when meshes are not finely discretized. Previous work has attempted to address this issue by simulating up to fine scales [18] or applying constraints to solving dynamics [1, 5]. This problem is particularly pronounced in real-time applications where coarse meshes are necessary for computational efficiency.

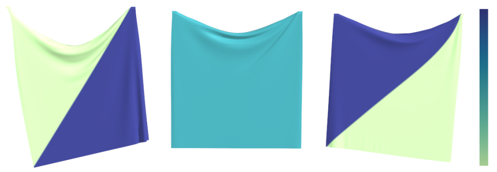
Figure 1. Equilibrium configurations of a square piece of cloth with spatially heterogeneous (left, right) vs. using homogeneous material (middle). Purple regions are stiffer than yellow ones. For the homogeneous cloth all triangles share the same stiffness, as in previous work, which is taken to be the average stiffness of triangles in the heterogeneous cloth. Despite identical initial and boundary conditions, stiffness variation leads to distinct behaviors. Our method can capture such variation.

In this work, we introduce a learning-based framework, called *Mass-Spring Net*, which addresses physical accuracy and efficiency using mass-spring-based systems, similar to recent recent work such as PhysTwin [4] and Spring-Gaus [21]. Our main contributions include: (1) Ability to model spatially-varying constitutive properties of fabrics; (2) Resistance to membrane locking problems that plague FEM-based simulations; (3) Minimal training requirements using only point cloud data and total mass.

2. Method

Learning task definition. Fig. 2 shows the neural surrogate modeling pipeline and the architecture of our constitutive model. Our method takes as input a *reference system*: T frames of motion of a piece of cloth, with N landmark points on the cloth tracked from frame to frame. In addition to the positions $\mathbf{y}_i \in \mathbb{R}^{3N}$ of the landmarks at each frame i, we assume the reference system also provides the external force $\mathbf{f}_{\text{land},i}$ acting on each landmark at each frame (e.g. gravity), an estimate of the cloth's area density ρ, and a binary classification of each landmark as either free or

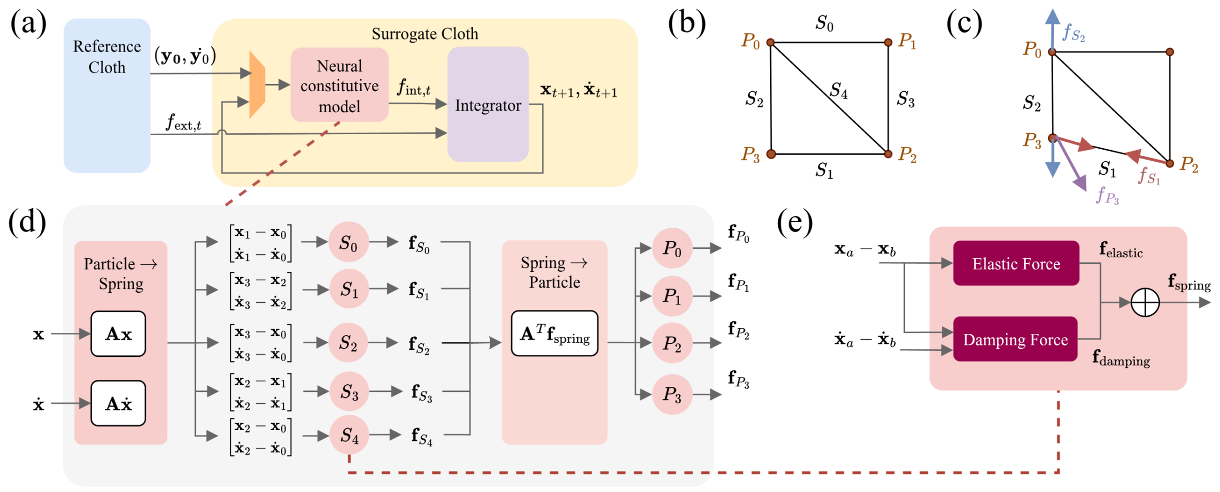
(a)
Reference Cloth
$(\mathbf{y}_0, \dot{\mathbf{y}}_0)$
Surrogate Cloth
Neural constitutive model
$f_{\text{int},t}$
Integrator
$\mathbf{x}_{t+1}, \dot{\mathbf{x}}_{t+1}$
$f_{\text{ext},t}$
(b)
(c)
(d)
Particle → Spring
Spring → Particle
$\mathbf{A}^T\mathbf{f}_{\text{spring}}$
(e)
$\mathbf{x}_a - \mathbf{x}_b$
$\dot{\mathbf{x}}_a - \dot{\mathbf{x}}_b$
Elastic Force
Damping Force
$\mathbf{f}_{\text{elastic}}$
$\mathbf{f}_{\text{damping}}$
$\mathbf{f}_{\text{spring}}$

Figure 2. Method overview. (a) The material modeling pipeline. We sample system state $(\mathbf{y}_0, \dot{\mathbf{y}}_0)$ from the reference cloth at the first time step of a rollout, resample following the scheme in Sec. 5 to get target particle positions, feed it to the neural constitutive model h_θ to predict neural particle force $\mathbf{f}_t$, then integrate. To further evolve the surrogate we use previously predicted states. (b)-(e) illustrates the architecture of our Mass-Spring Net which models a simple rectangular cloth, with rest configuration shown in (b) and a deformed configuration in (c). (d): Each neural spring (circle) takes the relative position and velocity of particles at its two ends, and predicts the spring's internal force. Then each particle accumulates forces applied by its surrounding springs. (e) Neural spring. At every time step, the module computes elastic and damping force and sums them up. Parts with learnable parameters are in dark red.

pinned in place. This reference system data might come from a high-fidelity finite element simulation, or from mocap markers pinned on a real-world cloth [2]. The surrogate system is a mass-spring network consisting of a user-specified number of particles P connected by S springs. The task is to learn stiffness and damping coefficient for each spring, from the reference system's landmark trajectories. As a preprocessing step, we map the given landmark trajectories to target positions $\hat{\mathbf{x}}_i \in \mathbb{R}^{3P}$ of each particle, and the external forces to forces $\mathbf{f}_{\text{ext},i}$ on each particle (Section 5). These quantities are used to supervise training of the neural constitutive model (Section 2.1).

Finally, given the predicted internal forces and the prescribed external forces and boundary conditions, we use semi-implicit Euler time integration to advance the state of the surrogate model from frame to frame. We learn the neural constitutive model parameters θ by comparing this simulated trajectory to the ground-truth motion observations $\hat{\mathbf{x}}_{i=1:T}$ (Section 2.2).

2.1. Neural Constitutive Modeling

As shown in Fig. 2 (d), we use the incidence matrix $\mathbf{A} \otimes I_{3\times3}$ to extract the displacement $\mathbf{d}$ and relative velocity $\mathbf{v}$ between each spring's two endpoints from the system position and velocity vectors $\mathbf{x}$ and $\dot{\mathbf{x}}$, as described in Liu et al. [6]'s method of fast simulation of mass-spring systems. The total internal force applied by each neural spring is computed as the sum of an elastic restoring force and viscous damping force (see Fig. 2 (e)):

$$\mathbf{f}_{\text{spring}} = \frac{k(\|\mathbf{d}\| - l_0)}{\|\mathbf{d}\|}\mathbf{d} + \frac{b(\mathbf{v}\cdot\mathbf{d})}{\|\mathbf{d}\|^2}\mathbf{d}, \tag{1}$$

where the elastic and damping coefficient $k_{s=1:S}$, $b_{s=1:S}$ are learnable parameters.

2.2. Simulation and Training

Forward dynamics. We set the surrogate system's initial conditions based on the target positions computed in Section 5, i.e.

$$\mathbf{x}_0 = \hat{\mathbf{x}}_0, \quad \dot{\mathbf{x}}_0 = \frac{\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0}{\Delta t},$$

where Δt is the reference system time step. We step these initial conditions forward in time with semi-implicit Euler integration. In particular, we update velocity using

$$\mathbf{M}(\dot{\mathbf{x}}_{j+1} - \dot{\mathbf{x}}_j) = \Delta t\ [\mathbf{f}_{\text{ext},j} + \mathbf{f}_{\text{springs}}(\mathbf{x}_j)], \tag{2}$$

where $\mathbf{M}$ is the $\mathbb{R}^{3P\times3P}$ diagonal mass matrix, and position using $\mathbf{x}_{j+1} = \mathbf{x}_j + \Delta t\, \dot{\mathbf{x}}_{j+1}$.

Loss formulation. We iterate the above time integration process to simulate an entire trajectory $\mathbf{x}_{j=1:T}$. We compare this trajectory to the target trajectory $\hat{\mathbf{x}}_{j=1:T}$ using the loss

$$L = \lambda_{\text{f}} L_{\text{f}} + \lambda_{\text{J}} L_{\text{J}} + \lambda_{\text{k neg.}} L_{\text{k neg.}} + \lambda_{\text{b neg.}} L_{\text{b neg.}}. \tag{3}$$

In Eq. 3, L_{f} denotes the force loss, which penalizes difference between target and simulated net force applied on each particle in the system at each time step. The target net force is estimated from the target positions and Newton's Second Law:

$$L_{\mathrm{f}} = \frac{1}{P(T-2)} \sum_{i=2}^{T-1} ||\mathbf{f}_i - \hat{\mathbf{f}}_i||_2^2 \tag{4}$$

$$\mathbf{f}_i = \mathbf{M} \left(\frac{\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{(\Delta t)^2} \right), \tag{5}$$

where $\mathbf{f}_i = \mathbf{f}_{\text{ext},i} + \mathbf{f}_{\text{springs}}(\mathbf{x}_i)$ denotes the net force (including gravity, environmental damping, and the spring forces) applied to each particle at time step i of forward dynamics. L_{J} denotes the impulse loss, which penalizes difference between target and predicted system impulse:

$$L_{\mathrm{J}} = \frac{1}{P} \left\| \mathbf{J} - \hat{\mathbf{J}} \right\|_2^2, \tag{6}$$

where $\mathbf{J}$ is the accumulated impulse on each particle from the start to the end of the trajectory, integrated using the Trapezoid Rule:

$$\mathbf{J} = \frac{1}{2} \sum_{j=1}^{T-1} (\mathbf{f}_j + \mathbf{f}_{j+1}) \Delta t, \tag{7}$$

and $\hat{\mathbf{J}}$ is computed analogously from the $\hat{\mathbf{f}}$. Intuitively, the force and impulse loss terms act analogously to the P and I terms in a PID controller, where the former penalizes discrepancy between predicted and target instantaneous forces applied to each particle and the latter penalizes error accumulation over time. We notice through experiments that the optimal weights λ_J, λ_f vary across different sources of data (i.e. reference systems) and need to be fine-tuned, just as the weights of a PID controller must be tuned when a mechanical system's parameters change.

The final two loss terms of Equation (3) are regularization terms that penalizes non-physical, negative spring stiffness or damping constants. Specifically,

$$L_{\text{k neg.}} = \sum_{s=1}^{S} \mathrm{ReLU}(-k_s) \tag{8}$$

$$L_{\text{b neg.}} = \sum_{s=1}^{S} \mathrm{ReLU}(-b_s), \tag{9}$$

where the k and b are the learned per-spring parameters of our neural constitutive model (see Sec. 2.1). The term stays at zero as long as the estimated parameter stays positive.

3. Experiments

3.1. Implementation

We implement our differential simulation pipeline using NVIDIA Warp [8] and PyTorch [10].

3.2. Reconstructing Spatially Varying Materials

Here we show that Mass-Spring Net can accurately estimate the stiffness (k) and damping (b) properties of a reference cloth with spatially varying stiffness, using the RMSE metric. We build the reference cloth with masses and springs, and we construct a surrogate system with the same resolution as the reference cloth. Bending and the complete set of shear springs are included in both the reference and the surrogate. This allows us to establish a one-to-one mapping from springs in the reference cloth to springs in the surrogate cloth. We also assess the quality of motion reconstruction using the RMSE of motion reconstruction.

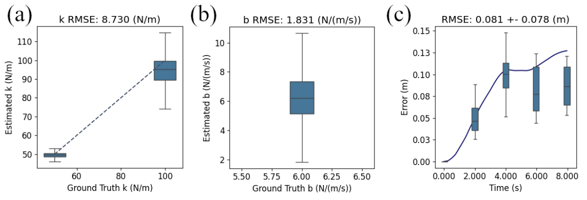

Figure 3. Predicted vs ground-truth (a) spring stiffness constants; (b) damping constants; (c) per-frame motion reconstruction RMSE. The curve shows mean RMSE across 16 test rollouts on each time step; we subsample five steps to visualize the distribution across rollouts. Each cloth contains $5,826$ springs.

Fig. 3 shows the accuracy of stiffness and damping estimates, and the reconstruction error obtained from a test set which contains 16 8 s-long rollouts. The reference cloth we used to generate training and test rollouts is a square-shaped, 15.5×15.5 m cloth discretized into a 32×32 particle grid connected by springs. The reference cloth contains springs with three stiffnesses: 10, 50 and 100 N/m, and all springs share the same damping constant of 6 $\mathrm{N/m/s}$. Fig. 3 (a) tells that most of our estimated stiffnesses are close to the ground-truth. The damping estimates are also close to the ground-truth on average, despite some variations: see (b).

3.3. Immunity to Membrane Locking

Finite element methods are susceptible to membrane locking, a phenomenon that produces artificial bending stiffness that is more significant at coarse resolutions [13]. An interesting consequence of our approach, which is learning material properties from force and impulse losses and not the shape of the draped cloth, is that we are able to model material properties even when the training data generated from a low-resolution FEM-based simulation is significantly affected by the numerical limitations of membrane locking. Thus, our simulation results match higher-resolution FEM simulations more closely than the training data– see Fig. 4.

Moreover, we offer a qualitative comparison with MeshGraphNet (MGN) [11] in terms of the two surrogate's be-

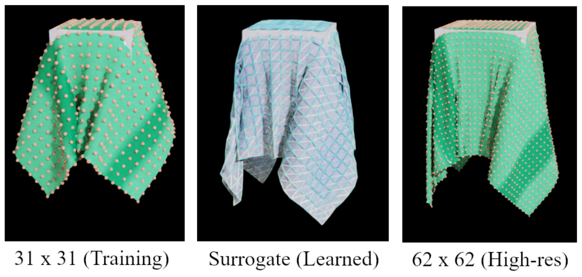

Figure 4. Our 32x32 surrogate (center) was trained on a data from a 31x31 FEM mesh (left) that exhibits significant membrane locking. This is apparent in the drape test of the cloth on a square table. Remarkably, our results more closely match the results of a higher resolution FEM cloth simulation (right), indicating that our model captures material properties well.

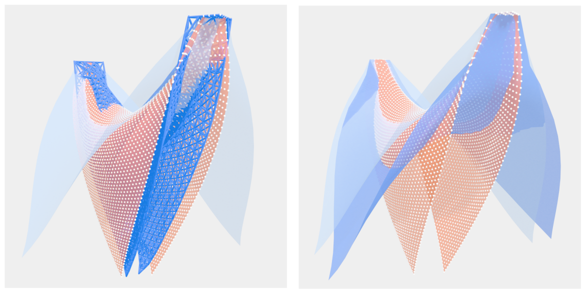

Figure 5. Mass-Spring Net (left, dark blue) vs MGN (right, dark blue) trained on kinematic data sampled from low-resolution (16×16) mesh (light blue), settled to the misaligned configuration in evaluation. A high-resolution (61×61) mesh (red) is placed in each scene for reference.

havior in the misaligned equilibrium configuration. Notice that in Fig. 5 MGN matches the low-resolution mesh's behavior more than the high-resolution mesh's, which makes sense, because the surrogate is trained to model the behavior of the low-resolution mesh when boundary conditions and trajectories are provided as inputs. Yet Mass-Spring Net is robust to numerical artifacts that exist in the training data. Moreover training of Mass-Spring Net takes orders of magnitude faster: 80 minutes vs 36 hours.

3.4. Interaction with Rigid Objects

We place the trained surrogate cloth models from the above experiment under a new scene, in which all particles on the surrogate cloth are free to move, and the surrogate cloth starts falling from time $t = 0s$ until it drops onto another object. We also instantiate the reference cloth in the same scene and set it to have the same initial state (position and orientation of the cloth) as the surrogate, and place an identical object below it for the reference cloth to drape onto. Figs. 4 and 6 show the steady state after each system settles. Note that as discussed in 3.3, the reference cloth used

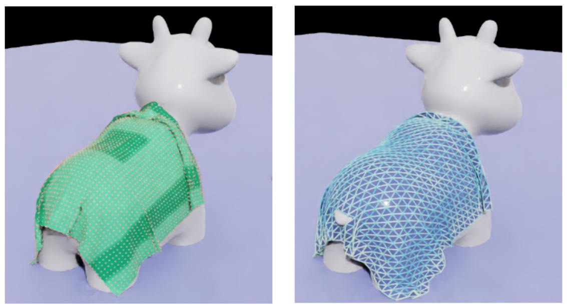

Figure 6. After simulation starts, under gravity the reference cloth (left) drapes onto a cow Crane et al. [3] beneath it and eventually settles to a steady state. Right: the surrogate cloth drapes onto a replica of the same cow.

here has the same physical dimension as the surrogate, but bears $2\times$ higher spatial resolution.

4. Conclusion and Future Work

We presented Mass-Spring Net, a surrogate model trained with a novel force-impulse loss and training curriculum that can learn complex, spatially-varying constitutive properties of a piece of simulated cloth without relying on a reference mesh. We demonstrated that our method can reproduce the observed behavior of both homogeneous and heterogeneous cloth sheets better than previous neural network approaches. It generalizes more readily to new scenarios, including those with previously unobserved contacts. Further, we showed that Mass-Spring Net's mass-spring-based architecture is immune to membrane locking that plagues FEM-based simulations.

Future Work. While our surrogate has shown promise, they have yet to capture the full complexity of nonlinear cloth behavior, and have yet been fitted to real-world data. Additionally, we aim to encapsulate our surrogate model within an Isaac Sim [9] plugin, enabling seamless integration with robotic assets for downstream manipulation tasks and facilitating more comprehensive robotics simulation workflows.

References

[1] Hsiao-yu Chen, Paul Kry, and Etienne Vouga. Locking-free simulation of isometric thin plates. *arXiv preprint arXiv:1911.05204*, 2019. 1

[2] Franco Coltraro, Júlia Borràs, Maria Alberich-Carramiñana, and Carme Torras. Tracking cloth deformation: A novel dataset for closing the sim-to-real gap for robotic cloth manipulation learning. *The International Journal of Robotics Research*, page 02783649251317617, 2025. 1, 2

[3] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 4

[4] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos. *arXiv preprint arXiv:2503.17973*, 2025. 1

[5] Ning Jin, Wenlong Lu, Zhenglin Geng, and Ronald P Fedkiw. Inequality cloth. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 1–10, 2017. 1

[6] Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6):1–7, 2013. 2

[7] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300, 2023. 1

[8] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, 2022. NVIDIA GPU Technology Conference (GTC). 3

[9] NVIDIA. Isaac Sim. 4

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3

[11] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 1, 3

[12] Yingdong Ru, Lipeng Zhuang, Zhuo He, Florent P Audonnet, and Gerardo Aragon-Caramasa. Can Real-to-Sim Approaches Capture Dynamic Fabric Behavior for Robotic Fabric Manipulation? 2025. 1

[13] Henryk Stolarski and Ted Belytschko. Membrane locking and reduced integration for curved elements. 1982. 1, 3

[14] Priya Sundaresan, Rika Antonova, and Jeannette Bohgl. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10828–10835. IEEE, 2022. 1

[15] Yuran Wang, Ruihai Wu, Yue Chen, Jiarui Wang, Jiaqi Liang, Ziyu Zhu, Haoran Geng, Jitendra Malik, Pieter Abbeel, and Hao Dong. DexGarmentLab: Dexterous Garment Manipulation Environment with Generalizable Policy. *arXiv preprint arXiv:2505.11032*, 2025.

[16] Xinyuan Yu, Siheng Zhao, Siyuan Luo, Gang Yang, and Lin Shao. Diffclothai: Differentiable cloth simulation with intersection-free frictional contact and differentiable two-way coupling with articulated rigid bodies. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 400–407. IEEE, 2023. 1

[17] Ziqiu Zeng, Siyuan Luo, Fan Shi, and Zhongkai Zhang. Fast but accurate: A real-time hyperelastic simulator with robust frictional contact. *arXiv preprint arXiv:2503.15078*, 2025. 1

[18] Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L James, and Danny M Kaufman. Progressive simulation for cloth quasistatics. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022. 1

[19] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-grid neural dynamics for learning deformable object models from rgb-d videos. *arXiv preprint arXiv:2506.15680*, 2025. 1

[20] Dongzhe Zheng, Siqiong Yao, Wenqiang Xu, and Cewu Lu. Differentiable cloth parameter identification and state estimation in manipulation. *IEEE Robotics and Automation Letters*, 9(3):2519–2526, 2024. 1

[21] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. *European Conference on Computer Vision (ECCV)*, 2024. 1

Learning Simulatable Models of Cloth with Spatially-varying Constitutive Properties for Robotics

Supplementary Material

5. Method - Spatial Discretization and Resampling

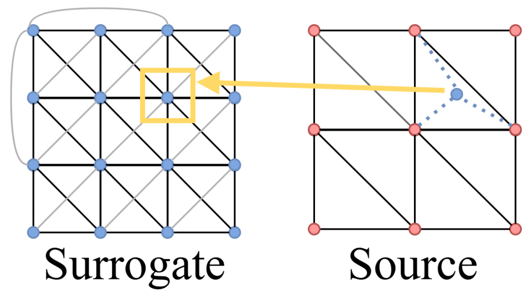

Figure 7. Spatial discretization of the surrogate system (left) and the reference system (right). Bending springs and diagonal springs from top-left to bottom-right can be optionally excluded on construction. For simplicity, we only show two bending springs; the full surrogate contains bending springs connected at stride $= 1$. The dotted lines show how we compute the target position of each surrogate particle from the positions of nearby reference landmarks via barycentric interpolation. Note that in training only landmarks from the reference are needed, and topolgical information are not necessary.

Given a desired number of surrogate particles P, we build our surrogate mass-spring network using an approach similar to how Sundaresan et al. [14] resamples reference point cloud: we isometrically unroll the reference system's cloth specimen into a rectangle in the plane, then discretize this rectangle with a P-vertex regular square grid, as shown in Fig. 7 (a). We place a surrogate spring at each edge of the grid, with the spring rest length l_0 determined by the edge length in this rest configuration.

As shown in Fig. 7 (b), the reference system landmarks do not necessarily correspond to the surrogate particles (and the resolution of the surrogate particle grid might be much coarser or finer than the density of landmark points). Therefore we must resample the given landmark positions $\mathbf{y}_i$ at each frame i to positions of the surrogate particles.

Let $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$ be the rest positions of the unrolled landmarks and surrogate particles within the 2D rectangle, respectively. For each surrogate particle p, we identify its three nearest neighbors $v_{\{1,2,3\}}$ among the $\bar{\mathbf{y}}_j$. Let $b_{\{1,2,3\}}$ be the barycentric coordinates of $\bar{\mathbf{x}}_p$ within triangle $\{\bar{\mathbf{y}}_{v_1}, \bar{\mathbf{y}}_{v_2}, \bar{\mathbf{y}}_{v_3}\}$. We set the target position $\hat{\mathbf{x}}_{p,i}$ of particle p at frame i to

$$\hat{\mathbf{x}}_{p,i} = b_0 \mathbf{y}_{v_0,i} + b_1 \mathbf{y}_{v_1,i} + b_2 \mathbf{y}_{v_2,i}. \qquad (10)$$

We map external forces to the surrogate particles using barycentric interpolation in an analogous way.

We also need to assign a mass m_p to each surrogate particle. We simply set $m_p = \rho A/P$, where A is the rectangle area.