Mechanics-Aware Modeling of Cloth Appearance

Zahra Montazeri, Chang Xiao, Yun (Raymond) Fei, Changxi Zheng, and Shuang Zhao

Abstract—Micro-appearance models have brought unprecedented fidelity and details to cloth rendering. Yet, these models neglect fabric mechanics: when a piece of cloth interacts with the environment, its yarn and fiber arrangement usually changes in response to external contact and tension forces. Since subtle changes of a fabric's microstructures can greatly affect its macroscopic appearance, mechanics-driven appearance variation of fabrics has been a phenomenon that remains to be captured.
We introduce a mechanics-aware model that adapts the microstructures of cloth yarns in a physics-based manner. Our technique works on two distinct physical scales: using physics-based simulations of individual yarns, we capture the rearrangement of yarn-level structures in response to external forces. These yarn structures are further enriched to obtain appearance-driving fiber-level details. The cross-scale enrichment is made practical through a new parameter fitting algorithm for simulation, an augmented procedural yarn model coupled with a custom-design regression neural network. We train the network using a dataset generated by joint simulations at both the yarn and the fiber levels. Through several examples, we demonstrate that our model is capable of synthesizing photorealistic cloth appearance in a mechanically plausible way.

Index Terms—Cloth appearance, cloth mechanics

1 Introduction

It is a universal phenomenon that a material's microstructure affects its macroscopic appearance profoundly. In light of this, computer graphics has made significant advances toward capturing realistic object appearance by modeling its microstructures (e.g., [9], [10], [16], [23]). Motivated by its utter prominence in virtual senses, cloth appearance has also been modeled microscopically. State-of-the-art *micro-appearance* models [20], [33], [35] can produce cloth renderings with stunning richness and details, by accounting for the fabric's *micro-geometric structures* (i.e., the arrangement of fibers and yarns).

We argue that the microstructures of cloth should be modeled not only geometrically but also *mechanically*. This is because the appearance-driving arrangement of a fabric's constituent fibers and yarns is by no means fixed. For instance, a simple stretch of a piece of fabric may thin its yarns, making the fabric more see-through and changing its shininess (see Figure 2). Even without external distortion, the interaction between individual yarns of a cloth can change yarn shapes and cause specific texture patterns to emerge [8], [18]. The mechanical response—the interplay between small-scale filament structures and their material properties—plays a significant role in a fabric's appearance.

From this vantage point, we introduce a cloth appearance model that incorporates mechanical responses of a fabric's microstructures for better capturing the fabric appearance and its changes under external forces.

A seemingly straightforward idea is to rely on physics-based simulation to reconfigure the structures of yarns and fibers. However, executing this idea needs to address a significant computational challenge. On the one hand, it has been shown that both the structure of fibers (i.e., how the individual filaments form a yarn) and the structure of yarns (i.e., how the individual yarns are interwoven) are instrumental for fabric appearance [6], [20], [27], [33], [35]. On the other hand, while there exist physics-based models that simulate individual yarns of a cloth [8], [19], simulating individual fibers is intractable, because even for a sizable cloth, a myriad of fibers must be included.

We tackle this challenge through a *bi-scale* approach: we simulate the cloth only at the *yarn level* and synthesize the *fiber-level* details in a mechanics-aware manner during post-processing. The synthesis of fibers is enabled by augmenting the procedural yarn model [27], [35] with spatially varying parameters derived from the simulated yarn states. These parameters are used to deform the procedurally generated fibers in the way that reflect the fabric's mechanical response. Since the relationship between the fiber-deforming parameters and the simulated yarn-level states is highly complex, we leverage a custom-design regression neural network to learn this relation.

The training dataset of our regression network is generated by running physics-based simulations at both the yarn level (where a cloth yarn is modeled as a single curve) and the fiber level (where a yarn is depicted as a bundle of fiber curves). To make the training simulations tractable, we design the neural network such that they involve only a small yarn segment. Furthermore, to ensure the consistency between the simulations at both scales, we develop a new parameter fitting algorithm that automatically determines the (homogenized) material properties at the yarn level using those from the fiber level.

At runtime, our method simulates the microstructures of a fabric at the yarn level. It then utilizes procedural yarn modeling coupled with the pre-trained regression network to generate the fabric's fiber-level geometries and adapt them to the simulated yarn states. In this way, we are able to produce diverse and physically plausible fabric appearance (Figure 1).

To our knowledge, this is the first mechanics-aware

- *Z. Montazeri and S. Zhao was with the Department of Computer Science, University of California, Irvine, CA, 92697. E-mail: zmontaze@uci.edu*
- *C. Xiao, Y. (R.) Fei and C. Zheng are with Columbia University, New York, NY, 10027*

Manuscript received May 12, 2019; revised May 12, 2019.

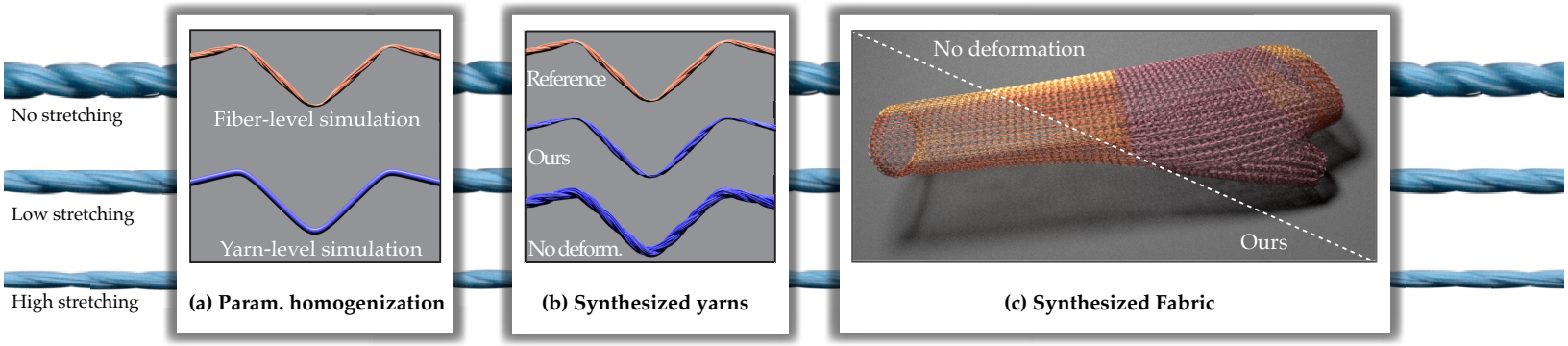


Fig. 1. We introduce a first-of-a-kind technique to model cloth appearance not only micro-geometrically but mechanically. (a) Our technique simulates the deformation of fibers at the yarn-level by leveraging a novel parameter homogenization process so that centerlines of a yarn simulated at the yarn and the fiber levels match. (b) By procedurally generating fibers around simulated yarn centerlines and deforming those fibers based on yarn-level external forces, our method synthesizes fiber microstructures that closely match reference fiber-level simulations. (c) Accurate reproduction of fiber-level micro-geometries is crucial as they greatly affect a fabric's overall appearance.

technique for modeling the appearance of cloth (or any material if that matters): existing techniques offering fiber-level details either rely on measured fiber geometries that are virtually impossible to animate [20], [33], or use procedurally generated fibers with fiber and yarn mechanics completely neglected [24], [27], [35]. Our technique is also independent of the underlying cloth simulation method: although we choose to use a state-of-the-art offline approach for maximized accuracy, other yarn-level simulation approach can be readily used as well. Our major technical contributions include:

- A *parameter homogenization* technique that fits yarn-level mechanical parameters so that yarn centerlines match between fiber-level and yarn-level simulations.
- A *custom-design regression neural network* that learns the mapping from simulated yarn-level states to fiber-deforming parameters.

We validate our method by comparing fiber-level simulations and photographs (Figures 13 and 16). To demonstrate the use of our method, we apply it to a few fabrics under various mechanical deformations, and show their appearance changes in response to the interaction with the environment (Figures 1, 17, and 18).

2 Related Work

Cloth appearance models: Modeling and reproducing the appearance of a cloth has been an active research area in computer graphics for decades. Traditionally, cloth are modeled as 2D thin sheets with the appearance described using generic [2], [30], [31] or specialized surface reflectance models [1], [15], [26]. These models offer adequate quality for reproducing cloth appearance when viewed from a distance where individual cloth yarns are barely visible. On the other hand, they lack the fidelity and details to produce plausible close-up renderings and the generality to capture fabrics' diverse visual effects (such as strong textures and grazing-angle highlights) usually arising from their characteristic microstructures.

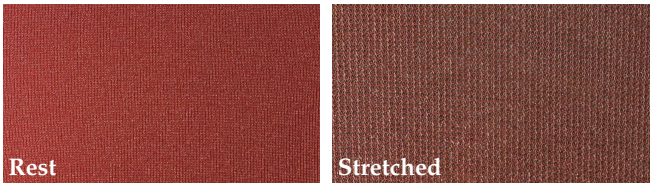


Fig. 2. **Appearance change of a real fabric** driven by external stretching. When a real piece of cloth is stretched, its internal yarn and fiber structures can change drastically, leading to significantly different overall appearance. In this example, we show photographs of a nylon fabric under rest and stretched states.

Recently, a family of micro-appearance models has been introduced to model fabric appearance [6], [20], [21], [24], [27], [32], [33], [34], [35]. Unlike earlier methods, these techniques explicitly model fabrics' fiber-level microstructures using high-resolution volumes or fiber meshes. Since these microstructures largely affect a fabric's overall appearance, micro-appearance models enjoy the unprecedented details and the generality to capture the appearance of a wide variety of fabrics.

Unfortunately, these models are mechanically agnostic: many of them rely on measured microstructures that cannot be mechanically deformed at all [20], [21], [33], [34]; others use specialized procedures that completely neglect fiber mechanics [24], [27], [32], [35], producing inaccurate overall appearance (e.g., see Figure 2). In short, these models are significantly limited when modeling animated fabrics.

Cloth and yarn simulation: Computer graphics has also seen a long history of simulating cloth and yarns. A thorough review can be found in the survey [29]. Oftentimes, a piece of cloth is modeled as a 2D surface, discretized with a triangular or quadrilateral mesh [3], [13]. Since a real cloth is woven by a set of fiber bundles (or *yarns*), a more faithful method is to model those yarns directly. The first of such models was introduced by Kaldor et al. [18], and later accelerated in [19].

The main challenge of simulating at yarn level is the high cost of resolving contacts. To address this issue, Cirio et al. [8] introduced sliding constraints to efficiently treat inter-yarn contacts. Our work is built on a Lagrangian/Eulerian approach of resolving contacts [17]. Extending the material point method [28], this approach represents yarns as poly-lines embedded in a background Eulerian grid, and resolves

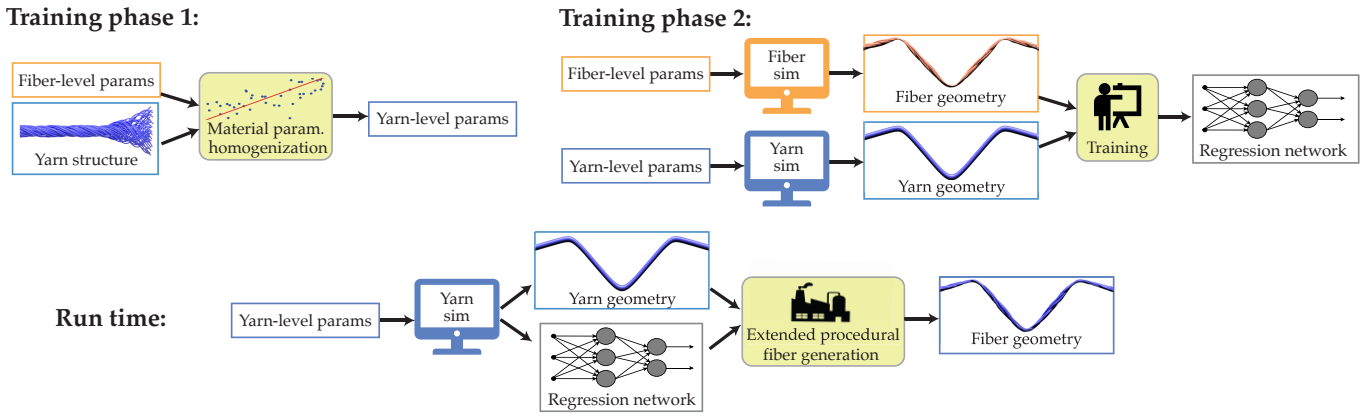


Fig. 3. **Method overview. (top)** In the preprocessing phase, our method has two sub-steps: homogenizing the material parameters for yarn-level simulation (left) and training the regression neural network (right). **(bottom)** At runtime, we capture the rearrangement of yarn-level structures in a fabric, and enrich it with fiber-level geometries with the neural network. The fiber-level geometries then determine the macroscopic appearance of the fabric.

collisions by modeling them as volumetric deformations. In the same framework, this method can also simulate thin shells, while ignoring bending and twisting forces. This limitation is addressed recently for mesh-based thin shells [14]. We address this limitation for simulating individual yarns by incorporating the elastic rod model [4], similar to the recent work by Fei et al. [11].

Despite of these advances, simulating individual fibers in a sizable cloth remains intractable, because an excessively large number of fibers is needed. While Jiang et al. [17] managed to simulate multiple "threads" per yarn in their examples, the number of threads is still much smaller than the typical number of fibers in a real yarn. In this work, we sidestep the challenge of simulating fibers fully. Our simulation still performs at the yarn level, but we design a deep neural network to "upsample" the yarn-level simulation results to obtain fiber-level details.

Cloth rendering vs. simulation: Historically, the research on cloth appearance modeling and physics-based simulation advance largely in parallel, probably because the two lines of research have been traditionally considered from different physical perspectives (optics vs. mechanics). Although cloth simulation results are often demonstrated with realistic rendering, it is very rare, if not at all, that a cloth appearance model has benefited from physics-based simulation. In this work, we show that the two seemingly separated types of models can work in tandem, yielding an improved cloth appearance model.

3 Method Overview

The structure of fabrics is complex at multiple scales. As a whole, a fabric consists of thousands of threads, or *yarns*, combined via manufacturing techniques like weaving and knitting. Each yarn is, in turn, comprised of tens to hundreds of micron-radius filaments, or *fibers*. Since a fabric's yarn and fiber arrangements significantly affects its overall appearance, our goal is to capture cloth's mechanical response down to the fiber level.

To this end, we leverage physics-based simulation to capture a fabric's yarn- and fiber-level mechanics. Unfortunately, simulating the dynamics of all fibers in an entire fabric is generally intractable. Instead, we propose to simulate at yarn level (at runtime), and then procedurally generate fibers around simulated yarns in a mechanics-aware manner to complete the appearance model.

To enable mechanics-aware generation of fiber structures, our method consists of three major components:

- *Training.* In our training phase, we simulate a single yarn under a range of conditions at both the yarn and fiber levels. For each pair of simulation results, we compare the simulated fiber curves and procedurally generated ones (guided by the simulated yarn centerline). The discrepancy between the two is learned by a custom-design regression network, one that maps the simulated yarn-level states (i.e., mechanical forces) to the discrepancy-compensating fiber deformations.
- *Material parameter homogenization.* The training step demands consistency between the yarn- and fiber-level simulations. Therefore, given the material properties of individual fibers, we determine their yarn-level counterparts to ensure the consistency between yarn centerlines simulated at both levels (under identical initial conditions). This is achieved by fitting (or homogenizing) yarn-level material parameters using fiber-level simulations.
- *Runtime.* At runtime, we simulate the dynamics of full fabrics only at the yarn level. With individual yarn centerlines and forces along them determined by the simulation, we procedurally generate fibers following these centerlines and use the trained regression network to further adjust the fibers so that they respond to yarn-level forces such as stretching and compression properly.

The pipeline described above is also summarized in Figure 3. In the following sections, we detail our training stage in §4, parameter fitting in §5, and runtime phase in §6.

4 Training Phase

The objective of our training phase is to acquire a mapping between yarn-level deformation and the corresponding fiber reconfiguration. This mapping will be used at run-time to procedurally generate fiber structures in a mechanics-aware manner. To learn the mapping, we simulate a single yarn under a few predetermined configurations at both the yarn

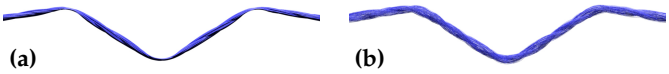


Fig. 4. **Deforming fibers using yarn-level deformation gradients** generally leads to unsatisfactory results (b) compared to the reference (a) since they effectively capture average deformation within full grid cells that are much larger than individual fibers.

and the fiber levels. These training simulations are set up under identical conditions to provide consistent dynamics across both levels.

We will elaborate our simulation models and parameter choices in §5. In this section, we focus on our main training phase. In §4.1, we briefly revisit the procedural approach for generating cloth models with fiber-level details. Then, in §4.2, we show how this procedure can be extended to become mechanics-aware by leveraging a custom-design regression neural network.

4.1 Background: Procedural Modeling of Fiber Geometry

To generate a cloth model with fiber-level details, a procedural approach originated in textile design has been introduced to graphics recently [27], [35]. This approach takes as input a set of yarn centerlines and synthesizes fiber curves around each yarn independently.

For a yarn with its centerline following the z-axis, each of the constituent fiber j is modeled as a circular helix:

$$\boldsymbol{h}_j(z) = \left[r_j \cos\left(\theta_j + \frac{2\pi z}{a}\right),\ r_j \sin\left(\theta_j + \frac{2\pi z}{a}\right),\ z \right]^\top, \tag{1}$$

where r_j, a and θ_j control the radius, pitch and initial phase of the helix $\boldsymbol{h}_j$, respectively. When building a procedural yarn, the radius r_j of each fiber is drawn independently from a probability distribution

$$p(r;\ \epsilon, \beta) \propto (1 - 2\epsilon_1) \left(\frac{\mathrm{e} - \mathrm{e}^r}{\mathrm{e} - 1}\right)^{\epsilon_2}, \tag{2}$$

where ϵ_1 and ϵ_2 are two parameters shared by all fibers.

To better model the migration of fibers, Eq. (1) can be extended to allow the helix radius r_j to vary with z. A typical implementation of this idea is to have $r_j(z)$ changing under a sinusoidal pattern between two predefined minimal and maximal values. Additionally, real-world yarns usually contain multiple substrands, or plies. In this case, each ply can be described using Eqs. (1, 2). Then, the yarn is formed by twisting all the constituent plies around a common centerline. Please refer to the prior works [27], [35] for more details on fiber migration and multi-ply yarn modeling.

Lastly, given a centerline $\boldsymbol{\alpha}$ expressed as a 1D curve parameterized by arc length z with normal $\boldsymbol{n}_{\boldsymbol{\alpha}}(z)$ and binormal $\boldsymbol{b}_{\boldsymbol{\alpha}}(z)$, a straight yarn centered at the z-axis can be warped to follow this centerline by transforming each fiber $\boldsymbol{h}_j$ to $\boldsymbol{h}_j^{\boldsymbol{\alpha}}$ given by

$$\boldsymbol{h}_j^{\boldsymbol{\alpha}}(z) = \boldsymbol{\alpha}(z) + [\boldsymbol{h}_j(z)]_x \boldsymbol{n}_{\boldsymbol{\alpha}}(z) + [\boldsymbol{h}_j(z)]_y \boldsymbol{b}_{\boldsymbol{\alpha}}(z), \tag{3}$$

where $[\boldsymbol{h}_j(z)]_x$ and $[\boldsymbol{h}_j(z)]_y$ denote the x- and y-components of $\boldsymbol{h}_j(z) \in \mathbb{R}^3$, respectively.

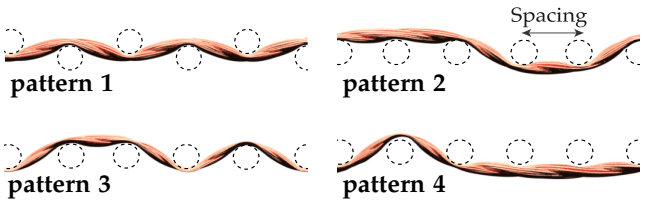


Fig. 5. **Simulation for training.** We simulate the deformation of a yarn under a range of configurations. In particular, we use four common patterns [34] and six different cylinder spacings per pattern. Please refer to the supplemental material for the complete set of configurations.

4.2 Learning How Fibers Deform

The procedural model outlined in Eqs. (1–3) does not capture the *deformation* of yarns due to their mechanical responses (e.g., contacting with other yarns and objects or being stretched).

To address this problem, we augment the procedural yarn model with a *cross-sectional affine transformation* $\boldsymbol{T} \in \mathbb{R}^{2\times 2}$ that varies spatially. Specifically, $\boldsymbol{T}$ transforms each fiber $\boldsymbol{h}_j$ to:

$$\boldsymbol{h}'_j(z) := \begin{bmatrix} \boldsymbol{T}(z) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{h}_j(z), \tag{4}$$

which can in turn be warped to follow arbitrary yarn centerlines as before using Eq. (3). Based on this formulation, the mechanics-aware modeling of a fabric's fiber microstructures boils down to finding proper cross-sectional transformations $\boldsymbol{T}$. This, however, is challenging since cloth fibers usually react to mechanical forces in complicated ways based on their material properties and geometric arrangements.

We aim to obtain the cross-sectional transformations $\boldsymbol{T}$ using only results simulated at the yarn level. During these simulations, a yarn is represented as a polyline where each segment is associated with a deformation gradient $\boldsymbol{F}$ that describes the local volumetric deformation. We provide more details on physics-based simulation of cloth yarns and fibers in §5.

One naïve approach is to directly use the yarn-level deformation gradients to transform the fibers. Unfortunately, this generally leads to unsatisfactory results, as demonstrated in Figure 4.

We instead leverage a custom-design regression neural network to infer the cross-sectional transformations $\boldsymbol{T}$ based on yarn-level deformation gradients $\boldsymbol{F}$. Our goal here is to learn a mapping between the deformation gradients $\boldsymbol{F}$ and the cross-sectional deformations $\boldsymbol{T}$ so that the latter can be used to deform the procedurally generated fibers via Eq. (4). Since different types of yarns (e.g., cotton vs. silk) generally exhibit distinctive fiber structures that cause the yarns to react to external forces differently, we learn the mapping for each yarn separately.

4.2.1 *Training data generation*

We generate training data by performing a series of simulations of a single yarn at both the fiber and the yarn levels. To ensure that the training dataset covers a wide range of yarn and fiber deformations driven by mechanical forces, we configure our training simulations to include a single yarn compressed by a few rigid cylinders. We use multiple cylinder arrangements for the simulations (Figure 5), and collect the results as our training dataset.

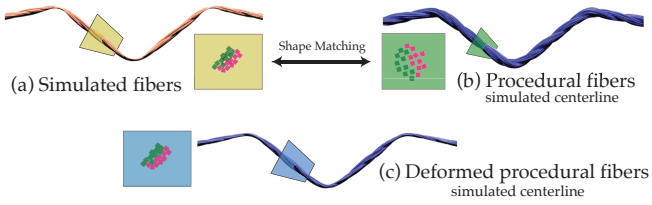


Fig. 6. **Computing cross-sectional fiber deformations.** We simulate a yarn at both fiber (a) and yarn (b) levels. At the yarn level, we procedurally generate fibers following the simulated centerline (which agrees with the fiber-level simulation thanks to the parameter homogenization described in §5). For each cross section along both yarns (from yarn- and fiber-level simulations), we compute the intersections of the fibers on the cross-sectional plane to obtain a 2D point cloud per yarn. Lastly, we perform 2D shape matching between the two point clouds, and obtain the desired fiber deformation matrix $\boldsymbol{T}$. By deforming the procedurally generated fibers, the resulting fiber microstructure (c) matches the results of fiber-level simulation closely.

Training simulations: For each simulation configuration, as shown in Figure 6-a, we use the procedural approach depicted in §4.1 to generate the constituent fibers of a straight yarn as the initial state of the simulation. Then, we quasi-statically move a few rigid cylinders toward the yarn and simulate the mechanical response of the fibers. We record the shape of all fiber curves after each time step.

At the yarn level, we simulate how a yarn centerline reacts to the same quasi-statically moving cylinders (Figure 6-b). Thanks to our parameter homogenization step (which will be detailed in §5), the simulated yarn centerline closely agrees with that from the fiber-level simulation. After each time step, we record the yarn centerline as well as the normal and binormal directions (computed using the simulated twisting angles) and the deformation gradients along the yarn. Using the recorded centerline and normals, we generate fiber curves following the simulated centerline using Eqs. (1–3) with no deformation.

Shape matching: Given results from the simulations at both levels, we now compute $\boldsymbol{T}(z)$ along the yarn centerline to compensate for the discrepancy between the simulated fiber curves and the procedurally generated ones. We discretize $\boldsymbol{T}(z)$ at the edge centers of the yarn polyline. At each edge center, we define a cross-sectional plane perpendicular to the tangential direction along the centerline. Then, we compute the intersections between the cross-sectional plane and the fiber curves (Figure 6-ab). Let $\mathcal{P}_{\text{fiber}}$ and $\mathcal{P}_{\text{yarn}}$ denote the cross-sectional fiber locations obtained using simulated fiber curves and procedurally generated ones, respectively. Then, the desired cross-sectional transformation $\boldsymbol{T}$ should warp $\mathcal{P}_{\text{yarn}}$ to match $\mathcal{P}_{\text{fiber}}$ as closely as possible. This can be formulated as a shape matching problem [25] in 2D. By applying the computed deformations $\boldsymbol{T}$ via Eq. (4), the procedurally generated fibers can closely match the fully simulated fibers (Figure 6-c).

4.2.2 Our regression network

Next, we train a regression neural network to learn the relation between deformation gradients simulated at the yarn level and cross-sectional transformations of cross-sectional fiber locations. At runtime, the latter will be estimated using the network *without* resort to fiber-level simulations.

Input and output: The deformation $\boldsymbol{T}$ at any cross section along the yarn mostly depends on the external forces

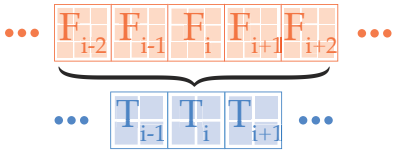


Fig. 7. **Input and output** to our neural network. Our neural network takes as input deformation gradients given by yarn-level simulation in a small neighborhood window around a cross section, and estimates the cross-sectional deformation at that cross section.

acting at a small neighborhood around the cross section. Thus, when our regression network estimates $\boldsymbol{T}$ at a cross section, it takes as input the simulated deformation gradients in a small window around that cross section (Figure 7). In other words, our network acts like a (nonlinear) 1D filter along yarn centerlines.

Transforming training data to local frames: Since simulated yarn curves can be rotated without affecting their internal structures, it is desired for the input and output of our regression network to be rotationally invariant. Yet, our deformation gradients do vary under rotations since they effectively transform individual segments of the yarn polyline from their canonical material spaces (where the tangential direction is aligned with the x-axis) to the world space. This observation suggests that we should transform the simulated deformation gradients into a *local* frame of reference to ensure rotational invariance.

Consider a local neighborhood of cross sections (Figure 7), we define the local frame of reference by specifying the tangent $\boldsymbol{t}$, normal $\boldsymbol{n}$, and binormal $\boldsymbol{b}$ at each cross section in the neighborhood. The tangent $\boldsymbol{t}$ naturally follows that of the yarn centerline. To determine the normal $\boldsymbol{n}$, we use the principal normal $\boldsymbol{n}_0$ at the central cross section. Let $\boldsymbol{t}_0$ be the tangent at this location and $\boldsymbol{b}_0 := \boldsymbol{n}_0 \times \boldsymbol{t}_0$. Then, $(\boldsymbol{t}_0, \boldsymbol{n}_0, \boldsymbol{b}_0)$ defines the local frame of reference[1] at the central cross section. Using this frame as a reference, we then determine the local frames for the rest of cross sections in the neighborhood as follows. For a cross section with a tangent direction $\boldsymbol{t}'$, we set its normal and binormal as

$$\boldsymbol{n}' := \boldsymbol{S}(\boldsymbol{t}_0 \to \boldsymbol{t}')\,\boldsymbol{n}_0 \text{ and } \boldsymbol{b}' := \boldsymbol{S}(\boldsymbol{t}_0 \to \boldsymbol{t}')\,\boldsymbol{b}_0, \tag{5}$$

where $\boldsymbol{S}(\boldsymbol{t}_0 \to \boldsymbol{t}') \in \mathbb{R}^{3\times 3}$ denotes the minimal rotation that aligns $\boldsymbol{t}_0$ with $\boldsymbol{t}'$. As a corner case, when the principal normal at the central cross section vanishes, we find a nearest neighbor with non-vanishing principal normal and generate the reference frame $(\boldsymbol{t}_0, \boldsymbol{n}_0, \boldsymbol{b}_0)$ there. As we define the normal directions in the rotation-minimizing fashion, the local frame of reference changes smoothly along the centerline within any neighborhood.

Next, we transform the deformation gradients into the local frames of reference using

$$\boldsymbol{F}^{\text{local}} = [\boldsymbol{t},\ \boldsymbol{n},\ \boldsymbol{b}]\,\boldsymbol{F}, \tag{6}$$

where $\boldsymbol{F}$ is the original deformation gradient provided by the yarn-level simulation. Then, all the $\boldsymbol{F}^{\text{local}}$ matrices to-

1. Note not to confuse these frames with ones used for procedural fiber generation (3): unlike the latter ones that typically vary smoothly along a yarn centerline, the former frames depend on (and only on) the local yarn geometry around individual cross sections and can differ considerably even between overlapping neighborhoods (e.g., when the yarn curvature changes sign).

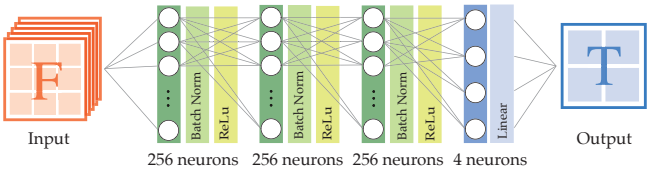


Fig. 8. **Network architecture.** Our regression neural network involves three fully connected hidden layers with 256 nodes each (shown in green). The network acts as a 1D nonlinear filter along a yarn centerline and turns simulated deformation gradients $\boldsymbol{F}$ into cross-sectional fiber deformations $\boldsymbol{T}$ (see Figure 7).

gether in a neighborhood serve as an input to the regression network.

Lastly, since the normal and binormal directions given by Eq. (5) generally differ from those given by the yarn centerline, we need to also transform our shape matching result $\boldsymbol{T}$ at each cross section into our newly defined local spaces. Let $\boldsymbol{S}(\boldsymbol{n}_\alpha \rightarrow \boldsymbol{n}) \in \mathbb{R}^{2\times 2}$ denote the rotation that transforms $\boldsymbol{n}_\alpha$ to $\boldsymbol{n}$. Then, the cross-sectional deformation local to the neighborhood is

$$\boldsymbol{T}^{\text{local}} = \boldsymbol{S}(\boldsymbol{n}_\alpha \rightarrow \boldsymbol{n})\, \boldsymbol{T}\, \boldsymbol{S}(\boldsymbol{n}_\alpha \rightarrow \boldsymbol{n})^\top, \tag{7}$$

which is also the expected output from our regression network.

Loss function for training: After obtaining the training data involving local deformation gradients $\boldsymbol{F}^{\text{local}} \in \mathbb{R}^{3\times 3}$ and fiber deformation $\boldsymbol{T}^{\text{local}} \in \mathbb{R}^{2\times 2}$, we train our regression network by minimizing the following loss function:

$$\underbrace{\left\| \mathcal{N}\left(\{\boldsymbol{F}^{\text{local}}\}_i\right) - \boldsymbol{T}_i^{\text{local}} \right\|_2^2}_{\text{parameter loss}} + \lambda \underbrace{\left\| \mathcal{N}\left(\{\boldsymbol{F}^{\text{local}}\}_i\right) \boldsymbol{H} - \boldsymbol{T}_i^{\text{local}} \boldsymbol{H} \right\|_2^2}_{\text{regularization}}, \tag{8}$$

where $\{\boldsymbol{F}^{\text{local}}\}_i$ denotes the local deformation gradients from the neighborhood around the i-th cross section, $\boldsymbol{T}^{\text{local}}$ is the desired fiber deformation at this cross section, $\mathcal{N}()$ indicates the network output (given the deformation gradients as input), $\boldsymbol{H}$ is a constant matrix containing the 2D location of each fiber center averaged over all cross sections, and $\|\cdot\|_2$ denotes the 2-norm of a matrix. Intuitively, this loss function involves a parameter loss term that captures the difference between network prediction and the desired output. Additionally, to regularize the network, we introduce an extra regularization term weighted by some $\lambda > 0$. As the $\boldsymbol{H}$ matrix is comprised of the cross-sectional locations of a set of fibers, $\mathcal{N}\left(\{\boldsymbol{F}^{\text{local}}\}_i\right) \boldsymbol{H}$ and $\boldsymbol{T}_i^{\text{local}} \boldsymbol{H}$ give the transformed fiber locations using the network prediction and the desired output, respectively. The regularization term then captures the difference between these two transformed 2D point clouds.

Data augmentation: To further improve the robustness of our regression network, we augment our training data by a factor of four. Given a set of local frames $\{(\boldsymbol{t}_k, \boldsymbol{n}_k, \boldsymbol{b}_k)\}$ in a neighborhood (where k indices each cross section in the neighborhood), we compute three additional sets of frames of reference given by $(\boldsymbol{t}_k, -\boldsymbol{n}_k, -\boldsymbol{b}_k)$, $(-\boldsymbol{t}_k, \boldsymbol{n}_k, -\boldsymbol{b}_k)$, and $(-\boldsymbol{t}_k, -\boldsymbol{n}_k, \boldsymbol{b}_k)$ for each k. For each set of frames, we then compute the corresponding local deformation gradients $\boldsymbol{F}^{\text{local}}$ and fiber transformations $\boldsymbol{T}^{\text{local}}$ via Eqs. (6, 7) and include them in the training dataset. In practice, we found that this data augmentation step greatly improves the stability of our regression network.

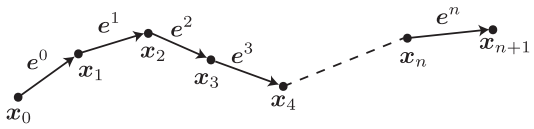


Fig. 9. **Yarn/fiber discretization.** In yarn-level simulation, each yarn is represented by a series of particles connected on a polyline. In fiber-level simulation, each fiber is represented as a polyline, and a group of fibers are twisted to form a yarn.

Network implementation: Our regression network consists of three fully connected hidden layers with 256 internal nodes each (see Figure 8). We implement this network using the Python-based Keras library [7] with Tensorflow [12]. We split our augmented data into a training set (85%) and a validation set (15%), and optimize the network weights using the Adam method [22].

5 Simulation Model

This section presents our numerical model for both yarn- and fiber-level simulation. The simulation model serves two purposes, i) to generate training data for the deep neural network enriching fiber-level geometries (§4.2), and ii) to perform at runtime yarn-level simulation providing input to the trained deep neural network (§6).

To this end, not only does the simulation model need to capture both yarn- and fiber-level dynamics, the simulated yarn dynamics must also be consistent across both scales. This poses a problem of fitting (or homogenizing) yarn-level material parameters from the fiber-level simulation. While several models have been proposed to simulate yarns, none of them addresses the problem of parameter homogenization—one that this section focuses on.

5.1 Unified Simulation at Yarn and Fiber Level

We simulate textile at both yarn and fiber level in a unified framework, which uses the elastic rod model [5] and a hybrid Lagrangian/Eulerian method based on [17]. We refer to their papers for more details, while briefly reviewing the key components here for describing our parameter homogenization.

Internal force: In fiber-level simulation, each fiber is represented as a polyline; and in yarn-level simulation, a polyline represents a yarn. In both cases, we compute the internal force using the elastic rod model [5]. The internal force is contributed by three types of elastic energies, namely, the stretching, bending, and twisting energy. Following the notation in [5] (summarized in Figure 9), the discrete stretching is defined as

$$E_{\text{stretch}} = \sum_{i=0}^{n} \frac{k}{2} \left(|\boldsymbol{e}^i| - |\bar{\boldsymbol{e}}^i| \right)^2, \tag{9}$$

where $|\boldsymbol{e}^i|$ is the (stretched) edge length while $|\bar{\boldsymbol{e}}^i|$ is the corresponding rest length, and k is the stretching coefficient. In addition, the discrete bending and twisting energy are given by

$$E_{\text{bend}} = \sum_{i=1}^{n} \frac{\alpha(\boldsymbol{\kappa}\boldsymbol{b}_i)^2}{2\bar{l}_i} \quad \text{and} \quad E_{\text{twist}} = \sum_{i=1}^{n} \frac{\beta m_i^2}{2\bar{l}_i}. \tag{10}$$

Here, $\bar{l}_i$ is the effective rest length of particle i, defined as $\bar{l}_i = \left(|\bar{\boldsymbol{e}}^{i-1}| + |\bar{\boldsymbol{e}}^{i}|\right)/2$. The scalar m_i is the twisted angle of the polyline at particle i, and $\boldsymbol{\kappa b}_i$ is its curvature binormal. We refer to [5] for their specific formulas. Lastly, α and β are the bending and twisting coefficients, respectively. The parameters α, β, as well as k (in Eq. (9)) are material related. They are different in fiber- and yarn-level simulations: the values in yarn-level simulation are the "homogenized" version of those in fiber-level simulation.

External force: Fibers (or yarns) are deformed by external forces such as collision and friction forces. Yet, resolving collision and friction is computationally prohibitive, especially in fiber-level simulation, wherein yarns are modeled explicitly as a large number of fibers. We therefore take a hybrid Lagrangian/Eulerian approach [17], which treats the deformation of fibers (or yarns) volumetrically. In this approach, collision forces are modeled as resistance of compression of the fiber (or yarn) volume, and friction force is viewed as the resistance of shearing of the fiber (or yarn) volume. To model these forces in the discretized setting, each edge $\boldsymbol{e}^i$ is associated with an elastic deformation gradient $\boldsymbol{F}^i$, which describes the local deformation of the fiber (or yarn) volume near $\boldsymbol{e}^i$. Particularly, the local material directions $\boldsymbol{D}^i = [\boldsymbol{D}_1\ \boldsymbol{D}_2\ \boldsymbol{D}_3]$ of $\boldsymbol{e}^i$ at the rest state is deformed into $\boldsymbol{F}^i\boldsymbol{D}^i$, the deformed material directions. Applying the QR-decomposition, $\boldsymbol{QR} = \boldsymbol{F}^i\boldsymbol{D}^i$, yields the orthonormal material directions $\boldsymbol{Q}$ and the matrix $\boldsymbol{R}$. The latter can be further decomposed into three components (i.e., $\boldsymbol{R} = \boldsymbol{R}_3\boldsymbol{R}_2\boldsymbol{R}_1$) given as follows:

$$\boldsymbol{R}_1 = \begin{bmatrix} r_{11} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_2 = \begin{bmatrix} 1 & r_{12} & r_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}. \tag{11}$$

$\boldsymbol{R}_1$ describes the stretch along the tangential direction of the curve (be it a fiber or yarn), $\boldsymbol{R}_2$ describes the shear along the curve, and $\boldsymbol{R}_3$ indicates the cross-sectional deformation. Similar to [17], we use two quadratic energy functions $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ to model the energy generated by collision and friction forces. Their specific forms will be described shortly in §5.2. Different from [17], we neglect the stretching component $\boldsymbol{R}_1$, as the stretching energy is already taken into account in the aforementioned elastic rod model, and the method of [17] does not consider the bending and twisting forces of the curves.

With these energy definitions, the internal and external forces can be computed by taking the derivatives of the energies with respect to particle positions of the polylines. Complete derivations of these forces are detailed in [5] and [17]. In our simulation, we use the Explicit Symplectic Euler (with a time step of 2×10^{-5}) to advance the states of the curves over time.

Remark: The volumetric view of the external forces offers another advantage for us. The deformation gradient $\boldsymbol{F}^i$ at each edge $\boldsymbol{e}^i$ encapsulates the local deformation (such as stretch, rotation, and shear) due to external forces. Thereby, in yarn-level simulation, $\boldsymbol{F}^i$ provides a concise clue about how the fibers that form the yarn should deform. Naturally, $\boldsymbol{F}^i$ will be fed into the deep neural network for estimating fiber-level geometries, as has been discussed in §4.2.



Fig. 10. **Centerline of a fiber bundle.** A yarn is simulated at fiber level with individual fibers shown in blue. The yarn's centerline is highlighted in red. We fit yarn-level parameters so that the yarn curve resulting from yarn-level simulation closely matches the centerline from fiber-level simulation.

5.2 Parameter Homogenization

We use the above force models in both fiber- and yarn-level simulations. To ensure consistent yarn dynamics across the two levels, we must set proper material parameters in Eqs. (9) and (10) and the energy functions, $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$. In fiber-level simulation, material parameters are set based on the physical properties of the fibers (such as silk and cotton). In yarn-level simulation, we need to set the parameters so that the simulated yarn curves, under the same initial condition, closely match the centerlines of the corresponding fiber bundles in the fiber-level simulation (Figure 10).

5.2.1 Fitting parameters of internal forces

We first decide elastic rod parameters (k, α, and β) in Eqs. (9) and (10) in yarn-level simulation. Because these parameters are independent from each other, we fit their values separately and in a similar fashion. To fit k, we quasistatically stretch a fiber bundle by a small distance Δx in the fiber-level simulation, and evaluate the collective stretching force $\boldsymbol{f}_{\text{s}}^*$ at the end of the fiber bundle. Meanwhile, we quasistatically stretch a yarn of the same length by Δx in the yarn-level simulation. We search for k such that the stretching force $\boldsymbol{f}_{\text{s}}$ at the end the yarn matches $\boldsymbol{f}_{\text{s}}^*$. Because $\boldsymbol{f}_{\text{s}}$ is proportional to k, the value of k can be quickly found through a binary search (Figure 11-a). In practice, we find that k is not sensitive to the choice of Δx, and we repeat this fitting process multiple times each with a different Δx value, and then take the average. The bending and twisting coefficients (α and β) are obtained in a similar way, as shown in Figure 11-b and -c.

5.2.2 Fitting parameters of external forces

As introduced in §5.1, the external forces are determined by two energy functions $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ both having quadratic forms. In fiber-level simulation, we use $g(\boldsymbol{R}_2) = \gamma(r_{12}^2 + r_{13}^2)$ and $h(\boldsymbol{R}_3) = \mu(r_{22}^2 + r_{33}^2) + \frac{\lambda}{2}(r_{22} + r_{33})^2$ similar to [17]. Here, r_{ij} are matrix elements defined in Eq. (11). γ, μ, and λ are parameters set according to specific type of fibers. In yarn-level simulation, we use

$$\begin{aligned} g(\boldsymbol{R}_2) &= a_1 r_{12}^2 + a_2 r_{12} r_{13} + a_3 r_{13}^2 \text{ and} \\ h(\boldsymbol{R}_3) &= b_1 r_{22}^2 + b_2 r_{22} r_{33} + b_3 r_{33}^2, \end{aligned} \tag{12}$$

where parameters a_i and b_i ($i = 1, 2, 3$) are what we need to fit.

Fitting these yarn-level parameters is nontrivial. Unlike the elastic rod parameters, the parameters a_i and b_i are related to the external forces through nonlinear derivatives of $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$; their relationships to the forces can not be decoupled, and neither are they proportional to the forces. Consequently, the binary search is not useful. When choosing a_i and b_i, we need to ensure the energy functions

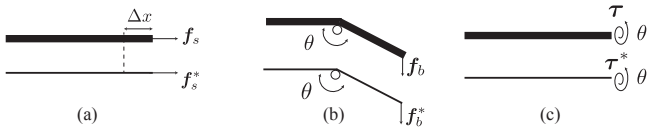


Fig. 11. **Fitting elastic rod parameters. (a)** Fixing one end of a yarn, we stretch it by Δx in both fiber-level (top) and yarn-level (bottom) simulation. The stretching coefficient k is found in a binary search to match the resulting stretching forces $\boldsymbol{f}_\mathsf{s}^*$ and $\boldsymbol{f}_\mathsf{s}$. **(b)** Fixing one end and the middle point of a yarn, we quasistatically move the other end to bend the yarn in both fiber-level (top) and yarn-level (bottom) simulations by the same amount. The bending coefficient α is found in a binary search to match the resulting bending forces $\boldsymbol{f}_\mathsf{b}^*$ and $\boldsymbol{f}_\mathsf{b}$ at the end. **(c)** Similarly, we fix one end of the yarn, and twist at the other end by an angle θ in fiber-level (top) and yarn-level (bottom) simulations. The twisting coefficient β is obtained in a binary search to match the resulting torques $\boldsymbol{\tau}^*$ and $\boldsymbol{\tau}$.

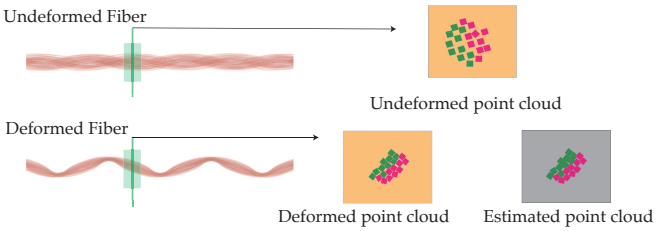


Fig. 12. **Estimating deformation gradient.** Consider a small segment of a yarn (highlighted in red box). We visualize the cross-sectional fiber distributions on the right. In fiber-level simulation, the fiber distribution can get severely deformed (bottom-right). We estimate a "homogenized" deformation gradient (i.e., a 3×3 transformation matrix) that transform the fibers in the rest material space to match the deformed fibers as closely as possible.

$g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ are positive semidefinite—a constraint known as the *conic constraint* in convex optimization. Moreover, in fiber-level simulation, the deformation gradients (and hence $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$) is defined on each finite elements $\boldsymbol{e}^i$ of the fibers, whereas in yarn-level simulation they are defined on the elements of yarns. Caution is needed in estimating yarn-level deformation gradients from fiber-level simulation.

Our parameter homogenization is based on a key observation. Consider a yarn resulted from a fiber-level simulation. Its centerline is what the yarn-level simulation should produce when material parameters are fit properly. Since the elastic rod parameters of yarn-level simulation have already been determined in the previous step, the expected internal forces in yarn-level simulation are known. In general, knowing the internal forces is insufficient to infer the yarn-level external forces. However, if the yarn is in *static equilibrium*, internal forces must be balanced by external forces. This means that the expected external forces are known as well.

In light of this observation, we use fiber-level simulation to obtain a yarn's static equilibrium state under collisions, and thereby compute the external forces needed in yarn-level simulation. We then estimate yarn-level deformation gradients, which are in turn used to fit the parameters a_i and b_i $(i = 1, 2, 3)$ in Eq. (12).

Estimating deformation gradients: We now describe how to estimate yarn-level deformation gradients from fiber-level simulation results. As illustrated in Figure 12, consider a small segment of a yarn. In fiber-level simulation, this segment consists of a set of fiber particles distributed over a group of fibers. Let $\{\bar{\boldsymbol{x}}_j\}_{j=1}^n$ denote the positions of these particles at the rest state, and $\{\boldsymbol{x}_j\}_{j=1}^n$ be their corresponding position at the deformed state. The yarn-level deformation gradient $\boldsymbol{F}$ at the same segment should be able to transform $\{\bar{\boldsymbol{x}}_j\}_{j=1}^n$ into $\{\boldsymbol{x}_j\}_{j=1}^n$. This is again a shape matching problem [25], [36] similar to that we encountered in §4.2.1. $\boldsymbol{F}$ is obtained by solving a small least-squares problem,

$$\boldsymbol{F} = \arg\min_{\boldsymbol{F},\boldsymbol{t}} \sum_{j=1}^{n} \|\boldsymbol{F}(\boldsymbol{x}_j - \boldsymbol{t}) - \bar{\boldsymbol{x}}_j\|_2^2, \tag{13}$$

where $\boldsymbol{t}$ is a vector to eliminate the translational difference, jointly optimized with $\boldsymbol{F}$.

Fitting a_i and b_i: External forces are related to a_i and b_i through the derivatives of $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ with respect to the deformation gradient $\boldsymbol{F}$, because $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$ are computed using $\boldsymbol{F}$. Once the yarn-level external forces and deformation gradients are estimated, we are able to fit the parameters a_i and b_i. A complete exposition of the fitting process involves detailed derivations from the Lagrangian/Eulerian approach [17]. Without interrupting the presentation flow, we defer it in the appendix, a six-dimensional semidefinite-quadratic-linear programming problem.

6 Run-Time Phase

With the preprocessing stages completed, we are now able to generate animated cloth models with fiber-level details. Our runtime pipeline takes the following inputs:

- The yarn-level material parameters obtained in the parameter fitting step (§5);
- The parameters for procedural generation of fibers via Eqs. (1) and (2);
- The pre-trained neural network that maps yarn-level deformation gradients to fiber-level deformations along yarn centerlines (§4).
- The initial condition of a fabric for performing yarn-level simulation.

The output of our system at runtime is a collection of animated fiber curves.

Our yarn-level simulation outputs the shape and arrangement of yarn centerlines together with the deformation gradients and normal directions along the yarn. With these information, we first procedurally generate fiber curves using the simulated yarn centerline and normals without deformation.

As demonstrated in Figure 2, neglecting fiber mechanics leads to inaccurate macroscopic appearance. Therefore, we update the cross-sectional fiber locations using deformation matrices $\boldsymbol{T}$ estimated by our neural network. Concretely, we apply our neural network as a 1D filter along every yarn of the fabric. For each cross section window, we first transform the simulated deformation gradients into corresponding local frames of reference (§4.2.2), and feed them to the neural network to estimate the cross-sectional deformation matrix $\boldsymbol{T}_{\text{local}}$. Via the inverse of Eq. (7), this matrix $\boldsymbol{T}_{\text{local}}$ is transformed back to $\boldsymbol{T}$, which is in turn used to transform the fiber curves in Eq. (4). Finally, the fiber-level geometries are ready for rendering the fabric.

7 Results

To generate the training dataset, we simulate a single yarn at both fiber and yarn levels. The material parameters used in fiber-level simulation is listed in Table 1. While we use the same set of fiber-level parameters, the fitted yarn-level parameter set may differ, because the yarn-level parameters also depend on the specific fiber structure in the yarn. Nevertheless, the parameters in yarn-level simulation are automatically decided using our parameter fitting algorithm (§5.2). At runtime, only yarn-level simulation is needed. In principle, our technique is independent of the underlying yarn-based simulation method. In practice, we use the anisotropic elastoplasticity model introduced by Jiang et al. [17] for its high physical accuracy. Our training fiber- and yarn-level simulations take 8 days to run on a single workstation (for all configurations). Upon obtaining the training dataset, our regression neural network is trained in 30 minutes.

TABLE 1
Material parameters of fiber-level simulation.

Param.	Value
k	1.87×10^{11}
α	4.07×10^{5}
β	1.08×10^{3}
γ	9.60×10^{5}
μ	3.51×10^{5}
λ	1.01×10^{6}

To render our mechanics-aware cloth models, we fully realize them by procedurally generating all the fibers and deforming them using cross-sectional transformations predicted by our regression network. Our method could be combined with previous realization-minimizing techniques [24] for better rendering performance.

7.1 Validation and Evaluation

Single-yarn results: We validate the accuracy of our method by comparing to photographs and reference simulations at the fiber level. In Figure 13, we show experimental results for three types of yarns, rayon, cotton, and polyester, each using fitted procedural parameters from the previous work [35]. We stretch all these yarns both directly and through a number of 3D-printed cylinders. Both situations are considerably different from our training simulations. Our method is able to accurately estimate the deformation of the yarns' fiber-level microstructures (Figure 13-c), which cannot be closely captured using previous methods that neglects fiber mechanics (Figure 13-d). Accurate estimation of fiber micro-geometry is crucial for producing fabric appearance at larger scales, as demonstrated at the bottom of the figure. Please refer to the accompanying video for the corresponding animations.

This experiment also validates our parameter homogenization algorithm. The (b1) column of Figure 13 is generated using fiber-level simulation, while the (c1) column is generated using yarn-level simulation with fitted parameters. It is evident that the yarn centerlines resulted from yarn-level simulation with the fitted material parameters is able to closely match those from the fiber-level simulation.

Alternative regression models: To learn the mapping between yarn-level simulated deformation gradients and desired transformations of fiber center, we chose to use a regression neural network due to the complexity of this mapping. Figure 14 demonstrates the performance of several simpler regression models include linear and polynomial regression and Gaussian process. Trained with the same data, our regression network behaves much more stably and provides results with superior visual quality.

TABLE 2
Simulation and processing time statistics. In Figure 13, fiber-level simulations are used only to generate references. Numbers in the right-most column indicate the total time spent on procedurally generating fiber curves and deforming them by evaluating our regression network. The timing statistics are collected on a workstation with four Intel Xeon E5-2620 v3 CPUs with six cores each running at 2.40GHz.

Scene	# vertices	Fiber-level simulation	Yarn-level simulation	Fiber generation + NN evaluation
Figure 13	300	8 h	15 min	20 sec
Figure 17-a	85832	-	20 h	1 min
Figure 17-b	85832	-	18 h	30 sec
Figure 17-c	58545	-	28 h	1 min
Figure 18-a	100000	-	9 h	1 min
Figure 18-b	100000	-	10 h	1 min
Figure 18-c	1266000	-	110 h	10 min

Multi-yarn results: Figure 15 shows a comparison between the reference (obtained via fiber-level simulations) and our result. Because explicitly simulating individual fibers is intractable at a large scale, we performed the simulation for a small patch with 4×4 yarns and tiled the resulting fiber geometries for both the reference (i.e., directly simulating at fiber level) and our method (i.e., enriching the yarn-level simulation) to provide a macroscopic comparison. As shown in the figure, our result resembles the reference at both micro and macro scales, while previous methods fail to capture the yarns' thinning effect (Figure 15-c).

Lastly, we perform a qualitative comparison between our results and photographs (Figure 16). In this experiment, we take photographs of a real knitted fabric under rest and stretched states. Additionally, we create a virtual model after the physical sample and tweaked the optical parameters to match the appearance in rest states (Figure 16-a1 and b1). Then, we simulate a stretching of this model at the yarn level and apply our technique to the stretched model. Our method (Figure 16-b2) successfully predicts the color change caused by yarn and fiber deformations. Please refer to the accompanying video for an animated version of this comparison.

7.2 Simulated Results

We now demonstrate the effectiveness of our technique via a number of experiments. Please see Table 2 for detailed statistics and performance numbers.

Our technique can be applied independently to individual yarns of a full fabric. Figure 17 shows rendered results of full knitted textiles. Figure 17-a shows a knitted fabric with a slip-stitch rib pattern being stretched outward, causing the entire fabric to become more see-through. In Figure 17-b, we show a rigid sphere being pushed toward another knitted fabric with the same pattern. The contact force applied by the sphere varies across the surface of this fabric, yielding spatially varying yarn and fiber deformations. Our model is able to reproduce this effect in a visually convincing manner. In Figure 17-c, we show the stretching of a knitted glove modeled after the work by Wu and Yuksel [32]. The mechanical responses of the yarns and fibers in this glove change its macro-scale appearance drastically: the glove not

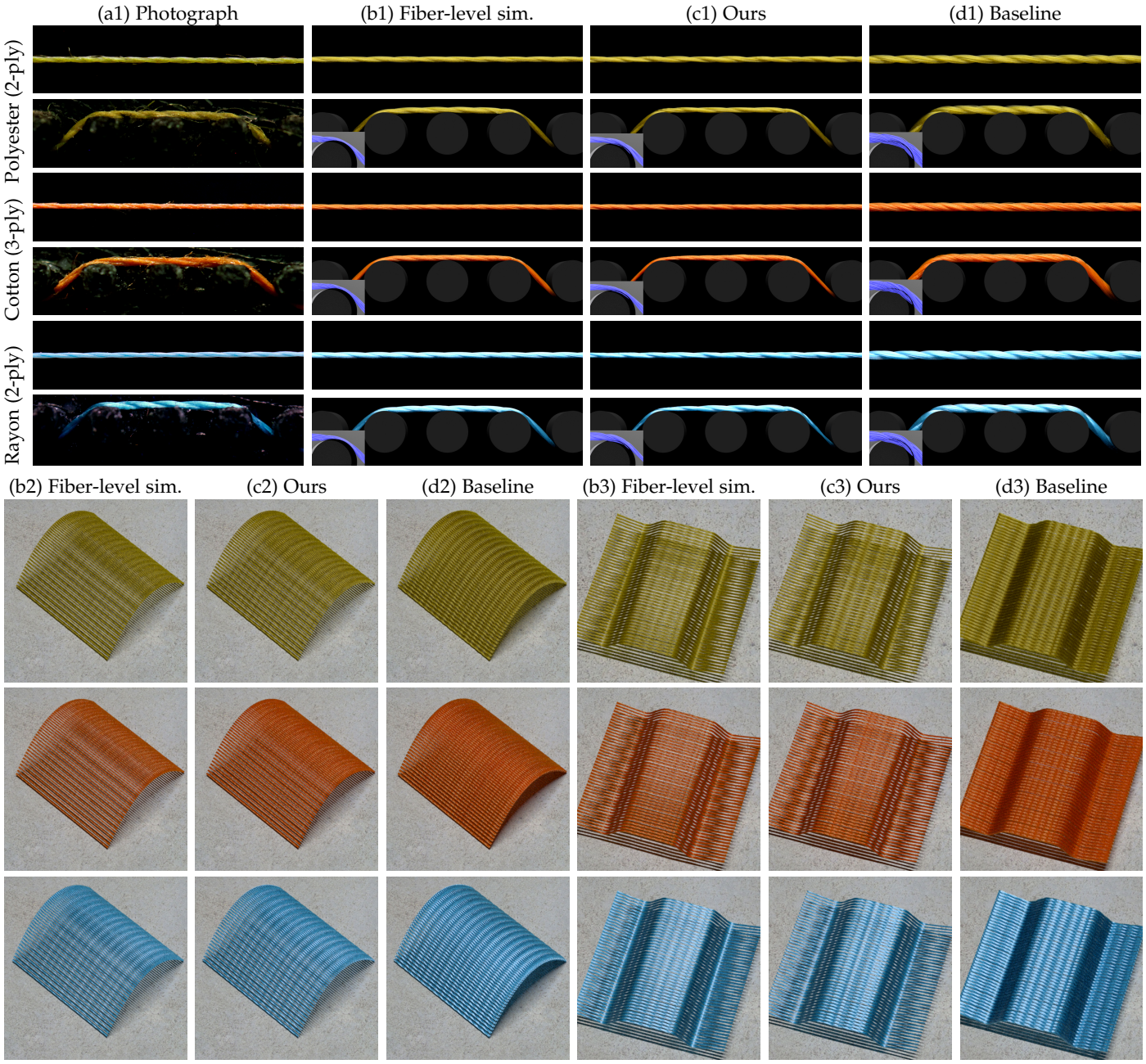


Fig. 13. **Generated models for single yarns.** We demonstrate the effectiveness of our method by comparing our results (c) to photographs (a) and reference fiber-level simulations (b) using three types of yarns: polyester, cotton, and rayon. Column (d) shows procedurally generated yarns with fiber mechanics neglected. The top images (1) show detailed fiber microstructures from individual yarns. The bottom images (2 and 3) use tiled versions of the top results to demonstrate how difference in those structures affect macro-scale appearance.

only becomes more see-through but also appears shinier due to the fibers being better aligned. Our technique successfully captures this complex change of appearance without require simulating individual cloth fibers.

Besides knitted fabrics, those fabricated via weaving are also ubiquitous in our daily lives. Although woven fabrics are generally more rigid (i.e., less deformable) compared to their knitted counterparts, our technique manages to capture their mechanics-driven appearance variations. Figure 18-a contains a small woven fabric with a flower logo resulting from its weave pattern. After being stretched horizontally, the weft yarns that follow the stretching direction are thinned more heavily than the warp yarns that are perpendicular to the stretching. This causes the entire fabric to appear less red and more white. Our technique successfully captures this effect (Figure 18-a2). In Figure 18-b, we show a fabric with a 2×2 twill pattern. This fabric is being pushed onto a static body with a capsule shape which causes the green yarns to stretch more than the yellow yarns. This change of fiber arrangement affects the fabric's overall appearance, making the regions touching the capsule to be more yellow. Our technique successfully captures this appearance change in colors. Lastly, in Figure 18-c, we show a jacquard fabric being stretched both horizontally and vertically, resulting in complicated yarn and fiber deformations. Our model manages to produce

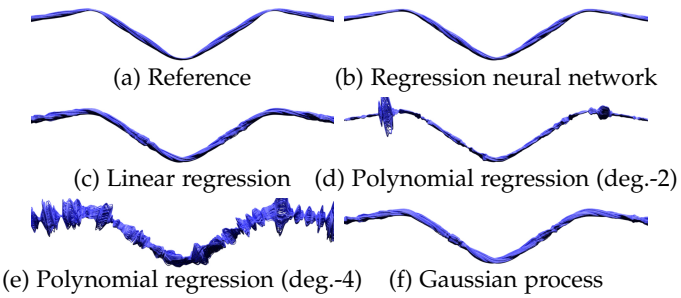


Fig. 14. **Alternative regression methods** such as linear and polynomial regressions as well as Gaussian processes have difficulties in capturing the complex relations between simulated yarn-level deformation gradients and desired cross-sectional transformations of fiber centers.

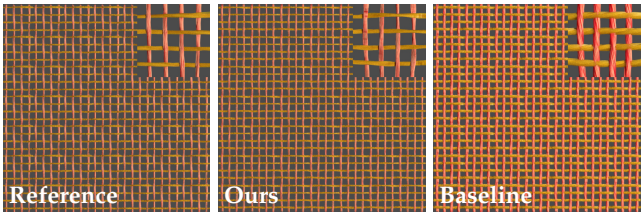


Fig. 15. **Generated models for a small textile.** We simulate the dynamics of eight fibers (four horizontal and four vertical), as shown in the insets. Our result generated by yarn-level simulation and the fiber-deforming neural network matches closely the reference fully simulated at the fiber level.

spatially varying appearance changes caused by these deformations (Figure 18-c2), which are mostly absent if fiber-level deformations are neglected (Figure 18-c3).

8 Discussion and Conclusion

Limitations and future work: Since our technique does not simulate fibers explicitly, it cannot capture visually significant effects involving complex fiber-level behaviors. For instance, brushing a yarn can pull out many of its fibers and make the yarn hairier. Capturing effects like this may require explicit simulation of individual fibers. While currently intractable, this problem remains interesting for future investigation. The same computational challenge also arises even in yarn-level simulation when the number of yarns are large. Our pipeline does not critically depend on the particular simulation model that we choose to use. Thus, more efficient yarn-level simulation methods will likely be beneficial for our pipeline.

Further, since yarns in our yarn-level simulation have no explicit "boundaries", the collision is resolved using volumetric elastic energies. Yet two yarn curves can get very close to each other, causing intersecting fibers. While how to resolve collisions of filament structures in a robust and efficient way remains a chronic challenge in general, improving our technique to reduce fiber intersections is an interesting topic for future research.

Lastly, the procedural model used by our method was previously tested mainly for simple yarns [27], [35]. Generalizing the model and our method to support a wider range of yarns (e.g., artificial and novelty ones) can benefit future applications.

Conclusion: We introduce the first mechanics-aware cloth appearance model capable of capturing the rearrangement of cloth yarns and fibers caused by external forces

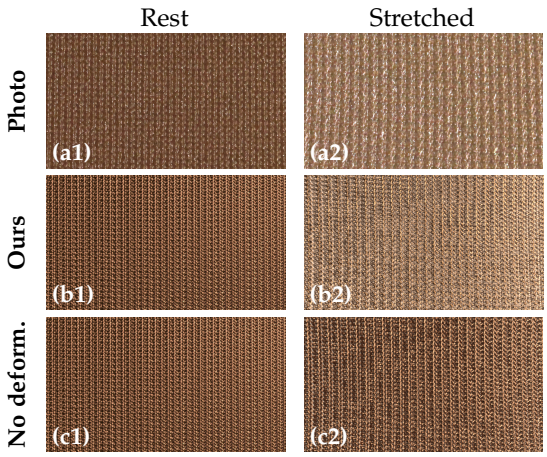


Fig. 16. **Qualitative validation** of our technique. We compare photographs of a real knitted fabric made of nylon in rest (a1) and stretched (a2) states. With a virtual model with a similar rest appearance (b1), our method predicts its appearance change when the fabric is stretched (b2). Existing methods neglect the mechanical responses of yarns and fibers, and thus fail to capture this phenomenon (c1, c2).

without simulating individual fibers explicitly. At the core of our technique is an extended modeling procedure that synthesizes cloth fibers and rearrange them based on simulated information only at the yarn level. We leverage a custom-design regression neural network that maps the simulated forces expressed as deformation gradients to cross-sectional affine transformations of fiber centers. To train this network, we use training simulations of a single yarn at both the yarn and the fiber levels and utilize a novel parameter fitting step to ensure the consistency at both scales. Results rendered with our technique demonstrate physically plausible cloth appearance at both micro and macro scales.

Acknowledgments

We thank the anonymous reviewers for their constructive suggestions and comments. This work was supported in part by NSF grants 1453101, 1717178, and 1813553.

References

[1] N. Adabala, N. Magnenat-Thalmann, and G. Fei. Visualization of woven cloth. In *Proceedings of the 14th Eurographics Workshop on Rendering*, pages 178–185, 2003.

[2] M. Ashikmin, S. Premože, and P. Shirley. A microfacet-based brdf generator. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 65–74, 2000.

[3] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM.

[4] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun. Discrete viscous threads. *ACM Transactions on Graphics (TOG)*, 29(4):116, 2010.

[5] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. *ACM transactions on graphics (TOG)*, 27(3):63, 2008.

[6] A. Carlos, C. Carlos, G. Diego, O. M. A., L. Jorge, and J. Adrian. An appearance model for textile fibers. *Comput. Graph. Forum*, 36(4):35–45, 2017.

[7] F. Chollet. Keras: The python deep learning library. https://keras.io, 2018.

[8] G. Cirio, J. Lopez-Moreno, D. Miraut, and M. A. Otaduy. Yarn-level simulation of woven cloth. *ACM Trans. Graph.*, 33(6), Nov. 2014.

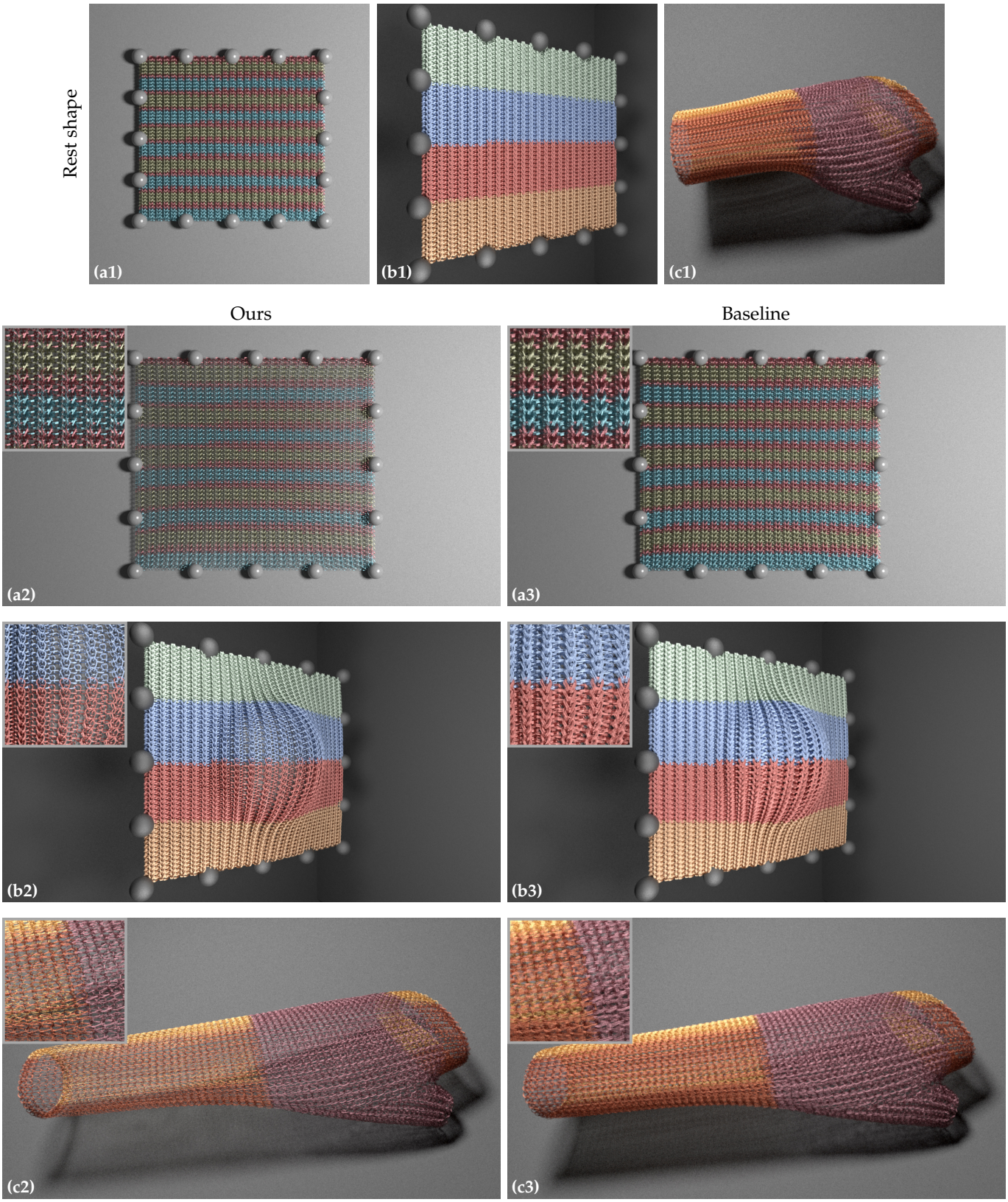


Fig. 17. **Generated models for knitted fabrics.** Our technique can also capture the change of appearance of knitted fabrics caused by their mechanical responses to external forces. (1) fabrics in their rest shapes; (2) deformed fabrics modeled with our techniques; (3) deformed fabrics with identical yarn centerlines but with fiber-level deformations neglected. The knitted fabrics in (a), (b), and (c) are comprised of cotton, polyester and rayon yarns, respectively.

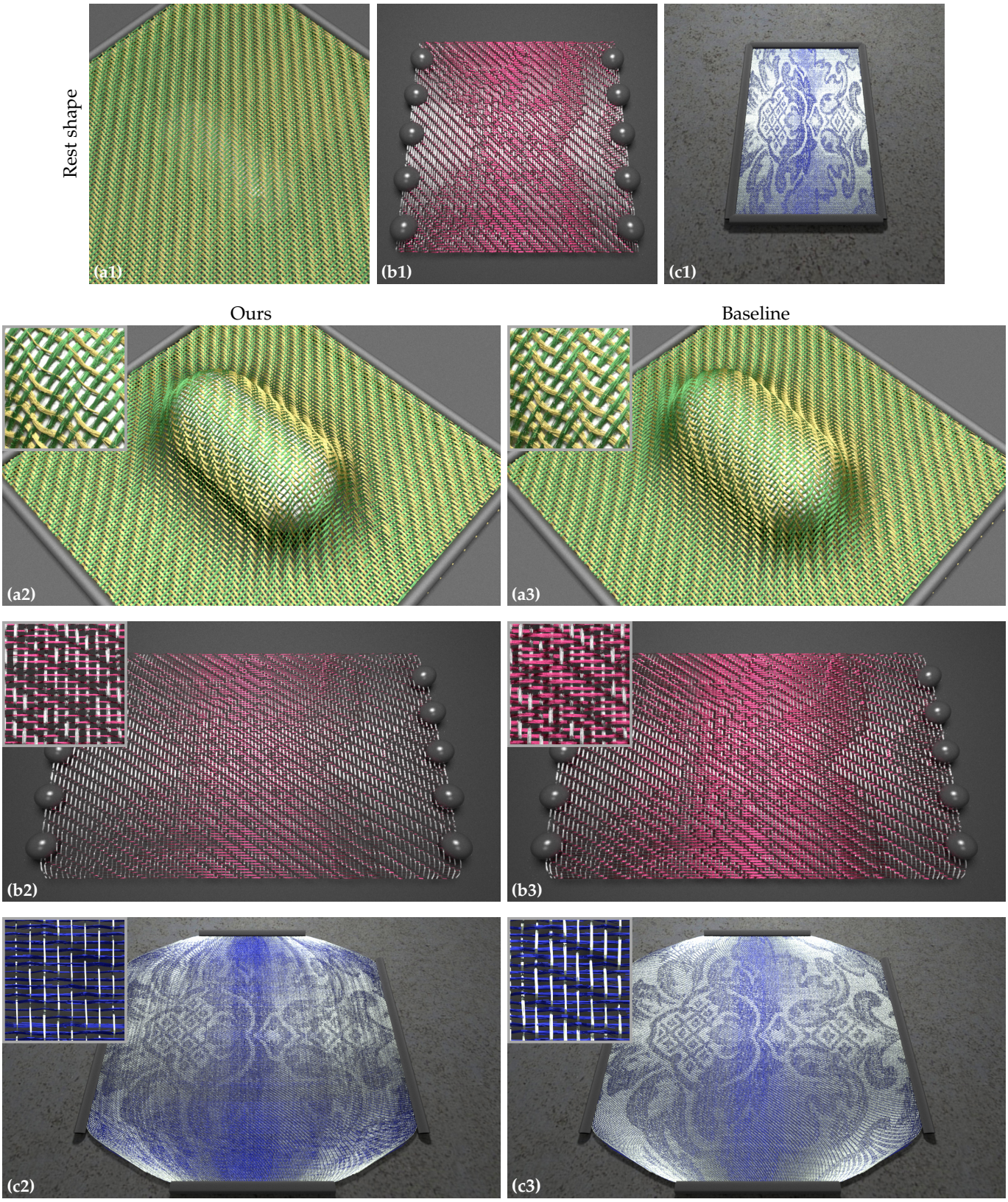


Fig. 18. **Generated models for woven fabrics:** when applied to full fabrics, our technique is able to capture the change of appearance, both dramatic and subtle, due to fiber mechanics at both macro and micro scales. (1) fabrics in their rest shapes; (2) deformed fabrics modeled with our techniques; (3) deformed fabrics with identical yarn centerlines but with fiber-level deformations neglected. The woven fabrics in (a), (b), and (c) are comprised of cotton, polyester and rayon yarns, respectively.

[9] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.
[10] Z. Dong, B. Walter, S. Marschner, and D. P. Greenberg. Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graph.*, 35(1):9:1–9:13, 2015.
[11] Y. R. Fei, C. Batty, E. Grinspun, and C. Zheng. A multi-scale model for simulating liquid-fabric interactions. *ACM Transactions on Graphics (TOG)*, 37(4):51:1–51:16, Aug. 2018.
[12] Google. Tensorflow. https://www.tensorflow.org, 2018.
[13] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
[14] Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics (TOG)*, 37(4):147:1–147:15, Aug. 2018.
[15] P. Irawan and S. Marschner. Specular reflection from woven cloth. *ACM Trans. Graph.*, 31(1):11:1–11:20, 2012.
[16] W. Jakob, M. Hašan, L.-Q. Yan, J. Lawrence, R. Ramamoorthi, and S. Marschner. Discrete stochastic microfacet models. *ACM Trans. Graph.*, 33(4):115:1–115:10, 2014.
[17] C. Jiang, T. Gast, and J. Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):152, 2017.
[18] J. M. Kaldor, D. L. James, and S. Marschner. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, 27(3):65:1–65:9, Aug. 2008.
[19] J. M. Kaldor, D. L. James, and S. Marschner. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.*, 29(4):105:1–105:10, July 2010.
[20] P. Khungurn, D. Schroeder, S. Zhao, K. Bala, and S. Marschner. Matching real fabrics with micro-appearance models. *ACM Trans. Graph.*, 35(1):1:1–1:26, 2015.
[21] P. Khungurn, R. Wu, J. Noeckel, S. Marschner, and K. Bala. Fast rendering of fabric micro-appearance models under directional and spherical gaussian lights. *ACM Trans. Graph.*, 36(6):232:1–232:15, 2017.
[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
[23] A. J. Liu, Z. Dong, M. Hašan, and S. Marschner. Simulating the structure and texture of solid wood. *ACM Trans. Graph.*, 35(6):170:1–170:11, 2016.
[24] F. Luan, S. Zhao, and K. Bala. Fiber-level on-the-fly procedural textiles. *Comput. Graph. Forum*, 36(4):123–135, 2017.
[25] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, July 2005.
[26] I. Sadeghi, O. Bisker, J. De Deken, and H. W. Jensen. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.*, 32(2):14:1–14:12, 2013.
[27] K. Schröder, A. Zinke, and R. Klein. Image-based reverse engineering and visual prototyping of woven cloth. *IEEE Trans. on Vis. and Comp. Graphics*, 21(2):188–200, 2015.
[28] D. Sulsky, Z. Chen, and H. L. Schreyer. A particle method for history-dependent materials. *Computer methods in applied mechanics and engineering*, 118(1-2):179–196, 1994.
[29] B. Thomaszewski, M. Wacker, W. Straßer, E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo, and N. Magnenat-Thalmann. Advanced Topics in Virtual Garment Simulation. In K. Myszkowski and V. Havran, editors, *Eurographics 2007 - Tutorials*. The Eurographics Association, 2007.
[30] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph.*, 26(2):255–264, 1992.
[31] H. Wu, J. Dorsey, and H. Rushmeier. Physically-based interactive bi-scale material design. *ACM Trans. Graph.*, 30(6):145:1–145:10, 2011.
[32] K. Wu and C. Yuksel. Real-time cloth rendering with fiber-level detail. *IEEE Transactions on Visualization and Computer Graphics*, PP(99), 2017.
[33] S. Zhao, W. Jakob, S. Marschner, and K. Bala. Building volumetric appearance models of fabric using micro ct imaging. *ACM Trans. Graph.*, 30(4):44:1–44:10, 2011.
[34] S. Zhao, W. Jakob, S. Marschner, and K. Bala. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph.*, 31(4):75:1–75:10, 2012.
[35] S. Zhao, F. Luan, and K. Bala. Fitting procedural yarn models for realistic cloth rendering. *ACM Trans. Graph.*, 35(4):51:1–51:11, 2016.
[36] C. Zheng. One-to-many: Example-based mesh animation synthesis. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 145–153, 2013.



Zahra Montazeri is a Ph.D. candidate in the Department of Computer Science at University of California, Irvine. She received her M.Sc. degree from University of California, Irvine in 2017, and her B.Sc degree from Sharif University of Technology in 2015. Her research focuses on appearance modeling for complex materials such as cloth and fabrics.



Chang Xiao is a Ph.D. student in Columbia University. He received the B.E. degree in Computer Science at Zhejiang University, China in 2016. His research interests are computer graphics, computer vision, and machine learning.



Yun (Raymond) Fei received his Ph.D. degree from Columbia University in 2019, and his B.Eng degree from Tsinghua University in 2013. As of 2019, his research focus on physics simulation of various materials, including but not limited to, hairs, clothes, liquid, sound, and the interactions between them.



Changxi Zheng is an Associate Professor in the Computer Science Department at Columbia University. Prior to joining Columbia, he received his M.S. and Ph.D. from Cornell University and his B.S. from Shanghai Jiaotong University. His research spans computer graphics, physically-based simulation, computational design, computational acoustics, scientific computing and robotics. He won the NSF CAREER Award and the Cornell CS Best Dissertation award in 2012.



Shuang Zhao is an Assistant Professor in the Department of Computer Science at University of California, Irvine. Before joining UCI, he was a postdoctoral associate at MIT. Shuang received his M.S. and Ph.D. in computer science from Cornell University. His research focuses on physics-based simulation of light transport, material appearance modeling/acquisition, and physically based rendering.