UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Modeling and Rendering Cloth

Permalink

https://escholarship.org/uc/item/5pm3p1tq

Author

Montazeri, Zahra

Publication Date

2021

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, availalbe at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Modeling and Rendering Cloth

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Zahra Montazeri

Dissertation Committee:
Assistant Professor Shuang Zhao, Chair
Professor Gopi Meenakshinsundaram
Professor Aditi Majumdar

2021

© 2021 Zahra Montazeri

DEDICATION

To my beloved husband, Ali Joonam.

TABLE OF CONTENTS

Page

LIST OF FIGURES **v**

LIST OF TABLES **x**

LIST OF ALGORITHMS **xi**

ACKNOWLEDGMENTS **xii**

VITA **xiii**

ABSTRACT OF THE DISSERTATION **xvii**

1 Introduction **1**
- 1.1 Contributions . . . 3
- 1.2 Outline . . . 5

2 Prior Works **7**
- 2.1 Background . . . 7
- 2.2 Related Works . . . 8

3 Mechanics-Aware Modeling of Cloth Appearance **12**
- 3.1 Method Overview . . . 15
- 3.2 Training Phase . . . 16
  - 3.2.1 Background: Procedural Modeling of Fiber Geometry . . . 17
  - 3.2.2 Learning How Fibers Deform . . . 18
- 3.3 Simulation Model . . . 26
  - 3.3.1 Unified Simulation at Yarn and Fiber Level . . . 26
  - 3.3.2 Parameter Homogenization . . . 29
- 3.4 Run-Time Phase . . . 32
- 3.5 Results . . . 34
  - 3.5.1 Validation and Evaluation . . . 35
  - 3.5.2 Simulated Results . . . 36
- 3.6 Discussion and Conclusion . . . 40

4 Ply-Based Appearance Modeling for Woven Fabrics **44**
4.1 Modeling Cloth Geometry . . . 46
4.1.1 Yarn Centerlines . . . 48
4.1.2 Ply-Level Geometry . . . 51
4.1.3 Fiber-Level Details . . . 51
4.2 Modeling Cloth Appearance . . . 53
4.2.1 Specular Components . . . 55
4.2.2 Body Components . . . 56
4.2.3 Round and Flat Yarns . . . 59
4.3 Implementation Details . . . 59
4.3.1 Ray-Ply Intersection . . . 60
4.3.2 Fiber-Level Details . . . 61
4.4 Results . . . 61
4.4.1 Validation with Measurements . . . 61
4.4.2 Validation with Photographs . . . 63
4.4.3 Comparison with Previous Methods . . . 65
4.4.4 Additional Results . . . 68
4.5 Discussion and Conclusion . . . 68

5 Ply-Based Appearance Modeling for Knitted Fabrics **72**
5.1 Our Method . . . 73
5.1.1 Generating the Flat Knits . . . 74
5.1.2 Base Geometry Transformation . . . 75
5.1.3 Appearance Modeling . . . 77
5.2 Implementation Details . . . 79
5.2.1 Modeling Geometry . . . 79
5.3 Results . . . 80
5.3.1 Comparison with photographs . . . 81
5.3.2 Main Results . . . 82
5.4 Discussion and Conclusion . . . 82

6 Conclusion **86**

Bibliography **88**

Appendix A Yarn Simulation **96**
A.1 Details of Parameter Homogenization . . . 96

LIST OF FIGURES

Page

1.1 **Cloth Applications.** Virtually modeling the fabrics are essential in our daily life and there are wide range of applications such as online retail, textile design and entertainment. 6

3.1 **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. 14

3.2 **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. 16

3.3 **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. 18

3.4 **Simulation for training.** We simulate the deformation of a yarn under a range of configurations. In particular, we use four common patterns [98] and six different cylinder spacings per pattern. Please refer to the supplemental material for the complete set of configurations. 20

3.5 **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 §3.3). 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. 21

3.6 **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. 22
3.7 **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 3.6). 24
3.8 **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. 26
3.9 **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. 29
3.10 **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 $\boldsymbol{f}_{\mathrm{s}}^{*}$ and $\boldsymbol{f}_{\mathrm{s}}$. **(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 $\boldsymbol{f}_{\mathrm{b}}^{*}$ and $\boldsymbol{f}_{\mathrm{b}}$ 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 $\boldsymbol{\tau}$. 31
3.11 **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. 33
3.12 **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. 37

3.13 **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. 38
3.14 **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. 38
3.15 **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). 40
3.16 **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. . . 41
3.17 **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. 42

4.1 **Cross section processing**. (a) We generate the cross section of a ply using a procedural model. (b) Within this cross section we perform 2D ray tracing to detect fibers visible from the exterior. We find the normals for these visible fibers using the same ray tracing process, and tangents are computed using derivative of the corresponding helix in 3D. Furthermore, local fiber shadowing is calculated for the visible fibers using 2D ray tracing. (c) Using the fiber normals, tangents, and shadowing textures we can shade the ply curve and account for shading details due to the individual fibers. Lastly, we add fiber irregularities as stated in §4.1.3. 52
4.2 **Illustration of our Ply BSDF model for round (left) and flat (right) plies.** At entrance point $\boldsymbol{x}$ and exit point $\boldsymbol{y}$, our model utilizes a specialized BSDF comprised of a *specular* and a *body* component where the former (denoted using a superscript S) captures (anisotropic) specular reflection and transmission, and the latter (denoted with a superscript B) approximates multiple-scattered light. 55

4.3 **Components of our Ply-BSDF:** specular reflection (top left), specular transmission (top right), body reflection (bottom left), body transmission (bottom right) 57
4.4 **Ply surface parameterization:** We parameterize the surface of a ply so that points with identical V coordinates correspond to individual ply cross sections. By assigning 1D textures specifying fiber normal and tangent information to a discrete set of cross sections, we can then obtain continuous fiber details on the ply surface by linearly interpolating the two 1D textures. 60
4.5 **Comparison with X-Rite TAC7 measurements:** Comparison of linear light measurements using an X-Rite TAC7 scanner. The sample was rotated such that the warps faced the linear light on the left plot whereas the wefts faced the linear light on the right. In this example, the pattern is satin. . . 62
4.6 **Matching photographed samples at the micro scale:** Our system is capable of representing multi-ply yarns along with the fiber-level details using an interactive user interface. Once the geometry and appearance parameters are set at this level, large scale models can be rendered with the same material as shown in Figure 4.7. 63
4.7 **Matching photographs at the macro scale for Ours and Fiber-based models:** Using the same samples and parameters as in figure 4.6 our model is able to simulate the anisotropic highlights seen in the photographs at macro scale. Note, how both the plain weave and the satin has very different appearances when the sample is front lit compared to the back lit sample; our model is able to simulate this difference and match the photographs. In the last three columns we use fiber-based geometry and BCSDF [44]. While BCSDF model can not match the front and back lighting simultaneously, our system found a descent fit for matching only one side. 64
4.8 **Appearance-matching volumetric models**. Despite the great difference between our model and the volumetric one that relies on anisotropic radiative transfer, our technique is able to produce appearances that are quite similar, thanks to its flexibility. 66
4.9 **Temporal consistency:** We show two frames of a rendered animation where a cloth drapes onto two spheres and the ground. Our technique manages to produce stable and temporally consistent renderings. Please see the supplementary video for the full animation. 66
4.10 **Anisotropic transmission for satin pattern in two orientations**. When the lamp is off, our model shows mainly the reflected lighting, showing the texture that is mapped to the surface color. And when the lamp is on, it shows the anisotropic transmission of the light as well and the change of color. Note the backlit highlight orientation is perpendicular to the flat yarns as also expressed in Figure 4.7-b and d. This is further demonstrated in the supplementary video. 67

4.11 **Editing geometry and appearance**. Starting from the Figure 4.7-b where we use fitted parameters to measurement: (a) Change the color (b) edit glossiness value, (c) change the weave pattern to basket with pronounced flyaways. Note, the subtle appearance of flyaways on the basket pattern which greatly increases realism along the edges. 71

5.1 **Method overview.** a) The input to our system is a geometry base mesh with per-vertex UV as well as a set of curves for a tilable knitted pattern. b) The knit geometry is generated in a flat plane also referred as 3D texture space. c) Then, warped to the object by extruding out of the base geometry and forming the ply-level cloth model. d) During rendering, fiber details are added using ply-based appearance model. 73

5.2 **Pattern tiling and stitching.** Coupling the endpoints of all the curves in the pattern cell among neighbor cells to compute how it is connected to the adjacent cells and generate the fully connected yarns. Each curve represents a node in the graph and the arrow shows the direction of the yarn which may need to be reversed based on the curves which it is connected to. 75

5.3 **Mapping grid.** The pre-computed mapping method for transferring the flat piece to the base geometry by extruding the yarns out of the base mesh. The same grid is also used to accelerate the rendering process and will be discussed in Section 5.1.3 76

5.4 (a) Illustrates the interpolation along one segment. (b) In order to map the fiber details we need to compute the angular phase around the interpolated ellipse at the intersected position, the arc length of the ellipse (β) is computed using the angle between the shading and geometry normal. 80

5.5 **Matching photographs of real sample knits with our model.** We use VGG loss to optimize the appearance parameters of the rendered image to match with the reference photos. Our system is capable of matching the appearance of the knits under both front and back lighting. 81

5.6 **Interactive editing geometry and appearance.** a) Changing the knitting pattern b) yarn density c) color d) and glossiness. 84

5.7 **Temporal consistency.** Our model is capable to generating detailed animations of knitted fabrics with only the base mesh simulated. In this example, we show four frames of an animation where a heavy piece of cloth is draped onto a spherical light (that is off in the top and on in the bottom). Our technique manages to produced visually plausible deformation of yarns (e.g., the stretching of yarns when they collide with the sphere). 84

5.8 Knitted fabrics with **common knitting patterns** simulated by Leaf et al. [51]. 85

LIST OF TABLES

Page

3.1 Material parameters of fiber-level simulation. 34
3.2 **Simulation and processing time statistics.** In Figure 3.12, 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. 39

4.1 A list of important symbols. 47
4.2 **Performance statistics** for renderings of our models in Figures 4.7–4.10. All rendering have been done using a dual Xeon E5-2699 system. 70

LIST OF ALGORITHMS

		Page
1	Transferring the flat textile to the base geometry	77
2	Rendering the cloth model	79

ACKNOWLEDGMENTS

I would like to deeply express my gratitude to my advisor, Shuang Zhao for his patience, motivation, enthusiasm and immense knowledge. He taught me not only the technical skills in computer graphics, but also an enlightening vision in problem solving in general, and inspirations in research problems. This thesis would not be possible without his continuous support.

My sincere thanks also goes to Henrik Wann Jensen for offering me the opportunity at Luxion Inc. during my 6–month internship in Denmark as well as our remote collaboration for almost two years. His insightful advices have been invaluable and his superb guidance and encouragement helped me shape my passion. I also feel extremely fortunate to work with Soren Gammelmark at Luxion, for his generous help not only in graphics concepts but also in general software engineering and answering my never-ending questions. His intellectual comments for improvement of the projects helped me tremendously.

I also thank Gopi Meenakshisundaram and Aditi Majumdar for teaching me the foundations of computer graphics and their support throughout my entire PhD program. Besides all my supervisors and mentors, I would like to thank my thesis committee: Shuang Zhao, Gopi Meenakshisundaram and Aditi Majumdar, Chatheir insightful feedback and constructive suggestions.

I thank my fellow labmates: Guo Yu, Cheng Zhang, Zhanhang Liang for the stimulating discussions and for the sleepless nights before deadlines. I feel extremely fortunate to have the chance of working with my fellow officemates: Mahdi Abbaspour Tehrani, Mehdi Rahimzadeh, Muhammad Twaha Ibrahim, and Samia Shafique. Also I thank my friends Marzieh Shariati, Fatemeh Moosapour, Negin Shariat, Forough Mehralian, Maryam Hassani for their kind support, without whom it was not possible to survive the pandemic.

I also thank IEEE and ACM verlogs for permission to include Chapter 3, 4 of my dissertation which were originally published in journal of Transaction of Visualization and Computer Graphics (TVCG 2019) and ACM Transaction on Graphics (SIGGRAPH 2021). Financial support was provided by University of California Irvine, NFS grants 1453101, 1717178, and 1813553.

Last but not least, I am deeply grateful to my family for their unfaltering support. I thank my husband Ali for his enormous amount of love and caring. Words can not express how grateful I am for his constant encouragement and the sacrifices, who has stood by me through all my travails. Special thanks to my father: it was him who motivated me to pursue graduate studies and his selfless efforts raising me along with my beloved mother. Sadly, she is no longer here to watch me completing my studies. Her prayer for me was what sustained me thus far.

VITA

Zahra Montazeri

EDUCATION

Ph.D Candidate in Computer Science, University of California, Irvine, **GPA: 4.0** expected 2021

Master in Computer Science, University of California, Irvine, **GPA: 3.9** 2017

B.Sc. in Computer Engineering Sharif University of Technology, Iran **GPA: 3.8** 2015

RESEARCH EXPERIENCES

Realistic Appearance Modeling for cloth and textile, PhD advisor: **Shuang Zhao**

- Multi-Scale and On-the-Fly Procedural Models for Cloth and Textiles current
- Mechanics-Aware Yarn Models for Cloth Rendering 2017 - 2019
  Physics-based interaction of individual yarns in presence of dynamics
  Using Neural-Network to enrich fiber-level details to the generated procedural deformed fibers
- Procedural Modeling for woven and knitted cloth in fiber-level 2017
- Surface-Based Modeling for Cloth Rendering of Volumetric Data 2016 - 2017
  Implemented a new medium pluging for Mitsuba renderer using openCV and Eigen.
- Separation of overlapping Microscopy images using texture synthesis 2016
- UI Design and Implementation for a Brain Analyzer Application 2015

Vision and Image Processing, Sharif University of Tech., Bachelors dissertation advisor: Prof. Sh. Kasaei

- Camera Pose Estimation by RGB-D cameras using Principal Component Analysis (PCA) 2014 - 2015
- Real-time Camera Tracking using Sports Pitch Markings, using OpenCV 2014

WORK EXPERIENCES

- **Software Engineer, Industrial Light and Magic (ILM) at LucasFilm**, London

Aug. - Oct. 2021

Part-time Research Intern, Luxion Inc (Makers of KeyShot), California
Practical Ply-based cloth modeling implemented in KeyShot renderer. Under supervision of **Professor Henrik W. Jensen** Jan. 2020 - June 2021

- **Research Intern, Luxion Inc (Makers of KeyShot)**, Denmark
  Modeling, shading and rendering curve-based cloth model and calculated fiber-level details on-the-fly. The model supports Multi-scale shading approach matching ply-based and surface-based models employing a simplified cloth BSSRDF. Under supervision of Professor Henrik Wann Jensen July-Dec 2019
- **Software Engineer Intern, Pixar Animation Studio**, lighting tool team, Emeryville
  Implemented two new tools for Hydra viewer in Katana: Snap-to-Cursor and Eye-Rig. Using Snap-to-Cursor, light is placed and aimed at cursor positions in 3D geometry helping artists to quickly create and modify all light kinds. And Eye-Rig tool is used to place eye-highlight at user defined position and the mode to place highlight at the relatively same position for all other eyes found in the character. This provides the quick modification of lights to leads highlights at desirable positions on eyes. Summer 2018
- **R&D Intern, DreamWorks Animation**, lighting team, Glendale
  Implemented a new plugin for Katana that extracts light rig from panorama stereo images into 3D geometries as light shapes. This provides ability to extract and modify the lights existed in the environment lights or even substitute them with neighbor objects, useful for light manipulation in live action movies. Summer 2017
- **R&D Intern, DreamWorks Animation**, Rendering team, Glendale
  Voluntarily worked with Moonray (Dreamworks Renderer) on implementation of Multiple-Importance-Sampling for emissive volumes that reduces noise in the scene contains only medium light-sources. Sep. 2017
- **Software Engineer Intern, xSeer Inc**, San Francisco
  Implementation and visualization of a big data (demographic) on world map using WebGL and Three.js both in 2D and 3D with multi-scale view ability. Summer 2016
- **Research Intern, Intelligence User Interface Lab**, Koc University, Istanbul
  Designing and Implementation of an Android User Interface Application, designed for teachers and students to solve and grade quizzes remotely. Feb - July 2015
- **Research Intern, Intelligence User Interface Lab**, Koc University, Istanbul,
  Visualization of Hierarchical Data of an Intelligent User Interface for Student Assessment through Free-Form Questions using Navigable Treemap Summer2014

PUBLICATIONS

- **Zahra Montazeri**, Soren B. Gammelmark, Henrik W. Jensen, Shuang Zhao. A Practical Ply-Based Appearance Model of Knitted Fabrics. *Eurographics Symposium on Rendering*, July 2021

- **Zahra Montazeri**, Soren B. Gammelmark, Shuang Zhao, Henrik W. Jensen. A Practical Ply-Based Appearance Model of Woven Fabrics. **ACM Transactions on Graphics (SIGGRAPH Asia 2020)**, 39(6), December 2020
- **Zahra Montazeri**, Chang Xiao, Yun Fei, Changxi Zheng, Shuang Zhao. Mechanical-Aware Modeling for Cloth appearance. **IEEE Transactions on Visualization and Computer Graphics**, 27 (1), January 2021 (Date of Publication: August 26, 2019)
- **Zahra Montazeri**, M. Gopi. Separating Overlapping Tissue Layers from Microscopy Images, Technical Report (arXiv:1905.09231) May 2019
- **Zahra Montazeri**, M.S. Mahdavi. Leila - Genghis's Daughter, illustrated poem book, Fasle Panjom Publication, 2013.

PATENTS

- **Zahra Montazeri**, Soren B. Gammelmark, Shuang Zhao, Henrik W. Jensen. Systems and Methods to Compute the Appearance of Woven and Knitted Textiles at the Ply-level. Attorney Docket No. 02AA-071001, June 2020
- M.S. Mahdavi, **Z. Montazeri**, B. Abdi, and Sh. Moghani, A Novel Eighteen-Step Tree-Phase Seven-Level Inverter Topology for a D-STATCOM Submitted to State Organization for Registration Of Deeds and Properties (SSAA), 2013

HONORS AND AWARDS

- **HopperX1 Scholarship Award**, CRA-W, Seatle 2019
- **AnitaB.org Scholarship**, Grace Hopper Celebration of Women in Computing, Florida 2017
- **CRA-W Scholarship**, Grad Cohort Workshop, San Diego 2016
- **Runner-up poster award at Computer Science show-case**, University of California, Irvine 2017
- **LightSpeed Venture Partners Summer Fellowship**, xSeer Inc, San Francisco Summer 2016
- **UC Irvine PhD Fellowship**, including five years of full financial support, UC Irvine Since 2015
- **Graduate Student Scholarship**, the Scientific and Technological Research Council of Turkey (TUBITAK), Koc University, Istanbul, Turkey 2015
- **Ranked 3rd** in cumulative GPA among 2010-beginners in Computer Engineering department, Sharif University of Technology
- **2nd Place** in Track and Field national Inter-University Competition, Tabriz, Iran 2011

EXTRACURRICULAR ACTIVITIES

- **UCI Student Center Board of Advisors**, UC Irvine 2020-2021
- **Chair of DECADE organization**, office of diversity and inclusion, UC Irvine 2018-2021
- **Student Volunteer at SIGGRAPH Asia**, Brisbane, Australia 2019
- **Painting Instructor at Visual-Art Organization**, Sharif University of Technology, Iran 2012 - 2015
- **Student Volunteer at SIGGRAPH**, Anaheim, California 2016
- **Member of Animation Club**, UC Irvine since 2018
- **3D-Printing Workshop**, FABWORKS, UC Irvine 2016
- **3D MAX Professional Summer Training**, Tehran, Iran 2012
- **Managing Editor for Computer Science articles** , IUT Literary Publication, Iran 2010 - 2011

ABSTRACT OF THE DISSERTATION

Modeling and Rendering Cloth

By

Zahra Montazeri

Doctor of Philosophy in Computer Science

University of California, Irvine, 2021

Assistant Professor Shuang Zhao, Chair

Modeling cloth appearance, an active research topics in computer graphics for decades, generally boils *down* to representing the geometry and optics of a fabric. Existing models can be broadly classified into two categories. *Surface-based models* depict cloth geometry using smooth 2D sheets (expressed as polygonal meshes) and optics using specialized reflectance models. *Micro-appearance models*, on the other hand, express cloth geometry at the micro-scale down to individual micron-diameter fibers and optics by utilizing volumetric or fiber-based light scattering models.

In practice, both categories have their own advantages and disadvantages. The surface-based models are typically light-weight, easy to edit, and can faithfully reproduce the appearance of a fabric at the macro-scale. Thus, they have been widely used in the computer graphics industry. However, these models lack the fine-grained details that are crucial for cloth rendering under close-up views. The micro-appearance models, in contrast, are capable of generating renderings with remarkably high fidelity and details. Unfortunately, as these models are highly data-intensive and usually difficult to manipulate, their practical use has been quite limited.

In this dissertation, we present a family of new techniques to efficiently build and render fabric models with fiber-level details. Our first contribution tackles the challenge an extension

of the micro-structure models by supporting fabric mechanics. In this light, we introduce a mechanics-aware model that adapts the micro structures of cloth yarns in a physics-based manner. Our second contribution focuses on a practical appearance model for woven fabrics that represents the cloth in ply-level and offer an interactive rendering process. Lastly, our third contribution aims for extending this practical model for knitted fabrics.

The approaches presented in this dissertation have created realistic renderings with fiber-level details. The results match the measurements and offer an accurate appearance. The remaining challenges include developing a more complex representation of cloth while preserving all the fiber-level details and allow transition between different level of details (i.e. surface-based, ply-based, and fiber-based models). We also believe the techniques discussed in this dissertation can inspire insights for other materials beyond fabrics.

Chapter 1

Introduction

Fabrics are essential in our daily lives, therefore modeling them virtually and their appearance is important and has been an active research in Computer Graphics. Thus, modeling the appearance of fabrics in a physically faithful fashion has applications in many areas such as following:

- **Online retail**: With accurate prediction for cloth appearance and deformation, virtual designing rooms are used to preview a cloth on 3D body models. This particularly can be used for online shopping so the shopper can try out cloth virtually.
- **Textile design**: Designers for fabrics can preview the final prototype of their product before physically creating the model. In addition to saving money and time, this offers quick editing and exploring possible designs.
- **Entertainment**: Realistic fabrics can be highly used in CG movies and game industries. Given that fabrics are very common in every scenes, realistic appearance of them enhance the quality of the entire scene.

Accurately modeling fabrics remains challenging because different types generate distinct

appearance based on their underlying geometry. It is difficult to generate all different appearance using one universal model. Furthermore, the structure of fabrics vary significantly and they naturally arise irregularities and these natural structures are difficult to be generated procedurally. Some thick fabrics such as velvet and felt have complicated structures that cannot be described using traditional surface-based models.

In this dissertation we focus on two main problems: practical representation for cloth geometry and appearance. Both representations centered around *reductionism* phenomena which means focusing on more fundamental concepts rather than details. While in general, especially in computer graphics detailed representations often carry more accurate information, we experimented the opposite. In this thesis we discuss that for cloth modeling, accurate data can be still unraveled using high level models. While recent works focus on detailed representation of fabrics down to the fiber-level, we used higher level representation to model woven and knitted fabrics as explained in Chapter 4 and 5.

Cloth Geometry. In Computer Graphics, cloth is mostly represented using surface-based models (e.g., [1, 29, 72]). These models computes light interactions with low resolution surface mesh and hence fast to render. However, they have difficulties to capture the fine details which greatly contributes to the realism of the overall appearance. Therefore, modeling the actual 3D structure attracted a great attention in the filed. Recently, a wide range of micro-appearance models are introduced and they are more faithful to the 3D geometry of the fabric and lead to more accurate appearance (using high-resolution volumes [96, 99] or fiber meshes [76, 44]). These advanced models are capable of capturing a fabric's fiber-level details, however, it remains difficult to create the high fidelity geometry in an automatic way. Accurate appearance using procedural methods for fabrics first introduced by Zhao et al. [96]. In Chapter 3, we develop an extensions to these models by making them mechanics-aware; that is, capable of capturing how cloth fibers deform under external forces. Then in Chapter 4 and 5, we propose a novel technique for modeling the cloth in which it offers the

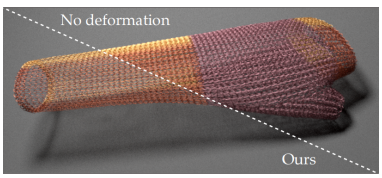
highly realistic appearance of micro-appearance models while allow efficient rendering.

Cloth Appearance. Surface-based reflectance models are widely used in computer graphics. While they offer efficient rendering, they have difficulties to capture the accurate appearance which is essential for the overall realism. On the other side, micro-appearance models provide high-fidelity appearance yet are very slow to render. To simulate how light interacts with the geometry, we build on recent work for appearance modeling of hair and fur [43, 104]. In these materials, same as fabrics, light intersects with the individual strands explicitly. Simulating light interaction with individual fibers leads to very slow rendering. Existing techniques either sacrificed the accuracy of the appearance [90] or very slow to render [44]. We introduce a new technique to accelerate light interactions while preserving the detailed appearance and we evaluate our system by comparing to the measurements and photographs. Please refer to Chapter 4 and 5.

1.1 Contributions

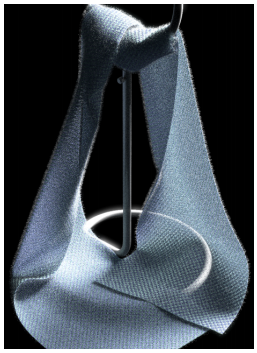
In this thesis, we propose a family of novel techniques for modeling and rendering cloth. These techniques tackle a range of problems from geometry modeling for several material types to photo-realistic appearance. The detailed model build by our techniques can produce rendered images closely matching photographs of real samples. This makes our methods to be useful particularly in virtual textile design and cloth prototyping for online retail.
In this section, we summarize the key contributions as listed below:

Contribution 1. Our first contribution (Chapter 3) is a mechanics-aware technique that adapts the micro-structures of fabrics in physics-based manner. Our system supports the

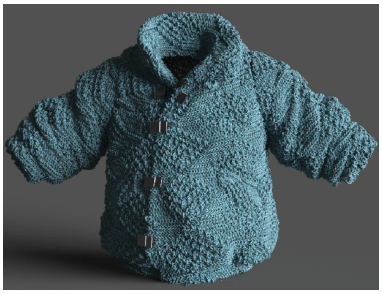


change of appearance for a deformed fabric. Our technique captures the changes of yarn and fiber arrangements in response to an external forces by only simulating the yarn curves and later add the fiber details. The fiber-level enrichment is made practical through a new parameter fitting algorithm, and an augmented procedural yarn model coupled with a custom-design neural network. We train the neural network using data-sets that are generated by jointly simulations of yarns and fibers. We demonstrate that our technique is capable of synthesizing photo-realistic cloth appearance in a mechanically plausible way.

Contribution 2. Our second contribution (Chapter 4) is a new approach for practical appearance modeling for woven fabrics. We model a piece of cloth at the ply level and simulate the local appearance of fibers making up each ply in an aggregated manner. Our method is capable of capturing both reflection and transmission of light and the resulting renderings accurately match photographs of real samples. Compared to the existing micro-appearance models, our method is light-weight and can scale to large pieces of clothing with minimum time and memory overhead in an interactive system.



Contribution 3. Fabrics modeled using our proposed technique briefly explained in previous contribution have offered a new level of practicalism for woven cloth. However, knitted fabrics express different behavior with light interactions due to their highly complex structures. Our third contribution (Chapter 5) is a generalization off Contribution 2 to knitted fabrics. Unlike woven fabrics that only have two states of yarn arrangements, knits offer a wide range of complex structures and require separate study. Our system can generate yarn geometries for a broad set of preset stitch types and common knit patterns. We demonstrate that our



method can match the structure and appearance of real knit sample pieces.

1.2 Outline

The rest of this dissertation is organized as follows.

- **Chapter 2** overviews background as well as the prior works of fabric geometry and appearance modeling.
- **Chapter 3** presents a novel technique for mechanics-aware cloth model.
- **Chapter 4** introduces our new system for practical appearance modeling for woven fabrics.
- **Chapter 5** presents the extend of our practical cloth rendering for knit fabrics.
- **Chapter 6** provides our conclusion and future research directions.

(a) Online retail (b) Textile design (c) Entertainment



Figure 1.1: **Cloth Applications.** Virtually modeling the fabrics are essential in our daily life and there are wide range of applications such as online retail, textile design and entertainment.

Chapter 2

Prior Works

In this chapter explain important terminology that will be used extensively throughout this dissertation and overview related materials and previous studies.

2.1 Background

Fabrics are generally divided into three categories: felt, woven and knitted. Felts are built by compressing fibers together that form a stable layer of fibers. Unlike to felt, woven and knitted fabrics are structured and have repeating patterns [40]. Understanding the underlying structure of a cloth provides insight of how light interacts with the geometry hence the appearance of the fabric would largely be affected. Fabrics are composed of *yarns*, which themselves are generated by twisting several *plies*. Each ply is a bundle of natural twisted *fibers*. All these factors have significant impact on a fabric's overall appearance.

Modeling Woven Fabrics. Woven fabrics are constructed by interlacing two sets of yarns, called *warps* and *wefts*. During weaving process, warp yarns are spun onto a loom, and weft

yarns are inserted from a perpendicular direction. At each crossing of weft and warp, one of the yarns goes on top of the other. This forms a binary configuration known as weave pattern. This regular structure of woven fabrics make them resistant to stretching unless the yarns are stretchable.

Modeling Knitted Fabrics. Knitted fabrics exhibit more complicated structures than woven. They are constructed by pulling yarn loops into another and forming a stitch. The two primary directions of knits are called *course* and *wale*. Where the course travels in the direction of a single row of loops and the wale travels in the direction of the stack of loops. Due to this construction, knits tend to show low resistant in case of stretching compared to woven, even if the yarns are not stretchable itself. There are a handful number of knit patterns that are the most common patterns in knitting industry.

2.2 Related Works

In Computer Graphics, there is a large body of works in cloth rendering and simulation. In this section, we summarize previous works related to modeling as well as the appearance of the fabrics.

Surface-based cloth models Traditionally, cloth has been modeled as thin 2D surfaces with light-cloth interactions depicted using bidirectional reflectance distribution functions (BRDFs) (e.g., [1, 29, 72]). These models are light-weight and can produce high-quality results. However, they usually lack fine-grained details such as thread and fiber structures necessary for close-up views.

Micro-appearance cloth models A number of *micro-appearance models* have been developed in the recent years that explicitly describe a fabric's micro-geometry down to the fiber level and have offered a new level of fidelity and details [96, 75, 76, 44, 101, 53, 62]. Due to the high complexity of these models, rendering them efficiently is challenging. To address this problem, several precomputation-based rendering methods have been developed. Zhao et al. [95] proposed a technique that exploits the modularity of woven cloth models by efficiently combining precomputed light-transport information on the fly. Wu and Yuksel [90] developed a GPU-based method for procedurally described fabrics by leveraging 'core fibers' with precomputed shadowing information. Khungurn et al. [46] extended precomputed radiance transfer [78] to handle micro-appearance cloth models. Unfortunately, these methods either lack performance [95, 46] or physical accuracy [90] for predictive rendering and editing micro-appearance cloth models. Furthermore, even though these models are very detailed they have not demonstrated the ability to simulate both reflection and transmission through cloth accurately.

Data-driven models Cloth appearance can also be modeled in data-driven ways, such as using Bidirectional Texture Functions (BTFs) [16]. However, these representations are cumbersome and difficult to manipulate. Further, they are normally too data-intensive to capture the level of details offered by micro-appearance models.

Fiber scattering models The reflectance profile used by our technique is related to various fiber [57, 104, 44, 3] and fur [93, 92] scattering models. Additionally, there are several multi-scattering studies for hair rendering [64, 65, 105]. These models utilize bidirectional curve scattering distribution functions (BCSDFs) to describe how light scatters off individual fibers or fur strands.

Granular materials models Similar to our aggregation ply BSDF, granular materials are composed of many dielectric grains where their aggregate behaviour determines the overall appearance [66, 60, 68]. They study the decomposition of a complex modeling/appearance problem to decouple individual grains from their arrangement to achieve high performance.

Micro-geometry models Rendering glints on micro-geometry surfaces liked brushed metal shares some of the complexity seen with bundles of fibers in cloth rendering [88, 31, 89, 71, 50]. These methods have not been used for the fiber structures seen in clothing, but the ideas could likely be extended to increase the efficiency of highly detailed cloth models.

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 [82]. Oftentimes, a piece of cloth is modeled as a 2D surface, discretized with a triangular or quadrilateral mesh [6, 23]. 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. [41], and later accelerated in [42].

The main challenge of simulating at yarn level is the high cost of resolving contacts. To address this issue, Cirio et al. [13] introduced sliding constraints to efficiently treat inter-yarn contacts. Our work is built on a Lagrangian/Eulerian approach of resolving contacts [37]. Extending the material point method [81], this approach represents yarns as polylines embedded in a background Eulerian grid, and resolves 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 [24]. We address this limitation for simulating individual yarns by incorporating the elastic rod model [7], similar to the recent work by Fei et al. [19].

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. [37] 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.

Chapter 3

Mechanics-Aware Modeling of Cloth Appearance

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., [15, 33, 18, 52]). Motivated by its utter prominence in virtual senses, cloth appearance has also been modeled microscopically. State-of-the-art *micro-appearance* models [97, 43, 100] 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 3.1). Even without external distortion, the interaction between individual yarns of a cloth can change yarn shapes and cause specific texture patterns to emerge [41, 13].

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 [97, 43, 77, 100, 9]. On the other hand, while there exist physics-based models that simulate individual yarns of a cloth [42, 13], 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 [77, 100] 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 **??**).

To our knowledge, this is the first mechanics-aware 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 [97, 43], or use procedurally generated fibers with fiber and yarn mechanics completely neglected [77, 100, 54]. 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.

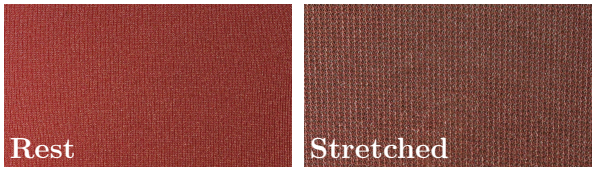


Figure 3.1: **Appearance change of a real fabric** driven by external stretching. When a real piece of cloth is stretched, its internal yarn and fiber structures can change drastically, leading to significantly different overall appearance. In this example, we show photographs of a nylon fabric under rest and stretched states.

We validate our method by comparing fiber-level simulations and photographs (Figures 3.12 and 4.7). 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 **??**, 3.16, and 3.17).

3.1 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.

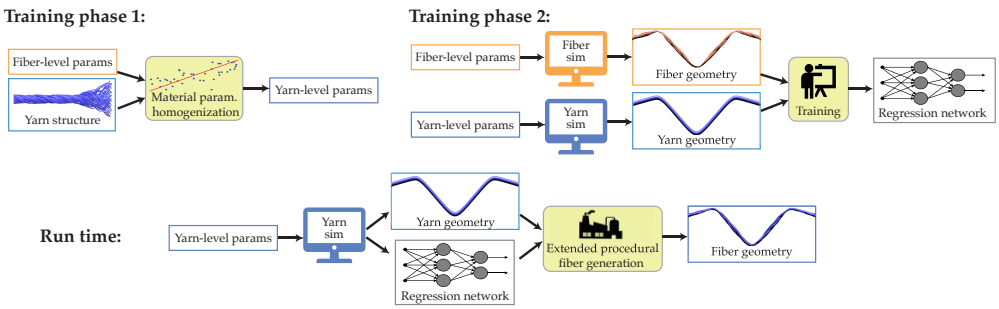


Figure 3.2: **Method overview. (top)** In the preprocessing phase, our method has two sub-steps: homogenizing the material parameters for yarn-level simulation (left) and training the regression neural network (right). **(bottom)** At runtime, we capture the rearrangement of yarn-level structures in 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.

- *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 3.2. In the following sections, we detail our training stage in §3.2, parameter fitting in §3.3, and runtime phase in §3.4.

3.2 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 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 §3.3. In this section, we focus on our main training phase. In §3.2.1, we briefly revisit the procedural approach for generating cloth models with fiber-level details. Then, in §3.2.2, we show how this procedure can be extended to become mechanics-aware by leveraging a custom-design regression neural network.

3.2.1 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 [77, 100]. 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:

$$\boldsymbol{h}_j(z) = \left[r_j \cos\left(\theta_j + \frac{2\pi z}{a}\right),\ r_j \sin\left(\theta_j + \frac{2\pi z}{a}\right),\ z \right]^\top, \tag{3.1}$$

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

$$p(r;\ \epsilon, \beta) \propto (1 - 2\epsilon_1) \left(\frac{\mathrm{e} - \mathrm{e}^r}{\mathrm{e} - 1}\right)^{\epsilon_2}, \tag{3.2}$$

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

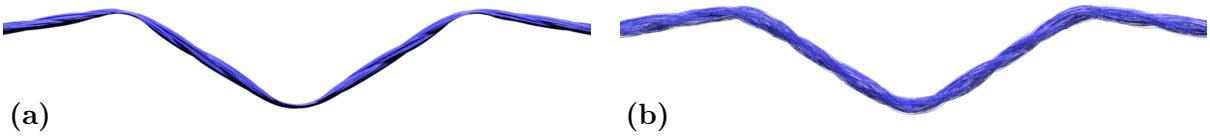


Figure 3.3: **Deforming fibers using yarn-level deformation gradients** generally leads to unsatisfactory results (b) compared to the reference (a) since they effectively capture average deformation within full grid cells that are much larger than individual fibers.

To better model the migration of fibers, Eq. (3.1) 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. (3.1, 3.2). Then, the yarn is formed by twisting all the constituent plies around a common centerline. Please refer to the prior works [77, 100] 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

$$\boldsymbol{h}_j^{\boldsymbol{\alpha}}(z) = \boldsymbol{\alpha}(z) + [\boldsymbol{h}_j(z)]_x \boldsymbol{n}_{\boldsymbol{\alpha}}(z) + [\boldsymbol{h}_j(z)]_y \boldsymbol{b}_{\boldsymbol{\alpha}}(z), \tag{3.3}$$

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.

3.2.2 Learning How Fibers Deform

The procedural model outlined in Eqs. (3.1–3.3) 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:

$$\boldsymbol{h}'_j(z) := \begin{bmatrix} \boldsymbol{T}(z) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{h}_j(z), \tag{3.4}$$

which can in turn be warped to follow arbitrary yarn centerlines as before using Eq. (3.3). 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 §3.3.

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 3.3.

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. (3.4). 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.

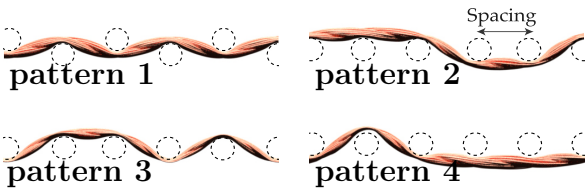


Figure 3.4: **Simulation for training.** We simulate the deformation of a yarn under a range of configurations. In particular, we use four common patterns [98] and six different cylinder spacings per pattern. Please refer to the supplemental material for the complete set of configurations.

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 3.4), and collect the results as our training dataset.

Training simulations For each simulation configuration, as shown in Figure 3.5-a, we use the procedural approach depicted in §3.2.1 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 3.5-b). Thanks to our parameter homogenization step (which will be detailed in §3.3), 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. (3.1–3.3) with no deformation.

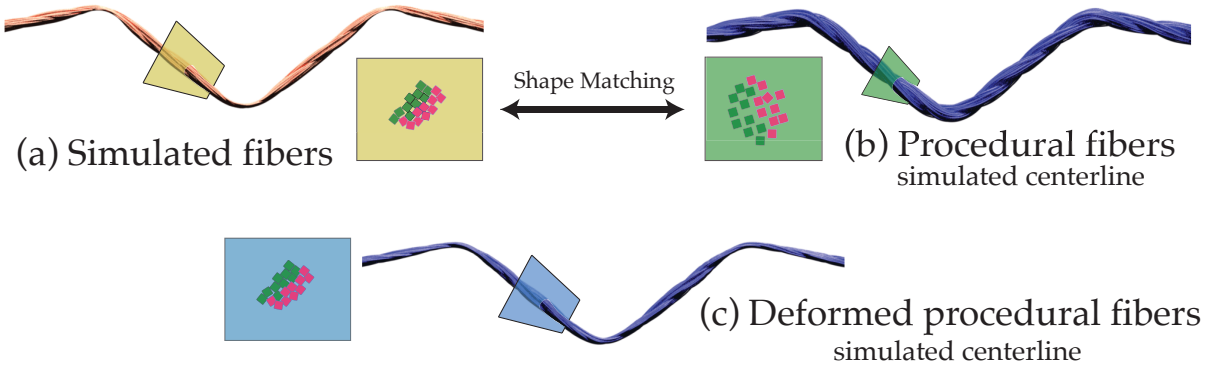


Figure 3.5: **Computing cross-sectional fiber deformations.** We simulate a yarn at both fiber (a) and yarn (b) levels. At the yarn level, we procedurally generate fibers following the simulated centerline (which agrees with the fiber-level simulation thanks to the parameter homogenization described in §3.3). 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.

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 3.5-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 [67] in 2D. By applying the computed deformations $\boldsymbol{T}$ via Eq. (3.4), the procedurally generated fibers can closely match the fully simulated fibers (Figure 3.5-c).

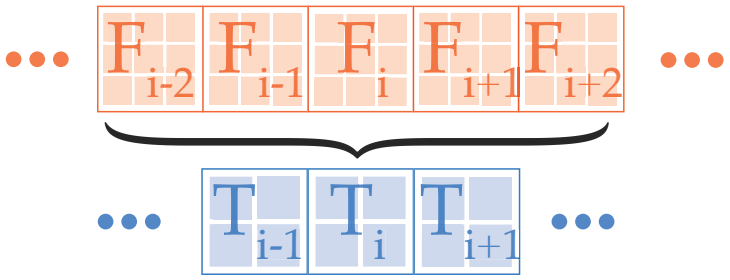


Figure 3.6: **Input and output** to our neural network. Our neural network takes as input deformation gradients given by yarn-level simulation in a small neighborhood window around a cross section, and estimates the cross-sectional deformation at that cross section.

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 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 3.6). 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 3.6), 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

$$\boldsymbol{n}' := \boldsymbol{S}(\boldsymbol{t}_0 \to \boldsymbol{t}')\, \boldsymbol{n}_0 \text{ and } \boldsymbol{b}' := \boldsymbol{S}(\boldsymbol{t}_0 \to \boldsymbol{t}')\, \boldsymbol{b}_0, \tag{3.5}$$

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

$$\boldsymbol{F}^{\text{local}} = [\boldsymbol{t},\ \boldsymbol{n},\ \boldsymbol{b}]\, \boldsymbol{F}, \tag{3.6}$$

where $\boldsymbol{F}$ is the original deformation gradient provided by the yarn-level simulation. Then, all the $\boldsymbol{F}^{\text{local}}$ matrices together in a neighborhood serve as an input to the regression network.

Lastly, since the normal and binormal directions given by Eq. (3.5) generally differ from

[1]Note not to confuse these frames with ones used for procedural fiber generation (3.3): 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).

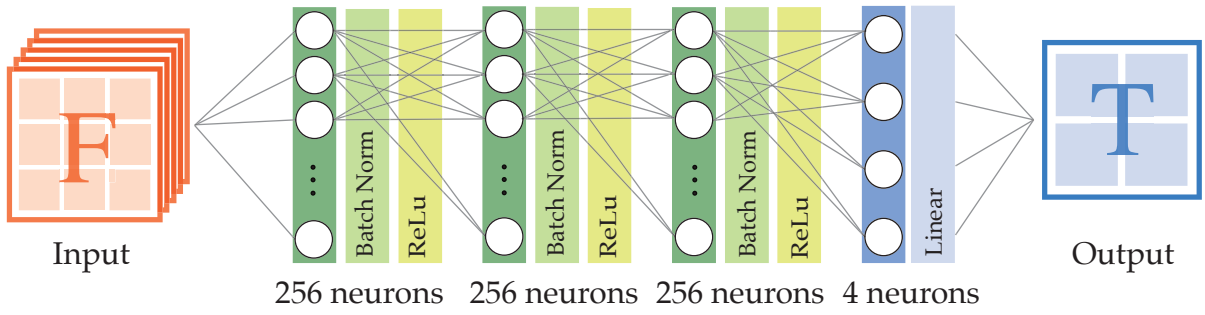


Figure 3.7: **Network architecture.** Our regression neural network involves three fully connected hidden layers with 256 nodes each (shown in green). The network acts as a 1D nonlinear filter along a yarn centerline and turns simulated deformation gradients $\boldsymbol{F}$ into cross-sectional fiber deformations $\boldsymbol{T}$ (see Figure 3.6).

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 $\boldsymbol{S}(\boldsymbol{n}_\alpha \to \boldsymbol{n}) \in \mathbb{R}^{2\times 2}$ denote the rotation that transforms $\boldsymbol{n}_\alpha$ to $\boldsymbol{n}$. Then, the cross-sectional deformation local to the neighborhood is

$$\boldsymbol{T}^{\text{local}} = \boldsymbol{S}(\boldsymbol{n}_\alpha \to \boldsymbol{n})\, \boldsymbol{T}\, \boldsymbol{S}(\boldsymbol{n}_\alpha \to \boldsymbol{n})^\top, \tag{3.7}$$

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:

$$\underbrace{\left\| \mathcal{N}\left(\{\boldsymbol{F}^{\text{local}}\}_i\right) - \boldsymbol{T}_i^{\text{local}} \right\|_2^2}_{\text{parameter loss}} + \lambda \underbrace{\left\| \mathcal{N}\left(\{\boldsymbol{F}^{\text{local}}\}_i\right) \boldsymbol{H} - \boldsymbol{T}_i^{\text{local}}\, \boldsymbol{H} \right\|_2^2}_{\text{regularization}}, \tag{3.8}$$

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. (3.6, 3.7) and include them in the training dataset. In practice, we found that this data augmentation step greatly improves the stability of our regression network.

Network implementation Our regression network consists of three fully connected hidden layers with 256 internal nodes each (see Figure 3.7). We implement this network using the Python-based Keras library [10] with Tensorflow [21]. We split our augmented data into a training set (85%) and a validation set (15%), and optimize the network weights using the Adam method [47].

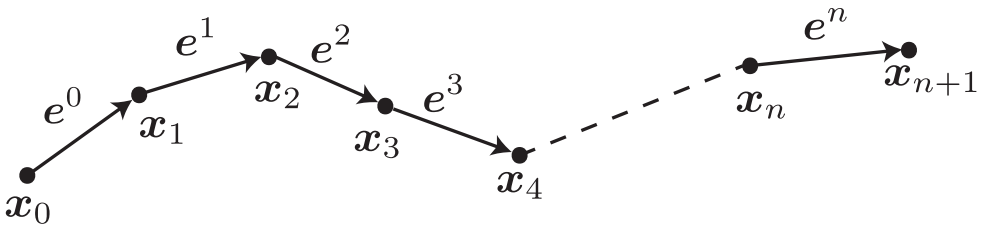


Figure 3.8: **Yarn/fiber discretization.** In yarn-level simulation, each yarn is represented by a series of particles connected on a polyline. In fiber-level simulation, each fiber is represented as a polyline, and a group of fibers are twisted to form a yarn.

3.3 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 (§3.2.2), and ii) to perform at runtime yarn-level simulation providing input to the trained deep neural network (§3.4).

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.

3.3.1 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 [8] and a hybrid Lagrangian/Eulerian method based on [37]. 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 [8]. The internal force is contributed by three types of elastic energies, namely, the stretching, bending, and twisting energy. Following the notation in [8] (summarized in Figure 3.8), the discrete stretching is defined as

$$E_{\text{stretch}} = \sum_{i=0}^{n} \frac{k}{2} \left(|\boldsymbol{e}^i| - |\bar{\boldsymbol{e}}^i| \right)^2, \tag{3.9}$$

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

$$E_{\text{bend}} = \sum_{i=1}^{n} \frac{\alpha(\boldsymbol{\kappa b}_i)^2}{2\bar{l}_i} \quad \text{and} \quad E_{\text{twist}} = \sum_{i=1}^{n} \frac{\beta m_i^2}{2\bar{l}_i}. \tag{3.10}$$

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 [8] 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. (3.9)) 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 [37], 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, $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{F}^i\boldsymbol{D}^i$, 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:

$$\boldsymbol{R}_1 = \begin{bmatrix} r_{11} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_2 = \begin{bmatrix} 1 & r_{12} & r_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}. \tag{3.11}$$

$\boldsymbol{R}_1$ 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 [37], 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 §3.3.2. Different from [37], 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 [37] 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 [8] and [37]. 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,

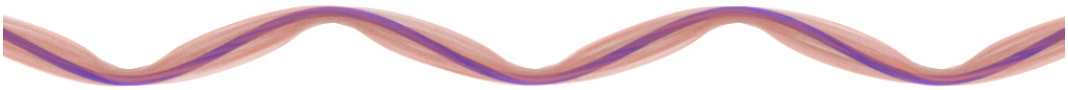


Figure 3.9: **Centerline of a fiber bundle.** A yarn is simulated at fiber level with individual fibers shown in blue. The yarn's centerline is highlighted in red. We fit yarn-level parameters so that the yarn curve resulting from yarn-level simulation closely matches the centerline from fiber-level simulation.

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 §3.2.2.

3.3.2 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. (3.9) and (3.10) 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 3.9).

Fitting parameters of internal forces

We first decide elastic rod parameters (k, α, and β) in Eqs. (3.9) and (3.10) 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 $\boldsymbol{f}_{\mathrm{s}}^{*}$ 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 $\boldsymbol{f}_{\mathrm{s}}$ at the end the yarn matches $\boldsymbol{f}_{\mathrm{s}}^{*}$. Because $\boldsymbol{f}_{\mathrm{s}}$ is proportional to k, the value of k can be quickly found through a binary search (Figure 3.10-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 3.10-b and -c.

Fitting parameters of external forces

As introduced in §3.3.1, 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 [37]. Here, r_{ij} are matrix elements defined in Eq. (3.11). γ, μ, and λ are parameters set according to specific type of fibers. In yarn-level simulation, we use

$$\begin{aligned} g(\boldsymbol{R}_2) &= a_1 r_{12}^2 + a_2 r_{12} r_{13} + a_3 r_{13}^2 \text{ and} \\ h(\boldsymbol{R}_3) &= b_1 r_{22}^2 + b_2 r_{22} r_{33} + b_3 r_{33}^2, \end{aligned} \tag{3.12}$$

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 $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ are positive semidefinite—a constraint known as the *conic constraint* in convex optimization. Moreover, in fiber-

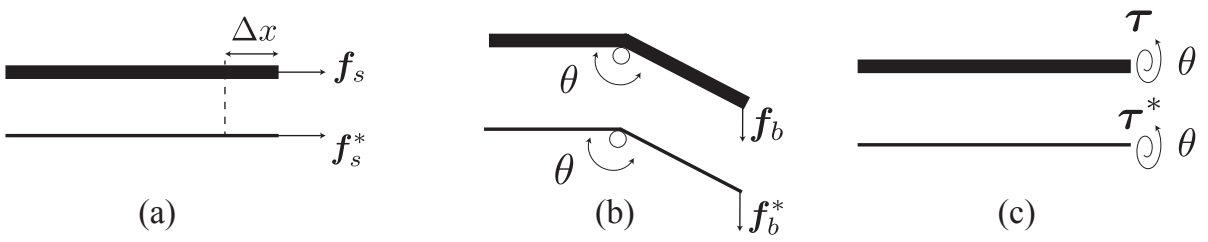


Figure 3.10: **Fitting elastic rod parameters. (a)** Fixing one end of a yarn, we stretch it by Δx in both fiber-level (top) and yarn-level (bottom) simulation. The stretching coefficient k is found in a binary search to match the resulting stretching forces $\boldsymbol{f}_{\mathrm{s}}^*$ and $\boldsymbol{f}_{\mathrm{s}}$. **(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 $\boldsymbol{f}_{\mathrm{b}}^*$ and $\boldsymbol{f}_{\mathrm{b}}$ 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 $\boldsymbol{\tau}$.

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. (3.12).

Estimating deformation gradients We now describe how to estimate yarn-level deformation gradients from fiber-level simulation results. As illustrated in Figure 3.11, 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 [67, 103] similar to that we encountered in §3.2.2. $\boldsymbol{F}$ is obtained by solving a small least-squares problem,

$$\boldsymbol{F} = \arg\min_{\boldsymbol{F},\boldsymbol{t}} \sum_{j=1}^{n} \|\boldsymbol{F}(\boldsymbol{x}_j - \boldsymbol{t}) - \bar{\boldsymbol{x}}_j\|_2^2, \tag{3.13}$$

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 [37]. Without interrupting the presentation flow, we defer it in the appendix, a six-dimensional semidefinite-quadratic-linear programming problem.

3.4 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 (§3.3);

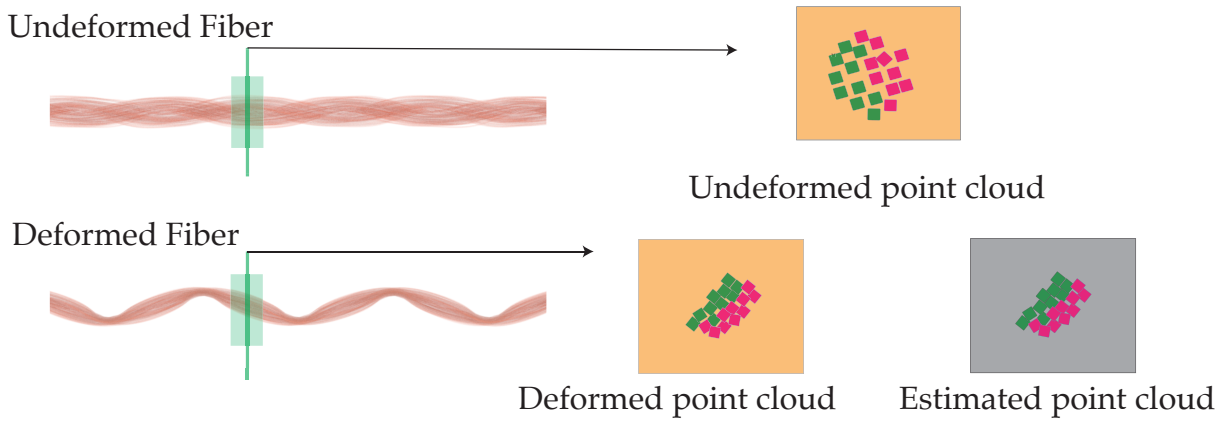


Figure 3.11: **Estimating deformation gradient.** Consider a small segment of a yarn (highlighted in red box). We visualize the cross-sectional fiber distributions on the right. In fiber-level simulation, the fiber distribution can get severely deformed (bottom-right). We estimate a "homogenized" deformation 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.

- The parameters for procedural generation of fibers via Eqs. (3.1) and (3.2);
- The pre-trained neural network that maps yarn-level deformation gradients to fiber-level deformations along yarn centerlines (§3.2).
- 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 3.1, 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 (§3.2.2), and feed them to the neural network to estimate the cross-sectional deformation matrix $\boldsymbol{T}_{\text{local}}$. Via the inverse of Eq. (3.7), this matrix $\boldsymbol{T}_{\text{local}}$ is transformed back to $\boldsymbol{T}$, which is in turn used to transform the fiber curves in Eq. (3.4). Finally, the fiber-level geometries are ready for rendering the fabric.

3.5 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 3.1. 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 (§3.3.2). 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. [37] 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 3.1: 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 [54] for better rendering performance.

3.5.1 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 3.12, we show experimental results for three types of yarns, rayon, cotton, and polyester, each using fitted procedural parameters from the previous work [100]. 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 3.12-c), which cannot be closely captured using previous methods that neglects fiber mechanics (Figure 3.12-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 3.12 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 3.13 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.

Multi-yarn results Figure 3.14 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 3.14-c).

Lastly, we perform a qualitative comparison between our results and photographs (Figure 4.7). 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 4.7-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 4.7-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.

3.5.2 Simulated Results

We now demonstrate the effectiveness of our technique via a number of experiments. Please see Table 3.2 for detailed statistics and performance numbers.

Our technique can be applied independently to individual yarns of a full fabric. Figure 3.16 shows rendered results of full knitted textiles. Figure 3.16-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 3.16-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

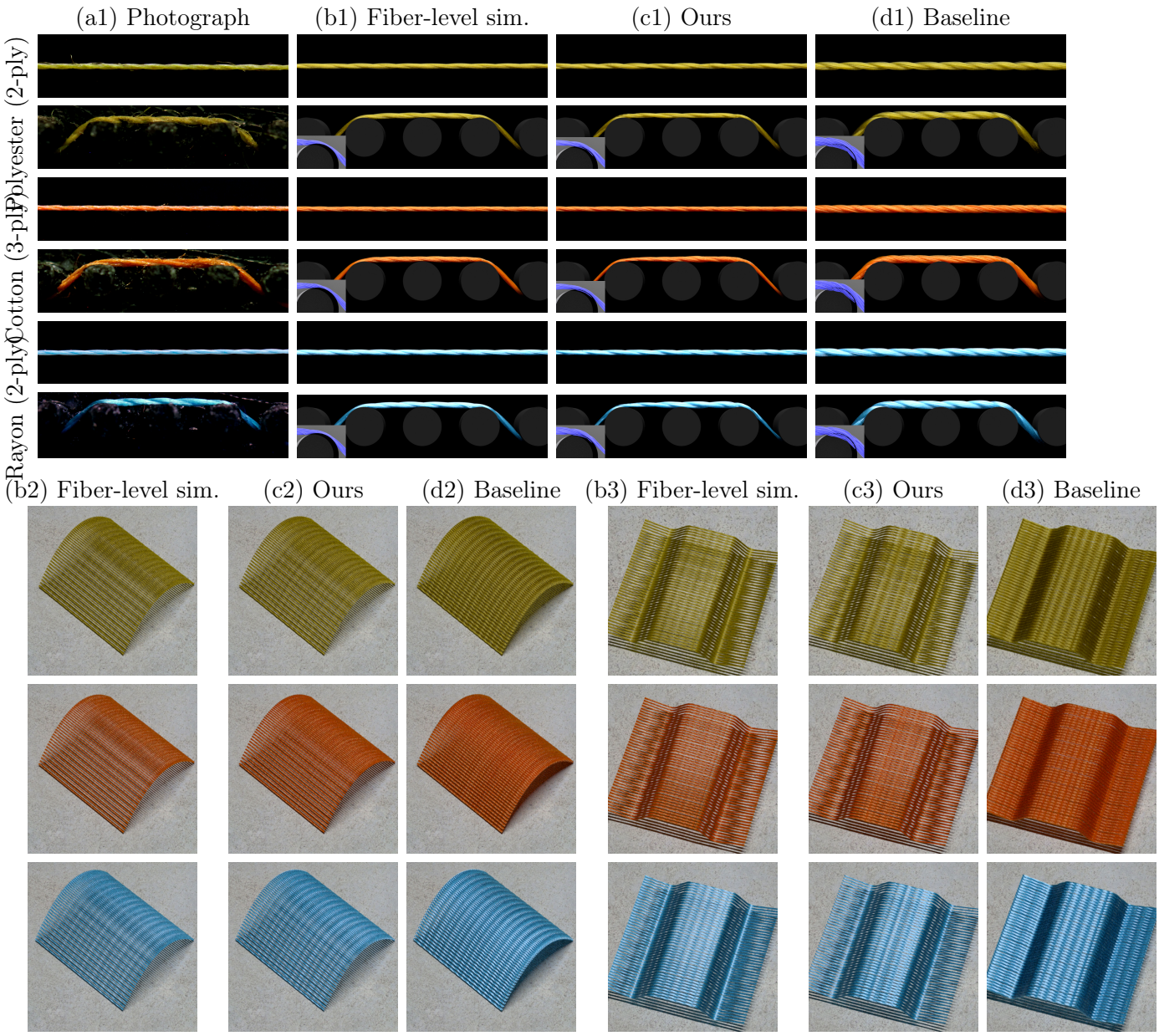
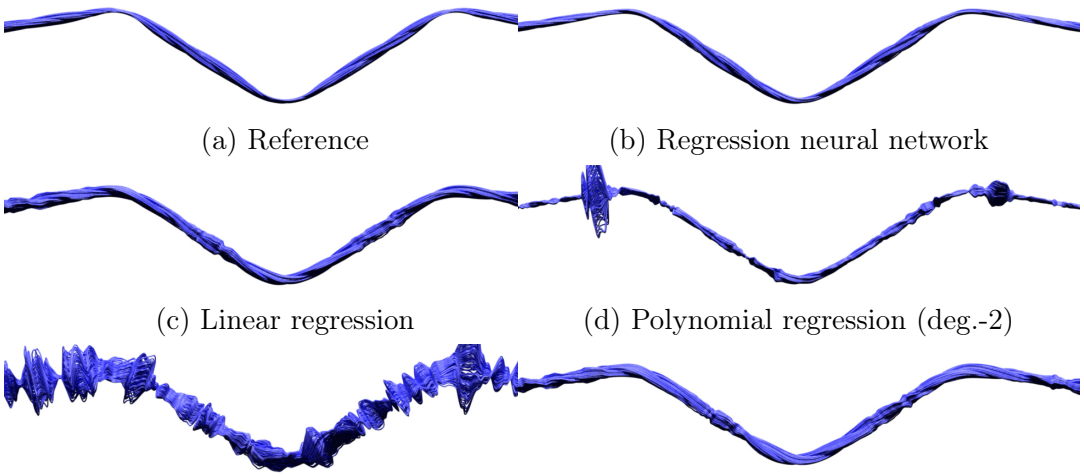


Figure 3.12: **Generated models for single yarns.** We demonstrate the effectiveness of our method by comparing our results (c) to photographs (a) and reference fiber-level simulations (b) using three types of yarns: polyester, cotton, and rayon. Column (d) shows procedurally generated yarns with fiber 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.

able to reproduce this effect in a visually convincing manner. In Figure 3.16-c, we show the stretching of a knitted glove modeled after the work by Wu and Yuksel [90]. The



(a) Reference (b) Regression neural network

(c) Linear regression (d) Polynomial regression (deg.-2)

(e) Polynomial regression (deg.-4) (f) Gaussian process

Figure 3.13: **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.

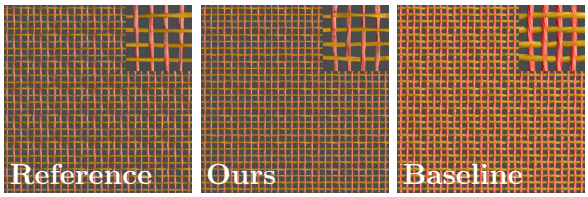


Figure 3.14: **Generated models for a small textile.** We simulate the dynamics of eight fibers (four horizontal and four vertical), as shown in the insets. Our result generated by yarn-level simulation and the fiber-deforming neural network matches closely the reference fully simulated at the fiber level.

mechanical responses of the yarns and fibers in this glove change its macro-scale appearance drastically: the glove not 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 3.17-a contains a small woven fabric with a flower logo resulting from its

Table 3.2: **Simulation and processing time statistics.** In Figure 3.12, 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 3.12	300	8 h	15 min	20 sec
Figure 3.16-a	85832	-	20 h	1 min
Figure 3.16-b	85832	-	18 h	30 sec
Figure 3.16-c	58545	-	28 h	1 min
Figure 3.17-a	100000	-	9 h	1 min
Figure 3.17-b	100000	-	10 h	1 min
Figure 3.17-c	1266000	-	110 h	10 min

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 3.17-a2). In Figure 3.17-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 3.17-c, we show a jacquard fabric being stretched both horizontally and vertically, resulting in complicated yarn and fiber deformations. Our model manages to produce spatially varying appearance changes caused by these deformations (Figure 3.17-c2), which are mostly absent if fiber-level deformations are neglected (Figure 3.17-c3).

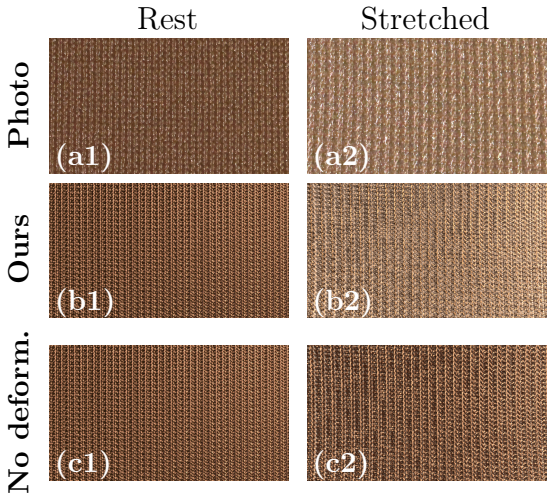


Figure 3.15: **Qualitative validation** of our technique. We compare photographs of a real knitted fabric made of nylon in rest (a1) and stretched (a2) states. With a virtual model with a similar rest appearance (b1), our method predicts its appearance change when the fabric is stretched (b2). Existing methods neglect the mechanical responses of yarns and fibers, and thus fail to capture this phenomenon (c1, c2).

3.6 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

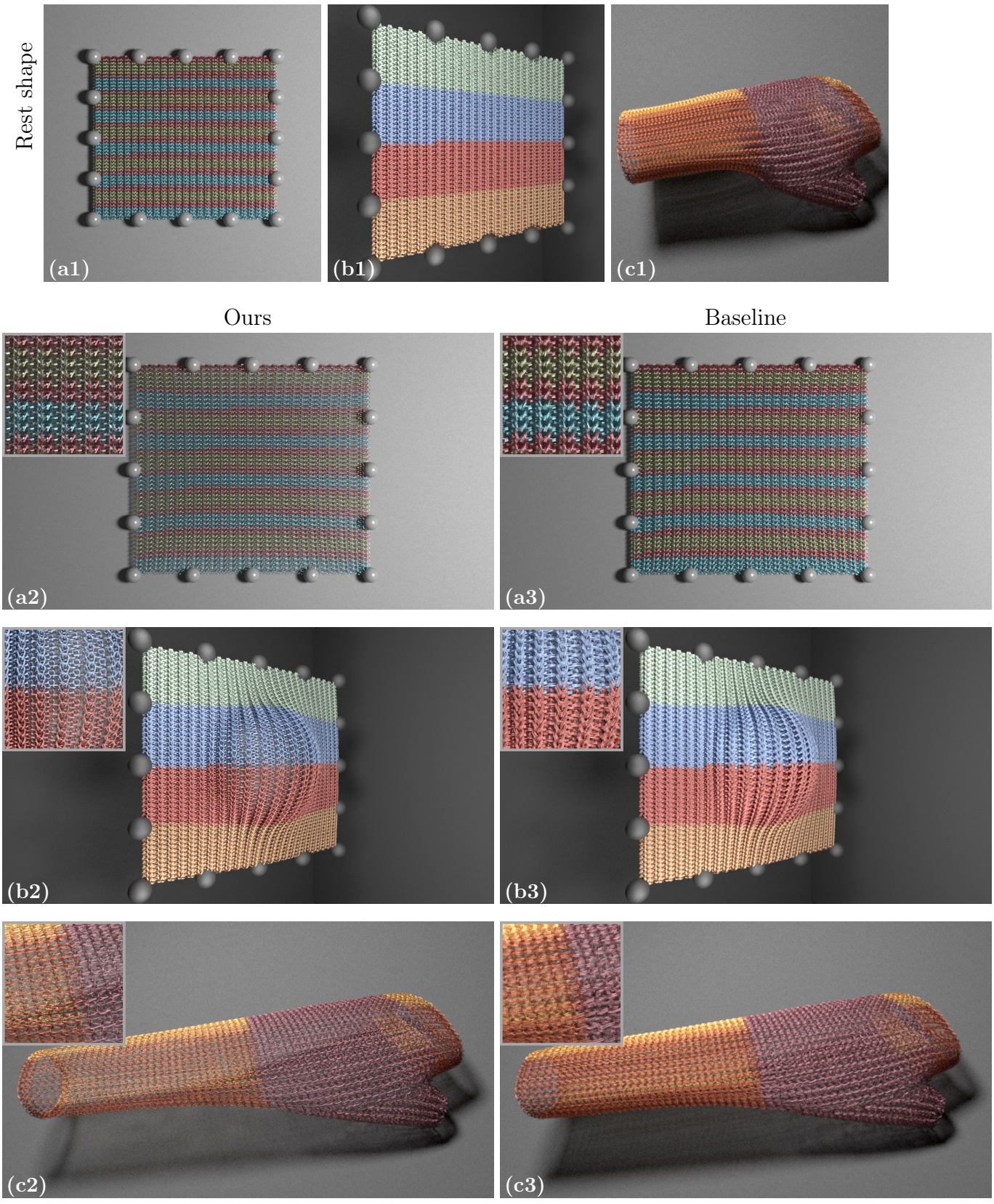


Figure 3.16: **Generated models for knitted fabrics.** Our technique can also capture the change of appearance of knitted fabrics caused by their mechanical responses to external forces. (1) fabrics in their rest shapes; (2) deformed fabrics modeled with our techniques; (3) deformed fabrics with identical 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.

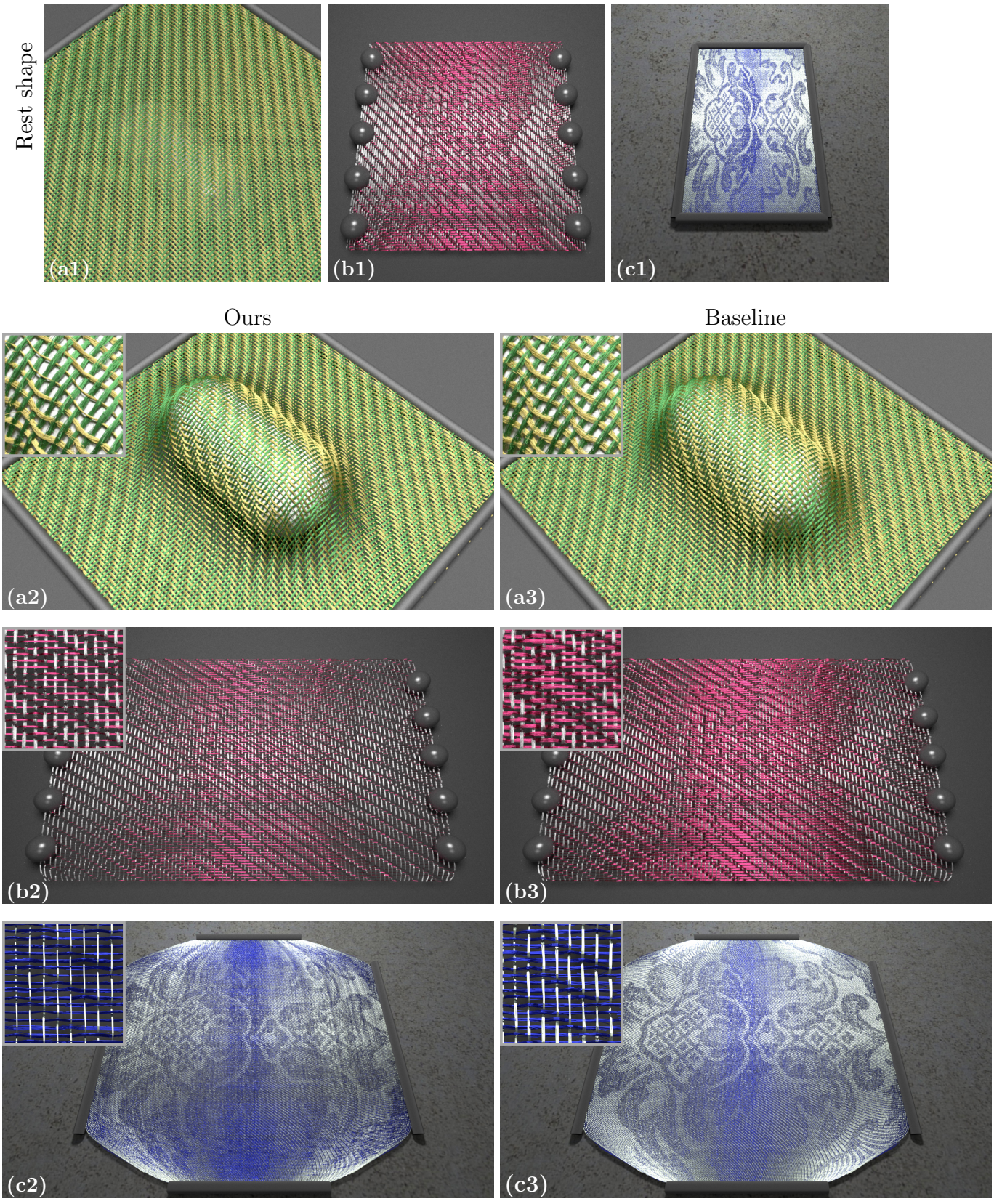


Figure 3.17: **Generated models for woven fabrics:** when applied to full fabrics, our technique is able to capture the change of appearance, both dramatic and subtle, due to fiber mechanics at both macro and micro scales. (1) fabrics in their rest shapes; (2) deformed fabrics modeled with our techniques; (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.

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 [77, 100]. 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 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.

Chapter 4

Ply-Based Appearance Modeling for Woven Fabrics

Modeling cloth appearance generally boils down to two distinct representation: *surface-based* and *micro-appearance*. In practice, both categories have their own advantages and disadvantages. The surface-based models are typically light-weight, easy to edit, and can faithfully reproduce the appearance of a fabric at the macroscale. Thus, they have been widely used in the computer graphics industry [59, 22]. However, these models lack the fine-grained details that are crucial for cloth rendering under close-up views. The micro-appearance models, in contrast, are capable of generating renderings with remarkably high fidelity and details. Unfortunately, as these models are highly data-intensive and usually difficult to manipulate, their practical use has been quite limited. Furthermore, both the surface-based and the micro-appearance techniques have mostly focused on modeling the reflection of incident light. *Validation against measurements of* the transmission of light, on the other hand, has been largely neglected *at the macroscale*.

There are many reasons why previous models have been unsuccessful in capturing both

reflection and transmission properties of textiles. In the quest to obtain very high fidelity renderings micro-appearance models relies on a very detailed representation of the textile. This is very useful for understanding how different aspects of the micro-geometry affect the overall appearance, but makes the models inflexible for dealing with textiles that do not fall within the idealized micro-properties of these models. Aliaga et al. [3] for example, demonstrate a wide range of optical properties for different textile fibers.

In the quest for more realistic cloth appearance models the addition of highly detailed fiber geometry can be seen as a form of reductionism [4]. While these models with increasingly detailed geometry offer incredible detail they also require more complex controls (fiber arrangement and shape as well as local fiber shading models). This added complexity makes the models difficult to control and edit, and previous work [44] as well as our results show that fiber based models cannot easily match both reflected and transmitted lighting in measured cloth samples. In this paper we argue that an aggregrate model offers a compromise where the appearance is easier to manipulate, while retaining most of the details from the fiber level.

Concretely, our main contributions include:

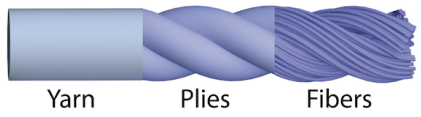
- A new geometric model (§4.1) that depicts a fabric as a collection of *plies* that are further enriched using procedurally generated fiber details. By working at the ply level, this representation offers a good balance between model complexity and fidelity.
- A new light scattering model (§4.2) that expresses how light interacts with a ply by capturing how, on average, light scatters off the constituent fibers. Thanks to this aggregate formulation, our cloth models can be rendered much more efficiently compared to the previous micro-appearance models. Additionally, by decoupling reflection and transmission properties, our light scattering model offers the flexibility to accurately reproduce both effects.

To demonstrate the practical usefulness of our technique, we have integrated it into an industrial rendering system that provides user-friendly material parameter tweaking.

4.1 Modeling Cloth Geometry

We now describe how our model depicts detailed geometry of woven cloth. Unlike most prior models that either lack sub-yarn-level information [29, 72] or rely on explicit fiber structures [96, 44, 101], our expression offers a good balance between performance and fidelity by mainly working at the ply level.

Preliminaries The geometric structure of a textile is complex at many scales. Fabrics are constructed from multiple *yarns* via manufacturing technologies like weaving and knitting. Each yarn, in turn, consists of few sub-strand bundles, or *plies*, that are created by twisting hundreds of micron-diameter fibers. The variations of these fiber details can impact the overall appearance [96, 29, 44].



Round yarn



Flat yarn

We consider two types of yarns, *round* and *flat*, as illustrated in the adjacent figure. Round yarns have cylindrical shapes and are very common in woven fabrics. Flat yarns, which are usually used in light and glossy fabrics such as silk satin, are ribbon-like and consist of well aligned fibers.

Overview We take as input a polygonal mesh (with per-vertex UV coordinates), which captures a fabric's macro-geometry, and a 2D weave pattern that specifies the arrangement of the yarns. Using a simple physical model, we compute a tilable set of yarn centerline curves

Table 4.1: A list of important symbols.

Geometry (§4.1)	P	Weave pattern
	Y	Yarn curve
	s	Parameter along yarn curve
	s_k	Piecewise yarn polynomial knot locations
	α_k, h_k	Yarn polynomial control points
	S_i	Length of yarn period
	h	Yarn height polynomial
	o	Yarn offset
	$\hat{\mathbf{t}}, \hat{\mathbf{b}}, \hat{\mathbf{w}}$	Yarn direction, offset and height basis
	k, k_c	Yarn strain and collision coefficient
	E	Yarn and weave pattern energy
	E_C	Yarn collision energy
Appearance (§4.2)	f	BSDF
	$f^{\mathrm{S}}_{\boldsymbol{x}}$	Specular reflection at $\boldsymbol{x}$
	$f^{\mathrm{S}}_{\boldsymbol{y}}$	Specular transmission at $\boldsymbol{y}$
	$f^{\mathrm{B}}_{\boldsymbol{x}}$	Body reflection at $\boldsymbol{x}$
	$f^{\mathrm{B}}_{\boldsymbol{y}}$	Body transmission at $\boldsymbol{y}$
	$\boldsymbol{x}, \boldsymbol{y}$	entrance and exit point
	τ	Volumetric transmittance of light
	σ_{t}	Material extinction coefficient (optical density)
	η	Material refractive index
	$\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}$	Incoming and outgoing light direction
	$\boldsymbol{\omega}_{\mathrm{h}}$	Half-way vector (normalized)
	$\boldsymbol{n}$	Surface normal
	G_{f}	Fiber-shadowing function
	G_2	Smith non-correlated masking-shadowing function
	F^{S}	Reflection fresnel term
	D	Microfacet Normal distribution function (NDF)
	β	Anisostropic roughness
	k^{S}	Specular tint
	k^{B}	Base color
	α	Reduced single-scattering albedo convertor
	B	Body energy
	r	Body reflected coefficient
	h	Thickness of virtual medium for flat yarns

(§4.1.1) that can be mapped to the input geometry in a manner analogous to displacement mapping. The plies are constructed as helices mapped from a yarn centerline space to fabric space similar to previous procedural techniques [101, 55] (§4.1.2). Finally, fiber-level details are added in a procedural texture-based fashion on individual plies (§4.1.3). We summarize this pipeline in Figure 5.1.

4.1.1 Yarn Centerlines

One strategy for obtaining yarn centerlines is to ensure the curves pass over and under each-other as dictated by the weaving pattern. While this may give reasonably looking results the resulting distribution of slopes will not match those computed by more complex simulations [51] nor those found by investigating the specular highlights [72]. In the following, we describe a physical model that can reproduce the slope distributions of yarns reported in [72, 51] and at the same time produce tileable curves.

Relaxation of yarn curves As [51], we assume that the textile is flat compared to the size of the yarns and perform yarn relaxation in flat space. In this space, the yarns in the pattern run along two primary orthogonal axes: the warp (x) and weft (y) directions. Each yarn centerline is a curve along one of these primary directions with the displacement (z) away from the tiling plane as the unknown quantity we need to compute. A weave pattern P with N warps and M wefts are placed in the rectangle $[0, N] \times [0, M]$ and the yarn offsets at half-integer coordinates. For example, the first warp curve is placed at $y = 0.5$ and runs along the x-axis from $[0, M]$. Further, we assume this pattern to be tiled over the entire plane, yielding periodic boundary conditions as detailed below.

In general, a yarn Y_i has a primary direction $\hat{\mathbf{t}}_i$ and a transverse direction $\hat{\mathbf{b}}_i$: for warps, $\hat{\mathbf{t}}_i$ is the x-direction and $\hat{\mathbf{b}}_i$ is the y-direction; for wefts, the opposite holds. Under our

formulation, a yarn curve can be expressed as

$$Y_i(s) = s\,\hat{\mathbf{t}}_i + o_i\,\hat{\mathbf{b}}_i + h_i(s)\,\hat{\mathbf{w}}, \tag{4.1}$$

where o_i is the half-integer offset of the yarn centerline and h_i is a function describing the deviation of the centerline from a straight line and $\hat{\mathbf{w}}$ is the direction orthogonal to both weave directions. Each yarn satisfy the periodic boundary conditions $h_i(s + S_i) = h_i(s)$ and $h_i'(s + S_i) = h_i'(s)$ where $S_i = M$ for warps and $S_i = N$ for wefts. This allows tiling the cell $[0, N] \times [0, M]$ with C^1-continuity.

For $h_i(s)$ we choose a piece-wise cubic polynomial with Q_i knots equidistantly placed between $s = 0$ and the period length ($S_i = N$ or $S_i = M$). At each knot s_k we specify the height $h_{i,k}$ and the slope $\alpha_{i,k}$ which uniquely defines the cubic polynomial within each segment[1] $[s_k, s_{k+1}[$. Note that the periodic boundary conditions imply that $h_{i,0} = h_{i,Q_i-1}$ and $\alpha_{i,0} = \alpha_{i,K_i-1}$ for all yarns Y_i.

Self-energy of a single yarn We opt for a simple strain-energy function of the deformed yarn given by

$$E[Y_i] = \frac{k}{2} \int_0^{S_i} \left(\frac{\partial h_i(s)}{\partial s} \right)^2 \mathrm{d}s, \tag{4.2}$$

where k is the yarn stiffness coefficient. Since h_i is a piece-wise cubic polynomial the energy $E[Y_i]$ and its derivatives can be expressed analytically in terms of the control points $\{h_{i,j}, \alpha_{i,j}\}$ which is detailed in Appendix **??**.

Collision energy To handle yarn-yarn collisions, we select the density of knots such that there is at least one knot at each yarn crossing and one knot in between each crossing. Let

[1] We use '$[s_k, s_{k+1}[$' to denote an interval that is closed on s_k and open on s_{k+1}.

P_C be the set of pairs of knots located at yarn crossing. For example if $(A, B) \in P_C$ denotes a yarn crossing index pair $A = (i, k)$, $B = (j, l)$ then yarn i and j intersect at control points k and l respectively.

We model the collision energy of these two control points as soft spheres. We denote the effective collision radii of these spheres by R_A and R_B which for round yarns is the radius of the yarns, but for flat yarns is reduced to allow the flat yarns to approach each other more closely. The collision energy is then given by

$$E_C(A, B) = k_c \, f \left[\left(\frac{h_A - h_B}{R_A + R_B} \right)^2 \right], \tag{4.3}$$

where k_c controls the magnitude of the collision energy and f is a collision potential [40] that equals $d^{-2} + d^2 - 2$ if $d < 1$ and is zero otherwise.

Energy of a woven fabric Adding the strain- and collision-energies up for all yarns gives the full energy functional for the pattern

$$E = \sum_{i \in P} \frac{k_i}{2} \int_0^{S_i} \left(\frac{\partial h_i(s)}{\partial s} \right)^2 \mathrm{d}s + \sum_{A,B \in P_C} E_C(A, B), \tag{4.4}$$

where we have allowed the stiffness coefficient to depend on the yarn i. Using a good non-linear optimization technique such as L-BFGS [20], this energy can be minimized at interactive speeds for small weaving patches even for a single-threaded CPU computation. As noted previously, yarn centerlines obtained this way matches the slopes reported by [72, 51] well and can therefore also reproduce specular highlights at the same locations.

With the yarn centerlines computed in a unit-cell, we can map the curves to a UV-mapped mesh. Let $p : \mathbb{R}^2 \mapsto S$ be the mapping from the mesh UV space to the surface and $\boldsymbol{n} : S \mapsto S^2$ be the normal field of the surface. Then each point (x, y, z) in yarn-space can be mapped to

the mesh space by computing

$$\phi(x, y, z) = p(x, y) + H\, z\, \boldsymbol{n}(p(x, y)), \tag{4.5}$$

where H is a mesh- and uv-mapping dependent scale factor to ensure ϕ is approximately isometric. In other words, we displace the yarn height along the normal of the mesh. In general, each point of the surface has a frame defined by the UV-tangent plane basis and the normal. This frame allows us to map any vector-field in weave-space to mesh space, allowing a natural transformation of yarn tangent and normal information to mesh space.

4.1.2 Ply-Level Geometry

Recall that a real yarn can consist of multiple twisted plies. After obtaining the yarn centerlines (§4.1.1), we generate the ply curves following the individual yarn curves. The plies add both geometric and shading complexity that greatly increases the realism compared to just using a single yarn.

Similar to previous works [58, 76, 101], we model ply curves as helices around each yarn curve with user-controllable parameters (e.g., ply count and pitch). This is shown in Figure 5.1-c. The ply-curves are subsequently rendered as piecewise cylinders. Flat yarns are not constructed from plies and are instead rendered as bilinear patches.

4.1.3 Fiber-Level Details

Our model enriches the ply surfaces by adding procedurally generated fiber-level details with normal, tangent, and shadowing maps. To this end, we utilize a procedural yarn model [76, 101] to drive the generation of these maps.

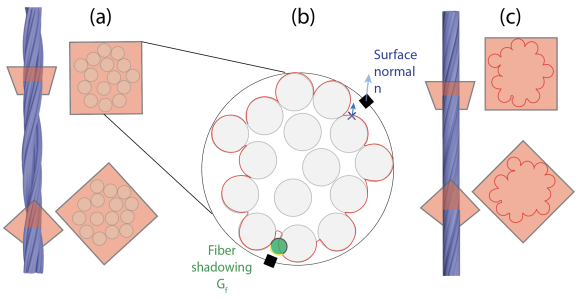
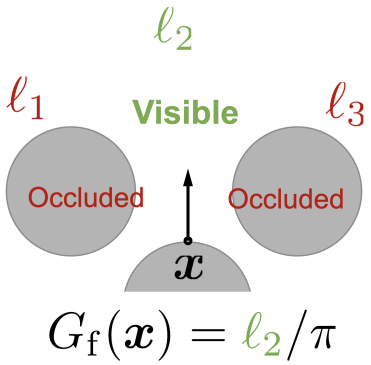


Figure 4.1: **Cross section processing**. (a) We generate the cross section of a ply using a procedural model. (b) Within this cross section we perform 2D ray tracing to detect fibers visible from the exterior. We find the normals for these visible fibers using the same ray tracing process, and tangents are computed using derivative of the corresponding helix in 3D. Furthermore, local fiber shadowing is calculated for the visible fibers using 2D ray tracing. (c) Using the fiber normals, tangents, and shadowing textures we can shade the ply curve and account for shading details due to the individual fibers. Lastly, we add fiber irregularities as stated in §4.1.3.

Specifically, we generate the cross section of a ply using the procedural model. Within this cross section, each fiber is a small 2D disc (with its location and radius specified by the model). Provided the cross-sectional arrangement of fibers, we perform ray tracing within the cross section to detect *visible fibers*—those visible from the exterior (see Figure 4.1-b).

Further, we extract the normal information of all visible fibers via the same ray tracing process and store the results as a 1D texture (see Figure 4.1-b). Tangent information of each visible fiber is calculated using the derivative of corresponding helix in 3D using the fiber distance from the ply center. Lastly, we compute local fiber shadowing, as shown in 4.1-b, that approximates the occlusion from neighboring fibers. Specifically, our inter-fiber shadowing term, which is a scalar ranging from 0 to 1, captures the ratio of incident directions that are not occluded. For fast computation, we consider the fibers to be locally straight and approximate the shadowing term by computing within ply cross sections as the ratio of the angle of the visible arc to 2π (see the adjacent figure). This gives the exact result for a point on the bottom of an infinite V-groove and is otherwise an approximation.



Introducing irregularities For round plies the fibers are circular helices around the ply centerlines, the cross-sectional normal and tangent information is almost invariant along a ply up to a global rotation (see Figure 4.1-a). This allows us to use a 1D texture, which adds minimal computational and storage overhead. However, the use of rotationally invariant ply cross sections can lead to results that suffer from visually disturbing periodic patterns. We tackle this challenge by switching randomly among a few precomputed fiber arrangements that are distributed randomly along the ply. We interpolate between two 1D fiber arrangements using a smoothstep function. This effectively mimics the migration of fibers (see the adjacent figure).



Without migration



With migration

Flyaway fibers Flyaway fibers are those that fray out from the textile. We use a technique similar to [96] by randomly modeling these fibers as separate pieces of geometry depicted as B-Spline curve (with user-controlled densities). Each flyaway is generated at a random location within a given ply. The initial direction of the flyaway is a randomly chosen angle away from the ply direction. The maximum angle and length are user controlled parameters. Subsequent flyaway segments are generated as continuations of the previous flyaway segment with a new random angle to create a flyaway curve with irregularities.

4.2 Modeling Cloth Appearance

To simulate how light is scattered by the generated ply-level geometry (§4.1) we build on recent work on appearance modeling of hair and fur [104, 17, 92]. One major difference to hair and fur rendering is that we compute light scattering with an aggregate bundle of cloth fibers rather than considering how light is scattered by each individual fiber.

Overview Our model works at the ply level with decoupled reflection and transmission components. This decoupling offers great flexibility and control than previous fiber-level micro-appearance models (this is demonstrated in Figure 4.7). We use a near field bidirectional scattering distribution function (BSDF) to simulate light interaction at a specific location on a ply.

At a high level, our model depicts how light is scattered (i.e., reflected and refracted) at two surface points jointly (see Figure 4.2). The first point $\boldsymbol{x}$ is the intersection between an incident light ray and a ply surface, which arises from standard path tracing. The second point, $\boldsymbol{y}$, is where a (randomly sampled) refracted light ray originating at $\boldsymbol{x}$ intersects the ply surface (from the interior). At each point, our model utilizes a specialized BSDF that combines a *specular* and a *body* component. The specular term (denoted using a superscript S) captures (anisotropic) specular reflection and transmission, and the body term (denoted with a superscript B) approximates multiple scattering within the bundle of fibers.

When a light ray intersects a ply from the exterior at $\boldsymbol{x}$, we use the following BSDF: $\boldsymbol{x}$

$$f_{\boldsymbol{x}} = f_{\boldsymbol{x}}^{\mathrm{S}} + f_{\boldsymbol{x}}^{\mathrm{B}}. \tag{4.6}$$

Subsequently, when the light ray follows the transmission component of $f_{\boldsymbol{x}}^{\mathrm{S}}$ and intersects the ply from the interior at $\boldsymbol{y}$, we use another BSDF:

$$f_{\boldsymbol{y}} = f_{\boldsymbol{y}}^{\mathrm{S}} + f_{\boldsymbol{y}}^{\mathrm{B}}. \tag{4.7}$$

In the following, we define in detail the specular and body components at both points.

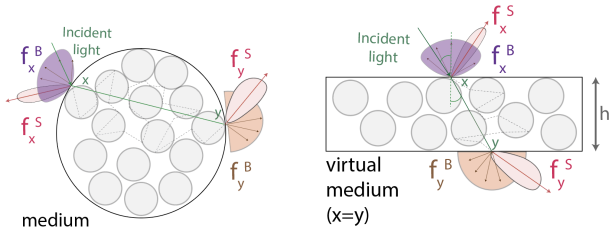


Figure 4.2: **Illustration of our Ply BSDF model for round (left) and flat (right) plies.** At entrance point $\boldsymbol{x}$ and exit point $\boldsymbol{y}$, our model utilizes a specialized BSDF comprised of a *specular* and a *body* component where the former (denoted using a superscript S) captures (anisotropic) specular reflection and transmission, and the latter (denoted with a superscript B) approximates multiple-scattered light.

4.2.1 Specular Components

Fabrics have evident specular and directional highlights due to the immediate reflection as well as light transmission from the other side of the cloth. We use the rough dielectric BSDF [84, 26] to model the specular components. The specular component of the BSDF at an entrance point $\boldsymbol{x}$ can be written as:

$$f_{\boldsymbol{x}}^{\mathrm{S}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = k_{\boldsymbol{x}}^{\mathrm{S}} \cdot \begin{cases} \frac{F_{\boldsymbol{x}}\, G_2(\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}},\boldsymbol{\omega}_{\mathrm{h}})\, D(\boldsymbol{\omega}_{\mathrm{h}};\boldsymbol{\beta}_{\boldsymbol{x}})}{4\,|\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{n}(\boldsymbol{x})|\,|\boldsymbol{\omega}_{\mathrm{o}}\cdot\boldsymbol{n}(\boldsymbol{x})|}, & (\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{\omega}_{\mathrm{o}} > 0) \\ \frac{|\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{\omega}_{\mathrm{t}}|\,|\boldsymbol{\omega}_{\mathrm{o}}\cdot\boldsymbol{\omega}_{\mathrm{t}}|}{|\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{n}(\boldsymbol{x})|\,|\boldsymbol{\omega}_{\mathrm{o}}\cdot\boldsymbol{n}(\boldsymbol{x})|}\, \frac{\eta^2\,(1-F_{\boldsymbol{x}})\, G_2(\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}},\boldsymbol{\omega}_{\mathrm{t}})\, D(\boldsymbol{\omega}_{\mathrm{t}};\boldsymbol{\beta}_{\boldsymbol{x}})}{[(\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{\omega}_{\mathrm{t}})+\eta\,(\boldsymbol{\omega}_{\mathrm{o}}\cdot\boldsymbol{\omega}_{\mathrm{t}})]^2}, & (\boldsymbol{\omega}_{\mathrm{i}}\cdot\boldsymbol{\omega}_{\mathrm{o}} < 0) \end{cases} \tag{4.8}$$

where $\boldsymbol{n}(\boldsymbol{x})$ is the surface normal at $\boldsymbol{x}$, $F_{\boldsymbol{x}}$ indicates the Fresnel reflection coefficient, η is the material's refractive index,[2] $\boldsymbol{\omega}_{\mathrm{h}}$ is the normalized half vector given by

$$\boldsymbol{\omega}_{\mathrm{h}} = {}^{(\boldsymbol{\omega}_{\mathrm{i}}+\boldsymbol{\omega}_{\mathrm{o}})}\!/_{\|\boldsymbol{\omega}_{\mathrm{i}}+\boldsymbol{\omega}_{\mathrm{o}}\|}, \qquad \boldsymbol{\omega}_{\mathrm{t}} = {}^{(-\boldsymbol{\omega}_{\mathrm{i}}-\eta\boldsymbol{\omega}_{\mathrm{o}})}\!/_{\|-\boldsymbol{\omega}_{\mathrm{i}}-\eta\boldsymbol{\omega}_{\mathrm{o}}\|}, \tag{4.9}$$

$D(\cdot;\boldsymbol{\beta})$ denotes the normal distribution function (NDF) where we use GGX distribution with width parameter $\boldsymbol{\beta} \in \mathbb{R}^2$ that captures the (anisotropic) roughness of a surface, and

[2] We have assumed the external refractive index to be one.

G_2 is the Smith uncorrelated masking-shadowing function. Additionally, $k_{\boldsymbol{x}}^{\mathrm{S}}$ in Eq. (4.8) is a user-controlled specular albedo.

Similarly, at the exit point $\boldsymbol{y}$, for all $\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{n}(\boldsymbol{y}) > 0$, the specular component uses the transmitted component of the rough dielectric model:

$$f_{\boldsymbol{y}}^{\mathrm{S}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = k_{\boldsymbol{y}}^{\mathrm{S}} \cdot \tau(\boldsymbol{x}, \boldsymbol{y}) \cdot \begin{cases} 0, & (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}} > 0) \\ \frac{|\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{t}}|\,|\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{\omega}_{\mathrm{t}}|}{|\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}(\boldsymbol{y})|\,|\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{n}(\boldsymbol{y})|} \frac{(1-F_{\boldsymbol{y}})\, G_2(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{t}})\, D(\boldsymbol{\omega}_{\mathrm{t}}; \boldsymbol{\beta}_{\boldsymbol{y}})}{[\eta\,(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{t}}) + (\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{\omega}_{\mathrm{t}})]^2}, & (\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}} < 0) \end{cases} \tag{4.10}$$

where $\boldsymbol{\omega}_{\mathrm{t}} = {}^{(-\eta\boldsymbol{\omega}_{\mathrm{i}} - \boldsymbol{\omega}_{\mathrm{o}})}/_{\|-\eta\boldsymbol{\omega}_{\mathrm{i}} - \boldsymbol{\omega}_{\mathrm{o}}\|}$, $F_{\boldsymbol{y}}$ indicates the Fresnel reflection coefficient, and $k_{\boldsymbol{y}}^{\mathrm{S}}$ is a user controlled specular transmission color, $\tau(\boldsymbol{x}, \boldsymbol{y})$ accounts for the transmission of light through the fiber volume between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\tau(\boldsymbol{x}, \boldsymbol{y}) := \exp(-\sigma_{\mathrm{t}} \|\boldsymbol{x} - \boldsymbol{y}\|), \tag{4.11}$$

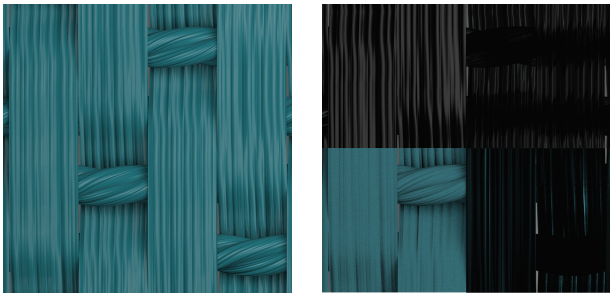
where σ_{t} is the material's extinction coefficient.

In Eq. (4.8), we use slightly higher surface roughness $\boldsymbol{\beta}_y$ to allow the transmission lobe (and the subsequent exiting location $\boldsymbol{y}$) to have a wider spread, approximating multiple internal reflections. This is similar to previous work on hair rendering [57].

4.2.2 Body Components

We approximate multiple scattering of light within a ply with body reflection and transmission terms $f_{\boldsymbol{x}}^{\mathrm{B}}$ and $f_{\boldsymbol{y}}^{\mathrm{B}}$ in Eqs. (4.6) and (4.7). We represent both terms using diffuse-like distributions[3].

[3]Although this is a rough approximation to the true single- and multiple-scattering phenomenon, we find it adequate in practice since cloth fibers are usually disorganized, causing light to diffuse quickly inside a



Full Components

Figure 4.3: **Components of our Ply-BSDF:** specular reflection (top left), specular transmission (top right), body reflection (bottom left), body transmission (bottom right)

We model the body of the ply as a homogeneous isotropic medium and approximate the scattering within using a BSDF approximation. This resulting BSDF body reflection term $f_{\boldsymbol{x}}^{\mathrm{B}}$ at the entrance point $\boldsymbol{x}$ is then comprised of two terms: The first term is a Lommel-Seeliger (LS) term that approximates single-scattered light within the ply medium [25, 35] and the second term approximates multiple scattering within the ply (that is, light enters the ply from $\boldsymbol{x}$, scattered many times internally before reemerging at $\boldsymbol{x}$); We observe that the introduction of the LS term to our reflection component is important for reproducing the smooth appearance of real-world plies.

$$f_{\boldsymbol{x}}^{\mathrm{B}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = r\, B(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})\, \mathrm{G_f}(\boldsymbol{x}) \Bigg[\frac{\alpha}{4\pi(|\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}(\boldsymbol{x})| + |\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n}(\boldsymbol{x})|)} + \frac{\alpha}{2\pi} \left(1 + \mathrm{e}^{-\frac{4}{3}A\sqrt{3(1-\alpha)}}\right) \mathrm{e}^{-\sqrt{3(1-\alpha)}} \Bigg], \quad (4.12)$$

where α is the (reduced) single-scattering albedo, which is computed from a user controlled diffuse base color, $k_{\boldsymbol{x}}^{\mathrm{B}}$, as described by Jensen and Buhler [34]. $\mathrm{G_f}(\boldsymbol{x})$ is the fiber-shadowing term discussed in §4.1.3 and $r \in [0, 1]$ is the portion of the *total body scattering energy* B that exits at point $\boldsymbol{x}$; assuming that all light is scattered within the fibers this value approximates forward and backward scattering. For details on the other terms in the equation please see

ply. A similar approach was used by Yan et al. [92] to account for scattering by the medulla in fur.

the work by Jensen et al. [35].

Multiple scattered light, $f_{\boldsymbol{y}}^{\mathrm{B}}$, that exits at the backside, y, is modeled using a simple Lambertian term:

$$f_{\boldsymbol{y}}^{\mathrm{B}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = (1 - r)\, B\, \mathrm{G_f}(\boldsymbol{y})\, \frac{k_{\boldsymbol{y}}}{\pi}. \tag{4.13}$$

where, $(1 - r)$ is the transmitted portion of body energy B (whose dependency on incident and outgoing directions at $\boldsymbol{x}$ are omitted for notational convenience), and $k_{\boldsymbol{y}}$ and $\mathrm{G_f}(\boldsymbol{y})$ are respectively, a user controlled diffuse transmission color and the fiber-shadowing term for point $\boldsymbol{y}$.

B is the fraction of the incident light that is multiple scattered. By assuming that absorption can be ignored we can find this fraction as the transmitted energy that is not also specularly transmitted at the backside:

$$B(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) \approx\ (1 - F_{\boldsymbol{x}})\,(1 - \tau(\boldsymbol{x}, \boldsymbol{y}) \cdot\ (1 - F_{\boldsymbol{y}})) \tag{4.14}$$

Note, that we have included user controlled specular and diffuse reflectance and transmission terms. Technically, only the fiber absorption and scattering should be used to control the color of the reflected and the transmitted light, but given the fact that the actual scattering within the fiber volume is approximated and that the non-linear behavior of scattering media is difficult to control we decided to add these explicit user controls. This decoupled control of the different emerging lobes is similar to current approaches for hair rendering [74].



Energy-conservation test

for round yarn



Energy-conservation test

for flat yarn

4.2.3 Round and Flat Yarns

We use the BSDFs expressed in §4.2.1 and §4.2.2 for both round and flat yarns. For round yarns where the plies are modeled as cylindrical pipes, we use ray tracing to find entrance and exiting points $\boldsymbol{x}$ and $\boldsymbol{y}$. For flat yarns where the plies are modeled as ribbon-like surfaces, on the other hand, we use the same location (i.e., where a light ray intersects a ply) for both $\boldsymbol{x}$ and $\boldsymbol{y}$. In this case, to model the volumetric transmittance of Eq. (4.11), we assume the ply to have some small thickness h and set

$$\tau = \exp(-\sigma_{\mathrm{t}}\, h/|\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{n}(\boldsymbol{x})|), \tag{4.15}$$

where $\boldsymbol{\omega}_{\mathrm{o}}$ is refracted direction of the light ray at $\boldsymbol{x}$. Our model satisfies energy conservation (i.e., not creating energy), which we verify by rendering plies under uniform white illuminations (see the adjacent figure).

4.3 Implementation Details

To render our ply-based models, we implement a path tracer in KeyShot [56] that supports environmental and local lighting, on-the-fly geometry generation, and next-event estimation (NEE). To accelerate ray-ply-intersection computations, we generate the ply curves and build

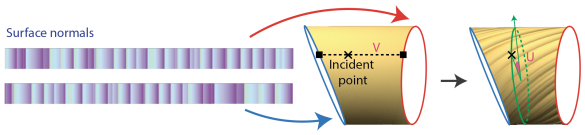


Figure 4.4: **Ply surface parameterization:** We parameterize the surface of a ply so that points with identical V coordinates correspond to individual ply cross sections. By assigning 1D textures specifying fiber normal and tangent information to a discrete set of cross sections, we can then obtain continuous fiber details on the ply surface by linearly interpolating the two 1D textures.

a bounding volume hierarchy (BVH) during preprocessing. At render time, we use a custom shader to both evaluate and importance sample our shading model described in §4.2. In what follows, we provide more details on our rendering process.

4.3.1 Ray-Ply Intersection

We store plies as polylines and store position, curve normal, and curve length information at each vertex. At render time, we compute ray-ply intersections using ply surfaces defined implicitly around their centerlines as follows.

- We treat *round plies* as cylindrical pipes. Specifically, we define a cylinder around each segment of a ply centerline. To ensure that two consecutive segments meet seamlessly, we make the two cylinders to join at an ellipse perpendicular to the average tangent of the two segments. To obtain shading frames continuous across the ply surface, we define these frames on the joints of adjacent segments and interpolate them for the interior points (see Figure 4.4).
- For *flat plies*, each segment is rendered as a bilinear surface whose orientation is determined using the curve normal. We use a standard bilinear patch intersection technique [70] to ray-trace these segments.

4.3.2 Fiber-Level Details

As explained in §4.1.3, we enrich the ply surfaces with fiber details using normal and tangent mapping. To this end, we UV-map the ply surface such that the V-direction is aligned with the ply tangent and all points belonging to the joint of two segments have identical V coordinates. In this way, all interior points along a segment with the same V coordinate give the boundary of a ply cross section. For round plies this is an ellipse and a line segment for flat plies. This ellipse or line determines our U coordinate as the phase around the ellipse or the parameter along the line segment.

Based on this surface parameterization, we use multiple 1D textures to specify the fiber normal and tangent at a few discrete locations (each with fixed V coordinates) along each ply. By interpolating these textures along the ply, as illustrated in Figure 4.4, we obtain normal and tangent directions that are C^0-continuous over the ply surfaces.

4.4 Results

In what follows, we show rendered results generated with our technique. Specifically, we validate the physical accuracy of our model in §4.4.1 and §4.4.2. Then, we demonstrate the effectiveness of our method by comparing to prior models in §4.4.3. Lastly, we show additional renderings generated with our model in §4.4.4.

4.4.1 Validation with Measurements

We compare our model against a detailed scan of a silk sample using an X-Rite TAC7 scanner, which is a sophisticated scanning device with built-in color calibration, multiple cameras, as well as directional and a linear illuminations. The device reconstructs the scanned sample

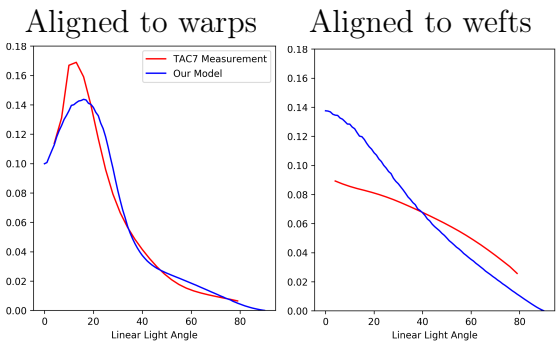


Figure 4.5: **Comparison with X-Rite TAC7 measurements:** Comparison of linear light measurements using an X-Rite TAC7 scanner. The sample was rotated such that the warps faced the linear light on the left plot whereas the wefts faced the linear light on the right. In this example, the pattern is satin.

in 3D and uses this reconstruction to re-project all camera images onto a unified coordinate system for all camera angles. By moving a linear light source in fixed angular increments, the device is capable to accurately recovering reflectance profiles of highly specular materials.

The pixel values $I(x, y)$ of the image output from the TAC7 (after re-projection) is determined by the ratio between radiance exiting each surface point and the incident irradiance at that point:

$$I(x,y) = \frac{L_o(\mathbf{p}, \boldsymbol{\omega}_\mathrm{o})|\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}|}{E(\mathbf{p})} = \frac{L_o(\mathbf{p}, \boldsymbol{\omega}_\mathrm{o})|\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}|}{\int |\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}|\, L_i(\mathbf{p}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\omega(\boldsymbol{\omega}_\mathrm{i})}. \tag{4.16}$$

To validate the physical accuracy of our model, we rebuild this measurement configuration virtually. Specifically, we use orthographic cameras to bypass the re-projection step needed by the TAC7 scanner. In this way, the rendered image directly gives the radiance L_o at the sample's surface. To obtain the incident irradiance $E(\mathbf{p}) = \int |\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}| L_i(\mathbf{p}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\omega(\boldsymbol{\omega}_\mathrm{i})$, we render a unit-white diffuse material, producing an image with the value $\frac{1}{\pi} \int |\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}|\, L_i(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\omega(\boldsymbol{\omega}_\mathrm{i}) = {}^{E(\mathbf{p})}/_{\pi}$.

In Figure 4.5, we plot the mean intensity of the measured and rendered image as a function of illumination direction. In this figure, the red curve are obtained using the scanned data

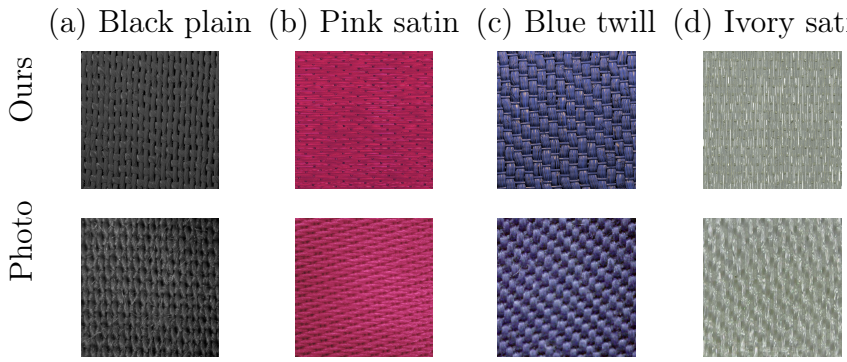


Figure 4.6: **Matching photographed samples at the micro scale:** Our system is capable of representing multi-ply yarns along with the fiber-level details using an interactive user interface. Once the geometry and appearance parameters are set at this level, large scale models can be rendered with the same material as shown in Figure 4.7.

and exhibits off-specular peaks due to the arrangement of the warps. The blue curve shows our model, which faithfully reproduces the same off-specular peak although the measured value is slightly higher and narrower. The weft orientation, on the other hand, is almost diffuse in its appearance, confirming the highly anisotropic nature of this material.

Given the difficulty of matching the unknown yarn-geometry of physical cloth samples and the corresponding light-scattering parameters, we consider the match between the measured data and the our model being quite reasonable. Recall that our yarn geometry is obtained using the relaxation discussed in §4.1.1, which is not expected to match the exact shape of the real-world sample.

4.4.2 Validation with Photographs

To further validate the physical accuracy of our model, we create virtual replica of a few real-world fabrics using our model. We manually model the yarn-level geometry (by examining the samples under a digital microscope) and rely on the procedural yarn model [101] (of the corresponding materials) to add fiber details. Given the cloth geometry, we search for the values of our light-scattering model parameters (§4.2) automatically by solving an inverse-

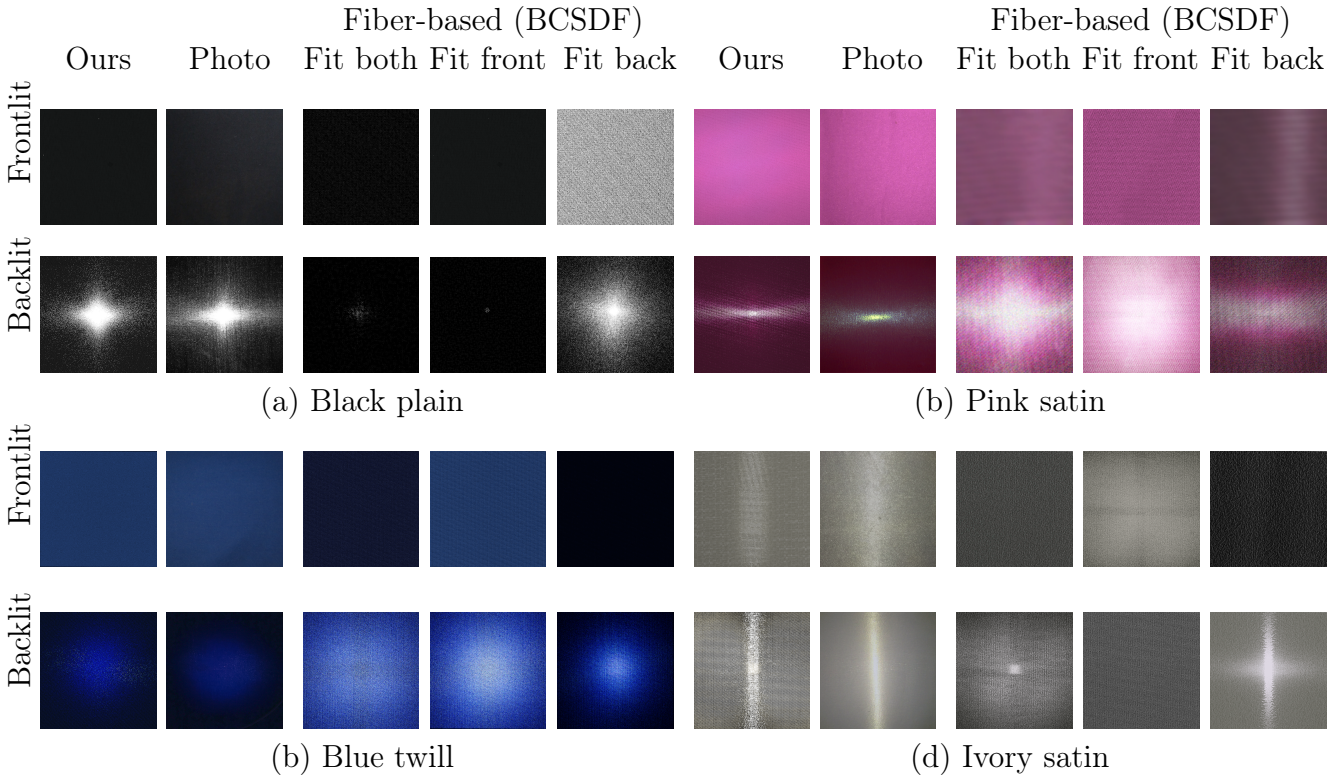


Figure 4.7: **Matching photographs at the macro scale for Ours and Fiber-based models:** Using the same samples and parameters as in figure 4.6 our model is able to simulate the anisotropic highlights seen in the photographs at macro scale. Note, how both the plain weave and the satin has very different appearances when the sample is front lit compared to the back lit sample; our model is able to simulate this difference and match the photographs. In the last three columns we use fiber-based geometry and BCSDF [44]. While BCSDF model can not match the front and back lighting simultaneously, our system found a descent fit for matching only one side.

rendering problem. Specifically, we minimize the L_2 distance between the average intensity of the rendered images and the photographs using the sequential least squares programming (SLSQP) method [49] implemented in SciPy [**?**]. The parameter fitting process can be improved by using gradient-based methods like Adam [48] with differentiable rendering of our model, although this is beyond the scope of the paper.

Figures 4.6 and 4.7 respectively compare renderings generated using our model and photographs at two greatly varying physical scales.

4.4.3 Comparison with Previous Methods

We demonstrate the flexibility of our model by comparing to the fiber-based representation introduced by Khungurn et al. [44], which depicts a fabric as a collection of micron-diameter fibers, in Figure 4.7.

We generate the fiber geometry using the same yarn centerlines. Unlike our method that expresses fiber details via normal and tangent mapping on ply surfaces, this fiber-based model directly stores all the procedurally generated fibers. To obtain the fiber-scattering parameters proposed in the same work, we solve inverse-rendering problems using three different objectives. In the "fit both" scenario, we minimize the difference of both front-lit and backlit renderings. In the "fit front" and "fit back" cases, we consider only the front-lit and the backlit rendering, respectively. Our experiment indicates that the fiber-based model, in its current form, has difficulties matching both reflection and transmission. This could be due to a number of simplifying assumptions being violated in reality (e.g., fibers having spatially invariant circular cross sections). In other words, although the fiber-based model is more detailed than our formulation, its provides insufficient flexibility for correcting numerous approximation errors introduced via the modeling process. Our model, on the other hand, is much more flexible and can easily match both front- and backlit photographs,

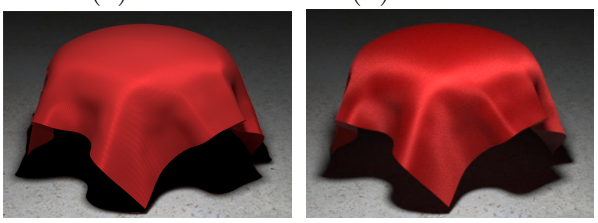


Figure 4.8: **Appearance-matching volumetric models**. Despite the great difference between our model and the volumetric one that relies on anisotropic radiative transfer, our technique is able to produce appearances that are quite similar, thanks to its flexibility.

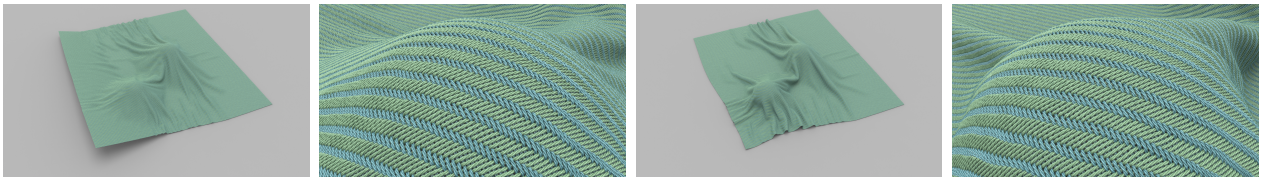


Figure 4.9: **Temporal consistency:** We show two frames of a rendered animation where a cloth drapes onto two spheres and the ground. Our technique manages to produce stable and temporally consistent renderings. Please see the supplementary video for the full animation.

thanks to its aggregate nature.

Lastly, we further demonstrate the flexibility of our model by fitting to the rendering of a volumetric cloth model expressed as a micro-flake-based anisotropic medium [96]. Notice that this model has been demonstrated to be less flexible than the fiber-based model [44]. This is mainly because the model relies on the radiative transfer framework to simulate light scattering in cloth. Without explicit fiber surfaces, phenomena like Fresnel reflection cannot be accurately modeled by this approach, yielding the lack of white highlights on colored fabrics. Additionally, this model has to model glossy appearances using high optical density, drastically decreasing the physical plausibility of the modeling of light transmission.

Despite the significant difference between our model and the volumetric one, we manage to obtain a close match, as demonstrated in Figure 4.8.

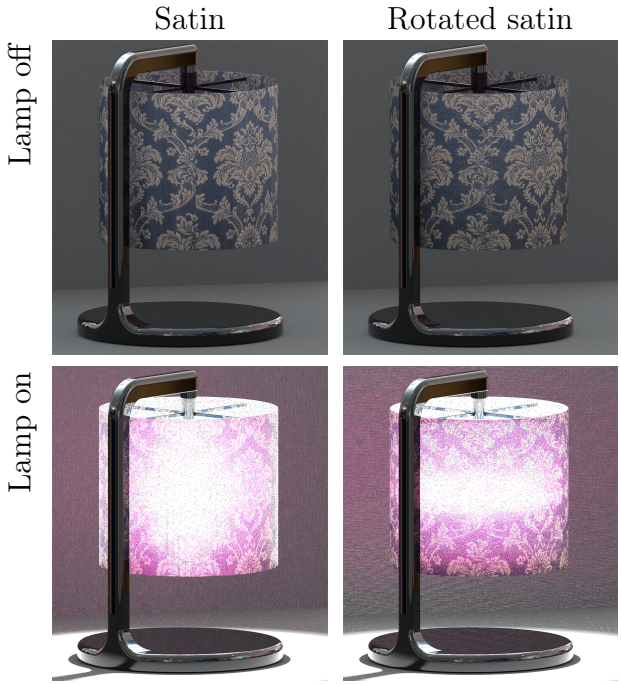


Figure 4.10: **Anisotropic transmission for satin pattern in two orientations**. When the lamp is off, our model shows mainly the reflected lighting, showing the texture that is mapped to the surface color. And when the lamp is on, it shows the anisotropic transmission of the light as well and the change of color. Note the backlit highlight orientation is perpendicular to the flat yarns as also expressed in Figure 4.7-b and d. This is further demonstrated in the supplementary video.

4.4.4 Additional Results

We now show additional rendered results produced using our model.

In Figure 5.7, we show two frames of an animation where a large piece of cloth drapes onto two spheres. Our model is capable of producing temporally consistent results down to individual flyaway fibers. Please refer to the accompanying video for the full animation.

In Figure 4.10 we show renderings of a lamp. When the lamp is off, our model shows mainly the reflected lighting, thus visualizing the texture that is mapped to the surface color. When the lamp is on, it shows the anisotropic transmission of the light as well as the change of color which is derived from the medium. We note that the backlit highlight orientation is perpendicular to the flat yarns as appeared in Figure 4.7-b and d as well. This is further demonstrated in the supplementary video.

Lastly, Figure 4.11 demonstrates that our model can be easily manipulated. Starting with the fitted parameters of the pink satin in Figure 4.7, we edit the model in the following steps: (a) changing yarn color, (b) reducing roughness, and (c) changing weave pattern (from satin to basket) and increasing the amount of flyaway fibers.

4.5 Discussion and Conclusion

Limitations and future work Our technique works only for woven fabrics due to the simple weave pattern with a warp and weft structure. Similar methodology can be employed for knitted fabrics to handle more complex patterns. Generalizing our practical ply-based model to knit textiles should be fairly straight forward with the main challenge being modeling the knit yarn geometry.

We would also like to explore a multi-resolution representation to further compress the

geometry for distant views where the ply and fiber detail is not visible. Ideally, this would be a gradual transition from our model to a coarse surface-based cloth model.

Our model follows previous procedural cloth models (e.g. [101]) and inherit the simplifying assumptions (i.e. fiber cross-sections are assumed to be circular while in reality they vary and may have concave shapes). Improving fiber details could lead to a more realistic detailed appearance.

Likewise, it would be interesting to explore a more accurate model for volume scattering within a bundle of fibers.

Conclusion We have presented an appearance model for woven fabrics that takes a surface mesh and a weave pattern as input and represents the clothing using interwoven ply curves with flyaway fibers. The ply curves are rendered using a BSDF model that accounts for both reflection and transmission. Fiber details are added to the ply BSDF model using precomputed fiber normals and shadows. Our results show that our model is able to match measured data for both reflection and transmission with intuitive physics-based parameters and artistically pleasing tool. Furthermore, our model matches previous surface-, volume-, and fiber-based models for reflection, and is at the same time practical for simulating transmission including the anisotropic highlights often seen through clothing. Our ply based representation is easy to edit, compact, and stable as the cloth surface is deformed making it practical to use for woven fabrics.

Table 4.2: **Performance statistics** for renderings of our models in Figures 4.7–4.10. All rendering have been done using a dual Xeon E5-2699 system.

Fig. 4.7-a	Fig. 4.7-b	Fig. 4.7-c	Fig. 4.7-d	Fig. 4.8	Fig. 5.7
139.8 MB	364.6 MB	100.3 MB	142.3 MB	308.3 MB	906.1 MB
191 sec	60 sec	93 sec	80 sec	52 sec	41 sec
Fig. 4.11-a	Fig. 4.11-b	Fig. 4.11-c	Fig. 4.11-d	Fig. 4.11-e	Fig. 4.10
449.9 MB	449.9 MB	899.9 MB	899.9 MB	99.2 MB	7.9 MB
76 sec	81 sec	81 sec	62 sec	68 sec	84 sec

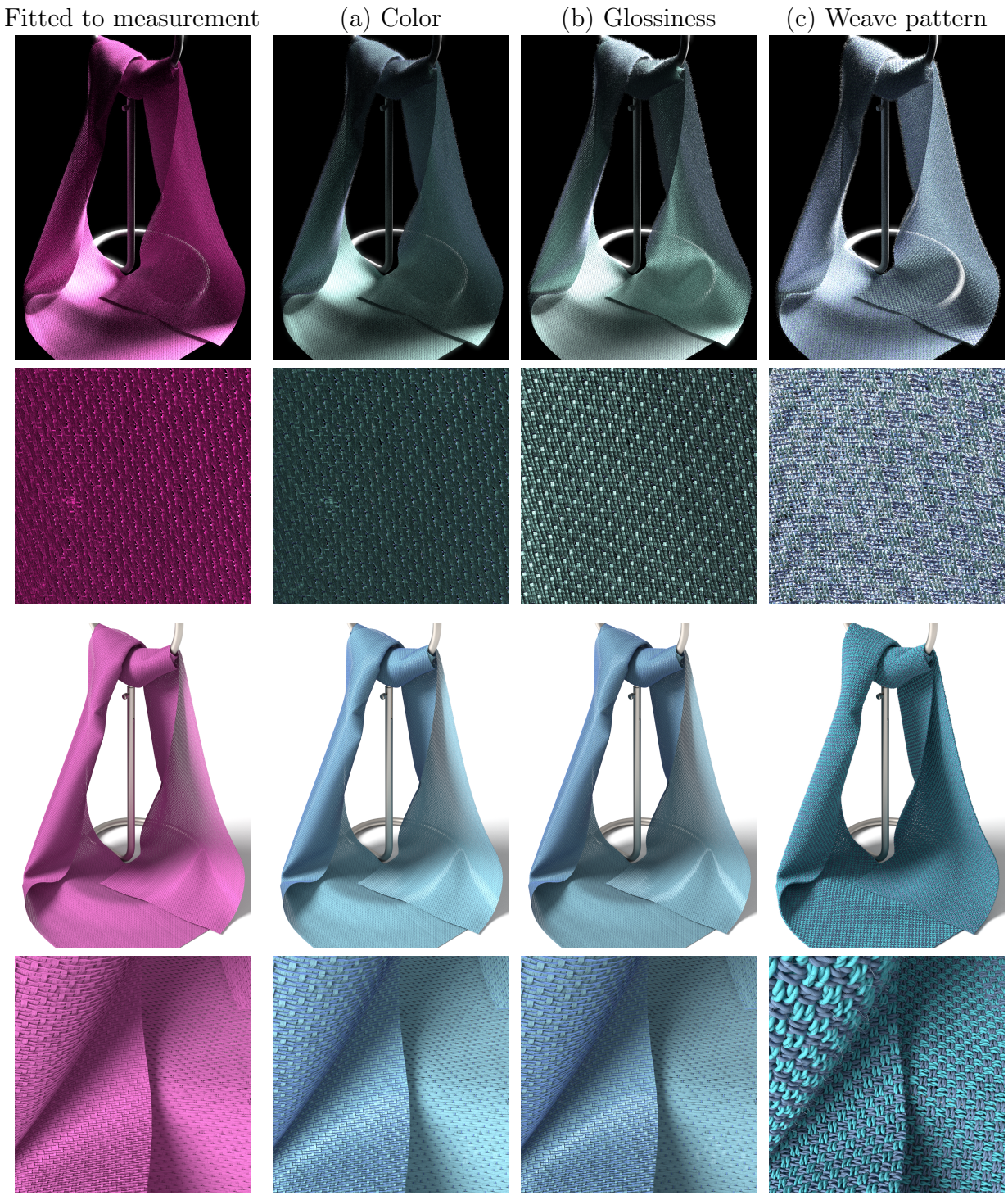


Figure 4.11: **Editing geometry and appearance**. Starting from the Figure 4.7-b where we use fitted parameters to measurement: (a) Change the color (b) edit glossiness value, (c) change the weave pattern to basket with pronounced flyaways. Note, the subtle appearance of flyaways on the basket pattern which greatly increases realism along the edges.

Chapter 5

Ply-Based Appearance Modeling for Knitted Fabrics

In previous chapter, we discussed the practicality and flexibility of our ply-level model, however that has been designed for woven fabrics only. Knitted fabrics—which are arguably more common than the woven ones since knitting has been the most popular method for manufacturing cloth—have been largely neglected by this model and many previous ones. Compared with woven fabrics, knitted ones typically exhibit more three-dimensional patterns that lead to more complex scatterings of light. Efficiently describing large pieces of knitted fabrics with fiber-level details has largely remained a challenge.

In this chapter, we extends Montazeri et al.'s model [61] to handle knitted fabrics. Concretely, our contributions include:

- A new pipeline that efficiently models the geometry of a knitted fabric at the ply level (Sections 5.1.1 and 5.1.2).
- An improved appearance model build upon ply-based model [61] that better match the

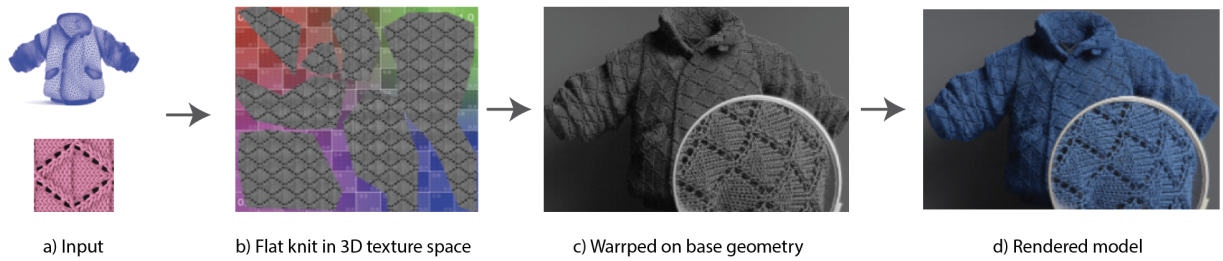


Figure 5.1: **Method overview.** a) The input to our system is a geometry base mesh with per-vertex UV as well as a set of curves for a tilable knitted pattern. b) The knit geometry is generated in a flat plane also referred as 3D texture space. c) Then, warped to the object by extruding out of the base geometry and forming the ply-level cloth model. d) During rendering, fiber details are added using ply-based appearance model.

reference photos (Section 5.1.3).

We qualitatively evaluate the accuracy of our models by comparing their renderings to photographs of physical samples with identical knitting patterns (Figure 5.5).

5.1 Our Method

We now describe our main pipeline for generating and rendering knitted fabrics in ply-level. Our ply-level representation is based on the work by Montazeri et al [61] that offers a good balance between efficiency and fidelity. Also, our approach simulates only the macro-scale base mesh then map the ply-level representation afterward. Our method for generating the knit geometry consists of two main stages: first generating the knit curves in a flat pieces in 3D texture space, then warp them to the 3D object by extruding the yarns out of the base geometry.

Overview. Our method takes as input a mesh with per-vertex UV along with a tilable relaxed yarn curves for a single knitted pattern released by Leaf et al. [51]. We first generate the knit geometry in a flat piece, throughout the paper we refer to the space where the cloth

is represented as *3D texture space* and then we extrude the yarns out from the base mesh to form the yarns in actual 3D object. Ply curves are generated twisting around the yarn curves given the parameters (e.g. number of plies, ply twist). Lastly, fiber geometry is added using normal/tangent and shadow mapping on the surface of ply-curve geometries same as Montazeri et al. [61].

5.1.1 Generating the Flat Knits

To generate the knit geometry, we first create the yarn curves in a flat piece. Given the density of the curves, number of plies and the ply twisting amount we can create the knit geometry in the 3D texture space. This method involves two stages: first tiling the knitted pattern given the number of repeating parameter as an input. Then, stitching the tilable cells together to make a connected piece of the entire knit textile.

Pattern tiling. Given the yarn curves of a knitted pattern, we construct a directed graph, in which each node represents a copy of the given pattern, and we refer to them as *cells*. Each cell includes the curves of the pattern and each curve has two endpoints connected to the two neighbor cells. As a pre-processing step, we store a dictionary of the *partner endpoints* — the couple endpoints of a curve from one of the four neighbor cells that they are connected to. Finding the partner of an endpoint can be done by searching for the closest endpoint among all four neighbor cells (i.e. top, bottom, right, row). Therefore, all endpoints of the cell are labeled which adjacent cell and what curve of that cells it is connects to. Figure 5.2 illustrates the labeling for each endpoint of the center cell. Since later we repeat the cell along the texture map to form the 3D geometry in a flat piece, the labels can be reused for all cells repeatedly.

Pattern stitching. Unlike the woven fabrics, knits are usually built using a small number of yarns. Therefore, after connecting yarn curves in adjacent cells they form long yarn curves. We need the full yarns because the yarn length is required to twist the plies around the yarn center. Also, the length of the ply curves are needed for adding the fiber details. Fiber details are computed using normal/tangent as well as shadow mapping which requires the ply surface to be fully parameterized, same method as Montazeri et al. [61]. We define a graph where each node represents each curve of every pattern cell. This establishes a degree-2 graph and we traverse the graph to generate the full curves. Figure 5.2 also illustrates the full curve generation. Note that the order of vertices along the curves might be reversed due to the connecting edges and this is depicted using an arrow inside the curve node. So using the correct connections between the tiled cells are computed to seamlessly generate the knit topology.

5.1.2 Base Geometry Transformation

Once the flat knitted fabric is generated in 3D texture space as explained in Section 5.1.1, to transform the model into the object space, we develop an efficient mapping technique to transform the flat piece from 3D texture space to the actual geometry in object space 5.1.2. The mapping requires a pre-computation phase with a simple implementation. The pre-computed mapping allows efficient rendering, and will be discussed in Section 5.1.3.

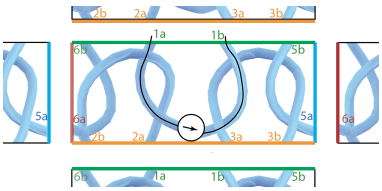


Figure 5.2: **Pattern tiling and stitching.** Coupling the endpoints of all the curves in the pattern cell among neighbor cells to compute how it is connected to the adjacent cells and generate the fully connected yarns. Each curve represents a node in the graph and the arrow shows the direction of the yarn which may need to be reversed based on the curves which it is connected to.

Pre-computation. We define a 2D grid to store the curve segments in 3D texture space along with their correspondent triangles from the base geometry. The 2D grid is created by subdividing the UV space into cells. Each UV cell holds two corresponding lists: first, the indices of the curve segments that lie within that particular cell, second, given the UV values of each vertex of the triangle mesh, a list of triangle indices that cover that particular cell in the base geometry. Hence, each grid cells c corresponds a list of triangles in object space (T) to a list of curve segments in 3D texture space (Γ), with a tuple $c = (s, \tau)$ where $s(c) \in \Gamma$ and $\tau(c) \in T$. The resolution of the grid is pre-defined proportionally to the mesh density. Figure 5.3 shows the dictionary to correspond the curve segments from the flat piece to the triangles of the base geometry.

Transformation. To transform the flat knit as explained in Section 5.1.1 from 3D texture space to the actual object space of the base geometry, we require the following two inputs. First the UV values of all the vertices of the segment $s(c) \in \Gamma$ for the flat textile (U)as well as the pre-computed 2D grid which holds the correspondent triangle set T and the segment set Γ for each cell c. For each $\vec{u} \in U$, given $\vec{u}$ we find the cell index c as well as the set of triangles T which belongs to the cell. Then using bilinear interpolation we find the exact triangle that corresponds to the segment which is $\tau(c) \in T$. Finally, Using trilinear interpolation we compute the 3D coordinate of the point on the based geometry $\vec{P} = (x, y, z)$. All vertices of

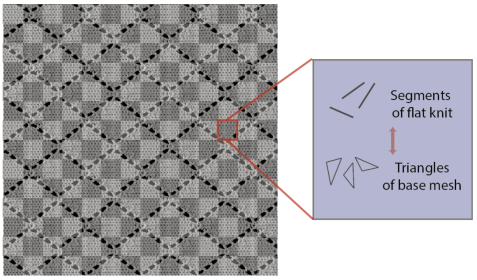


Figure 5.3: **Mapping grid.** The pre-computed mapping method for transferring the flat piece to the base geometry by extruding the yarns out of the base mesh. The same grid is also used to accelerate the rendering process and will be discussed in Section 5.1.3

.

the ply curve along with their normals as well as the curve length are stored which require 28 Bytes per vertex overhead in total. Algorithm 1 elaborates the steps further.

Algorithm 1 Transferring the flat textile to the base geometry

1: **Require:** The pre-computed 2D grid and U
2: **for all** Every $\vec{u} \in U$ **do**
3: Find $c(\vec{u})$
4: Find T given c
5: Find $\tau(c) \in T$ {using bilinear interpolation}
6: Compute $\vec{P}$ {using trilinear interpolation}
7: **end for**

5.1.3 Appearance Modeling

To faithfully re-produce the cloth appearance in ply-level, we use an aggregated BSDF model which first proposed by Montazeri et al. [**?**] to model the light reflection as well as the light transmission. It is a known fact that thick yarns which happen to be more common in knit manufacturing (compared to woven fabrics), are consist of several plies. Hence, using a ply-level appearance modeling plays an important role in the knit geometry.

Parameter fitting. To fit the parameters of the light-scattering model automatically, we solve an inverse-rendering problem same as Montazeri et al. [**?**] by minimizing the loss between the rendered results and the photo references. While they used the average intensity of the rendered image compared to the reference, we chose to leverage a learning-based method loss function to improve the accuracy of the fitting process. In addition to L2 distance between two images as loss function, a popular method is using the pre-trained neural network for imaging as VGG16 [**?**] to compute the distance between the rendering and the reference in 10% resolution of the original image.

Rendering

We use a custom pathtracer for the rendering of our cloth models. To efficiently render the textile, we perform the raytracing in two stages, *global ray-tracing* and *local ray-tracing*, which are explained in the following two paragraphs, respectively. We again require the pre-computed 2D grid which holds the correspondent triangle set T and the segment set Γ for each cell c same as Section 5.1.2.

Ray-surface intersection. First, we execute a simple ray intersection with the base mesh using conventional acceleration hierarchy similar to the existing ray-tracing renderers, which returns the intersected triangle τ along with the UV value of the hit point in the object space $\vec{v}$. We refer to the *ray-surface intersection* as the global stage.

Ray-ply intersection. By looking-up in the pre-computed 2D mapping grid, the cell where $\vec{v}$ belongs to is found. Given that there are already pre-stored set of segments Γ belong to cell c, we compute the intersection of the ray only for those segments belonged to c locally. Each segment forms a cylinder with given ply radius and the ray intersection is computed only for few segments which makes the rendering process much faster rather than finding the ray intersection with the entire segmented curves. This *ray-cylinder intersection* is the local step. Once we find the hit point on the surface of a cylinder we check if it is in the trimmed section where two adjacent segments meet. In that case we compute the correct intersected point with the adjacent segment and further smooth out the normal of the hit point as explained in detail in Section 5.2.1. The angular phase β is needed to add the fiber details and finally, the normals and tangents are smoothed out by interpolating along the segment. This step returns $\vec{Q}$ which is the 3D position, the normal and the length of the intersected point on the surface of the ply.

Algorithm 2 Rendering the cloth model

Require: The pre-computed 2D grid and Base geometry mesh // Global stage: Ray-surface intersection:
Find τ and $\vec{v}$ by raytracing the base mesh
// Local stage: Ray-ply intersection:
Find Γ belongs to $c(\vec{v})$
for all Every $s \in \Gamma$ **do**
    Check intersection with the cylinder formed along the segment s
    Check if it hits at the joints
    Compute β needed for fiber details
    Compute $\vec{Q}$
    Smooth $\vec{Q}$ along the s
end for

5.2 Implementation Details

We implemented a custom raytracer that supports next-event estimation for both area and environment lighting. To accelerate ray-ply-intersection we perform two stages for rendering: global and local raytracing described in Section 5.1.3. In what follows, we provide more details on modeling process.

5.2.1 Modeling Geometry

Given the pre-stored vertices of the piece-wise linear ply curves we compute the ray-ply intersection such that each segment of the ply curve defines a cylinder where the overall shape of the plies are round pipes. At the joints where two cylinders meet, we model the intersected ellipse perpendicular to the average tangent of the adjacent segments. To add the fiber details, 1D textures warps around the endpoint ellipses of the segment as explained in Section 3.3 of [**?**]. The interpolation of the textures along the surface of the cylinder is non-trivial and was not mentioned in the previous work. In addition to the interpolation between two ellipses, to compute the angular phase of each point on the surface, the arc length of the ellipse is required. To address these challenges, we clarify our method for

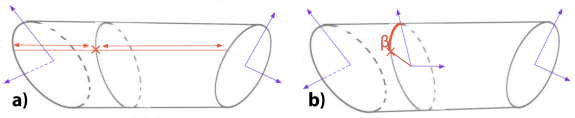


Figure 5.4: (a) Illustrates the interpolation along one segment. (b) In order to map the fiber details we need to compute the angular phase around the interpolated ellipse at the intersected position, the arc length of the ellipse (β) is computed using the angle between the shading and geometry normal.

smoothing the fiber texture in the following.

Smoothing fiber details To generate continuous shading frames across the ply surface, we linearly interpolate frames of the both ends of the cylinder. However since the end caps are elliptical, computing the normals along the circumference of the ellipse is challenging, hence we compute the intersection of the line parallel to the yarn center on the surface of the cylinder with the two elliptical caps to obtain the normals of the both ends along the circumference of the ellipses. Then, tangents of the frames are updated using rotation minimizing frames [**?**]. Figure 5.4(a) illustrates the interpolation along one segment between two elliptical endpoints. Later, in order to add the fiber details we compute the angular phase β around the interpolated ellipse at the intersected position, the arc length of the ellipse is computed by computing the angle between the *shading* and *geometry* normal as shown in 5.4(b). Where shading normal is the perturbation of the geometry normal based on the 1D fiber texture.

5.3 Results

In this section, we first evaluate the physical accuracy of our technique by comparing renderings to photographs of real samples (Section 5.3.1). Then, we show more rendered results generated with our method for common knitting patterns on complex base geometries (Section 5.3.2).

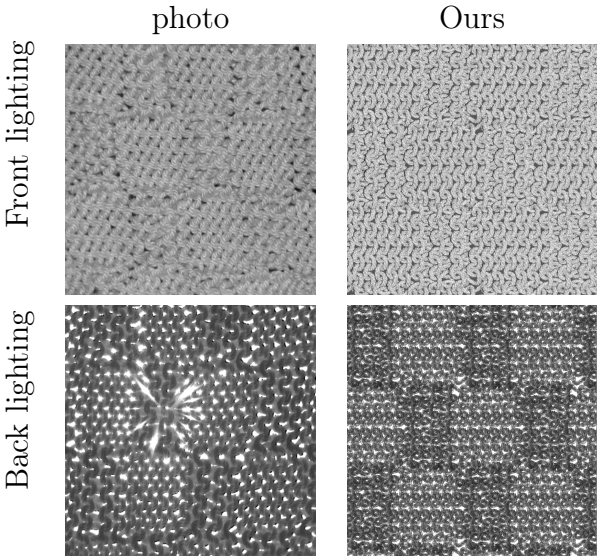


Figure 5.5: **Matching photographs of real sample knits with our model.** We use VGG loss to optimize the appearance parameters of the rendered image to match with the reference photos. Our system is capable of matching the appearance of the knits under both front and back lighting.

5.3.1 Comparison with photographs

We create virtual replica of real samples by examining each sample under a digital microscope and manually setting the geometry parameters. Then, we set the material appearance parameters automatically by solving an inverse-rendering problem. Specifically, we minimize the difference between rendered images and photographs measured using the VGG16 loss —a learning-based metric [**?**] with 10% resolution of the original image— in addition to a regular L2 distance.

Fig. 5.5 shows photographs of a real sample (on the left) and rendered results using our system with optimized parameters (on the right). We illuminated the cloth sample from both the front and the back using a small area light. Both front and back lighting references are needed for the system to optimize the appearance parameters. Since having these two setups captures both reflection and transmission of light, and all of the parameters were fitted given the two references simultaneously.

5.3.2 Main Results

We now show additional rendered results using our technique for complex geometries.

Our model enjoys the flexibility to represent knitted fabrics with widely yarn structures and optical properties. Fig. 5.6 shows such an example with several different knitting patterns, yarn densities, colors, and glossiness. Additionally, we show in the accompanying video an example where these properties are edited interactively.

Fig. 5.7 illustrates four frames extracted from an animation in which a rectangular piece of knitted fabric is draped onto a solid sphere. The model is rendered under both front (top) and back lighting (bottom). Our modeling technique allows detailed and physically plausible animations to be obtained with only the base mesh being simulated (that is, no yarn-level simulation). Please see the supplementary video for the full video.

Lastly, in Fig. 5.8 we show knitted fabrics with a wide array of patterns simulated by Leaf et al. [51].

5.4 Discussion and Conclusion

Limitations and future work. Our technique only works for limited tilable patterns that are pre-relaxed (using yarn-level simulation). Extending our system to support arbitrary knitting patterns is an interesting topic for future research.

Additionally, developing multi-resolution or level-of-detail versions of our geometric and appearance models could enable interactive high-fidelity rendering of knitted fabrics for VR/AR applications.

Conclusion. We have presented a new technique that efficiently models knitted fabrics with tilable patterns and arbitrary base geometries. Our model depicts a knitted fabric at the ply level—similar to the work by Montazeri et al. [**?**]—and add fiber-level details using tangent and normal mapping. Taking as input a base mesh and yarn curves of a tilable knitting pattern, we introduced a new pipeline that build our ply-based model automatically. Additionally, we developed a system to efficiently render our models.

We demonstrate qualitatively that our models to reproduce the appearance of real-world knitted fabrics under both front- and back-lit configurations. Further, our model allows knitted fabrics with fiber-level details to be animated in a physically plausible fashion by only simulating its base geometry.



Figure 5.6: **Interactive editing geometry and appearance.** a) Changing the knitting pattern b) yarn density c) color d) and glossiness.

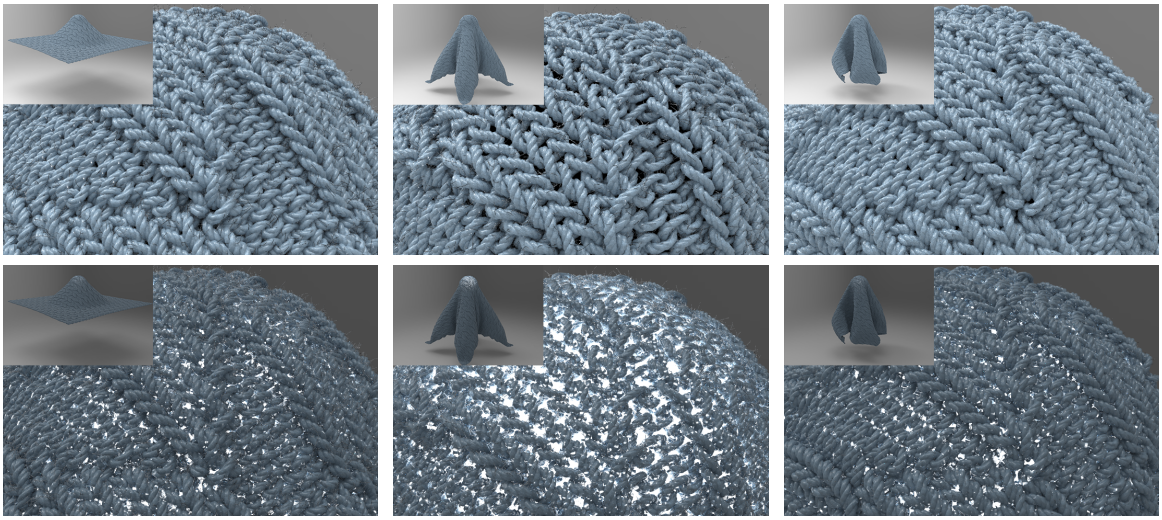


Figure 5.7: **Temporal consistency.** Our model is capable to generating detailed animations of knitted fabrics with only the base mesh simulated. In this example, we show four frames of an animation where a heavy piece of cloth is draped onto a spherical light (that is off in the top and on in the bottom). Our technique manages to produced visually plausible deformation of yarns (e.g., the stretching of yarns when they collide with the sphere).

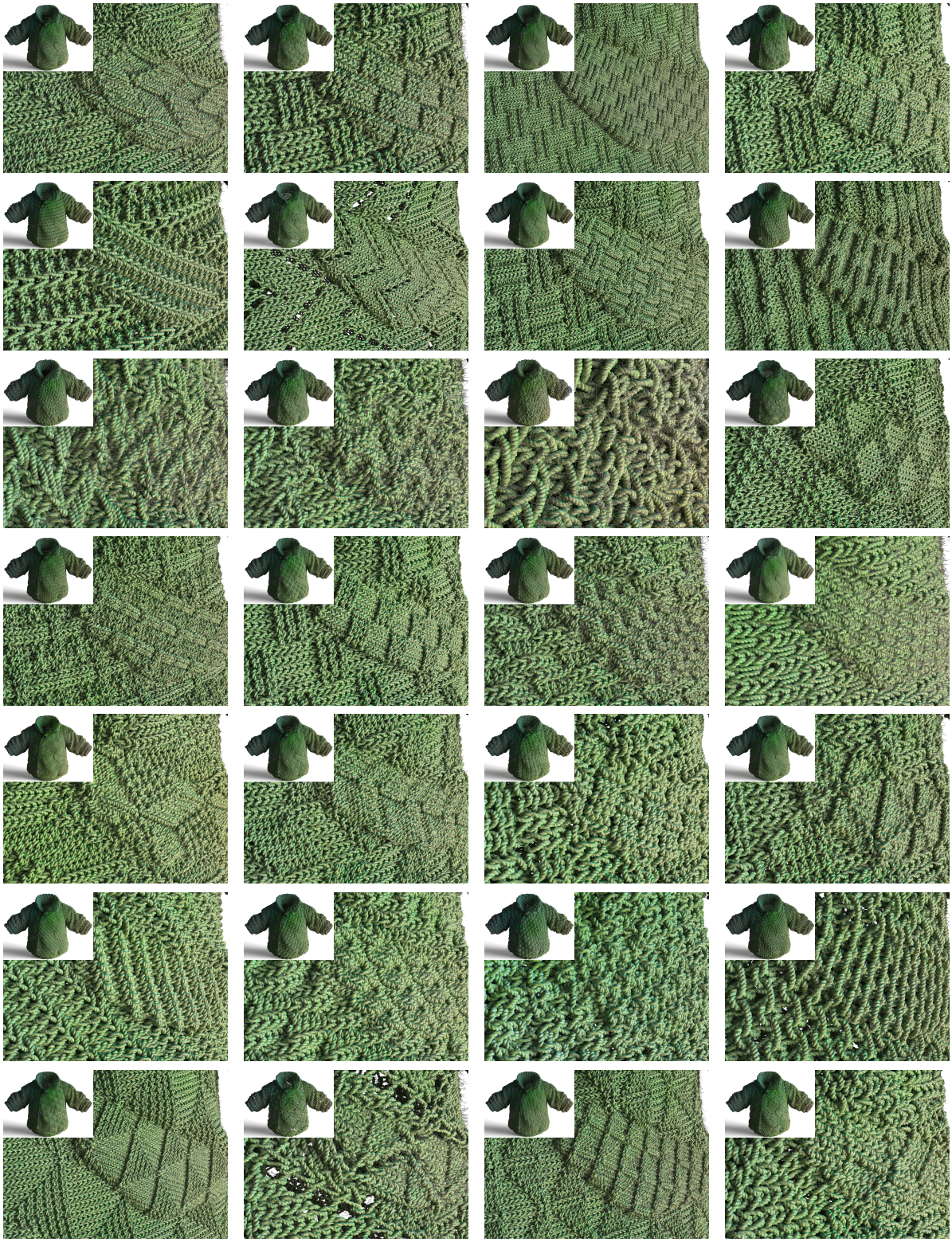


Figure 5.8: Knitted fabrics with **common knitting patterns** simulated by Leaf et al. [51].

Chapter 6

Conclusion

Fabrics are one of the most common materials in daily life so modeling and visualizing them digitally is essential and benefits not only computer graphics but also textile and retail industry. The main two types of textiles are woven and knitted fabrics and each has a distinct manufacturing which yields to very different appearances as well. Woven fabrics are often constructed from two sets of perpendicular yarns called warp and wefts and they are more resilient to stretching. While knitted fabrics are more elastic and their structure allow the entire surface to be formed from a single piece of yarn using interlocked stitches.

We have introduced a first-of-a-kind mechanics-aware technique that captures the fabric's appearance when deformed by external forces without simulating individual fibers explicitly. We leverage a multi-resolution simulation scheme which synthesized cloth in fiber-level and rearrange them based on simulation data only in yarn-level. A custom-design neural network is used to map the simulated forces to yarn cross-sectional shape. This technique allows physically plausible cloth appearance for macro scales which would not be practical using previous methods.

To model and render woven fabrics practically, we have presented a novel technique that

represents the clothing in ply-level and add fiber-details during render time. We use a custom BSDF model to accurately render ply curves capturing both reflection and transmission of light. Our results match measured data and offer an interactive user interface to further edit both the geometry parameters as well as the appearance of the fabric.

We generalized the aforementioned practical appearance model to support knitted fabrics as well as woven clothing. Knits express a very different appearance due to their complex structure under various lighting setups. Our model captures both reflection and transmission of light and qualitatively match with photographs of real pieces of knits with different patterns. To our knowledge, our ply-based appearance modeling for woven and knit fabrics is the most practical end-to-end system with interactive rendering. To demonstrate the practical usefulness of our system, we have integrated it into an industry renderer called KeyShot and provides user-friendly interactive editing interface.

With the advancements presented in this thesis, it is possible to embed the mechanics-aware cloth modeling into our practical ply-based system to obtain a general-purpose practical cloth model for woven and knit fabrics under external forces. Additionally, our interactive system can be further improved to generate the model and update the appearance in real-time.

Bibliography

[1] N. Adabala, N. Magnenat-Thalmann, and G. Fei. Real-time rendering of woven clothes. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '03, page 41–47, New York, NY, USA, 2003. Association for Computing Machinery.

[2] 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.

[3] C. Aliaga, C. Castillo, D. Gutierrez, M. A. Otaduy, J. Lopez-Moreno, and A. Jarabo. An appearance model for textile fibers. In *Computer Graphics Forum*, volume 36, pages 35–45, 2017.

[4] P. W. Anderson. More is different. *Science (New York, N.Y.)*, 177(4047):393–396, 1972.

[5] 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.

[6] 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.

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

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

[9] 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.

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

[11] J. Chu, N. B. Zafar, and X. Yang. A schur complement preconditioner for scalable parallel fluid simulation. *ACM Trans. Graph.*, 36(5), July 2017.

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

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

[14] G. Cirio, J. Lopez-Moreno, and M. A. Otaduy. Yarn-level cloth simulation with sliding persistent contacts. *IEEE Transactions on Visualization and Computer Graphics*, 23(2):1152–1162, 2017.

[15] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.

[16] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, Jan. 1999.

[17] E. d'Eon, G. Francois, M. Hill, J. Letteri, and J.-M. Aubry. An energy-conserving hair reflectance model. In *Proceedings of the Twenty-Second Eurographics Conference on Rendering*, EGSR '11, page 1181–1187, Goslar, DEU, 2011. Eurographics Association.

[18] 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.

[19] 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.

[20] R. Fletcher. *Practical Methods of Optimization; (2nd Ed.)*. Wiley-Interscience, USA, 1987.

[21] Google. Tensorflow. `https://www.tensorflow.org`, 2018.

[22] Google. Physically based rendering in filament. `https://google.github.io/filament/Filament.md.html`, 2019.

[23] 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.

[24] 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.

[25] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 165–174, New York, NY, USA, 1993. Association for Computing Machinery.

[26] E. Heitz. Sampling the ggx distribution of visible normals. *Journal of Computer Graphics Techniques (JCGT)*, 7(4):1–13, November 2018.

[27] E. Heitz, J. Dupuy, C. Crassin, and C. Dachsbacher. The sggx microflake distribution. *ACM Trans. Graph.*, 34(4), July 2015.

[28] C. Horvath and W. Geiger. Directable, high-resolution simulation of fire on the gpu. *ACM Trans. Graph.*, 28(3), July 2009.

[29] P. Irawan and S. Marschner. Specular reflection from woven cloth. *ACM Trans. Graph.*, 31(1), Feb. 2012.

[30] P. Irawan and S. Marschner. Specular reflection from woven cloth. *ACM Trans. Graph.*, 31(1):11:1–11:20, 2012.

[31] K. Iwasaki, Y. Dobashi, and T. Nishita. Interactive bi-scale editing of highly glossy materials. *TOG*, 31(6):144:1–144:7, Nov. 2012.

[32] W. Jakob, A. Arbree, J. T. Moon, K. Bala, and S. Marschner. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph.*, 29(4), July 2010.

[33] 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.

[34] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, page 576–581, New York, NY, USA, 2002. Association for Computing Machinery.

[35] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 511–518, New York, NY, USA, 2001. Association for Computing Machinery.

[36] C. Jiang, T. Gast, and J. Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans. Graph.*, 36(4), July 2017.

[37] 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.

[38] C. Jiang, T. Gast, and J. Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact supplementary technical document. 2017.

[39] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, page 271–280, New York, NY, USA, 1989. Association for Computing Machinery.

[40] J. M. Kaldor, D. L. James, and S. Marschner. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, 27(3):1–9, Aug. 2008.

[41] 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.

[42] 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.

[43] 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.

[44] P. Khungurn, D. Schroeder, S. Zhao, K. Bala, and S. Marschner. Matching real fabrics with micro-appearance models. *ACM Trans. Graph.*, 35(1), Dec. 2016.

[45] 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.

[46] 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), Nov. 2017.

[47] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[48] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[49] D. Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

[50] A. Kuznetsov, M. Hašan, Z. Xu, L.-Q. Yan, B. Walter, N. K. Kalantari, S. Marschner, and R. Ramamoorthi. Learning generative models for rendering specular microgeometry. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[51] J. Leaf, R. Wu, E. Schweickart, D. L. James, and S. Marschner. Interactive design of periodic yarn-level cloth patterns. *ACM Transactions on Graphics*, 37(6):1–15, jan 2019.

[52] 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.

[53] G. Loubet and F. Neyret. A new microflake model with microscopic self-shadowing for accurate volume downsampling. In *Computer Graphics Forum*, volume 37, pages 111–121, 2018.

[54] F. Luan, S. Zhao, and K. Bala. Fiber-level on-the-fly procedural textiles. *Comput. Graph. Forum*, 36(4):123–135, 2017.

[55] F. Luan, S. Zhao, and K. Bala. Fiber-level on-the-fly procedural textiles. *Comput. Graph. Forum*, 36(4):123–135, July 2017.

[56] Luxion. Keyshot. `https://www.keyshot.com`, 2020.

[57] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Trans. Graph.*, 22(3):780–791, July 2003.

[58] N. Max. Cone-spheres. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, page 59–62, New York, NY, USA, 1990. Association for Computing Machinery.

[59] S. McAuley, S. Hill, N. Hoffman, Y. Gotanda, B. Smits, B. Burley, and A. Martinez. Practical physically-based shading in film and game production. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, New York, NY, USA, 2012. Association for Computing Machinery.

[60] J. Meng, M. Papas, R. Habel, C. Dachsbacher, S. Marschner, M. Gross, and W. Jarosz. Multi-scale modeling and rendering of granular materials. *SIGGRAPH*, 34(4):49:1–49:13, July 2015.

[61] Z. Montazeri, S. B. Gammelmark, S. Zhao, and H. W. Jensen. A practical ply-based appearance model of woven fabrics. *ACM Trans. Graph.*, 39(6), Nov. 2020.

[62] Z. Montazeri, C. Xiao, Y. Fei, C. Zheng, and S. Zhao. Mechanics-aware modeling of cloth appearance. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2019.

[63] Z. Montazeri, C. Xiao, Y. Fei, C. Zheng, and S. Zhao. Mechanics-aware modeling of cloth appearance. *IEEE Transactions on Visualization and Computer Graphics*, 27(1):137–150, 2021.

[64] J. T. Moon and S. R. Marschner. Simulating multiple scattering in hair using a photon mapping approach. *SIGGRAPH*, 25(3):1067–1074, July 2006.

[65] J. T. Moon, B. Walter, and S. Marschner. Efficient multiple scattering in hair using spherical harmonics. *SIGGRAPH*, 27(3):31:1–31:7, Aug. 2008.

[66] J. T. Moon, B. Walter, and S. R. Marschner. Rendering discrete random media using precomputed scattering solutions. In *EGSR*, pages 231–242, June 2007.

[67] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, July 2005.

[68] T. Müller, M. Papas, M. Gross, W. Jarosz, and J. Novák. Efficient rendering of heterogeneous polydisperse granular media. *SIGGRAPH*, 35(6):168:1–168:14, Nov. 2016.

[69] K. Perlin and E. M. Hoffert. Hypertexture. SIGGRAPH '89, page 253–262, New York, NY, USA, 1989. Association for Computing Machinery.

[70] S. D. Ramsey, K. Potter, and C. Hansen. Ray bilinear patch intersections. *Journal of Graphics Tools*, 9(3):41–47, 2004.

[71] B. Raymond, G. Guennebaud, and P. Barla. Multi-scale rendering of scratched materials using a structured sv-brdf model. *ACM Trans. Graph.*, 35(4), July 2016.

[72] I. Sadeghi, O. Bisker, J. De Deken, and H. W. Jensen. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.*, 32(2), Apr. 2013.

[73] 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.

[74] I. Sadeghi, H. Pritchett, H. W. Jensen, and R. Tamstorf. An artist friendly hair shading system. *ACM Transactions on Graphics, SIGGRAPH 2010*, 29(4):56:1–56:10, July 2010.

[75] K. Schröder, R. Klein, and A. Zinke. A volumetric approach to predictive rendering of fabrics. In *Proceedings of the Twenty-Second Eurographics Conference on Rendering*, EGSR '11, page 1277–1286, Goslar, DEU, 2011. Eurographics Association.

[76] K. Schroder, A. Zinke, and R. Klein. Image-based reverse engineering and visual prototyping of woven cloth. *IEEE Transactions on Visualization and Computer Graphics*, 21:188–200, 02 2015.

[77] 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.

[78] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, 2002.

[79] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. Augmented mpm for phase-change and varied materials. *ACM Trans. Graph.*, 33(4), July 2014.

[80] S. Sueda, G. L. Jones, D. I. W. Levin, and D. K. Pai. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph.*, 30(4), July 2011.

[81] 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.

[82] 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.

[83] R. H. Tütüncü, K.-C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical programming*, 95(2):189–217, 2003.

[84] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195–206, Goslar, DEU, 2007. Eurographics Association.

[85] H. Wang. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Trans. Graph.*, 37(4), July 2018.

[86] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph.*, 26(2):255–264, 1992.

[87] H. Wu, J. Dorsey, and H. Rushmeier. Physically-based interactive bi-scale material design. *ACM Trans. Graph.*, 30(6):145:1–145:10, 2011.

[88] H. Wu, J. Dorsey, and H. Rushmeier. Physically-based interactive bi-scale material design. *SIGGRAPH Asia*, 30(6):1, Dec. 2011.

[89] H. Wu, J. Dorsey, and H. Rushmeier. Inverse bi-scale material design. *SIGGRAPH Asia*, 32(6):163:1–163:10, Nov. 2013.

[90] K. Wu and C. Yuksel. Real-time cloth rendering with fiber-level detail. *IEEE Transactions on Visualization and Computer Graphics*, PP(99), 2017.

[91] Y.-Q. Xu, Y. Chen, S. Lin, H. Zhong, E. Wu, B. Guo, and H.-Y. Shum. Photorealistic rendering of knitwear using the lumislice. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 391–398, New York, NY, USA, 2001. Association for Computing Machinery.

[92] L.-Q. Yan, H. W. Jensen, and R. Ramamoorthi. An efficient and practical near and far field fur reflectance model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, 36(4), 2017.

[93] L.-Q. Yan, C.-W. Tseng, H. W. Jensen, and R. Ramamoorthi. Physically-accurate fur reflectance: Modeling, measurement and rendering. *ACM Trans. Graph.*, 34(6), Oct. 2015.

[94] Yanyun Chen, S. Lin, Hua Zhong, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Realistic rendering and animation of knitwear. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):43–55, 2003.

[95] S. Zhao, M. Hašan, R. Ramamoorthi, and K. Bala. Modular flux transfer: Efficient rendering of high-resolution volumes with repeated structures. *ACM Trans. Graph.*, 32(4), July 2013.

[96] S. Zhao, W. Jakob, S. Marschner, and K. Bala. Building volumetric appearance models of fabric using micro ct imaging. *ACM Trans. Graph.*, 30(4), July 2011.

[97] 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.

[98] 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.

[99] S. Zhao, W. Jakob, S. Marschner, and K. Bala. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph.*, 31(4), July 2012.

[100] 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.

[101] S. Zhao, F. Luan, and K. Bala. Fitting procedural yarn models for realistic cloth rendering. *ACM Trans. Graph.*, 35(4), July 2016.

[102] S. Zhao, L. Wu, F. Durand, and R. Ramamoorthi. Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph.*, 35(6), 2016.

[103] 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.

[104] A. Zinke and A. Weber. Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):342–356, March 2007.

[105] A. Zinke, C. Yuksel, A. Weber, and J. Keyser. Dual scattering approximation for fast multiple scattering in hair. *SIGGRAPH*, 27(3):32:1–32:10, Aug. 2008.

Appendix A

Yarn Simulation

A.1 Details of Parameter Homogenization

Here we present the details of our parameter fitting process discussed in §3.3.2. First, we define the energy function for external force, $\psi(\boldsymbol{R}) = g(\boldsymbol{R}_2) + h(\boldsymbol{R}_3)$. Again, we note that here, unlike [37], we neglect the elastic stretching energy, because it is already considered in the elastic rod model together with the bending and twisting energies, which the work of [37] ignores.

Now, consider a yarn at static equilibrium in fiber-level simulation. Its centerline indicates the expected static equilibrium state of the same yarn under the same external load in yarn-level simulation (when the material parameters are properly set). In this case, the yarn's external forces are balanced against the internal forces, which, as described in §3.3.2, can be estimated using the elastic rod model applying to the centerline. We denote the external force estimated in this way as $\boldsymbol{f}_v(\boldsymbol{x})$, which is defined on each yarn vertex v located at $\boldsymbol{x}$ (recall Figure 3.8).

In the Lagrangian/Eulerian approach [37, 19], the simulation also exploits a background Eulerian grid. Forces on yarn vertices are transferred to the grid using kernel interpolation,

$$\tilde{\boldsymbol{f}}_i^* = \sum_v w_{iv}(\boldsymbol{x})\boldsymbol{f}_v(\boldsymbol{x}), \tag{A.1}$$

where $\tilde{\boldsymbol{f}}_i$ is the estimated external force at a grid node i, and $w_{iv}(\boldsymbol{x})$ is the kernel function for interpolating the force from yarn vertex v to grid node i. On the other hand, the external force on the grid can be expressed by taking the derivative of $\psi(\boldsymbol{R})$ with respect to vertex positions of the yarn and then transferring the force to the grid. The expression, derived in [37], is

$$\tilde{\boldsymbol{f}}_i = -\sum_v \sum_{\beta=2}^{3} V_v^0 \frac{\partial \psi}{\partial \boldsymbol{F}_\beta} \boldsymbol{d}_\beta^T \nabla w_{iv}(\boldsymbol{x}), \tag{A.2}$$

where the scalar V_v^0 is the effective volume of vertex v, $\boldsymbol{d}_\beta$ is the βth column of the matrix $\boldsymbol{d} = \boldsymbol{F}\boldsymbol{D}$. Here recall that $\boldsymbol{F}$ is the deformation gradient and $\boldsymbol{D}$ (introduced in §3.3.2) is the 3×3 matrix indicating local material directions. Both are defined on the finite elements of the polylines. Equating (A.2) with (A.1) results in a system of linear equations with respect to $\frac{\partial \psi}{\partial \boldsymbol{F}_\beta}$. Here, the number of equations is a triple of the grid size, and the number of unknowns is six times of the total number of yarn vertices (because $\frac{\partial \psi}{\partial \boldsymbol{F}_2}$ and $\frac{\partial \psi}{\partial \boldsymbol{F}_3}$ have six elements). In practice, this is an over-constrained linear system, and $\frac{\partial \psi}{\partial \boldsymbol{F}_\beta}$ can be obtained uniquely through a least-squares solve.

Next, we consider each edge $\boldsymbol{e}$ on yarn polylines. From the relationship $\boldsymbol{d} = \boldsymbol{F}\boldsymbol{D}$, we notice that $\frac{\partial \psi}{\partial \boldsymbol{d}} = \frac{\partial \psi}{\partial \boldsymbol{F}} \boldsymbol{D}^{-T}$ and

$$\frac{\partial \psi}{\partial \boldsymbol{d}} = \boldsymbol{Q}\left(\mathsf{T}[\boldsymbol{K}] + \mathsf{T}[\boldsymbol{K}]^T - \mathsf{D}[\boldsymbol{K}]\right)\boldsymbol{R}^{-T}. \tag{A.3}$$

The latter is derived in [38], in which $\mathsf{T}[\cdot]$ and $\mathsf{D}[\cdot]$ are matrix operators that return the upper

triangular and diagonal parts of the matrix, respectively, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are results from the QR-decomposition of $\boldsymbol{FD}$, as introduced in §3.3.2, and $\boldsymbol{K} = \frac{\partial \psi}{\partial \boldsymbol{R}} \boldsymbol{R}^T$.

At this point, we have obtained $\frac{\partial \psi}{\partial \boldsymbol{F}}$ from the above least-squares solve. Using Eq. (A.3), we compute the upper triangular part of $\boldsymbol{K}$ by

$$\tilde{\boldsymbol{K}} = \boldsymbol{Q}^T \frac{\partial \psi}{\partial \boldsymbol{F}} \boldsymbol{D}^{-T} \boldsymbol{R}^T. \tag{A.4}$$

In the resulting $\tilde{\boldsymbol{K}}$, only its upper triangular part is same as $\boldsymbol{K}$ (i.e., $\mathsf{T}[\tilde{\boldsymbol{K}}] = \mathsf{T}[\boldsymbol{K}]$). Fortunately, we observe the following relationships,

$$\mathsf{T}[\tilde{\boldsymbol{K}} \boldsymbol{R}^{-T}] = \mathsf{T}[\boldsymbol{K} \boldsymbol{R}^{-T}] = \mathsf{T}[\frac{\partial \psi}{\partial \boldsymbol{R}}], \tag{A.5}$$

which can be easily verified by noticing that $\boldsymbol{R}$ is an upper triangular matrix. Furthermore, because $\frac{\partial \psi}{\partial \boldsymbol{R}}$ is also an upper triangular matrix, we can compute this matrix by evaluating $\tilde{\boldsymbol{K}} \boldsymbol{R}^{-T}$ and ignoring its lower triangular part.

Finally, we express $\frac{\partial \psi}{\partial \boldsymbol{R}}$ analytically. Since $\psi(\boldsymbol{R}) = g(\boldsymbol{R}_2) + h(\boldsymbol{R}_3)$ is a quadratic form, its derivative with respect to $\boldsymbol{R}$ is a linear function with respect to its coefficients, a_i and b_i $(i = 1, 2, 3)$. We denote this linear function using $L_{\boldsymbol{R}}(a_i, b_i)$. This leads to a system of linear equations,

$$L_{\boldsymbol{R}}(a_1, a_2, a_3, b_1, b_2, b_3) = \frac{\partial \psi}{\partial \boldsymbol{R}},$$

where the value on the right-hand side is computed from the previous step. Each of the equations corresponds to a $\frac{\partial \psi}{\partial \boldsymbol{R}}$ value evaluated at an edge $\boldsymbol{e}$. When solving this system in a least-squares fashion, we must also ensure that $g(\boldsymbol{R}_2)$ and $h(\boldsymbol{R}_3)$ are both positive semidefinite, or the matrices,

$$\boldsymbol{A} = \begin{bmatrix} a_1 & \frac{1}{2}a_2 \\ \frac{1}{2}a_2 & a_3 \end{bmatrix} \text{ and } \boldsymbol{B} = \begin{bmatrix} b_1 & \frac{1}{2}b_2 \\ \frac{1}{2}b_2 & b_3 \end{bmatrix},$$

must be positive semidefinite. This imposes a conic constraint, and the problem is a classic semidefinite-quadratic-linear programming problem, which we solve using the SDPT3 solver [83].