Fitting Procedural Yarn Models for Realistic Cloth Rendering

Shuang Zhao
University of California, Irvine

Fujun Luan
Cornell University

Kavita Bala
Cornell University

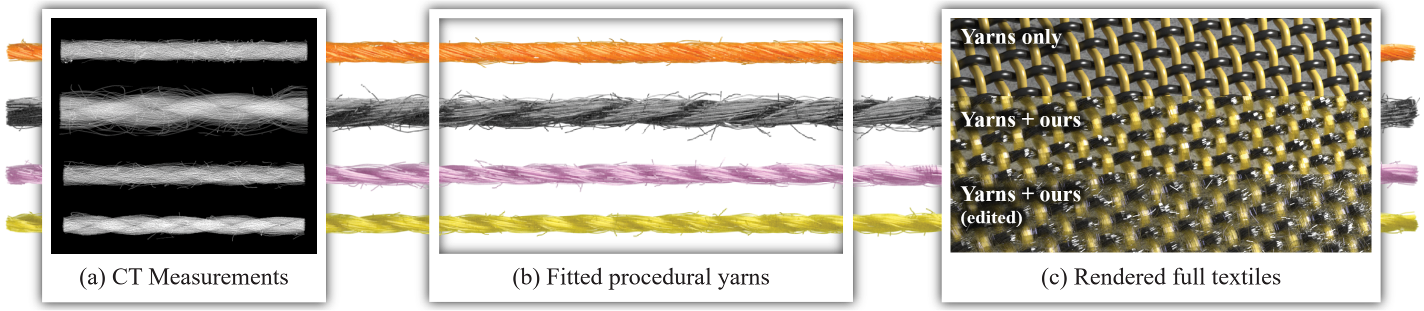


Figure 1: *We present a new technique to automatically generate procedural representations of yarn geometry. Based on geometric measurements of physical yarn samples (a), our approach fits statistical representations of fiber geometry that closely match reality (b). The four yarns in (a, b) from top to bottom are cotton, rayon, silk, and polyester. Our fitted models can populate realistic fiber-level details into yarn-based fabric models (generated using textile design software or physically-based yarn simulation) to significantly improve the quality of the rendered fabrics (c-top vs. c-middle (ours)). Our procedural models carry high-level synthetic information (e.g., twisting and hairiness) which offers easy editability (c-bottom).*

Abstract

Fabrics play a significant role in many applications in design, prototyping, and entertainment. Recent fiber-based models capture the rich visual appearance of fabrics, but are too onerous to design and edit. Yarn-based procedural models are powerful and convenient, but too regular and not realistic enough in appearance. In this paper, we introduce an automatic fitting approach to create high-quality procedural yarn models of fabrics with fiber-level details. We fit CT data to procedural models to automatically recover a full range of parameters, and augment the models with a measurement-based model of flyaway fibers. We validate our fabric models against CT measurements and photographs, and demonstrate the utility of this approach for fabric modeling and editing.

Keywords: appearance modeling, procedural geometry, textile

Concepts: •**Computing methodologies** → **Rendering;**

1 Introduction

Fabrics are essential to our daily lives. Designing and modeling them virtually is important for many applications such as online retail, textile design, and in entertainment applications like games and movies. Despite a rich history in computer graphics, accurately modeling fabrics remains challenging; fabrics are structurally and optically very complex, resulting in dramatic variation in their appearance.

SIGGRAPH '16 Technical Paper, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925932

Representing the richness of fabrics in virtual models has actively driven research in a variety of fabric appearance models. Recently, *micro-appearance based* fabric models [Zhao et al. 2011; Zhao et al. 2012; Khungurn et al. 2015; Schröder et al. 2015] that explicitly capture a fabric's micro-scale geometry have been introduced. Compared to traditional methods, these models describe fabrics at unprecedented detail, and thus offer superior generality as the complex appearance of a fabric is largely a direct consequence of its small-scale structures.

Since fiber-level details greatly affect large-scale fabric appearance, accurately capturing them is critical for predictive applications such as textile design. Previously, these detailed structures were measured through volume imaging (e.g., computed microtomography), leading to high-resolution 3D volumes [Zhao et al. 2011; Zhao et al. 2012] or large numbers of unorganized fiber curves [Khungurn et al. 2015]. Neither of these representations is easy to manipulate. On the other hand are yarn-based models [Kaldor et al. 2008; Cirio et al. 2014] which are much more amenable to editing and simulation [Yuksel et al. 2012], but do not capture the richly detailed fiber-level structures that are needed to achieve realism. See Figure 1-c (top), where the yarns appear to be made of plastic because they miss many detailed highlights generated by fiber-level structures. Bridging the gap between convenience and easy modeling on the one hand, and complex appearance on the other, is critical for the adoption of richly detailed fabric models.

This paper introduces an automated modeling process which takes physical CT measurements (Figure 1-a) and computes procedural descriptions for fabric yarns (Figure 1-b). The output can be used to populate realistic fiber-level details into yarn-based models (Figure 1-c, middle), significantly improving the realism of the final output. Further, the parameterized models are easy to edit, producing significant variations of appearance that match the real-world behavior of yarns (Figure 1-c, bottom).

Micro-geometry of fabrics has been studied extensively in textile research. Recently, Schröder et al. [2015] introduced a *procedural yarn model* to computer graphics leveraging state-of-the-art results from this field [Morris et al. 1999; Tao 1996; Keefe 1994]. Their model represents fabric yarns based on statistical distributions

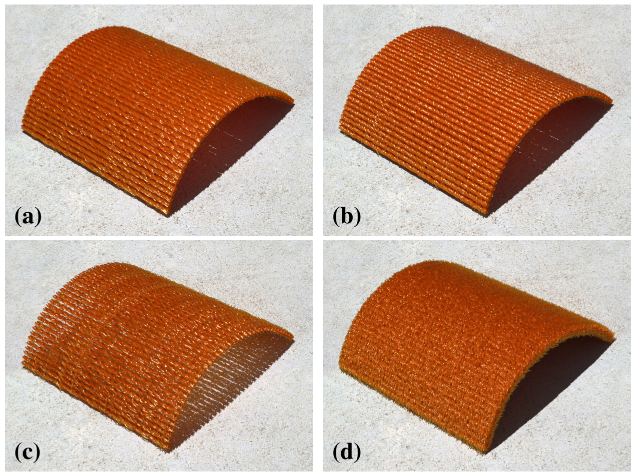


Figure 2: **Rendered fabrics** *with identical yarn-level geometry but varying fiber-level properties: (a) original; (b) modified fiber twisting; (c) modified fiber distribution; (d) increased flyaway variance. These fiber-level variations change appearance at much larger scales.*

which can be edited easily. However, how to set parameter values for this model to closely match real fabrics remains a significant challenge. In [Schröder et al. 2015], creating procedural yarns was a manual process. But this fitting can be tricky. As demonstrated in Figure 2, changes in fiber-level properties, from twisting to level of hairiness, can greatly affect a fabric's overall appearance. Thus, properly fitting these parameters is essential to correctly reproduce the visual characteristics of real-world fabrics. But this is very challenging to do manually as fiber-level structures are too small to observe, and setting corresponding parameters properly requires careful analysis of 3D micro-geometries that are difficult to obtain without volumetric measurements.

In this paper, we present a novel technique to create procedural yarn models *automatically* based on physical measurements of real-world yarns. Our contributions include:

- An end-to-end pipeline to automatically build our model from physical measurements acquired using micro CT imaging (§4).
- An improved flyaway fiber model which is not only statistically more meaningful, but also easier to fit to (§5).
- Validation of our fitted results by comparing with CT measurements as well as photographs (§6).

We are releasing our implementation, measured data, and fitted parameters (Table 2). Our tool and data allow fiber-level details to be added to yarn-based cloth designs produced by textile software [Pointcarré 2016] and simulation tools [Kaldor et al. 2008; Cirio et al. 2014]. The added details can lead to significantly higher rendering quality (Figures 1 and 17) and greatly benefit the design applications.

2 Related Work

Fabric appearance. Fabric appearance has been an active research topic in computer graphics for decades. Traditionally, fabrics are treated as infinitely thin 2D sheets. Many surface reflectance models have been proposed to model fabric appearance [Irawan and Marschner 2012; Sadeghi et al. 2013]. Although these models provide high-quality renderings for fabrics viewed from a distance, they lack the power to closely reproduce the appearance of fabrics with thick yarns or fuzzy silhouettes, or the generality to handle different fabrics with wildly varying appearances.

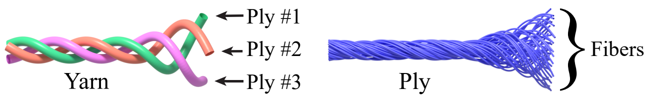


Figure 3: **Fabric structure:** *a* yarn *normally consists of multiple sub-stands called* plies. *Each ply in turn contains tens to hundreds of micron-diameter* fibers.

Recent advances in cloth appearance modeling have led to the development of volumetric [Xu et al. 2001; Jakob et al. 2010; Zhao et al. 2011; Zhao et al. 2012; Zhao et al. 2013] and fiber-based [Schröder et al. 2015; Khungurn et al. 2015] cloth models. Unlike traditional methods, they describe fabrics at 3D volumes with varying densities (volumetric) or collections of fiber curves (fiber-based). Compared to traditional surface-based methods, these models are better at capturing the thickness and fuzziness of fabrics, which add significantly to their visual realism, and thus have brought the quality of computed rendered cloth to the next level. However, these techniques either provide automated creation of highly realistic models (achieved by measuring real-world samples) but carry little high-level synthetic information (e.g., yarn twisting) [Zhao et al. 2011; Khungurn et al. 2015], or offer compact procedural representations but rely on manually configured models which require nontrivial parameter tweaking to match reality closely [Schröder et al. 2015]. We introduce a technique that takes the strength of both approaches while avoiding their weaknesses.

Fabric geometry. The geometric structure of fabrics have been studied by the textile research community for decades. State-of-the-art results from this field [Morris et al. 1999; Tao 1996; Keefe 1994; Shinohara et al. 2010] have resulted in the introduction of some aforementioned micro-appearance modeling techniques [Zhao et al. 2011; Schröder et al. 2015].

Yarn-based fabrics. Yarn-based representations describe fabrics at the yarn-level and are used almost exclusively in textile design [Pointcarré 2016]. Recently, yarn-based simulation [Kaldor et al. 2008; Cirio et al. 2014] and editing [Yuksel et al. 2012] have been developed in computer graphics. These models, however, usually offer limited rendering quality as they lack fiber-level details that contribute significantly to a fabric's overall appearance. Procedural models generated by our method bridge this gap: they can be used to populate fiber-level structure in the yarn models, significantly improving the virtual realism of their appearance.

3 Background

We discuss prior work in fabric micro-appearance modeling techniques (§3.1) and procedural yarn models (§3.2).

3.1 Fabric Micro-Appearance Modeling

Fabrics are constructed by combining multiple yarns via manufacturing technologies such as weaving and knitting. Each yarn, in turn, is created by twisting hundreds of micron-diameter fibers. Real-world yarns usually consist of multiple sub-strands or *plies* (Figure 3). Recent fiber-based models achieve visual accuracy by representing the real structure of yarns and plies explicitly. These models can be classified into two major categories: volumetric and fiber-based.

Volumetric models. Volumetric models describe a fabric's micro-geometry using high-resolution 3D volumes obtained by processing micro CT measurements [Zhao et al. 2011]. At each point in the volume, local fiber density and orientation are stored.

Fiber-based models. Fiber-based models treat fabrics as collections of individual fibers described as 1D curves [Khungurn et al. 2015; Schröder et al. 2015]. These fiber curves are then combined

Algorithm 1 procedural yarn generation

```
1: for each ply $i$ do
2:     generate regular fibers using Eq. (1), Eq. (2), Eq. (3)
3:     add flyaway fibers
4:     scale all fibers in the ply to obtain elliptical cross-sections
5: end for
6: twist all plies together around the yarn center
```

with *bidirectional curve scattering distribution functions* (BCSDFs) to generate rendered images.

Compared to volumetric models, fiber-based models offer a similar level of realism [Khungurn et al. 2015] while being more compact and providing more synthetic information. Therefore, in this paper, we choose to generate fiber-based representations procedurally.

3.2 Procedural Modeling of Yarn Geometry

In this paper, we build on a procedural yarn model proposed by Schröder et al. [2015] based on state-of-the-art results from textile research [Morris et al. 1999; Tao 1996; Keefe 1994]. This model statistically describes how individual yarns are formed by underlying fibers. The key parameters are:

- *For fibers:* cross-sectional fiber distribution, fiber twisting, and fiber migration.
- *For plies:* ply cross section, ply twisting, and fiber count.

Additionally, this model has a separated step, which we improve upon in §5, that handles the important-for-realism effect of flyaway fibers. See [Schröder et al. 2015] for more details.

Cross-sectional fiber distribution. A key component of the procedural yarn model is a *cross-sectional fiber distribution* that captures the likelihood of a fiber's existence given its distance $R \in [0, 1)$ from the ply center. This distribution uses the following (unnormalized) density function:

$$p(R) = (1 - 2\epsilon)\left(\frac{e - e^R}{e - 1}\right)^{\beta} + \epsilon, \tag{1}$$

which is used with rejection sampling (Algorithm 2) to draw cross-sectional fiber locations. Given a sampled location for the i-th fiber (x_i, y_i), the fiber curve (as a circular helix parameterized by θ) can be generated as follows, assuming the ply center to be the Z-axis:

$$x(\theta) = R_i \cos(\theta + \theta_i),\ y(\theta) = R_i \sin(\theta + \theta_i),\ z(\theta) = \frac{\alpha\theta}{2\pi}, \tag{2}$$

where $R_i := \|(x_i, y_i)\|_2$, $\theta_i := \text{atan2}(y_i, x_i)$, and α is a constant determining the fiber's twist (i.e., the helix's pitch).

Fiber migration. In Eq. (2), the distance between a generated fiber and the ply center stays constant. But this is unnatural: fibers typically migrate from such a fixed distance. This *fiber migration* is modeled by allowing the distance to change continuously between two given constants $R_{\min}$ and $R_{\max}$. That is, by replacing R_i in Eq. (2) with:

$$R_i(\theta) := R_{\min} R_i + \frac{(R_{\max} - R_{\min}) R_i}{2}[\cos(s\theta + \theta_i^{(0)}) + 1], \tag{3}$$

where s is a constant controlling the length of a rotation, and $\theta_i^{(0)}$ is a per-fiber parameter indicating the rotation's initial "phase".

Ply cross section. Plies generated with Eq. (1) and Eq. (3) always have circular cross-sections. A simple generalization is to support elliptical cross-sections by scaling a ply along the X- and Y-directions by factors of e_X and e_Y respectively.

Algorithm 2 Sampling cross-sectional fiber location

```
Require: cross-sectional fiber distribution parameters $\epsilon$, $\beta$
1: procedure SampleFiberLocation($\epsilon$, $\beta$)
2:     repeat
3:         draw $(x, y)$ uniformly in a unit disc
4:         draw $\xi$ from $U[0, 1]$
5:     until $\xi < p(\|(x, y)\|_2)$                ▷ $p$ defined in Eq. (1)
6:     return $(x, y)$
7: end procedure
```

Flyaway fibers. Real yarns usually contain *flyaway fibers* that do not follow the flow of normal fibers. These irregularities not only contribute greatly to the realism of yarn appearance, but are also crucial for reproducing fuzzy silhouettes of real-world fabrics. Previous work generated flyaway fibers in a somewhat simplified manner using 3D perlin noise [Schröder et al. 2015]. We introduce an improved flyaway model (§5) which is statistically more meaningful and easier to fit to.

Ply twisting. The final step to build a procedural yarn model is twisting the component plies. For a yarn centered at the Z-axis, each ply is twisted to follow a circularly helical curve

$$\boldsymbol{S}(z) := (S_x(z),\ S_y(z),\ z)$$

with its pitch controlled by α^{ply}:

$$\begin{aligned} S_x(z) &= R^{\text{ply}} \cos(2\pi z/\alpha^{\text{ply}} + \theta^{\text{ply}}), \\ S_y(z) &= R^{\text{ply}} \sin(2\pi z/\alpha^{\text{ply}} + \theta^{\text{ply}}). \end{aligned} \tag{4}$$

Besides the cross section and twisting information, each ply has an integer m associated with it that specifies the number of component fibers. The entire pipeline for procedural generation of yarn geometry is summarized in Algorithm 1.

Challenges in parameter fitting. Although the fiber generation process (Algorithm 1) is straight-forward given the model parameters, its inverse problem of parameter fitting is far from trivial. Many fiber-level parameters, such as cross-sectional fiber distribution and fiber migration, are based on the statistical properties of a yarn's micro-geometry. These properties are difficult to acquire due to their small scale, and challenging to fit to because of their naturally existing irregularities.

In this paper, we address these challenges by acquiring a yarn's micro-geometry using micro CT imaging, and introducing an end-to-end pipeline to automatically and robustly fit procedural representations to the measured data (§4). In addition, we introduce a new model for capturing flyaway fibers (§5) which contributes significantly to photorealism.

4 Model Fitting

We present our end-to-end pipeline for fitting procedural yarns to physical measurements of micro-geometries. he parameters needed to be fit are summarized in Table 1. The challenge is converting volumetric CT data with no synthetic information to a procedural yarn model with correct ply-level and fiber-level properties. Many parameters must be identified properly to match the appearance of physical yarns. Ultimately this approach yields a compact and editable representation capturing the rich irregularities of real-world yarns.

Our pipeline, from a high level, is analogous to executing Algorithm 1 in *reverse* order (Figure 4). In particular, given measured yarn micro-geometry (§4.1):

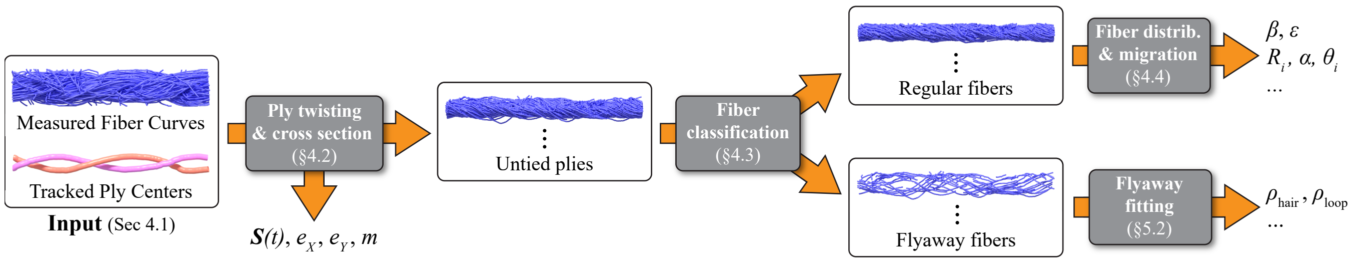


Figure 4: Our parameter fitting pipeline. *This pipeline is approximately in the opposite order of the procedural yarn generation algorithm (Algorithm 1) since it is fitting parameters.*

Parameter Type	Parameter Name	Defined	Fitted
Cross-Sectional Fiber Distribution	$\beta,\ \epsilon$	Eq. (1)	§4.4
Fiber Twisting	α $R_i,\ \theta_i$	Eq. (2)	§4.4
Fiber Migration	$R_{\min},\ R_{\max},\ s$ $\theta_i^{(0)}$	Eq. (3)	§4.4
Ply Cross Section	$e_X,\ e_Y$	§3.2	§4.2
Ply Twisting	$R^{\text{ply}},\ \alpha^{\text{ply}}$	Eq. (4)	§4.2
Per-ply Fiber Count	m	§3.2	§4.2
Improved Flyaway Fiber Distribution	$\rho^{\text{loop}},\ \rho^{\text{hair}},\ R^{\text{loop}}_{\max},$ $R^{\text{hair}}_{\min},\ R^{\text{hair}}_{\text{span}},\ z^{\text{hair}}_{\min},$ $z^{\text{hair}}_{\text{span}},\ \theta^{\text{hair}}_{\min},\ \theta^{\text{hair}}_{\text{span}}$	Eq. (11)	§5.2

Table 1: List of parameters *used by the procedural yarn model. Parameters shown in purple are per-ply attributes. Those shown in blue are specified for every fiber by independently sampling the corresponding statistical distributions. In particular, per-fiber flyaway parameters (i.e., $R^{loop}_{\max}$, $R^{hair}_{\min}$, R^{hair}_{span}, and θ^{hair}_{span}) are sampled from normal distributions (see §5 and Table 2) while others (i.e., R_i, θ_i, and $\theta_i^{(0)}$) are drawn uniformly (§3.2).*

- First, we estimate ply twisting as well as the component plies' cross-sectional shapes (§4.2). Using this information, we 'untie' and deform all component plies so that they are centered on the Z-axis and have circular cross-sections.
- Then, we analyze each untied ply and classify the constituent fibers into two categories: flyaway and regular (§4.3).
- Next, based on the obtained flyaway fibers, we determine the parameter values required by our improved model (§5.2).
- Lastly, we fit the cross-sectional fiber distribution and migration parameters using the previously obtained regular fibers (§4.4).

In the rest of this section, we first describe our CT imaging process stage leveraging state-of-the-art techniques [Zhao et al. 2012; Khungurn et al. 2015] to obtain clean input geometry to our main fitting pipeline (§4.1). Then, we explain our pipeline following the flow of Algorithm 1 (i.e., from right to left in Figure 4).

4.1 Input

We acquire micro geometries of physical yarns using micro CT imaging. As shown in Figure 5-a, multiple yarns are packed together for faster acquisition since they can be scanned simultaneously. Given micro CT measurements with volumetric fiber densities (Figure 5-b), we process them using the approach developed by Khungurn et al. [2015] to extract fiber curves (Figure 5-c). In addition, we perform yarn tracking [Zhao et al. 2012] to extract the center curve for each component ply (Figure 5-d). The recovered fiber curves and ply centers (Figure 5-cd) act as input to our main parameter fitting pipeline.

4.2 Ply Twisting and Cross Section Estimation

The first step of our parameter fitting pipeline is to recover the input yarn's plying (i.e., ply twisting and cross-sectional shape) so that we can separate individual plies and perform later per-ply analysis. In this paper, we assume all plies in a yarn are *identical* and *evenly distributed* around the yarn center.

Ply twisting. For each given ply center i (represented as a polyline), we fit a helical curve $\mathbf{S}_i$ in the form of Eq. (4) which requires estimating the ply radius R_i^{ply}, the pitch α_i^{ply}, and initial angle θ_i^{ply}. Given our assumption of identical and evenly distributed plies, this boils down to finding:

- One set of R^{ply} and α^{ply} shared by all piles;
- θ_1^{ply}, the initial angle of the first ply, which can be used to determine those of all other plies by setting $\theta_i^{\text{ply}} = \theta_1^{\text{ply}} + 2\pi\frac{i-1}{K}$.

To determine these parameters, we optimize the L2 distance between the given and fitted ply centers by minimizing:

$$E^{\text{ply}}(R^{\text{ply}}, \alpha^{\text{ply}}, \theta_1^{\text{ply}}) := \sum_{i=1}^{K} \int_{z_0}^{z_1} \left\| \boldsymbol{S}_i\left(z \mid R^{\text{ply}}, \alpha^{\text{ply}}, \theta_i^{\text{ply}}\right) - \boldsymbol{S}_i^{\text{tracked}}(z) \right\|_2^2 \,\mathrm{d}z, \tag{5}$$

where K is the number of plies, $\boldsymbol{S}_i$ is given by Eq. (4), and $\boldsymbol{S}_i^{\text{tracked}}$ is the i-th input ply center (Figure 5-d) whose two endpoints give z_0 and z_1.

To minimize Eq. (5), we use an open source implementation [Hutt 2011] of the Nelder-Mead Simplex Method [1965]. One could also leverage more advanced optimization methods such as the floating tangent algorithm [Derouet-Jourdan et al. 2013], possibly leading to better accuracy and performance. However, since our acquisition setup keeps yarn samples straight (Figure 5-a), the input curves

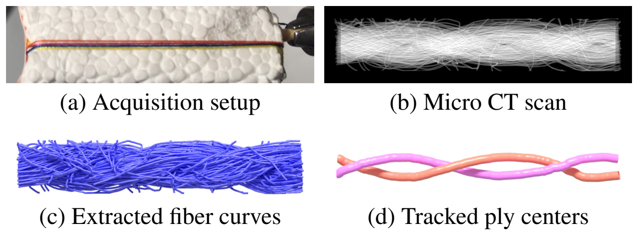


Figure 5: Acquisition of yarn geometries. *(a) We stack multiple yarns and CT scan them. (b) Volumetric density information for one of the scanned yarns. (c) Extracted fiber-based representation of the yarn. (d) Tracked ply centers.*

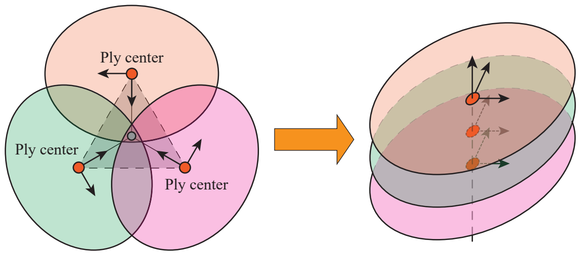


Figure 6: Fitting ply cross-sections *(3-ply example). (left) We place an ellipse at each ply center. The short axis of this ellipse points toward the center of a polygon (shown in gray with dashed boundaries) formed by all ply centers. (right) We then rotate and translate all plies (in plane) with the corresponding fiber points. These transformed 2D fiber locations will then be used to fit the ply cross-sections (Figure 7).*

(i.e., $\boldsymbol{S}_i^{\text{tracked}}$) are already close to circular helices. We found that our simple approach was fast, well-behaved (i.e., there were no convergence issues), and produced high-quality results for rendering purposes.

Cross-section estimation. As in [Schröder et al. 2015], we model the cross-sections of the plies as ellipses that are allowed to intersect, to mimic ply compression. Given a cross-sectional plane, we place an ellipse at the center of each ply with its short axis pointing towards the center of a polygon formed by all the ply centers (see Figure 6 for a 3-ply example). These ellipses are used to determine the lengths of the ellipse axes (i.e., e_X and e_Y). We assume identical plies, and therefore obtain one set of e_X and e_Y values based on information from all plies. In particular, we rotate and translate each ply (with all its constituent fiber centers) in the plane, making it axis-aligned and centered at the origin (Figure 6, right). By stacking all transformed plies from all cross-sectional planes,[1] we obtain an accumulated 2D point cloud consisting of fiber centers (Figure 7). Then, we set e_X and e_Y to twice the standard deviation of the X- and Y-coordinates of all these points, respectively. The resulting ellipse covers approximately 95% of the fiber centers. We rely on our flyaway fiber model to capture the remaining, mostly irregular, fibers.

Besides ply twisting and cross-section parameters, we estimate the number of fibers per ply as $m = \lfloor L_{\text{total}}/(L_{\text{ply}}\, K) \rfloor$, where L_{total} indicates the total length of fibers in the measured geometry (Figure 5-c), L_{ply} denotes the length of a fitted ply center (all $\boldsymbol{S}_i$ curves have identical lengths), and K is the number of plies.

4.3 Fiber Classification

Using the plying information obtained in §4.2, we untie the plies so that they are all centered around the Z-axis. All fibers in these piles then form a 'fiber soup' which will be analyzed by the following steps of our pipeline. In this step, we classify these fibers into *regular* versus *flyaway*.

To classify each fiber into one of the two categories, we consider its minimal and maximal distances denoted as $d_{\min}$ and $d_{\max}$ to the ply center (i.e., Z-axis). Given a fiber with n vertices $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$, we have

$$d_{\min} := \min_{1 \le i \le n} \|(x_i, y_i)\|_2, \quad d_{\max} := \max_{1 \le i \le n} \|(x_i, y_i)\|_2.$$

CT measurements of real yarns usually contain alien components (e.g., dust) that do not belong to the yarn. They can be distinguished by large $d_{\min}$ values since alien materials tend to stay far from the

[1]In practice, we take around 1000 cross-sectional planes for each yarn, similar to the resolution of micro CT measurements.

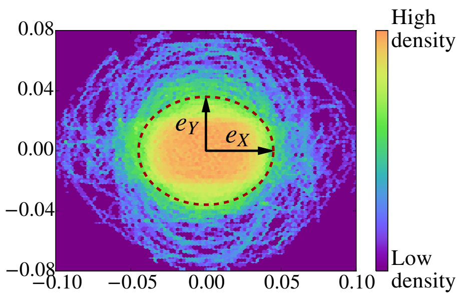


Figure 7: Fitting ply cross-sections (cont'd). *A density visualization of transformed fiber centers (Figure 6) from a sequence of cross-sectional planes. We use this density information to determine the shape of ply cross-sections.*

ply center. In practice, we consider all fibers with $d_{\min}$ beyond some threshold as alien and simply ignore them. We then categorize the remaining fibers into regular and flyaway based on their $d_{\max}$ values. Intuitively, fibers with small $d_{\max}$ values stay close to the ply center and are likely to be regular. On the other hand, those with large $d_{\max}$ values are at some point far from the center. Thus, they are considered flyaway.

To robustly obtain the thresholds for respectively identifying alien and flyaway fibers, we consider the means (denoted as $\mu_{\min}$, $\mu_{\max}$) and standard deviations (denoted as $\sigma_{\min}$, $\sigma_{\max}$) of all $d_{\min}$ and $d_{\max}$ values respectively. Precisely, we treat all fibers satisfying $d_{\min} > \mu_{\min} + c_{\min}\sigma_{\min}$ as alien, and the remaining ones with

$$d_{\max} > \mu_{\max} + c_{\max}\sigma_{\max} \tag{6}$$

as flyaway (where $c_{\min}$ and $c_{\max}$ are user-specified constants). In practice, we use $c_{\min} = c_{\max} = 2$ and perform fiber classification separately for each (untied) ply. Figure 8 shows an example classification.

4.4 Fitting Fiber Distribution and Migration

Given the set of regular fibers obtained in §4.3, we now present the last step of our pipeline which fits the fiber distribution Eq. (1) and migration Eq. (3) parameters. Recall that all these fibers belong to untied plies, meaning that they are all centered around the Z-axis.

In theory, fiber migration parameters $R_{\min}$, $R_{\max}$, and s, can be recovered from a single regular fiber. However, due to irregularities in real data, using only one fiber generally yields highly unreliable results. Furthermore, many fibers in the input geometry are short due to limitations of the CT imaging processing step [Khungurn et al. 2015], making migration parameter estimation even more challenging.

We tackle this problem by minimizing a reconstruction error defined as:

$$\begin{aligned} &E_{\text{mig}}(R_{\min},\, R_{\max},\, s) \\ &:= \sum_i \min_{R_i,\, \theta_i^{(0)}} E_i\left(R_i,\, \theta_i^{(0)} \,\middle|\, R_{\min},\, R_{\max},\, s\right), \end{aligned} \tag{7}$$

where the summation is over all regular fibers, and E_i indicates the squared L2 difference between fiber i (represented as a polyline) and the helix generated with R_i, $\theta_i^{(0)}$, $R_{\min}$, $R_{\max}$, and s via Eq. (1) and Eq. (3). Namely,

$$E_i\left(R_i,\, \theta_i^{(0)} \,\middle|\, R_{\min},\, R_{\max},\, s\right) := \int_z \|\boldsymbol{F}_i(z) - \tilde{\boldsymbol{F}}_i(z)\|_2^2 \, \mathrm{d}z, \tag{8}$$

where $\boldsymbol{F}_i$ and $\tilde{\boldsymbol{F}}_i$ respectively denote the input and generated fibers, both of which are parameterized by z. The limits of this 1D integral

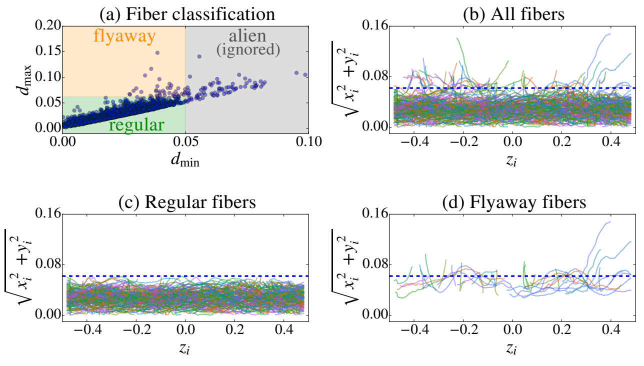


Figure 8: Fiber classification: *we classify all input fibers into two categories: regular and flyaway (with alien ones removed). (a) Scatter plot of $(d_{\min}, d_{\max})$ for each fiber. (b, c, d) 2D fiber visualizations where the horizontal and vertical axes show z_i and $\|(x_i, y_i)\|_2$ for each fiber vertex, respectively. The dashed line indicates the threshold beyond which a fiber is considered flyaway.*

are given by the Z-coordinates of $\boldsymbol{F}_i$'s two endpoints. Then, we set $(R^*_{\min},\, R^*_{\max},\, s^*) = \arg\min E_{\text{mig}}(R_{\min},\, R_{\max},\, s)$.

Minimizing this reconstruction error Eq. (7), however, is non-trivial since the error metric itself includes minimization over R_i and $\theta_i^{(0)}$. Thus, we enumerate a densely sampled set of $R_{\min}$, $R_{\max}$, and s values. For each combination, we solve the inner optimization problem (i.e., RHS of Eq. (7)). Similar to minimizing Eq. (5), we found it easy to find optimizers for this inner problem and used the same implementation.

After determining the fiber migration parameters, only the cross-sectional fiber distribution remains unknown. Let R_i^* and $\theta_i^{(0)*}$ be the minimizers of $E_i\left(R_i,\, \theta_i^{(0)} \;\middle|\; R^*_{\min},\, R^*_{\max},\, s^*\right)$ for each fiber i. We then apply Maximum-Likelihood Estimation (MLE) over all R_i^* values to obtain the fiber distribution parameters ϵ and β. Notice that the density function Eq. (1) is essentially conditional since it is used within a rejection sampling framework (Algorithm 2) where R is not uniformly distributed. The unconditional density for the MLE should be:

$$p^{\text{MLE}}(R) = 2R\, p(R) = 2R \left[(1-2\epsilon) \left(\frac{e - e^R}{e - 1} \right)^{\beta} + \epsilon \right]. \quad (9)$$

Let $q(\epsilon, \beta)$ be the normalization term for Eq. (9)

$$q(\epsilon, \beta) := \int_0^1 p^{\text{MLE}}(R)\, \mathrm{d}R, \quad (10)$$

we have the normalized density function $p^{\text{MLE}}_{\text{norm}}(R \mid \epsilon, \beta) := p^{\text{MLE}}(R)/q(\epsilon, \beta)$, which can be used in MLE. In practice, we use Matlab's mle() function with q evaluated numerically which can sometimes be prone to (numerically similar) local optima. However, this has caused no visual difference in our experiments.

At this point, we have obtained a set of ply-based and fiber-based parameters which can be used to procedurally generate regular fiber curves. With only regular fibers, however, yarns look unrealistic because they lack irregularities. In the next section, we present our improved flyaway fiber model in §5.1, and in §5.2, we describe how to fit the model parameters given real flyaway fibers obtained in §4.3.

5 Our Flyaway Fiber Model

Natural cloth yarns normally contain flyaway fibers which deviate from most other fibers, causing great irregularity in a yarn's micro-geometry. Although there are much fewer flyaways fibers comparing to regular ones, they contribute significantly to a fabric's fuzziness. Thus, to accurately reproduce the appearance and structure of real-world fabrics, these fibers need to be properly modeled.

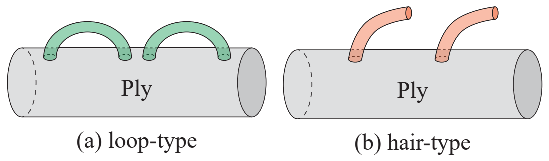


Figure 9: The two types of flyaway fibers *in our model.*

5.1 Model Description

Flyaway fibers have been studied by textile researchers. In particular, Voborova et al. [2004] categorize flyaway fibers into five types. Inspired by their work, we introduce an improved flyaway fiber model over the one developed by Schröder et al. [2015] based on Perlin noise.

Our model classifies flyaway fibers into two categories: *loop* and *hair* (Figure 9). Our hair-type fibers correspond to the first two flyaway classes in [Voborova et al. 2004], while our loop-type to their third class.

Loop-type. Loop-type flyaway fibers have both endpoints inside the main ply body. Each of these fibers was originally regular but part of it has been accidentally pulled out during the manufacturing process (Figure 9-a). The density of these fibers and the distance each loop deviates from the ply center provide an important visual cue on how tight the fibers are twisted.

Schröder et al. [2015] do not explicitly model these type of fibers. Instead, they use very strong fiber migrations to achieve similar effects. Unfortunately, this approach cannot easily capture occasionally occurring loop-type fibers, which are the common case in reality. The resulting geometry normally contains regularly appearing loops (Figure 10-a).

We generate loop-type flyaway fibers by modifying the regular ones built previously (Line 2 of Algorithm 1). When turning a regular fiber into a loop-type flyaway, we randomly pick one of its vertices with Eq. (3) maximized and raise its radius to $R^{\text{loop}}_{\max}$ (by scaling its X- and Y- coordinates). We also scale up the radii of neighboring vertices belonging to the same migration period (i.e., one period of $R_i(\theta)$ in Eq. (3), see Figure 11). To create a set of loop-type flyaway fibers (based on existing regular ones), we draw $R^{\text{loop}}_{\max}$ from a normal distribution for each of them.

For each ply, we use ρ^{loop} to capture the density of loop-type fibers. Given an untied ply centered around $\boldsymbol{Z_0 Z_1}$ with $\boldsymbol{Z_0} = (0, 0, z_0)$ and $\boldsymbol{Z_1} = (0, 0, z_1)$, we repeat the aforementioned process to generate $\lfloor \rho^{\text{loop}}(z_1 - z_0) \rfloor$ loop-type flyaway fibers.

Hair-type. As shown in Figure 9-b, each hair-type flyaway fiber has one endpoint outside the body of its hosting ply. This type of fiber contributes most significantly to a yarn's hairy appearance. We create a hair-type fiber (for Line 3 of Algorithm 1) by adding its

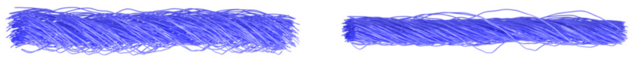


Figure 10: Comparison of loop-type flyaway fiber models: *(a) the migration-based approach [Schröder et al. 2015] lacks the generality to represent occasionally occurring loops; (b) our model has sufficient representative power to capture such fiber geometry.*

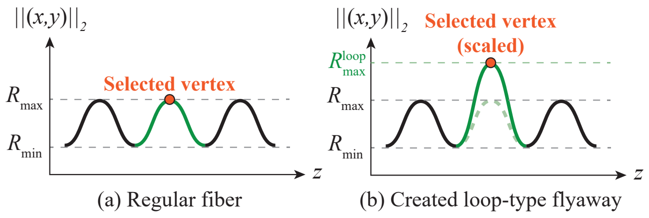


Figure 11: Generation of loop-type flyaway fibers. *(a) We randomly pick a vertex maximizing the distance to the ply center (indicated with an orange circle). (b) We then select all vertices belonging to the same migration cycle (shown in green) and scale their radii so that the selected vertex has radius $R^{loop}_{\max}$ afterwards.*

visible (flyaway) part explicitly. That is, we generate an 'arc' determined by its starting and ending radii $R^{\text{hair}}_{\min}$, $R^{\text{hair}}_{\max}$ (i.e., distance to the ply center), azimuthal angles $\theta^{\text{hair}}_{\min}, \theta^{\text{hair}}_{\max}$, as well as locations $z^{\text{hair}}_{\min}, z^{\text{hair}}_{\max}$. Let $R^{\text{hair}}_{\text{span}} := R^{\text{hair}}_{\max} - R^{\text{hair}}_{\min}$, $\theta^{\text{hair}}_{\text{span}} := \theta^{\text{hair}}_{\max} - \theta^{\text{hair}}_{\min}$, and $z^{\text{hair}}_{\text{span}} := z^{\text{hair}}_{\max} - z^{\text{hair}}_{\min}$, a hair-type flyaway fiber parameterized by $t \in [0, 1]$ can be represented as

$$\begin{aligned} x^{\text{hair}}(t) &= R^{\text{hair}}(t) \cos\left[\theta^{\text{hair}}(t)\right], \\ y^{\text{hair}}(t) &= R^{\text{hair}}(t) \sin\left[\theta^{\text{hair}}(t)\right], \\ z^{\text{hair}}(t) &= z^{\text{hair}}_{\min} + z^{\text{hair}}_{\text{span}}\, t, \end{aligned} \tag{11}$$

with $R^{\text{hair}}(t) = R^{\text{hair}}_{\min} + R^{\text{hair}}_{\text{span}}\, t$ and $\theta^{\text{hair}}(t) = \theta^{\text{hair}}_{\min} + \theta^{\text{hair}}_{\text{span}}\, t$.

To generate a set of hair-type fibers around a common center with endpoints z_0 and z_1, we draw $\theta^{\text{hair}}_{\min}$ from $U[z_0, z_1]$ and $z^{\text{hair}}_{\min}$ from $U[0, 2\pi)$ for each of them. The other parameters $R^{\text{hair}}_{\min}$, $R^{\text{hair}}_{\text{span}}$, $\theta^{\text{hair}}_{\text{span}}$, and $z^{\text{hair}}_{\text{span}}$ are sampled from separate normal distributions. Similar to loop-type, we use ρ^{hair} to describe the density of hair-type fibers.

In [Schröder et al. 2015], hair-type fibers are generated by randomly adding short curves with tangent directions disturbed by 3D Perlin noise (Figure 12-c). Although able to generate hair-like fibers, the oversimplified nature of this approach makes it tricky to match reality and, more importantly, difficult to fit to given measured yarn geometries. Our model, on the other hand, carries clear statistical meaning and is easy to fit to (see §5.2).

Fiber perturbations. Until this point, all regular and flyaway fibers were perfectly smooth curves. This level of smoothness rarely exists in reality: real fibers generally contain many small but irregular turns. To mimic this effect, we add small perturbations to all fibers by randomly scaling the radius (i.e., X- and Y-coordinates) of each vertex.

Discussion. Our flyaway fiber models make two notable simplifications. In particular, our loop-type flyaway fibers always "span" exactly one migration cycle (as demonstrated in Figure 11), which is generally not the case in reality. Moreover, our hair-type flyaways all have "arc" shapes due to Eq. (11), while real hair-type fibers normally have more complex trajectories (Figure 13). Despite these simplifications we match overall yarn appearance well (see Figure 15). They are introduced to ensure that our model can be reliably fitted from measurements containing limited number of flyaway fibers (e.g., Figure 5-b). In the future, with greater amount of measurements, flyaway fiber models based on more complex statistical distributions could be designed.

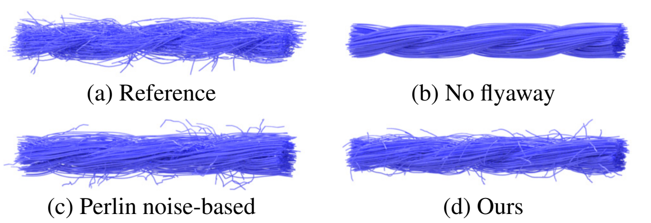


Figure 12: Comparison of flyaway fiber models (hair-type): *(a) measured yarn geometry (ground truth); (b) procedural yarn with no flyaway fibers; (c) procedural yarn with Perlin noise-based flyaways [Schröder et al. 2015]; (d) procedural with flyaways generated with our model. Fiber migration is increased in (c) for simulating loop-type effects.*

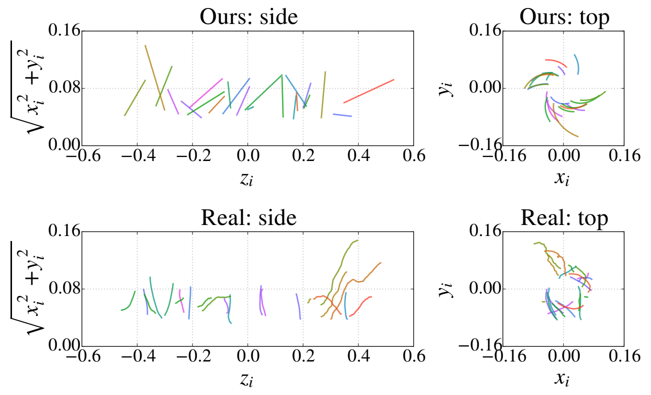


Figure 13: Simplifications *made by our hair-type flyaway fiber model: (top) our hair-type flyaways all have arc shapes; (bottom) real hair-type fibers have more complex trajectories.*

5.2 Model Fitting

Recall that in §4.3, fibers from untied plies are classified as regular and flyaway. Given our flyaway fiber model (§5.1), we now describe how to fit model parameters, i.e., ρ_{hair}, $R^{\text{hair}}_{\min}$, $R^{\text{hair}}_{\text{span}}$, $z^{\text{hair}}_{\text{span}}$, and $\theta^{\text{hair}}_{\text{span}}$ for hair-type ones, as well as ρ_{loop} and $R^{\text{loop}}_{\max}$ for loop-type ones. Notice that $z^{\text{hair}}_{\min}$ and $\theta^{\text{hair}}_{\min}$ are not fitted as we sampled them uniformly in the generation process.

After obtaining flyaway fibers for each untied ply, we start the fitting process by identifying the loop and hair components of each fiber. As illustrated in Figure 14-a, one flyaway fiber can contain both components. Thus, for each of them, we start by checking if any of its endpoints has a radius (i.e., $\|(x, y)\|_2$) above the flyaway threshold Eq. (6). If so, we cut off the corresponding end of the fiber at vertices where the radius is both at a local minimum and below the threshold and add it to the collection of hair-type flyaway fibers. The remaining part (if any) is then added to the set of loop-type fibers. This fiber-splitting operation is illustrated in Figure 14-b.

After all fibers have been processed, we obtain the flyaway densities ρ^{loop} and ρ^{hair} using the number of corresponding fibers divided by the length of the untied ply (i.e., $z_1 - z_0$). Then, for every loop-type fiber, we compute the radii of all its vertices and set $R^{\text{loop}}_{\max}$ as the maximum. Similarly, $R^{\text{hair}}_{\min}$, $R^{\text{hair}}_{\text{span}}$, $\theta^{\text{hair}}_{\text{span}}$, $z^{\text{hair}}_{\text{span}}$ can be evaluated for

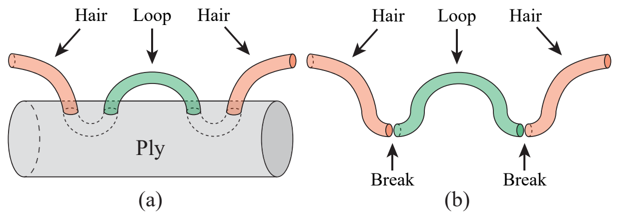


Figure 14: Processing Flyaway Fiber. *(a) One fiber can contain both hair and loop components. (b) We split each flyaway fiber into multiple hair and loop segments.*

each hair-type fiber by examining its vertices. We then compute the sample mean and standard deviation for each of these quantities, which can be used to generate hair-type fibers via Eq. (11).

With flyaway fiber parameters fitted, we finally have recovered the values of all parameters in Table 1. Using Algorithm 1, realistic yarn geometries can now be generated without the need for cumbersome CT measurement.

6 Results

We measured the micro-geometries of nine real-world yarns through three micro CT scans. These yarns are made from cotton, rayon, silk, and polyester, a good sampling of commonly used yarn materials. Our acquisition setup is shown in Figure 5-a. All yarns were scanned in an XRadia MicroXCT scanner under a resolution of 2.5 μm, and each scan took approximately three hours. All rendered images are created using a modified version of the Mitsuba physically based renderer [Jakob 2013].

Performance. Our fitting algorithm is fast: given the processed CT data (Figure 5-cd), it takes fewer than ten CPU core minutes to fit each yarn. Compared to the CT data processing step (§4.1) which normally takes several core hours, our fitting cost is negligible. Our method is generally well-behaved: despite the presence of non-convex optimization (e.g., §4.4), we did not observe any visual difference caused by local minima.

Validation. Figure 15 shows all our measured yarn geometries (a) and our fitting results (b). Procedural yarns generated with our method closely capture the statistics of the measured geometries, conveying a convincing impression of the corresponding yarn types. To validate this, we compare full renderings of our models (d) with photographs of those yarns (c). To describe the optical properties of our procedurally generated yarns, we use the fiber scattering model introduced in [Khungurn et al. 2015]. See Table 3 for corresponding parameter values. Notice that our models can sometimes appear slightly less hairy since our measured geometries (a), which come from small (around 0.5 cm in length) yarn segments, might not capture all irregularities the photos have. This is not a limitation of our approach though, since one can scan multiple copies of the same type of yarn for variety and feed all the information to our parameter fitting algorithm.

Non-repetitive yarns. Figure 16 shows long yarns generated with our approach. Compared to many existing micro-appearance models [Zhao et al. 2011; Khungurn et al. 2015], our approach does not rely on replicating (i.e., tiling) small pieces of fabrics. Thus, we produce realistic, non-repetitive details.

Full textiles. To describe full textiles, existing micro-appearance models usually take tens of megabytes (fiber-based) or several gigabytes (volumetric) of data. Our procedural model representation, on the other hand, is extremely compact: before instantiating fibers using Algorithm 1, only 22 numbers (see Table 2) need to be stored for each type of yarn. By combining these distributional parameters with a sparse set of yarn center curves, highly detailed textile models can be generated. This is demonstrated in Figure 17 where column (a) shows yarn curves that we use to populate fiber-level structures.[2] Without these small-scale details, the renderings offer limited quality. With added micro-geometry, on the other hand, the textiles appear much more realistic (b). The model constructed by our approach is fully procedural, offering easy editability. This is demonstrated in column (c) of Figure 17. For the silk scene, in particular, we adjusted fiber twisting so that the fibers are better aligned. Such a fiber-level modification not only results in significantly stronger anisotropic highlights, but also causes the fabric to become less transparent (which one can tell from the darker shadow on the ground). This is because when the fibers are better aligned, they tend to block light more effectively by having fewer 'gaps'. The second example in Figure 17 shows how changing the flyaway distribution affects overall appearance of a fabric globally. By increasing the density of flyaway fibers, the entire cloth becomes significantly fuzzier, resulting in less structured highlights but more small random ones or 'glints'. The last result in Figure 17 is a homogeneous basket weave. We use the procedural geometry of "Polyester 1" with altered optical parameters. When making the shape of ply cross-sections more elliptical (c), the underlying fiber structure changes dramatically, which leads to reduced overall glossiness. This further demonstrates the versatility of our approach and the significance of precise fitting of model parameters.

Limitations and future work. Our fitting pipeline requires solving several non-convex optimization problems (§4.4). In the future, advanced solution techniques could be explored for better numerical accuracy and performance. Currently, rendering textiles with yarns represented by our model requires realizing all fibers (using Algorithm 1). We intend to develop efficient rendering algorithms that directly work on our procedural geometry. In addition, our yarn model makes some simplifying assumptions:

- Ply cross-sections are assumed to be elliptical and spatially invariant, causing small geometric mismatches for Rayon 3, Silk 1, and Silk 2 in Figure 15-ab;
- Flyaway fibers have relatively simplified shapes (discussed in §5.1).

Thus, further generalizations of the yarn model and developing corresponding fitting techniques will be valuable. Finally, the level of geometric variance our method can capture is limited by the maximal sample size a micro CT scanner can take. Thus, large yarn-level variations that are difficult to measure are missing from our model.

7 Conclusion

Accurately modeling the appearance of fabrics is important for many design, prototyping, and entertainment applications. To accurately match reality, prior micro-appearance models either rely on data-intensive data sets or require tricky, manual parameter tweaking. We introduce a technique which bridges this gap, and thus enjoys advantages of both worlds while avoiding their weaknesses. Our method automatically fits procedural yarn models to accurate physical measurements obtained using volume imaging. The resulting models, which involve only 22 floating numbers for each yarn type, can be used to synthesize large textiles with high visual quality and without repetitive patterns. The statistical meaningfulness of these models also allow intuitive editing.

Acknowledgments

We are grateful to Pramook Khungurn for sharing his fiber extraction tool (§4.1). We thank Kai Schröder for helpful discussions and for sharing his implementation of procedural yarn generation (Algorithm 1). Funding for this work was provided by the National Science Foundation (IIS 1513967).

References

Cirio, G., Lopez-Moreno, J., Miraut, D., and Otaduy, M. A. 2014. Yarn-level simulation of woven cloth. *ACM Trans. Graph. 33*, 6, 207:1–207:11.

[2] We generated the yarn curves by uniformly sampling in the UV space of a given base mesh, and shifting the vertices along local normal directions following the weave pattern. In the future, one can leverage yarn-based simulation methods [Kaldor et al. 2008; Cirio et al. 2014] to obtain more physically accurate curves.

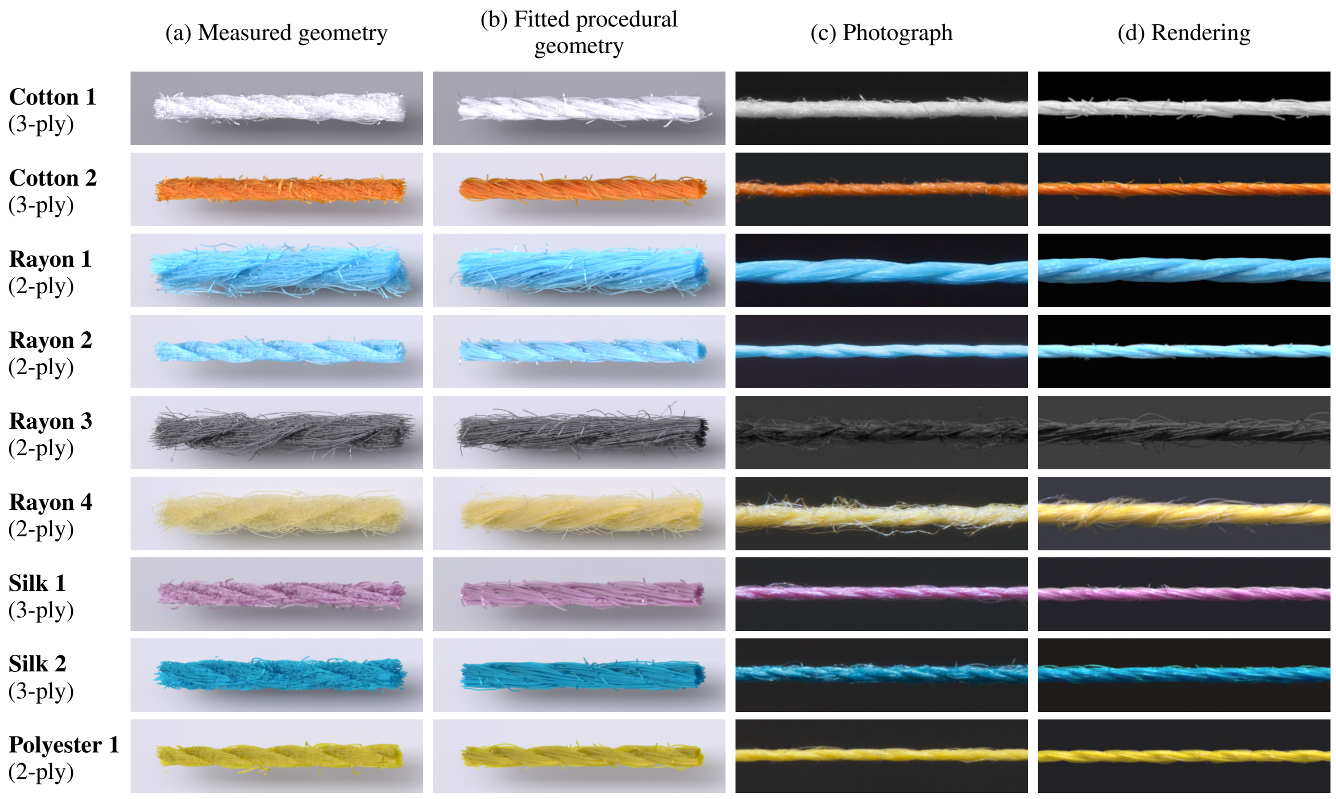


Figure 15: Parameter fitting results. *(a) CT-measured yarn geometries. (b) Procedural yarn geometries fitted using our approach. (c) Photographs of real-world yarns. (d) Full renderings of our procedural yarns under similar lighting/viewing conditions. Our fitted parameter values are shown in Table 2.*

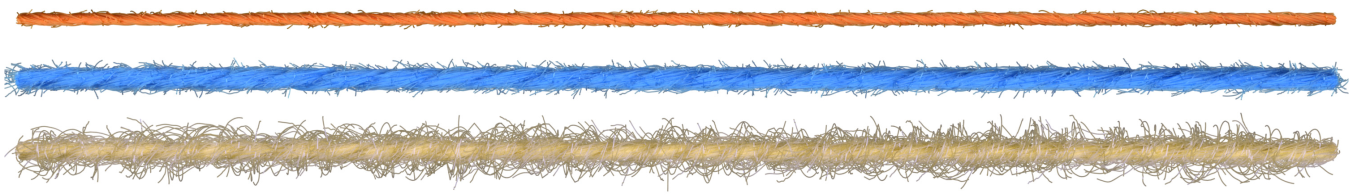


Figure 16: Long fabric yarns *generated procedurally with our fitted parameters. From top to bottom are "Cotton 2" (original parameters), "Rayon 1" (original parameters), and "Rayon 4" (increased hairiness). Unlike many prior micro-appearance modeling techniques, our model does not rely on tiling and thus can offer high-quality, non-repetitive yarn geometries.*

DEROUET-JOURDAN, A., BERTAILS-DESCOUBES, F., AND THOLLOT, J. 2013. Floating tangents for approximating spatial curves with G1 piecewise helices. *Computer Aided Geometric Design 30*, 5, 490–520.

HUTT, M., 2011. C++ implementation of the Nelder-Mead simplex method. http://www.mikehutt.com/neldermead.html.

IRAWAN, P., AND MARSCHNER, S. 2012. Specular reflection from woven cloth. *ACM Trans. Graph. 31*, 1, 11:1–11:20.

JAKOB, W., ARBREE, A., MOON, J. T., BALA, K., AND MARSCHNER, S. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph. 29*, 4, 53:1–53:13.

JAKOB, W., 2013. Mitsuba renderer. http://www.mitsuba-renderer.org.

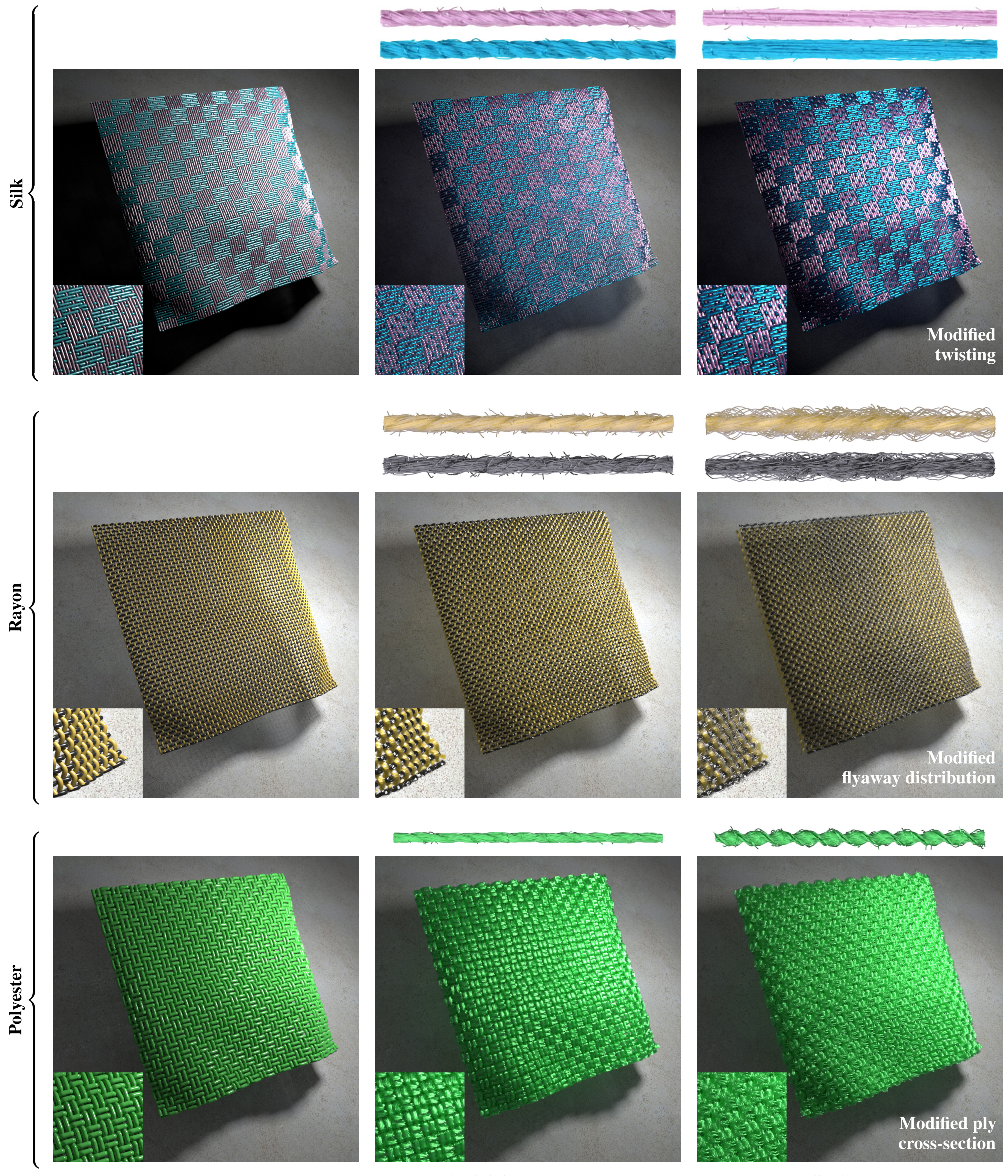
KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27*, 3, 65:1–65:9.

KEEFE, M. 1994. Solid modeling applied to fibrous assemblies. Part I: Twisted yarns. *Journal of the Textile Institute 85*, 3, 338–349.

KHUNGURN, P., SCHROEDER, D., ZHAO, S., BALA, K., AND MARSCHNER, S. 2015. Matching real fabrics with micro-appearance models. *ACM Trans. Graph. 35*, 1, 1:1–1:26.

MORRIS, P., MERKIN, J., AND RENNELL, R. 1999. Modelling of yarn properties from fibre properties. *Journal of the Textile Institute. Part 1, Fibre science and textile technology 90*, 3, 322–335.

NELDER, J. A., AND MEAD, R. 1965. A simplex method for function minimization. *The computer journal 7*, 4, 308–313.



(a) Yarn curves only (b) Original parameters (c) Edited parameters

Figure 17: Full textiles *with fiber-level geometries added procedurally using our method. The original parameters used in the three rows are: "Silk 1" + "Silk 2" (first), "Rayon 3" + "Rayon 4" (second) and "Polyester 1" (third). The edited silk example is obtained by multiplying ply and fiber twisting (i.e., α^{ply} and α) by 10×. The modified rayon model results from tripling flyaway densities ρ^{loop} and ρ^{hair} as well as doubling $R^{loop}_{\max}$ and R^{hair}_{span}. The altered polyester example is created by halving e_Y (the short axis of the ellipse) while doubling e_X (the long axis).*

Yarn ID	**Fiber-level parameters**					**Ply-level parameters**					**Flyaway fiber distribution**						
	Distrb & twisting			**Mig.**		**Cross sec.**		**Twisting**		**Fiber #**							
	β	ϵ	α	$R_{\min}$	s	e_X	e_Y	R^{ply}	α^{ply}	m	ρ^{loop}	ρ^{hair}	$R^{\text{loop}}_{\max}$	$R^{\text{hair}}_{\min}$	$R^{\text{hair}}_{\text{span}}$	$z^{\text{hair}}_{\text{span}}$	$\theta^{\text{hair}}_{\text{span}}$
Cotton 1	0.148	0.000	-0.369	0.80	1.1	0.026	0.020	0.038	0.453	75	22.17	33.77	(0.024, 0.005)	(0.020, 0.006)	(0.016, 0.009)	(-0.003, 0.057)	(0.377, 0.326)
Cotton 2	0.430	0.033	-0.436	0.95	1.9	0.025	0.018	0.033	0.336	37	13.34	19.50	(0.026, 0.007)	(0.021, 0.005)	(0.014, 0.006)	(0.019, 0.048)	(0.480, 0.404)
Rayon 1	0.682	0.139	0.710	0.85	1.9	0.053	0.036	0.051	-0.639	150	62.40	52.52	(0.050, 0.009)	(0.041, 0.007)	(0.020, 0.018)	(0.016, 0.059)	(0.403, 0.437)
Rayon 2	0.293	0.052	-0.381	0.85	1.6	0.028	0.021	0.029	0.372	80	39.08	50.39	(0.026, 0.006)	(0.018, 0.005)	(0.012, 0.007)	(0.001, 0.025)	(0.439, 0.498)
Rayon 3	0.640	0.000	0.609	0.95	1.9	0.044	0.034	0.042	-0.638	143	44.51	53.72	(0.041, 0.015)	(0.030, 0.005)	(0.028, 0.017)	(-0.014, 0.110)	(0.821, 0.707)
Rayon 4	0.671	0.361	0.560	0.70	1.7	0.048	0.034	0.045	-0.498	122	59.40	65.23	(0.043, 0.008)	(0.038, 0.007)	(0.021, 0.016)	(0.009, 0.048)	(0.326, 0.287)
Silk 1	0.462	0.000	-0.458	0.95	1.8	0.025	0.022	0.038	0.526	33	20.32	23.82	(0.024, 0.004)	(0.021, 0.004)	(0.014, 0.011)	(0.010, 0.050)	(0.407, 0.390)
Silk 2	0.554	0.405	-0.556	0.70	1.2	0.031	0.025	0.045	0.702	47	18.89	23.09	(0.029, 0.004)	(0.024, 0.005)	(0.010, 0.008)	(0.008, 0.038)	(0.417, 0.446)
Polyester 1	0.258	0.202	-0.379	0.75	1.5	0.029	0.021	0.029	0.370	60	35.69	30.44	(0.027, 0.004)	(0.021, 0.005)	(0.011, 0.009)	(-0.003, 0.036)	(0.356, 0.400)

Table 2: Our fitted parameter values *for all results shown in Figure 15. Positive and negative α values indicate counter-clockwise and clockwise (ply/fiber) twisting, respectively. We assume, without loss of generality, that $R_{\max} = 1$. For flyaway distribution parameters, paired numbers in parentheses indicate the mean and standard deviation of the corresponding normal distributions. When generating yarns using Algorithm 1, all parameters not mentioned here (e.g., R_i, θ_i, $\theta_i^{(0)}$) are drawn uniformly from their domains.*

Yarn ID	**Colors**		**Long. & azi. widths**		
	C_{R}	C_{TT}	β_{R}	β_{TT}	γ_{TT}
Cotton 1	(0.90, 0.90, 0.93)	(0.90, 0.90, 0.93)	1.0	27.0	38.0
Cotton 2	(0.10, 0.10, 0.05)	(0.93, 0.53, 0.01)	1.0	27.0	38.0
Rayon 1	(0.15, 0.25, 0.25)	(0.45, 0.82, 0.95)	1.2	10.0	26.0
Rayon 2	(0.25, 0.25, 0.30)	(0.45, 0.86, 0.95)	1.2	10.0	26.0
Rayon 3	(0.20, 0.20, 0.20)	(0.05, 0.05, 0.05)	1.0	10.0	26.0
Rayon 4	(0.05, 0.05, 0.10)	(0.95, 0.91, 0.60)	1.5	10.0	26.0
Silk 1	(0.10, 0.10, 0.10)	(0.73, 0.50, 0.67)	1.0	10.0	20.0
Silk 2	(0.10, 0.10, 0.10)	(0.05, 0.55, 0.70)	1.0	10.0	20.0
Polyester 1	(0.10, 0.10, 0.05)	(0.88, 0.83, 0.01)	4.0	10.0	20.0

Table 3: Fiber scattering parameters *using the model introduced by Khungurn et al. [2015] for generating all renderings in Figure 15. We manually adjusted the parameter values since finding optimal ones automatically is orthogonal to our current work. For better appearance matching, we recommended using advanced inverse rendering techniques such as [Khungurn et al. 2015].*

POINTCARRÉ, 2016. Pointcarré textile software. http://www.pointcarre.com.

SADEGHI, I., BISKER, O., DE DEKEN, J., AND JENSEN, H. W. 2013. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph. 32*, 2, 14:1–14:12.

SCHRÖDER, K., ZINKE, A., AND KLEIN, R. 2015. Image-based reverse engineering and visual prototyping of woven cloth. *Visualization and Computer Graphics, IEEE Transactions on 21*, 2, 188–200.

SHINOHARA, T., TAKAYAMA, J.-Y., OHYAMA, S., AND KOBAYASHI, A. 2010. Extraction of yarn positional information from a three-dimensional CT Image of textile fabric using yarn tracing with a filament model for structure analysis. *Textile Research Journal 80*, 7, 623–630.

TAO, X. 1996. Mechanical properties of a migrating fiber. *Textile research journal 66*, 12, 754–762.

VOBOROVA, J., GARG, A., NECKAR, B., AND IBRAHIM, S. 2004. Yarn properties measurement: an optical approach. In *2nd International textile, clothing and design conference*, 1–6.

XU, Y.-Q., CHEN, Y., LIN, S., ZHONG, H., WU, E., GUO, B., AND SHUM, H.-Y. 2001. Photorealistic rendering of knitwear using the lumislice. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 391–398.

YUKSEL, C., KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph. 31*, 3, 37:1–37:12.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph. 30*, 4, 44:1–44:10.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2012. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph. 31*, 4, 75:1–75:10.

ZHAO, S., HAŠAN, M., RAMAMOORTHI, R., AND BALA, K. 2013. Modular flux transfer: Efficient rendering of high-resolution volumes with repeated structures. *ACM Trans. Graph. 32*, 4, 131:1–131:12.