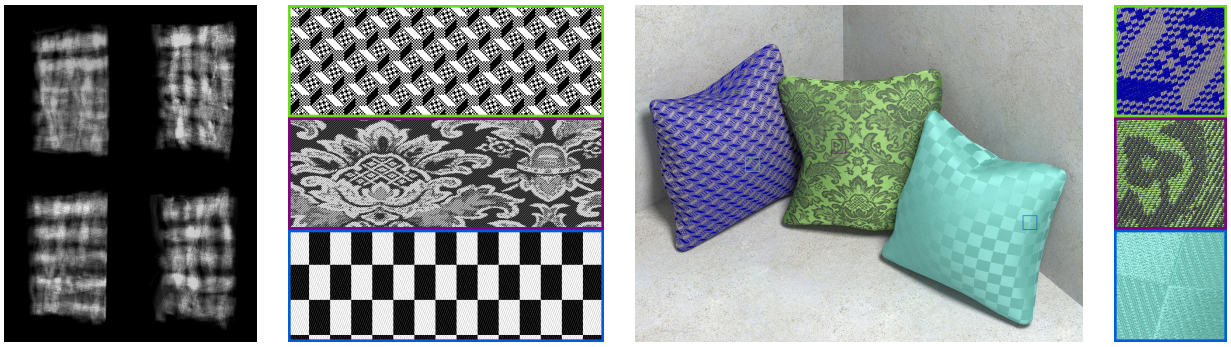
Appears in the SIGGRAPH 2012 Proceedings.

Structure-aware Synthesis for Predictive Woven Fabric Appearance

Shuang Zhao Wenzel Jakob Steve Marschner Kavita Bala
Cornell University*



(a) (b) (c) (d)

Figure 1: *We synthesize volumetric appearance models of fabrics with complex designs using a small set of exemplars: (a) density information of exemplars obtained using micro CT imaging; (b) fabric designs specified by weave patterns; (c) rendered results using synthesized volume data; (d) insets showing details: see, for example, blue yarns (top inset) hidden beneath the gray ones that are visible through the gaps.*

Abstract

Woven fabrics have a wide range of appearance determined by their small-scale 3D structure. Accurately modeling this structural detail can produce highly realistic renderings of fabrics and is critical for predictive rendering of fabric appearance. But building these yarn-level volumetric models is challenging. Procedural techniques are manually intensive, and fail to capture the naturally arising irregularities which contribute significantly to the overall appearance of cloth. Techniques that acquire the detailed 3D structure of real fabric samples are constrained only to model the scanned samples and cannot represent different fabric designs.

This paper presents a new approach to creating volumetric models of woven cloth, which starts with user-specified fabric designs and produces models that correctly capture the yarn-level structural details of cloth. We create a small database of volumetric exemplars by scanning fabric samples with simple weave structures. To build an output model, our method synthesizes a new volume by copying data from the exemplars at each yarn crossing to match a weave pattern that specifies the desired output structure. Our results demonstrate that our approach generalizes well to complex designs and can produce highly realistic results at both large and small scales.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

Keywords: appearance modeling, volume rendering, textiles

*E-mail: {szhao, wenzel, srm, kb}@cs.cornell.edu

Links: DL PDF WEB

1 Introduction

Woven fabrics are common in everyday life and display highly varied appearance, with very fine detail and subtle directional effects that are created by the interplay of geometric structure with fiber properties. Capturing these effects in predictive renderings for arbitrary woven fabrics is a major challenge.

Realistic cloth is important for graphics applications from entertainment to apparel rendering, and it is also important in textile design. Textile designers use software such as Pointcarré [2001] to design weave patterns for fabrics, and then drive industrial looms using the output. However, these packages do not provide realistic previsualization of the design before fabrication, thus forcing designers to fabricate "in the dark." Since loom time can be expensive and difficult to schedule, refining a design requires slow and costly iteration. By providing the ability to predictively preview the appearance of fabric designs before they are woven, we can minimize the need for test weaving, saving considerable time, raw materials, and cost.

Volume modeling and rendering techniques [Kajiya and Kay 1989; Perlin and Hoffert 1989] have recently been quite successful in capturing the diverse appearance of fabrics [Xu et al. 2001; Jakob et al. 2010; Zhao et al. 2011]. In particular, Zhao et al. [2011] built volumetric models for fabrics using yarn and fiber geometry information from micro computed tomography (CT) imaging and optical information from a photograph. For materials with simple repeating structure and a single type of yarn, they tiled the volume data to produce large areas of fabric, with highly realistic results. The key to this approach is to accurately model the geometric structure of the surface layer of the cloth, from which the many appearance phenomena of different fabric types emerge automatically. However, this method is limited to materials containing only a single type of yarn, and it can only reproduce the exact material that was scanned. But real fabrics are complex—including intricate weave patterns, large scale designs, and multiple yarn types for warp and weft, each with its own reflectance; ideally, flexibility to render such fabrics is desired. Further, in design applications, the real usefulness of rendering comes from predicting the appearance of new fabrics that have not been scanned.

This paper presents a new technique to create volumetric models of fabrics with complex, spatially varying structures and to predict the appearance of specific weave patterns. A small set of exemplars obtained from micro CT scans of simple fabrics are used to model new fabrics defined by 2D binary images representing each fabric's weave structure. Our structure-aware volume synthesis algorithm efficiently copies regions of the exemplars to assemble a volumetric model that matches the fabric's structure, without visible seams or periodic patterns.

We demonstrate our technique with synthesized results for a range of common structures that convey very different appearance. In some cases we use real weave patterns and compare to photographs of samples woven from the same patterns, and in other cases we use patterns generated to achieve a particular rendered appearance. This work can have impact on graphics applications in entertainment, e-commerce and apparel visualization, and textile design.

2 Related Work

2.1 Cloth modeling and rendering

Several methods producing procedural textures with BRDFs [Gröller et al. 1996; Adabala and Magnenat-Thalmann 2003; Adabala et al. 2003; Irawan and Marschner 2012] have been developed to simulate cloth with a variety of weave patterns. Drago and Chiba [2004] proposed a method to procedurally model different kinds of woven canvases using spline surfaces. Like ours, these methods account for the cloth's weave pattern and produce texture at scales where yarns are visible, but the level of realism produced is less than with the detailed volume models we use.

In addition, many techniques have been developed for acquiring and modeling spatially varying BRDFs [Marschner et al. 2005; Wang et al. 2008; Ghosh et al. 2009; Dong et al. 2010; Ghosh et al. 2010]. In particular, Wang et al. [2008] and Dong et al. [2010] both use BRDFs based on tabulated normal distributions to represent a variety of materials including a Jacquard silk satin. These models do an excellent job capturing the spatially varying anisotropy of the material, but resolution is limited to that of the photos used for capture. Also, only captured fabrics can be represented; they are not intended for rendering new fabrics.

Gröller et al. [1995] proposed a volumetric approach for modeling knitwear. Xu et al. [2001] introduced *lumislice* rendering, a procedural volume modeling method, to synthesize realistic close-ups. These methods addressed knits, rather than wovens, and did not consider fabrics with complex designs. Also, using a procedural model with only isotropic scattering limited the realism and range of materials that could be handled.

Commercial textile design software packages, such as Pointcarré [2001], include proprietary visualization tools, but these are generally 2D and mainly limited to predicting the overall color of the material, not its complete appearance.

Jakob et al. [2010] proposed a general framework for simulating light scattering within anisotropic media including cloth, along with the *microflake* model for phase functions in such media. Using this model, a piece of cloth can be represented with a volume in which each voxel contains four parameters: material density, local fiber orientation, single-scattering albedo, and fiber alignment. The first two parameters describe the geometric structure; the last two provide optical properties of the yarns.

Zhao et al. [2011] developed an approach that builds microflake models for fabrics using micro CT imaging. We build upon their method, using similar scans of a small set of exemplars, rather than just one fabric, and provide a mechanism to produce models of a wide variety of fabrics from this input.

2.2 Synthesis

Example-based texturing. There is a large body of work in the field of texture synthesis. For a comprehensive survey, see [Wei et al. 2009]. Example-based texture synthesis techniques create a large output texture using small exemplars. Those algorithms can be performed based on pixels [Heeger and Bergen 1995; Efros and Leung 1999; Wei and Levoy 2000; Lefebvre and Hoppe 2005] or patches [Cohen et al. 2003; Efros and Freeman 2001; Kwatra et al. 2003; Wu and Yu 2004] and can also synthesize solid textures [Kopf et al. 2007]. Some texture synthesis algorithms take additional constraints [Ashikhmin 2001; Kwatra et al. 2005; Ramanarayanan and Bala 2007], but the forms of such constraints are quite different from the one in our problem. Many approaches including [Ashikhmin 2001] also aim at preserving continuity across synthesized pixels. But the optimizations are normally performed locally and do not ensure global continuities.

Synthesizing appearance/geometry. Several approaches synthesize polyhedral meshes [Merrell and Manocha 2008], voxelized volumes [Zhou et al. 2006], and appearance models [Tong et al. 2002; Chen et al. 2004] onto arbitrary surfaces. However, all these methods focus on a very different problem: extending complex exemplars over a non-trivial domain. The technique proposed by Zhou et al. [2006], in particular, includes a deformation step to solve a similar problem as the one introduced in Section 5.5. Unfortunately, this method needs to be performed at the voxel level and thus does not scale to the size of our problem.

Discrete element synthesis. Recently, techniques that fill a volume with a set of discrete elements have been proposed [Hurtut et al. 2009; Ma et al. 2011]. Like texture synthesis, these methods usually do not support constraints, or they take constraints that are quite different from ours. And as the name suggests, the basic elements in these methods are "disconnected" from each other, which is not the case in our problem.

3 Background

Weaving is a process of interlacing two perpendicular sets of yarns, called the *warp* and the *weft*, to form a fabric. During the weaving process, warp yarns are fixed to the loom while weft yarns are inserted crossways, and different subsets of warp yarns are raised above or lowered below each inserted weft yarn so that the yarns become interlaced and the fabric holds together into a sheet by friction. In this paper, we follow the convention that warps go vertically and wefts horizontally.

Depending on the pattern in which warps are raised, fabrics with very different appearance and mechanical properties are produced. The pattern is described very simply using a binary image called a *weave pattern*, with the number of columns and rows equal to the number of warps and wefts in the cloth; a black pixel means the warp is above the weft at the corresponding yarn crossing, while white means it is below. Depending on the mechanics of the loom, only certain kinds of patterns may be achievable, but in the most general case of Jacquard looms, every yarn crossing is individually controlled by a computer. Figure 2 shows four different weave patterns from the *twill* and *satin* families that we use as exemplars.

Twill is one of the most common weave patterns, in which each row is shifted by one yarn from the previous row. As shown in the top row of Figure 2, fabrics created with twills convey characteristic diagonal lines. Simple twills are denoted "m/n twill" meaning a

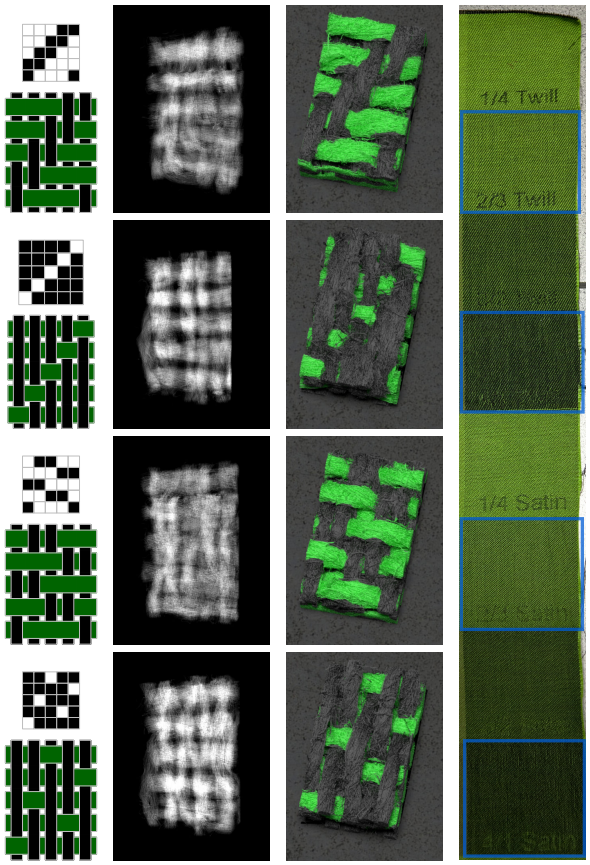


Figure 2: *Example weave patterns: twill (top two rows), satin (bottom two rows); (a) weave patterns and the corresponding 2D illustrations where warps and wefts are respectively drawn in black and green; (b) CT data of fabric samples with the same weave patterns; (c) colored visualizations of the CT data; (d) a photograph of our example fabric in which the four examples used in this figure are marked with blue rectangles.*

warp goes over m wefts, then under n wefts, then repeats. A twill pattern that repeats every k yarns is called a "k-end twill" or just "k-twill." The two 5-twill patterns shown in Figure 2 are 2/3 twill and 4/1 twill.

As opposed to twill, satin weaves shift each row by more than one yarn, with the aim of creating a more distributed pattern that does not call attention to the repeating structure. Satins build smooth surfaces and can be used for creating fabrics with glossy appearance. Satins are named in the same way as twills; the bottom rows of Figure 2 show a 2/3 and a 4/1 satin. Much larger satin patterns are often used with fine yarns when a glossy surface is desired.

In Jacquard fabrics with many-colored graphic patterns, the variety of available colors can be increased by using a multi-layer weave, so that some yarns can be hidden at the back of the cloth in areas where their color is not desired on the surface. The structures of double- and triple-cloth fabrics can be intricate, but for our purposes we are primarily interested in the yarns visible on the surface. Therefore in this work we treat cloth as if it were single-layered, with yarns that can change color along their length. In reality, of course, yarns do not change color but rather are substituted with other yarns that were previously hidden on the back, but the errors induced by this simplification are subtle.

4 Overview

The goal of our cloth modeling process is to produce volume models of woven materials, suitable for realistic close-up renderings, from two inputs: a description of the material to be simulated, and a few examples of similar but simpler fabrics. Our system accomplishes this in two phases. In the first phase, which only needs to be done once for a whole class of materials, CT scans of the example fabrics are used to build *exemplars* that contain all the information needed to synthesize large areas of complex fabrics. In the second phase, which is done once per material to be simulated, the exemplars are used in a new structure-aware volumetric texture synthesis method to synthesize a volume model according to the colors and weave pattern of the target material.

Exemplar creation phase. The purpose of the exemplar creation phase is to turn raw volume data into a database that can be used to synthesize volume models of a range of fabrics that are made from materials similar to the example materials, but have different structure. Normally the example materials are a set of simple weaves made using particular types of yarns.

As input we assume volume data showing the geometric structure of the input fabric. While other data sources, such as magnetic resonance imaging (MRI), could be used, in this paper we focus on volume datasets that come from CT scans. Thus, our input is the raw volume containing density information on a fine voxel grid covering a small patch of a fabric. High resolution scans are required to resolve fiber orientation and flyaway fibers, so each scan observes an area on the order of 5mm across, which, after cropping, typically produces exemplars with about 6×9 yarn crossings.

A processing pipeline takes this data and produces output by denoising the input density data, automatically tracking yarns in the data to detect the yarn trajectories, segmenting the voxels to match them to the appropriate yarns, and then automatically detecting the pattern of yarn crossings.

Each exemplar in the resulting database includes a voxel grid (containing density, fiber orientation, and yarn ID) and a small binary image representing the weave pattern. Section 6 gives details.

Synthesis phase. The input describing a new fabric to be simulated includes a 2D binary array giving the weave pattern for the whole cloth, and 2D arrays specifying the type of warp and weft yarn present at each yarn crossing. The synthesis phase, detailed in Section 5, creates an output volume that respects the input specification while displaying local structure and details that match the exemplars.

How the fabric specification will be created depends on the application. When predicting the appearance of a new fabric as part of the textile design process, this description can be extracted directly from the actual design, using the data that would be sent to the loom to make the fabric. In a graphics context, the weave and color pattern can be computed from a posterized image by a very simple process, since the constraints of actual weaving do not need to be observed, as described in Section 5.2.

5 Structure-aware Synthesis

Given appropriate exemplars, structure-aware synthesis produces a detailed model of a fabric from a description of the required design.

5.1 Input Specification

The input to our algorithm (Figure 3) consists of three components:

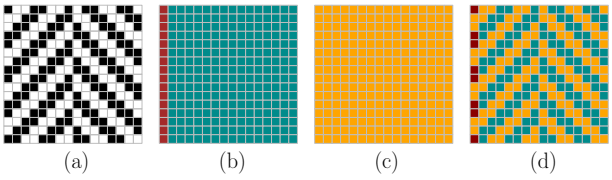


Figure 3: *Inputs to our algorithm: (a) shows the weave pattern; (b) and (c) show warp and weft ID maps encoded in colors, indicating that all but the left-most warp share the same optical properties while all wefts are identical; (d) illustrates the visible yarn ID at each crossing.*

1. A binary image W representing the weave pattern, where each binary value represents a yarn crossing.
2. A 2D array specifying the warp and weft types, represented with IDs, for every yarn crossing.
3. Color and gloss information for each type of yarn.

As discussed in Section 3, we model multilayer weaves by allowing a single warp or weft to change color along its length. This also provides flexibility in graphics applications where the cloth need not be manufacturable, because the color constraints of the actual weaving process can be discarded if desired. For this reason the yarn color arrays are two-dimensional rather than one-dimensional. In reality, there are usually no more than ten different kinds of yarns in a fabric.

5.2 Input Data Creation

The input above can be created in several ways. A textile designer would normally use design software such as Pointcarré [2001] to design a new material, and the software can simply output the required binary weave pattern and yarn color information.

For applications where constraints of producing actual cloth are less of a concern, the following simple method mimics the design process. Begin with a posterized image I and a set of q different elementary weave patterns $V_1, V_2, \ldots, V_q$ (repeated until they match the size of I) associated with warp and weft ID maps. Note that the set of elementary patterns can be arbitrary and does not need to be contained in our exemplar database. Let the set of discrete colors in I be $\mathcal{P}$. The user then assigns a weave pattern to each of the colors by specifying a mapping $p : \mathcal{P} \mapsto \{1, 2, \ldots, q\}$. To produce the patterned cloth, take each pixel from that of one of the elementary weave patterns as follows: $W(x, y) = V_{p(I(x,y))}(x, y)$.

5.3 Synthesis at the Yarn Level: The Problem

The goal of our synthesis process is to generate a large volume matching the given weave pattern. At the *voxel level*, the problem is to solve for the value of each voxel of output. Since the total number of voxels can be very large (a $1\text{m} \times 1\text{m} \times 2\text{mm}$ cloth sampled at $5\mu\text{m}$ resolution has 1.6×10^{13} voxels), computing or even storing the solution is costly and must be avoided. At the *yarn level*, the algorithm instead considers one pixel in the weave pattern (which represents a warp-weft yarn crossing) at a time, and "copies" the corresponding volume data, which we call a *block*, by referencing a rectangular box in an exemplar volume. This is much more tractable, so given the costs involved, we solve the problem at the yarn level, and then apply a post-process to effectively adjust the data at the voxel level to get a high quality synthesized result.

The core problem of the yarn-level synthesis process is to locate a block in the exemplar volumes for each pixel in the weave pattern to copy its volume contents from. Let A be a weave pattern

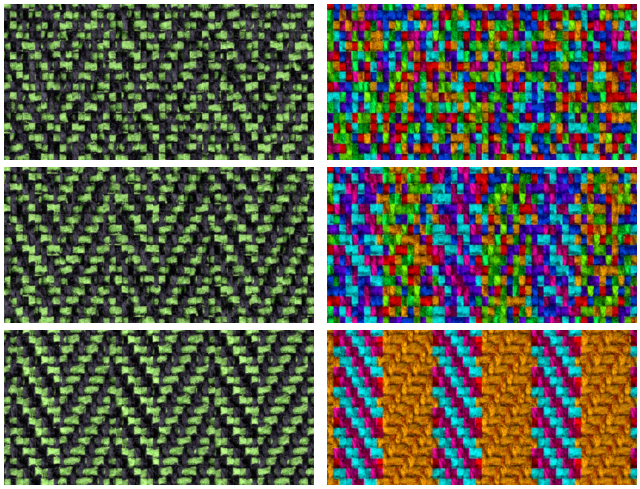


Figure 4: *Synthesized results using (top) naive algorithm, (middle) greedy algorithm, (bottom) our approach: the left column shows renderings using synthesized models; the right column shows from which exemplar each block copies its content (encoded in false colors).*

associated with a volume $\tilde{A}$, and let $\tilde{A}(i, j)$ denote the block in $\tilde{A}$ corresponding to pixel $A(i, j)$. Then the yarn-level synthesis problem can be formulated as follows: given a set of k example weave patterns $\{S_1, S_2, \ldots, S_k\}$ associated with k exemplar volumes and a target pattern W, determine an *assignment function* $c : \mathbb{N}^2 \mapsto \mathbb{N}^3$ where $c(i, j) = (u, x, y)$ indicates that $\tilde{W}(i, j)$ copies its data from $\tilde{S}_u(x, y)$. Note that c can be implemented simply as a 2D array.

One possibility is to randomly copy blocks that have the desired yarn (warp or weft) on the top, by assigning $c(i, j)$ a random triple (u, x, y) under the constraint that $W(i, j) = S_u(x, y)$. Unfortunately, this works poorly, as shown in the top row of Figure 4, since there is no consistency across block boundaries. The shape of the yarn passing through a given block is affected strongly by whether it passes under or over the next yarn, and for this reason it is critical to ensure that the binary values of the four neighboring pixels match when selecting an exemplar block to copy.

Thus our method follows three principles when selecting an exemplar block for each output block, enforcing them in priority order:

1. *Correctness.* The correct yarn (warp or weft) must be on top.
2. *Consistency.* The four neighbors in the desired weave pattern should match the neighbors in the exemplar's weave pattern.
3. *Continuity.* Choices that copy neighboring blocks in the exemplar into neighboring blocks in the output are preferred.

Next we define the term *consistency*. For every element $(i, j) \rightarrow (u, x, y)$ of c, which we call an *assignment*, consider the four neighbors of $W(i, j)$ and $S_u(x, y)$. If all these neighbors match in binary values, we say the mapping is *consistent*. Unfortunately, it is not always possible to find c such that all assignments are consistent. Thus the problem can be described as an optimization: find an assignment function such that the total number of matches is maximized. Figure 5 shows an example where the block in $\tilde{S}_3$ maximizes the number of matching neighbors.

Note that to maximize the total number of matches, the choice for one assignment is independent of that for another. This fact suggests a greedy algorithm: for each pixel $W(i, j)$, select the triple with the maximum number of matches. Although this simple algorithm can generate much better results (see the middle row of

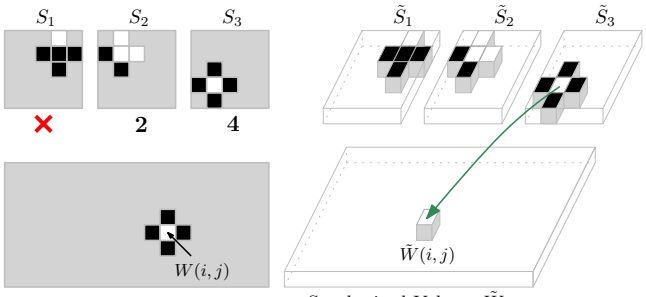


Figure 5: *The block in $\tilde{S}_1$ is not a valid candidate since it does not satisfy the correctness constraint; the blocks in $\tilde{S}_2$ and $\tilde{S}_3$ satisfy the constraint and respectively have 2 and 4 matching neighbors.*

Figure 4), it does not provide local continuity since the algorithm does not know how to break a tie when there are multiple candidates with the same number of matches. This is unfortunately a very common situation over uniform regions of a fabric.

Given an assignment function c, let the *continuity* at (i, j) be the total number of immediate neighbors (i', j') satisfying the following conditions: let $c(i, j) = (u_0, x_0, y_0)$ and $c(i', j') = (u_1, x_1, y_1)$, then $u_0 = u_1$ and

$$(x_1, y_1) - (x_0, y_0) = (i', j') - (i, j).$$

Our problem is to find an assignment maximizing the total consistency, using the total continuity to break any ties.

5.4 Synthesis at the Yarn Level: Our Algorithm

While matching consistency can be done using a greedy algorithm, maximizing the total continuity on a 2D grid is in general a very hard combinatorial optimization problem. Fortunately, the 1D version of this optimization problem, where continuity is defined by considering the two immediate neighbors for each yarn crossing, can be solved efficiently. And our experiments indicate that solving the 1D problem for every column (along the warps) is a good approximation of the 2D problem when handling weave patterns.

For convenience, we associate a unique index to each triple (u, x, y) used by the assignment function c. Beyond this point, we assume that $c(i, j)$ returns a single integer instead of a triple.

For a column y_0 in the weave pattern, let $f(i, t)$ denote the maximal total continuity for the first i rows in this column under the constraint that $c(i, y_0) = t$. And $f(i, t)$ is defined only when t maximizes consistency at $W(i, y_0)$. To compute $f(i, t)$, we can solve recursive sub-problems over the first $(i - 1)$ rows, namely computing $f(i - 1, t')$ for all feasible t' values, and then pick the best one to form $f(i, t)$ by computing

$$f(i, t) = \max_{t'} \{f(i - 1, t') + \text{gain}(t', t)\} \qquad (1)$$

where $\text{gain}(t', t)$ captures the 1D continuity and equals 1 if t' and t come from the same column of one exemplar volume and t' lies next to t and 0 otherwise. The base case of this recursion is $f(0, t) = 0$ for all t.

Note that the total number of states is polynomial, and this optimization problem can be solved efficiently using dynamic programming. Figure 6 illustrates the process of computing $f(i, t)$ by enumerating t' values.

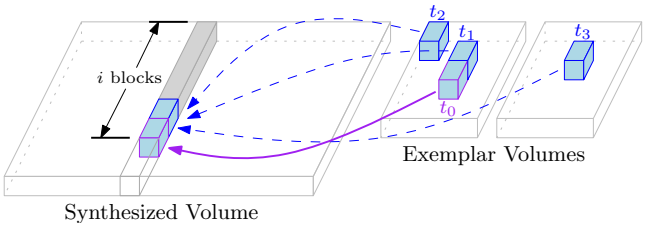


Figure 6: *The dynamic programming process: computing $f(i, t_0)$ by enumerating different possible t' values using Equation 1. For example, here we have $\text{gain}(t_1, t_0) = 1$ whereas $\text{gain}(t_2, t_0) = \text{gain}(t_3, t_0) = 0$.*

Algorithm complexity. Given a target weave pattern of size $M \times N$, and k example weave patterns each of size $m \times m$, the dynamic programming algorithm runs in $O(km^2MN)$ time. In our experiments, $m = 5$, $k = 8$, and M, N can be as large as several thousands.

Algorithm optimization. Many basic weave patterns are translationally symmetric: each column is a translated version of the previous one (with wrapping around the edges). It therefore suffices to consider just one column, reducing the time to $O(kmMN)$.

Randomization. The above optimization could result in a loss in variation by not considering the other $m - 1$ columns. Further, a side effect of maximizing local continuity is that it may create periodic patterns. To solve this problem, we randomly shift each block based on translational symmetries of the corresponding exemplar. And we shift every $\hat{m} \times \hat{m}$ block by the same amount to avoid destroying the continuity obtained by solving the optimization problem. Since $m = 5$ in our experiments, we have picked $\hat{m} = 3$.

Discussion. The 1D dynamic programming can be also performed along the weft direction or even alternating between the two directions. In our experiments, these schemes all produced very similar results. This is because the input weave pattern itself has strong 2D structure, and the 2D neighborhoods greatly restrict the set of candidate blocks at each yarn crossing. As shown in Figure 4, our algorithm provides good continuity in both directions.

5.5 Edge Fixing

Recall that each block represents one warp-weft yarn crossing. When two adjacent blocks copy their contents from disconnected blocks, visible seams may be created, as shown in Figure 9. We introduce an efficient method to fix the seams by shifting entire stacks of voxels along the Z direction.

Fixing a Single Block. Figure 7a illustrates the 2D case where both blocks contain a single yarn and there is a seam on the edge between them. Assume that the right end of yarn 1 and the left end of yarn 2 have depths h_1 and h_2, respectively. Then the seam can be fixed by shifting up yarn 1 by $\Delta h := (h_2 - h_1)/2$ on the right, and yarn 2 by $(-\Delta h)$ on the left.

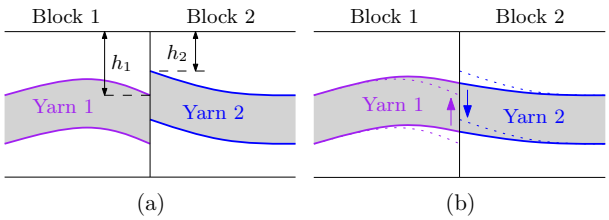


Figure 7: *Fixing the edges by moving stacks of voxels.*

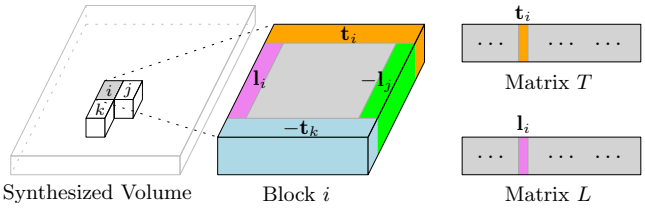


Figure 8: *Edge fixing: constructing matrices T and L. The structure of block i is shown in the middle, and assume that the blocks to its right and bottom are respectively block j and k.*

For the 3D case, the desired amount of shifting is defined along the 2D boundary of each block. However, only shifting the boundary stacks will create new seams. We also need to update the inner stacks so that the entire adjustment is smooth. We do this smooth adjustment of depths by solving a 2D discrete Poisson equation.

Note that we focus on matching the top-most yarns. In the case of a multi-layered fabric, this may cause errors (seams) for the yarns underneath. Fortunately, they can be safely ignored since we cannot see those yarns directly and the amount of shifting involved by this process is generally much smaller than the radius of a single yarn.

Fixing All Blocks. Although fixing one block is relatively easy, given that the total number of blocks is in millions, the time and storage needed to compute and store the solutions become prohibitive. However, the problem can be solved efficiently if all blocks have identical resolutions $b_1 \times b_2 \times b_3$, which is normally the case if the fabric samples are scanned using the same resolution.

For every block, say block i, let the boundary condition defined on its upper edge be $\mathbf{t}_i$ and that on its left edge be $\mathbf{l}_i$. Stack all $\mathbf{t}$ and $\mathbf{l}$ vectors as columns to form two matrices T and L with dimensions $b_1 \times MN$ and $(b_2 - 2) \times MN$, respectively. The boundary conditions defined on a block's right and bottom edges can be represented using those on its right neighbor's left edge and bottom neighbor's top edge, so they do not need to be stored. Figure 8 illustrates this process.

Next, we compute rank-r approximations for the matrices T and L: $T \approx B_T \times C_T$; $L \approx B_L \times C_L$ where B_T, B_L respectively have dimensions $b_1 \times r$, $(b_2 - 2) \times r$, while C_T, C_L are both $r \times MN$. In our experiments, r is picked manually and equals 15. Then we solve the $2r$ Discrete Poisson equations whose boundary conditions are given by each column of B_T and B_L (and setting all other edges to 0). We store the solutions $\mathbf{s}_t^T$ and $\mathbf{s}_t^L$ for $t = 1, 2, \ldots, r$. It follows that the solution of the Poisson equation defined on each block is (approximately) a linear combination of the stored values, since the solution of a Poisson equation is a linear function of the boundary condition.

For block i, assume its right and bottom neighbors are block j and k, respectively. Then the solution $\mathbf{s}_i$ at block i equals

$$\sum_{t=1}^{r} \left(C_T(t,i)\,\mathbf{s}_t^T + C_L(t,i)\,\mathbf{s}_t^L - C_T(t,k)\,\tilde{\mathbf{s}}_t^T - C_L(t,j)\,\tilde{\mathbf{s}}_t^L \right)$$

where $\tilde{\mathbf{s}}_t^T$ and $\tilde{\mathbf{s}}_t^L$ are the vertically and horizontally mirrored copies of $\mathbf{s}_t^T$ and $\mathbf{s}_t^L$, respectively.

In our implementation, we precompute C_T, C_L, $\mathbf{s}_t^T$, $\mathbf{s}_t^L$ and obtain $\mathbf{s}_i$ at runtime. In our experiments, storing this information takes roughly 150 MB of space. Finally, when the voxel contents at location $\mathbf{p} = (p_x, p_y, p_z)$ need to be fetched, we adjust p_z by $\mathbf{s}_i(p_x, p_y)$ before performing the volume lookup.

We have shown how to synthesize the volume data using several exemplars. Next we will provide the details on creating those exemplars.

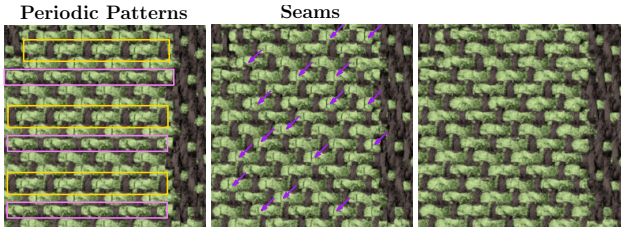


Figure 9: *Randomization and edge fixing: (left) maximizing consistency and continuity without randomization results in periodic patterns; (center) introducing randomization removes such patterns; (right) edge fixing significantly improves the seams.*

6 Exemplar Creation

In this section we describe our pipeline of CT image processing. The goal is to create a set of exemplar volumes in which each voxel contains three parameters: material density, local fiber orientation, and a yarn ID. The material density and fiber orientation can be passed directly to the microflake model [Jakob et al. 2010], and the ID can be used to tell the type of yarn (warp or weft) and then obtain the corresponding optical information from the input images. This information combined yields a complete volumetric appearance model which can then be rendered.

This stage starts with a basic processing step following [Zhao et al. 2011] which takes raw CT data and produces density and orientation information for each voxel. Next we compute per-voxel yarn IDs and regularize the exemplar volumes.

6.1 Yarn Tracking

To obtain the yarn ID for each voxel, we reconstruct the center curve, a discrete line strip, for each yarn in the volume; this process is called *yarn tracking*.

Tracking. Given a yarn passing 3D point $\mathbf{p}$, we can track it by iteratively computing the tangent direction $\mathbf{t}$ at the current location and moving in that direction by a small step d. The tangent direction $\mathbf{t}$ can be computed by averaging the local fiber orientation over a small region around $\mathbf{p}$:

$$\mathbf{t} = \text{normalize}\left(\sum_{\mathbf{v} \in V_{\mathbf{p},\mathbf{t}_0}} \omega_f(\mathbf{v}) \right)$$

where $\mathbf{t}_0$ is the tangent direction computed in the previous iteration, $\omega_f(\mathbf{v})$ is the local fiber orientation at $\mathbf{v}$, and $V_{\mathbf{p},\mathbf{t}_0}$ is a volume around $\mathbf{p}$ defined by

$$V_{\mathbf{p},\mathbf{t}_0} = \left\{ \mathbf{v} \in \mathbb{R}^3 \;:\; \|\mathbf{v} - \mathbf{p}\|_2 \le R \;\text{ and }\; |\omega_f(\mathbf{v}) \cdot \mathbf{t}_0| \ge c_0 \right\}$$

where d and c_0 are constants representing the size of a yarn (in voxels) and its degree of deformation. In our experiments, $d = 15$, $c_0 = 0.7$, and R respectively equals 20 and 30 when tracking warps and wefts.

If the input orientation field contains too much noise, the estimated $\mathbf{t}_0$ will be inaccurate, which may cause the tracking step to fail. We therefore need to ensure that the CT scans have not only sufficient resolution but also acceptable signal-to-noise ratio, which can be ensured by using longer exposures.

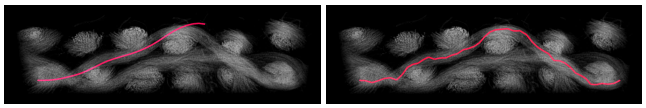


Figure 10: *Center correction: (left) without the correction, the tracking that starts from the left fails due to the yarn center leaving the volume; (right) with the correction, the tracking process becomes more robust.*

Endpoint Detection. To start the tracking process, an endpoint of every yarn is needed. We detect the endpoints automatically using K-means clustering (with the total number of clusters as user input).

We assume that the warps approximately go vertically (along the Y-axis) while the wefts run horizontally. This can be easily satisfied by roughly aligning the samples during the scanning process. Also, we assume that the user knows the number of warps and wefts in the scanned volume.

To detect the warp centers on a 2D slice perpendicular to the Y-axis, we perform a K-means clustering among those voxels on the slice whose fiber orientations are close enough to the Y direction; the center of each cluster indicates the warp centers. Similarly, the centers of the wefts can be detected by running the same algorithm for any slice perpendicular to the X-axis.

Note that the computed yarn centers can be unreliable: noise and voxel orientation errors can result in poor accuracy. We therefore run this process for every slice along each axis and pick the one which minimizes the maximal cluster radius.

Correction. Tracking a yarn by simply following the tangent direction causes $\mathbf{p}$ to leave the yarn in highly curved regions. Therefore, we introduce a correction step after each tracking iteration. Every time $\mathbf{p}$ is updated with $(\mathbf{p} + d \cdot \mathbf{t})$, we iteratively move $\mathbf{p}$ to the center of mass of a small volume around it:

$$\mathbf{p} \leftarrow \frac{\sum_{\mathbf{v} \in S_{\mathbf{p},\mathbf{t}}} \rho(\mathbf{v}) \cdot \mathbf{v}}{\sum_{\mathbf{v} \in S_{\mathbf{p},\mathbf{t}}} \rho(\mathbf{v})}$$

where $\rho(\mathbf{v})$ denotes the material density at $\mathbf{v}$, $S_{\mathbf{p},\mathbf{t}}$ is a 2D region surrounding $\mathbf{p}$ on the slice which is perpendicular to the main direction of the yarn and contains $\mathbf{p}$. In our experiments, we made $S_{\mathbf{p},\mathbf{t}}$ elliptical to provide more freedom for $\mathbf{p}$ to move along the Z-direction while preventing it from accidentally jumping to a neighboring yarn. As shown in Figure 10, the correction step significantly stabilizes the tracking process.

Discussion. Shinohara et al. [2010] proposed a similar yarn tracking approach. However, their method requires the user to enter the endpoint for each yarn and does not have the correction step which has proven crucial for tracking the yarns in our experiments.

6.2 Weave Pattern Detection

With the tracked yarns in hand, we would now like to infer their associated weave pattern. For each yarn crossing, this entails determining whether the weft passes above or below the warp. Mathematically, this property is captured by the *linking number* of the yarn curves [Rolfsen 2003]. Reminiscent of the winding number in two dimensions, the linking number counts the signed number of times a space curve wraps around another. Given parameterizations $\mathbf{p}_1(t_1)$ and $\mathbf{p}_2(t_2) : [0,1] \to \mathbb{R}^3$ of a warp and weft, respectively, we numerically compute their linking number using the Gaussian linking integral

$$L_{1\leftrightarrow 2} = \frac{1}{4\pi} \int_{[0,1]^2} \left\langle \frac{\mathbf{p}_1(t_1) - \mathbf{p}_2(t_2)}{\|\mathbf{p}_1(t_1) - \mathbf{p}_2(t_2)\|_2^3}, \frac{\partial \mathbf{p}_1}{\partial t_1} \times \frac{\partial \mathbf{p}_2}{\partial t_2} \right\rangle \mathrm{d}t_1 \mathrm{d}t_2$$

Assuming that the warps and wefts are parameterized so that the projection of their tangents into the plane of the weave pattern forms a positively oriented basis of $\mathbb{R}^2$, we assign the value 0 to this yarn crossing if $L_{1\leftrightarrow 2} > 0$ and 1 otherwise.

Using this technique we can automatically detect weave patterns for single-layered fabrics. For those with multiple layers, we must manually reason about the layered structure to find the top-most yarn for each weave grid location before computing the linking number.

6.3 Voxel Segmentation

Based on the tracked yarns, we can tell which yarn each voxel belongs to. This information is needed in the later steps of the pipeline. For each voxel, we assign it to the yarn whose center curve minimizes the distance to the voxel. Note that because the warp and weft yarns have different radii and stiffness, this simple approach can cause small errors at the yarn crossings. Those errors, however, are hardly visible in rendered results since they are hidden beneath the surface.

6.4 Volume Alignment

Since the samples are not perfectly registered during the scanning process, they need to be aligned before being used for synthesis.

We solve for a global rotation around the Z-axis for each scanned dataset. For the yarn i, let Y_i be the set of voxels it contains. We perform PCA on $\{(v_x, v_y) \ : \ \mathbf{v} = (v_x, v_y, v_z) \in Y_i\}$ to detect the principal direction of the yarn. Let ω_0 and ω_1 be the average principal directions of the warps and the wefts respectively. We rotate the whole volume around the Z-axis so that $(\omega_0 + \omega_1)/2$ matches the diagonal line $y = x$.

6.5 Weaving Grid Registration

Given a volume with the associated weave pattern, we need to crop out the incomplete yarns on the boundary and make the remaining part aligned with the weaving grid. This sub-yarn level registration is crucial for the quality of synthesized data since our algorithm works at the yarn level and may copy incomplete parts of yarn crossings if the yarns are not centered in the blocks.

We first estimate the size (w, h) of the crop window by multiplying the number of complete yarns and their average sizes. Then the problem becomes that of finding a translation (x, y) such that the content in the crop window agrees with the weave pattern best.

Assume that the scanned volume has size $s_1 \times s_2 \times s_3$. We compute an $s_1 \times s_2$ binary image b in which $b(i, j)$ is set to 0 if the topmost non-blank voxel located at (i, j) is part of a warp yarn and 1 otherwise. For each grid in the window corresponding to one pixel in the weave pattern, we assign it a score that equals the fraction of pixels in b that agree with the weave pattern value. The best crop window maximizing the total score of all grids can be computed in $O(s_1 \cdot s_2)$ time using a summed area table.

For our scanned data, $s_1 = s_2 = 1000$, $s_3 = 300$, $w = 575$, and $h = 350$. Using this configuration, each cropped volume contains 5×5 yarn crossings. Thus every block has the resolution $115 \times 70 \times 300$.

6.6 Summary

We create our exemplar database by taking the processed CT data, tracking the yarns, computing per-voxel yarn IDs, detecting the weave pattern, and regularizing the volume. Our pipeline requires a

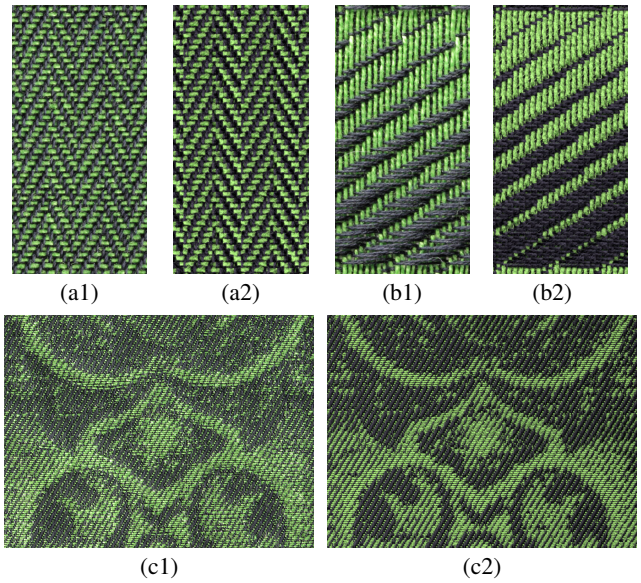


Figure 11: *Comparisons between photographs of fabricated cloth samples (left) and rendered images with the synthesized data (right): (a) a Herringbone fabric; (b) a fabric containing all 9-twill patterns; (c) a Jacquard fabric (design courtesy of Brooks Hagan).*

small amount of user input including the thresholds for tracking, the number of yarns for endpoint detection, yarn grouping for weave pattern detection, and the resolution of a single block for weaving grid registration. For creating each of our exemplars, the entire process (excluding the basic processing step from Zhao et al. [2011]) runs in seconds.

7 Experimental Results

We show two types of results to demonstrate our technique: comparisons of our synthesized results with real fabricated cloth samples, and new designs synthesized using our algorithm. Our renderings are generated using Monte Carlo volume path tracing implemented in the Mitsuba renderer [Jakob 2010].

First, to demonstrate the accuracy of our approach, we compare our results with real fabricated cloth samples. Several designs, including a "test blanket" of 5-twill and 5-satin example patches (shown in Figure 2d), were woven on an industrial Jacquard loom (see Figure 12) at Rhode Island School of Design (RISD). Solving for the optical properties of multiple kinds of yarns in a fabric is beyond the scope of this paper, so we manually picked colors for the black warp and green weft to obtain a rough match in appearance.

The results demonstrate our ability to correctly predict the structure and overall appearance of woven fabrics before they are fabricated, meaning that our methods are useful for textile designers who currently design "in the blind" without seeing any realistic preview of the cloth before fabrication.

We first pick two example weaves, Herringbone and a 9-twill pattern, that are not in our input set of exemplars, thus demonstrating the power of our approach to generalize to complex weaves. These are shown in Figure 11ab (in the 9-twill results, the fabric is rotated by 90° with the wefts going vertically). While our results are more regular than the photo, we successfully capture the key structure of both patterns. As shown in Figure 4, our algorithm copies big chunks of data if similar structures are contained in the database (the orange region) and synthesizes other regions from smaller pieces. Note that we are not allowed to rotate the exemplars, because the warp and weft yarns are dissimilar, or to mirror them, because that would reverse the twist of the yarns, destroying the consistency of local fiber orientations.

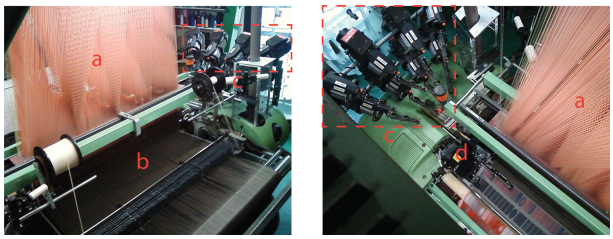


Figure 12: *The industrial Jacquard loom at Rhode Island School of Design used to weave our samples: (a)* harnesses *used to lift the warps; (b) the warp yarns; (c) spools of multi-colored weft yarns; (d) the* shuttle *for carrying and inserting wefts.*

Next we synthesized a model using the weave pattern that was used to weave the Jacquard sample. Figure 11c shows a comparison between our synthesized result and the real sample.

Finally, Figure 13 show more results synthesized using our technique and mapped to arbitrary surfaces using shell mapping [Porumbescu et al. 2005] following the approach of Zhao et al. [2011]. In each image the weave pattern or the posterized image used to create the weave pattern is shown at the top left corner, and magnified insets appear at the bottom right. Please see the accompanying video for animated renderings.

All weave patterns used to synthesize these results have the resolution of 900×1500. Our synthesis algorithm runs in no more than 10 seconds to create each output volume containing 3.26×10^{12} effective voxels. Each rendered image has the resolution 2560×1440, and the rendering time varies from 15 to 40 minutes on a QSSC-S4R Intel Server with 32 logical cores.

Figure 13a shows a fabric created with a 96×96 wavy twill pattern. This input is very different from the patterns in our exemplar set; thus it is difficult to copy large pieces of continuous structure. Our approach does a good job of synthesizing the fabric with smooth shading across the surface.

In Figure 13b, we show a fabric containing alternating 1/15 and 15/1 satin blocks with all yarns assigned identical optical properties. This kind of same-color patterning is often used in fabrics for bed linens and draperies. Because the yarns go in perpendicular directions, the fabric has highly anisotropic appearance. By correctly synthesizing the structure of the yarn we are able to automatically capture this characteristic appearance. In addition, 3D structures created by the transitions between the two weave patterns can be easily observed even at a large scale.

Figure 13c shows a Jacquard cloth with 1/7 twill and 7/1 satin respectively forming the golden and the red area. The posterized image used to generate the pattern is shown on the upper-left. Again, the different structure of the two weaves results in distinctive anisotropic behavior that is captured well by our method.

Figure 13d shows a complex Jacquard cloth with two weft colors (white and golden). The design is provided as an example in Pointcarré [2001]. The synthesized result conveys a richly detailed surface structure.

8 Conclusion and Future Work

We have demonstrated an approach to generate highly realistic volumetric models with spatially varying appearance and complex designs. Unlike previous techniques like BTFs, our method can pro-

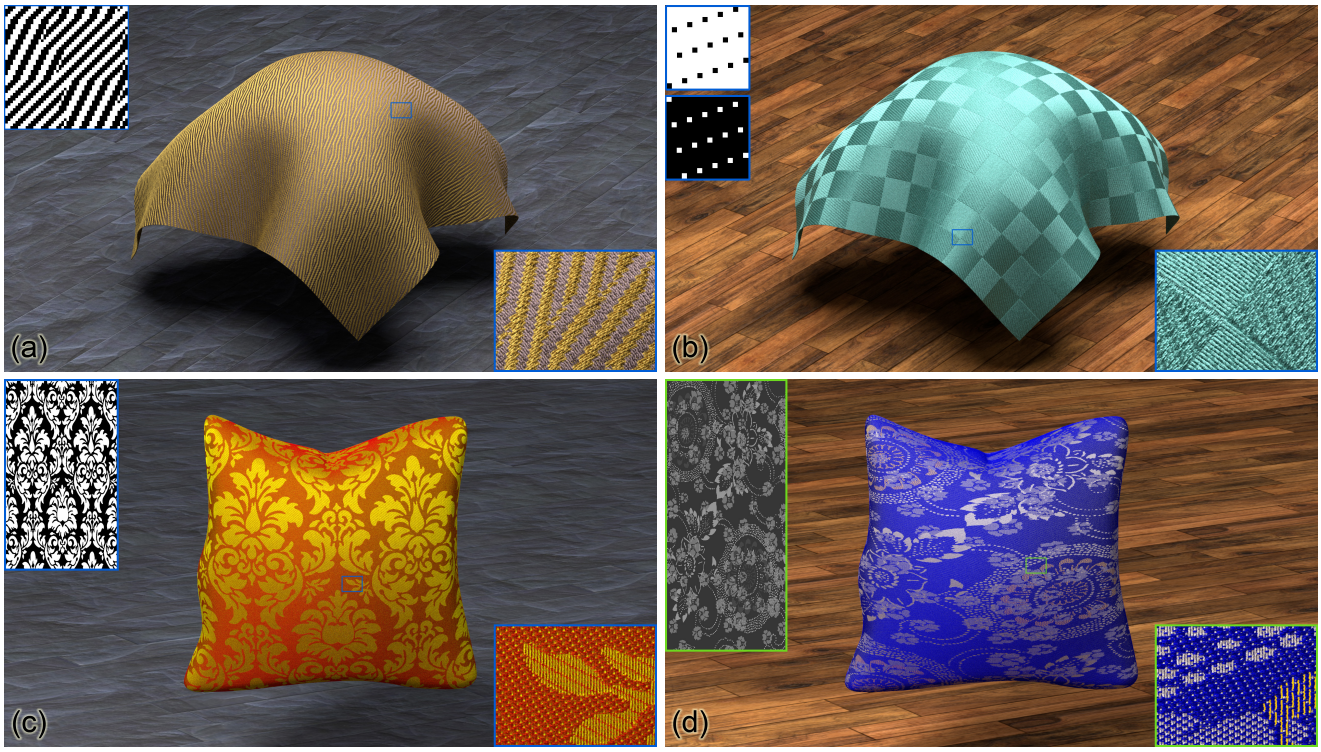


Figure 13: *Synthesized results with different weave patterns: (a) a wavy twill; (b) a fabric composed using alternating blocks with 1/15 and 15/1 satin patterns; (c) and (d) Jacquard brocades mapped onto pillows. Top-left: weave patterns, Bottom-right: insets.*

duce fabrics with different designs using a fixed set of exemplars. Our approach takes advantage of micro CT imaging to measure highly detailed 3D structure and introduces a synthesis process to generalize the measured data to model fabrics with complex larger-scale structures. Our contributions include a robust pipeline for rapidly creating exemplars for fabrics, and a fast synthesis algorithm to create complex volumetric models. We have validated our synthesized results by comparing them to fabricated real samples.

We believe that this technique is very useful for both the computer graphics community and the textile design community. Using our exemplars, users can now create high quality fabric models with their own designs without having to write specialized code. And textile designers can use our approach to predictively visualize their designs without physically creating them.

There are multiple areas of future work. One limitation of our current work is that our method can only synthesize models with a grid-like weave structure. Although such structure is very common in real-world fabrics, those of other cloth (such as knitwork) are different. We would like to extend our framework to support more structures and ultimately go beyond fabrics. For more automated, predictive rendering of existing fabrics, better appearance matching methods that solve for the optical properties of multiple kinds of yarns are needed. Finally, better optical models may be required to perfectly match the appearance of fabric samples.

Acknowledgements

The authors would like to thank Brooks Hagan from RISD for many enlightening discussions about weaving and textile design, and for weaving the fabric samples used in our results. We also thank Jessie Maisano from the UTCT facility for scanning the example fabrics. This research was conducted in conjunction with the Intel Science and Technology Center—Visual Computing. Funding was provided by the National Science Foundation under awards CCF-0644175 and IIS-1011919. We thank Amazon for the donation of EC2 time.

References

ADABALA, N., AND MAGNENAT-THALMANN, N. 2003. A procedural thread texture model. *Journal of Graphics Tools 8*, 3, 33–40.

ADABALA, N., MAGNENAT-THALMANN, N., AND FEI, G. 2003. Visualization of woven cloth. In *Proceedings of the 14th Eurographics Workshop on Rendering*, 178–185.

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 217–226.

CHEN, Y., TONG, X., WANG, J., LIN, S., GUO, B., AND SHUM, H.-Y. 2004. Shell texture functions. *ACM Transactions on Graphics 23*, 3, 343–353.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Transactions on Graphics 22*, 3, 287–294.

DONG, Y., WANG, J., TONG, X., SNYDER, J., LAN, Y., BEN-EZRA, M., AND GUO, B. 2010. Manifold bootstrapping for SVBRDF capture. *ACM Transactions on Graphics 29*, 4, 98:1–98:10.

DRAGO, F., AND CHIBA, N. 2004. Painting canvas synthesis. *The Visual Computer 20*, 5, 314–328.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual*

Conference on Computer Graphics and Interactive Techniques, 341–346.

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, vol. 2, IEEE, 1033–1038.

GHOSH, A., CHEN, T., PEERS, P., WILSON, C. A., AND DEBEVEC, P. 2009. Estimating specular roughness and anisotropy from second order spherical gradient illumination. *Computer Graphics Forum 28*, 4, 1161–1170.

GHOSH, A., CHEN, T., PEERS, P., WILSON, C. A., AND DEBEVEC, P. 2010. Circularly polarized spherical illumination reflectometry. *ACM Transactions on Graphics 29*, 6, 162:1–162:12.

GRÖLLER, E., RAU, R. T., AND STRASSER, W. 1995. Modeling and visualization of knitwear. *IEEE Transactions on Visualization and Computer Graphics 1*, 4, 302–310.

GRÖLLER, E., RAU, R. T., AND STRASSER, W. 1996. Modeling textiles as three dimensional textures. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, 205–214.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 229–238.

HURTUT, T., LANDES, P.-E., THOLLOT, J., GOUSSEAU, Y., DROUILLHET, R., AND COEURJOLLY, J.-F. 2009. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, 51–60.

IRAWAN, P., AND MARSCHNER, S. 2012. Specular reflection from woven cloth. *ACM Transactions on Graphics 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 Transactions on Graphics 29*, 53:1–53:13.

JAKOB, W., 2010. Mitsuba physically based renderer. mitsuba-renderer.org.

KAJIYA, J. T., AND KAY, T. L. 1989. Rendering fur with three dimensional textures. *SIGGRAPH Computer Graphics 23*, 3, 271–280.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics 26*, 3.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3, 795–802.

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics 24*, 3, 777–786.

MA, C., WEI, L.-Y., AND TONG, X. 2011. Discrete element textures. *ACM Transactions on Graphics 30*, 4, 62:1–62:10.

MARSCHNER, S. R., WESTIN, S. H., ARBREE, A., AND MOON, J. T. 2005. Measuring and modeling the appearance of finished wood. *ACM Transactions on Graphics 24*, 3, 727–734.

MERRELL, P., AND MANOCHA, D. 2008. Continuous model synthesis. *ACM Transactions on Graphics 27*, 5, 158:1–158:7.

PERLIN, K., AND HOFFERT, E. M. 1989. Hypertexture. *SIGGRAPH Computer Graphics 23*, 3, 253–262.

POINTCARRÉ, 2001. Pointcarré textile software. pointcarre.com.

PORUMBESCU, S. D., BUDGE, B., FENG, L., AND JOY, K. I. 2005. Shell maps. *ACM Transactions on Graphics 24*, 3, 626–633.

RAMANARAYANAN, G., AND BALA, K. 2007. Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics 13*, 1, 167–178.

ROLFSEN, D. 2003. *Knots and links*. American Mathematical Society.

SHINOHARA, T., TAKAYAMA, J., 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.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3, 665–672.

WANG, J., ZHAO, S., TONG, X., SNYDER, J., AND GUO, B. 2008. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Transactions on Graphics 27*, 3, 41:1–41:9.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 479–488.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*.

WU, Q., AND YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics 23*, 3, 364–367.

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*, 391–398.

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

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics 25*, 3, 690–697.