Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail

Cem Yuksel[*†] Jonathan M. Kaldor[*‡] Doug L. James[*] Steve Marschner[*]
[*]Cornell University [†]University of Utah [‡]Facebook

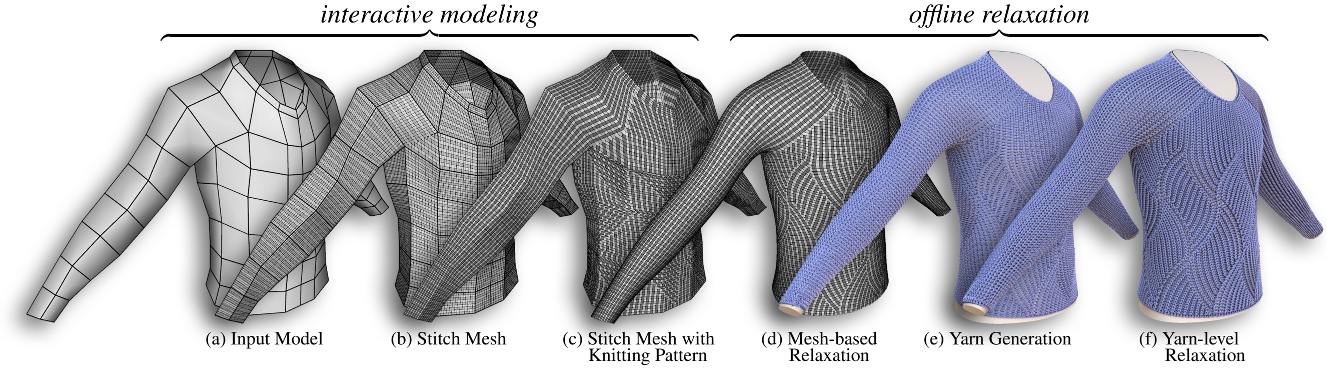
interactive modeling
offline relaxation
(a) Input Model
(b) Stitch Mesh
(c) Stitch Mesh with Knitting Pattern
(d) Mesh-based Relaxation
(e) Yarn Generation
(f) Yarn-level Relaxation

Figure 1: ***Stages of our knitted garment modeling system:*** *(a) We begin our interactive modeling process with a polygonal mesh that specifies the global shape of the cloth model; (b) using this polygonal mesh we produce a high-resolution stitch mesh that serves as a canvas-like abstraction of the yarn model; (c) then, we specify the desired knitting pattern over the stitch mesh's surface. (d) Following the interactive modeling process, the model goes through offline relaxation, beginning with a mesh-based relaxation that moves the stitch mesh to the subdivision surface of the input model and slides its vertices over this surface based on the topology of the knitting pattern; finally, (e) we generate the yarn curves and (f) use a physically based relaxation process at the yarn level to compute the final realistic shape.*

Abstract

Recent yarn-based simulation techniques permit realistic and efficient dynamic simulation of knitted clothing, but producing the required yarn-level models remains a challenge. The lack of practical modeling techniques significantly limits the diversity and complexity of knitted garments that can be simulated. We propose a new modeling technique that builds yarn-level models of complex knitted garments for virtual characters. We start with a polygonal model that represents the large-scale surface of the knitted cloth. Using this mesh as an input, our interactive modeling tool produces a finer mesh representing the layout of stitches in the garment, which we call the *stitch mesh*. By manipulating this mesh and assigning stitch types to its faces, the user can replicate a variety of complicated knitting patterns. The curve model representing the yarn is generated from the stitch mesh, then the final shape is computed by a yarn-level physical simulation that locally relaxes the yarn into realistic shape while preserving global shape of the garment and avoiding "yarn pull-through," thereby producing valid yarn geometry suitable for dynamic simulation. Using our system, we can efficiently create yarn-level models of knitted clothing with a rich variety of patterns that would be completely impractical to model using traditional techniques. We show a variety of example knitting patterns and full-scale garments produced using our system.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

1 Introduction

Knitted fabrics are commonplace and their complicated structure is visually apparent. Stitches can be formed using various knitting operations, leading to a rich variety of possible knitting patterns with drastically different appearances. Since the final shape of a particular stitch depends on the types of stitches around it, constructing realistic knitted clothing requires a simulation at the yarn level.

Unlike sheet-based cloth simulators that approximate the cloth as a thin surface, yarn-level simulators directly simulate the individual yarns and interactions resulting in significantly richer dynamics and more realistic results, especially for knitted clothing. Unfortunately, authoring models for yarn-level simulation is challenging with existing modeling practices. Yarn-level simulations do not tolerate even the slightest topological mistake in the input yarn curves, since one defective stitch can cause many other stiches to unravel when the simulation starts. While modeling yarn curves manually is too complicated for any practical purpose, automatically generating stitch patterns over a surface using modified texturing methods provides no topological guarantee that it will not unravel during simulation. An alternative approach is to simulate real-world knitting processes, but this would be very expensive using current simulation techniques, and the simulation outcome would likely be hard for an artist to predict, thus complicating garment design.

In this paper we address these challenges by proposing an efficient framework for modeling knitted clothing with realistic yarn-level details. Figure 1 shows the stages of our system, starting from a low-resolution polygonal mesh to the final knitted cloth model. Our stitch mesh structure provides a novel mesh-based representation of the knitted yarn geometry. Our stitch-mesh modeling framework provides an efficient workflow with both low-level and high-level design control, enabling artists to create a rich variety of knitting patterns and full-scale garments for virtual characters. To reduce yarn-level relaxation costs, we propose a novel mesh-based relaxation operation using the stitch mesh that can properly handle large-scale sliding of stitches. Finally, we improve the yarn-level relaxation process to support more stitch types for increased realism, and introduce optimizations, such as yarn pull-through detection, which accelerate and ensure the correctness of yarn-level simulations.

2 Prior Work

Robust and efficient cloth simulation has long been a research focus of the graphics community. Much of the work has been directed towards sheet approximations of cloth [Baraff and Witkin 1998; Bridson et al. 2002; Grinspun et al. 2003; Goldenthal et al. 2007; Volino et al. 2009], where the yarn-yarn interactions in fabric are abstracted away and instead treated as aggregate elastic forces on a 2D sheet. Some more recent work has looked at simulating cloth with explicit yarn interactions [Chu 2005; Kaldor et al. 2008; Kaldor et al. 2010]. While more expensive than elastic sheet approximations, these models can replicate yarn-level details in different styles of fabric. Our relaxation simulator is a two-phase hybrid approach, with the first mesh-based relaxation bearing similarities to sheet-level cloth simulation, and the subsequent relaxation based on yarn-level simulation. This has similarities to work like Nocent et al. [2001], which embeds knit stitches within a thin volume and simulates the forces of the yarns in the reduced volume space. However, we simulate our yarns directly in the unreduced space, with our yarn-level relaxation allowing significant local yarn deformation while providing guarantees that the fabric does not unravel, e.g., through a failure of the yarn-yarn contact forces to adequately resist yarns pulling through one another.

In both sheet-based and yarn-based models, the initial input to the simulator must be the original configuration of the model; in sheet-based approaches the input is a mesh defining the cloth surface, while in yarn-based approaches it is a curve defining the yarn geometry. Similar to actual manufacturing processes, sheet-based cloth modelers have often modeled cloth as panels to be sewn together [Carignan et al. 1992; Volino and Magnenat-Thalmann 2000; Volino et al. 2009]. These initial meshes are then simulated using the sheet-based simulator to generate the relaxed shape of the clothing on a human form with the user making further changes to the initial pattern based on the relaxed result. Since the simulation process can be slow, Luo and Yuen [2005] efficiently rebuilt relaxed 3D meshes from edits to 2D patterns. Other research has looked at designing useable interfaces for rapid and intuitive design of new patterns [Volino and Magnenat-Thalmann 2005; Umetani et al. 2011]. Researchers have also looked at sketch-based interfaces for designing garments and other 3D objects composed of 2D pieces [Decaudin et al. 2006; Turquin et al. 2007; Mori and Igarashi 2007; Igarashi et al. 2008b; Volino et al. 2009; Robson et al. 2011].

For yarn-based simulators, the problem of generating initial geometry is more complicated, since the entire yarn curve must be specified and any topological errors can propagate through the cloth in unpredictable and potentially disastrous ways, e.g., unraveling. Researchers have looked at replicating the manufacturing processes used by knitting machines [Eberhardt et al. 2000; Duhovic and Bhattacharyya 2006]. These processes produce correct knitted patterns, but are typically extremely slow (limiting them to small patches of material), and support only a limited set of stitch types much like real knitting machines. Meißner and Eberhardt [1998] estimated the appearance of some knitting patterns by using input data of a knitting machine to help generate and simulate simplified topologies. Kaldor [2011] describes a semi-automated process involving models of individual knit loops that can be tiled together to form cloth; however, constructing a new garment requires the user to write code specifying how loops are laid down, to create geometric models for any new types of loops used, and to verify that the final result is topologically correct.

There has also been work in the textile community on geometric modeling of knitted materials. Several works generate knit geometry using spline curves and focus on the plain knit stitch [Demiroz and Dias 2000; Göktepe and Harlock 2002; Choi and Lo 2003; Choi and Lo 2006; Renkens and Kyosev 2011]. Other research has looked at modeling representative cells of complex knitted fabric [Kurbak and Alpyildiz 2008; Kurbak 2009; Kurbak and Soydan 2009], but depending on the pattern, these cells might have to contain many individual stitches, and combining multiple patterns in a single fabric requires the boundaries of each representative cell to be carefully and correctly matched up.

Generating knitted yarn geometry also bears some similarity to texture synthesis [Heeger and Bergen 1995; Kwatra et al. 2003]. In both, a large scale output is to be generated from a small set of inputs and constraints on where they can appear in the output. In particular, extensions of texture synthesis-like methods have been proposed such as Mesh Quilting [Zhou et al. 2006], constrained structure preserving reshaping of architectural models [Cabral et al. 2009], and Celtic knots on surfaces [Lai et al. 2010]. However, unlike these applications, our intent is to generate curve models suitable for simulation, and the nature of knitted fabric means that errors in the construction affect the behavior of the material or can lead to unraveling. Unfortunately, texture synthesis based approaches can typically make no guarantees that the resulting model is topologically valid.

Recently, Igarashi et al. [2008a] proposed a method on the opposite problem of computing a set of knitting instructions for a given virtual model, Akleman et al. [2009] developed a system that converts any manifold mesh to a plain-woven object, and Igarashi and Mitani [2010] introduced an interactive layer operation that can produce woven patterns by flipping the depth order of deformable objects.

3 Overview

The real-world knitting process begins by placing (casting on) stitches on a knitting needle. These *cast-on* stitches form the first *row* of stitches for the garment. Then, an additional row of stitches are knitted by pulling the yarn through the stitches on the first row. This operation is continued by knitting an additional row onto the previous row of stitches until a desired number of rows are knitted.

Examining these stitches, we see that the yarn typically takes the shape of the curve shown in Figure 2a that we call a *yarn loop*. Yarn loops are placed side-by-side forming a row along the *course* direction. During knitting, these rows are formed one by one in order, such that the yarn loops on each row are pulled through the ones on the previous row in the *wale* direction as show in Figure 2b.

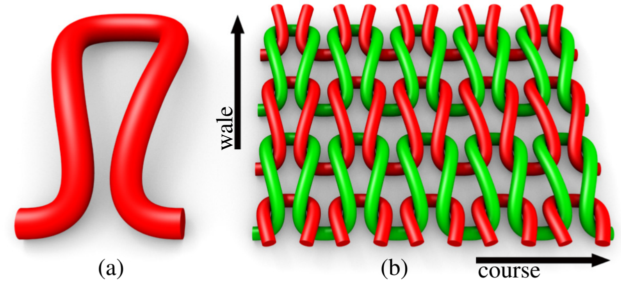

Figure 2: ***Yarn loops:*** *(a) An isolated yarn loop; (b) interlocked yarn loops forming a piece of knitted cloth.*

The relatively simple structure shown in Figure 2 can easily get complicated when loops are pulled through others in different ways to form various knitting patterns. Therefore, it is very difficult to manually prepare the yarn curves for a full cloth model in any but the simplest fabrics. However, while the yarn curves have complicated shapes, they are produced by a small number of stitch types, which are described by relatively simple knitting instructions.

To separate the pattern of stitches, which the user needs to specify, from the resulting yarn geometry, which should be generated automatically, we propose a mesh-based representation of the knitted yarn structure that we call the *stitch mesh* (Section 4). The stitch mesh is essentially a high-resolution polygonal mesh where each face corresponds to a stitch of the yarn-level model, and it is the foundation of our mesh-based modeling system. The user models the stitch mesh using an interactive modeling interface (Section 5) and the yarn-level cloth model is generated automatically from the stitch mesh through offline relaxation (Section 6).

During the interactive modeling process we produce a stitch mesh from a given polygonal model. This stitch mesh includes all the necessary information for generating yarn curves with a valid topological structure, including the knitting pattern and the stitch types used everywhere on the model. In that respect, the stitch mesh is an abstraction for representing the desired yarn curves that also allows easier visualization and interactive editing.

While the stitch mesh represents a valid topological structure for the yarn curves, the yarn curves generated from the stitch mesh do not necessary have a realistic shape. Therefore, following the interactive modeling process, an offline simulation is used to estimate physically based deformations of the knitted structure. To enforce that the offline relaxation stage produces predictable and useful results, we use soft and hard constraints to preserve the garment's overall shape.

4 The Stitch Mesh

The stitch mesh structure is an abstraction for the yarn-level geometry that enables easier modeling. Each stitch-mesh face is a building block which represents a portion of the knitted garment. The obvious building block for a knitted fabric is a yarn loop, as has been used in other work, but this is not a good choice since it does not encode topological state, and its shape changes depending on how it interlocks with adjacent loops. Instead we propose a new approach that uses a cell where two yarns interact. Our building block is the top part of one yarn loop, and the bottom part of another yarn loop that is pulled through forming a *stitch* as shown in Figure 3a. These individual units are much more independent since they connect only at single points between cells, and it is simple to create complicated structures by composing such building blocks. For example, by tiling the quad in Figure 3a over a surface we can easily form a patch of knitted cloth as shown in Figure 3b.

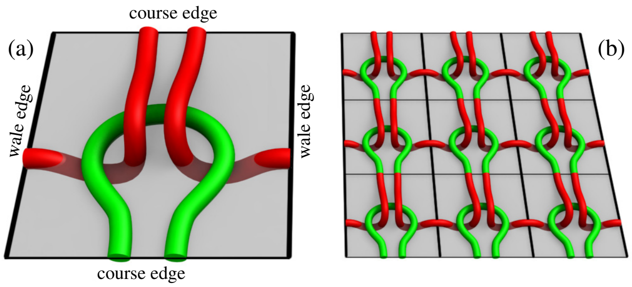

Figure 3: ***A quad stitch-mesh face*** *(a) with associated yarn pieces, and labeled wale and course edges; (b) a stitch mesh (and associated yarn curves) formed by tiling the quad face.*

The yarn curves that correspond to each face of the stitch mesh are constructed using a 2.5D representation. Each control point of the yarn curves is represented using mean value coordinates [Floater 2003] that specify the projection of the control point on the corresponding stitch-mesh face, and an offset value in the surface normal direction at this point.

We refer to face edges that are aligned with the course direction as *course edges*, and edges that are aligned with the wale direction as *wale edges* (Figure 3a). Each edge of the stitch mesh is either a course edge or a wale edge. Notice that each wale edge has a single yarn crossing it, whereas each course edge has two.

4.1 Increases and Decreases

In general, stitch-mesh faces do not have to be quads, but each face must have exactly two wale edges that connect it to adjacent stitches on the same row. Considering the edges of a face, the two wale edges separate the course edges into two groups: *top* and *bottom*, such that the wale direction on the face points from the bottom edges towards the top edges. A face can have any number of top and bottom course edges. Some examples of faces with different number of top and bottom edges are shown in Figure 4.

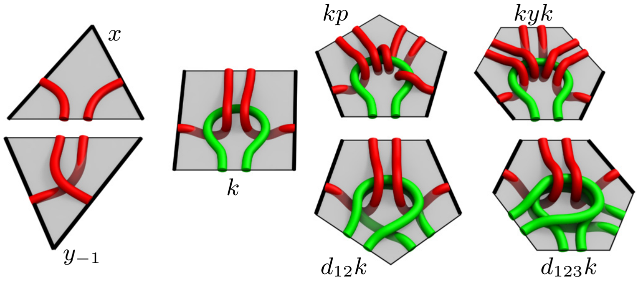

Figure 4: ***Example stitch-mesh polygons*** *demonstrate yarn models corresponding to faces with different numbers of edges: (top row) faces with a single bottom edge, and (bottom row) faces with a single top edge. Wale edges are shown thicker than course edges.*

A stitch-mesh face with multiple top edges represents a single stitch that is connected to multiple stitches on the next row. Such stitches are typically used for increasing the number of stitches on the next row; therefore, they are referred to as *increases*. Similarly, stitch-mesh faces with multiple bottom edges decrease the number of stitches on the current row as compared to the previous row, so they are called *decreases*. Increases and decreases are used to knit non-planar shapes, as well as to produce various patterns.

4.2 Stitch Types

There are three fundamental actions for generating stitches while knitting: pulling the yarn through an existing loop either from the back side towards the front (*knit*) or from the front side towards the back (*purl*), and simply wrapping the yarn around the knitting needle without pulling it through a loop (*yarn-over*). We represent these actions using k, p, and y symbols, respectively. Increases are formed by using combinations of k, p, and y on one existing loop. On the other hand, decreases are formed by using k or p through multiple existing loops. While forming a decrease stitch, the order in which the existing loops are stacked together is important, as it determines the direction that the stitch tends to twist. We represent decreases with the symbol d followed by a subscript that determines the stacking order. Some examples of possible stitch types are shown in Figures 4 and 5, along with their symbols. Ta-

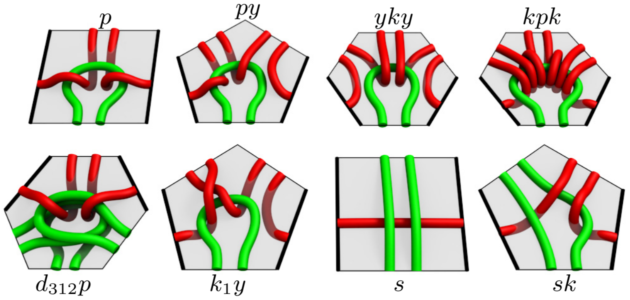

Figure 5: ***Example stitch types*** *for stitch-mesh faces.*

Table 1: *Stitch Types used in our system*

# bottom edges	# top edges	possible stitches
1	1	k, p
1	2	kp, ky, yk, py, yp
1	3	kpk, kyk, yky, kpy, ykp, pkp, pyp, ypy, pky, ypk
2	1	$d_{12}k$, $d_{21}k$, $d_{12}p$, $d_{21}p$
3	1	$d_{123}k$, $d_{132}k$, $d_{213}k$, $d_{231}k$, $d_{312}k$, $d_{321}k$, $d_{123}p$, $d_{132}p$, $d_{213}p$, $d_{231}p$, $d_{312}p$, $d_{321}p$

ble 1 includes all stitch types used in our system. Furthermore, the top edge connections of any k, p, and y can be twisted in clockwise (positive) or counterclockwise (negative) directions, as y_{-1} in Figure 4 and k_1y in Figure 5.

In addition to these standard stitches, the stitch mesh can be used for representing other stitch types, such as the x stitch in Figure 4, or s and sk in Figure 5. In general, the stitch mesh structure permits any stitch model that has a single yarn crossover on both wale edges, and a double yarn crossover on each course edge.

4.3 Borders of the Cloth Model

The borders of the cloth model correspond to the boundary edges of the stitch mesh. The simple example in Figure 6 shows how the loose ends of the yarn curve can be handled. Notice that the loose ends that correspond to the wale edges are either tied (the bottom-left and top-right faces in Figure 6) or connected to another loose end on a nearby wale edge (the bottom-right and top-left faces in Figure 6). The faces of the stitch mesh that are on the first row (the bottom row in Figure 6) represent *cast-on* stitches. In our implementation we merely use y_1 or y_{-1} for cast-on stitches (ignoring the bottom edges). As for the last row (the top row in Figure 6), we add yarn pieces that form the *bind-off* stitches.

Tube-shaped portions of knitted clothing are typically formed by placing stitches on a 3D spiral as shown in Figure 7a. The stitch mesh structure also permits having separated rows of stitches as shown in Figure 7b, which corresponds to having a separate piece of yarn on each row and each piece of yarn is tied to itself forming a *closed* yarn curve around the surface. While such structures are not really used (since they would need a separate knot on each row), they form topologically valid yarn structures for our purposes and they produce only a minor visual difference from the spiral layout. Therefore, in our system we use separate rows and avoid the spiral layout to keep the topology of the stitch mesh simpler.

4.4 Mismatched Wale Direction

Typically the wale directions on neighboring faces of a stitch mesh are aligned with each other. Yet, sometimes it is necessary to bind stitches with opposing wale directions. In reality, these special cases require either placing special stitches, or sewing together those stitches with opposing wale direction. However, the stitch mesh structure can automatically handle opposing wale directions (see Figure 8) by forming multiple separate small yarn pieces. While these pieces correspond to sewing the two patches together, they do not produce a real-world yarn structure.

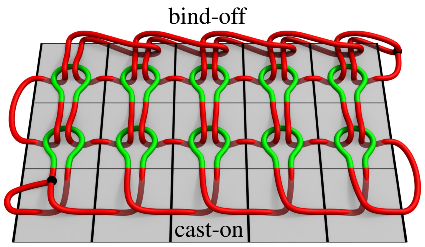

Figure 6: ***Handling the borders of the stitch mesh*** *with cast-on and bind-off stitches.*

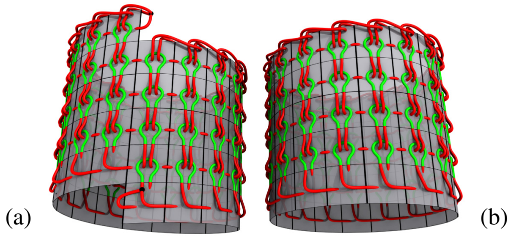

Figure 7: ***Two possible stitch meshes for a tube:*** *(a) A spiral structure that is analogous to real-world knitting approaches, and (b) a simpler ring-like structure formed by separate rows but which is much harder to knit in reality.*

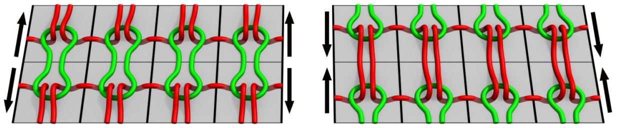

Figure 8: ***Mismatched wale directions*** *result in closed yarn loops between rows with mismatched directions.*

4.5 Cables

During a real-world knitting process, instead of pulling loops through the ones on the previous row in order, one can change the order by skipping a few stitches, pulling loops through the next few stitches, and then going back and pulling loops through the ones that were skipped. Such operations create *cables*. They are common in knitting, and result in interesting surface deformations (see Figure 9(left)). In our system we handle cables by replacing a group of selected course edges with a special face that we call the *cable face* (see Figure 9(right)). A cable face is placed in between two consecutive rows and has no wale edges. The yarn model that corresponds to the cable face determines the order in which the loops are pulled through the ones on the previous row. In our implementation, cable faces are introduced right before generating the yarn curves from the stitch mesh, and until then the selected course edges are merely marked as *cable edges*.

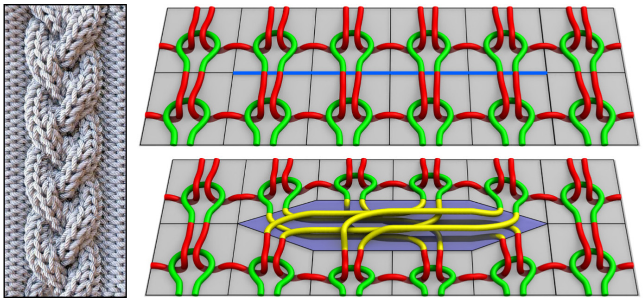

Figure 9: ***Knitting cables:*** *(Left) An example cable pattern; (Right) a cable stitch is modeled by replacing a set of "cable edges" in blue (top) by the blue-shaded "cable face" (bottom).*

5 Interactive Modeling

The aim of the interactive modeling stage is to design a stitch mesh that represents the yarn-level model that we would like to generate. We begin with an input polygonal mesh model that defines the subdivision surface of the desired garment model. In addition, we use the topological structure of this input mesh to determine the desired knitting direction over the surface.

The modeling process begins with labeling the input mesh to specify the knitting direction over each face of the input mesh. Using this information, we produce a high-resolution stitch mesh, which is used as a canvas for easily defining the desired stitch pattern over the surface. In the rest of this section we explain the stages of our interactive modeling process.

5.1 Labeling the Input Mesh Model

Given an input mesh that represents the surface of the garment model, there are often many ways it can be knitted. To provide the user direct control over the knitting direction over the surface, we begin our modeling process with *labeling*, during which each edge of the input mesh is labeled as either a *wale edge* or a *course edge*. We require that each face of the input mesh is assigned exactly two wale edges, and the rest of its edges are labeled as course edges. This way, we can uniquely define the knitting direction over the entire face, such that the yarn curves "enter" the face from one of the wale edges and "exit" the face from the other wale edge.

The collection of faces that are connected by wale edges are called a *row*. Hence, the labeling process conceptually separates the input mesh into a number of rows, as shown in Figure 10. Note that there is no guarantee that an input mesh with an arbitrary topology can be separated into a number of rows in this fashion. On the other hand, there might be multiple ways to separate the input mesh into rows as shown in Figure 10. If the input mesh cannot be separated into a number of rows according to the user's intentions, we require that the input mesh topology be modified manually as desired. Therefore, the input mesh must be modeled with a desired row structure in mind.

In our implementation we use a number of high-level tools for the labeling process, so that a typical input mesh can be easily labeled in a matter of seconds. In our system the user begins labeling by selecting a border edge of the input mesh as shown in Figure 11. Then, the entire border is automatically marked as course edges, all faces that use any of the the border edges form the first row, all unmarked edges of this first row that have a vertex on the border are labeled as wale edges, and the remaining edges are labeled as the course edges of the first row. Afterwards, the user can select any course edge on this row or another border edge to automatically label another row, until the entire input mesh is labeled. We also store the labeling order to determine the wale direction on the faces. Moreover, we allow the user to flip the wale direction of an entire row by selecting a face on that row, which automatically changes the labeling order accordingly.

5.2 Generating the Stitch Mesh

Once the labeling is completed, we generate a high-resolution stitch mesh by tessellating each face of the input mesh. We begin by determining each edge's tessellation value, n_i, which determines how many stitches will be placed along the edge. We initialize the n_i values based on edge length and the stitch size specified by the user, then the user can modify the n_i values as desired.

We enforce that the two wale edges of a polygon are assigned the same tessellation value, hence all wale edges on the same row get the same tessellation value, thereby dividing each row of the labeled input mesh into a number of stitch-mesh rows. On the other hand, the tessellation values of the top and bottom edges of a polygon can be different. In that case the polygon must contain increases or decreases (Figure 12). Increases and decreases are typically placed in the same locations on consecutive rows, resulting in a line pattern forming a "seam." Based on this observation, we place the increases and decreases on a polygon near its wale edges, so that the user can easily control the positions of these seams. We also allow the user to mark a wale edge such that all increases and decreases on the polygon are automatically placed along that wale edge only.

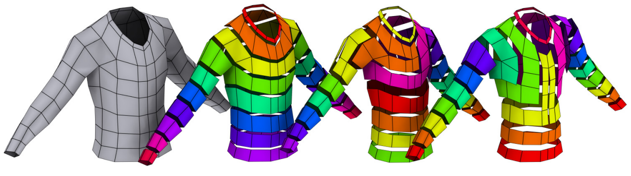

Figure 10: ***Labeling input meshes into knitting rows:*** *(a) input model; (b), (c), and (d) examples of possible rows each of which is displayed as a disjoint piece with a different color.*

The tessellation operation begins by tessellating all the edges by inserting vertices that are uniformly spaced along the edge (see Figure 12). Then, we determine the number of vertices to be placed for each stitch-mesh row over each input mesh face by interpolating the total tessellation values of the course edges on either side of the face. Finally, we place vertices on the input-mesh face using mean value coordinates [Floater 2003], and we generate stitch-mesh faces using these vertices, such that increase/decrease faces (ones with more than 4 vertices) are placed near the marked wale edge. Once the stitch mesh is generated, the user can modify it as we explain in the next section.

5.3 Editing the Stitch Pattern

It is crucial that a knitted garment modeling system allows the user to modify any individual stitch, since a single "wrong" stitch can cause highly visible changes in the final model.

We have implemented a number of low-level operations that modify the topology and attributes of the stitch mesh. *Change stitch type* allows specifying the desired stitch model for each face. In our implementation each stitch type has a particular color, and changing the stitch type changes the color of the face in the interactive window. *Set cable edges* is used for marking a number of connected course edges as cable edges (with the desired shift and order values), which are converted to cable faces right before generating the yarn curves, as explained in Section 4.5. Finally, *split/collapse edge* can be used on course edges and *add/remove/shift* can be used on wale edges to specify increases and decreases (see Figure 13).

Many knitted garments include repeated knitting patterns. Therefore, we have also developed a number of high-level operations that modify multiple faces of the stitch mesh at once based on a desired pattern. For example, the stitch type of multiple faces on neighboring rows and/or columns of the stitch mesh can be changed all at once. This feature can be used to specify common patterns that use alternating k and p stitches (such as rib, garter, and moss).

As for more complicated patterns that include increases and decreases, we replace a selected portion of the stitch mesh with a tiled *pattern*. In our implementation, a pattern is merely a stitch mesh generated on a quad with equal tessellation values on both course edges. Its internal topology and the stitch types of the faces define

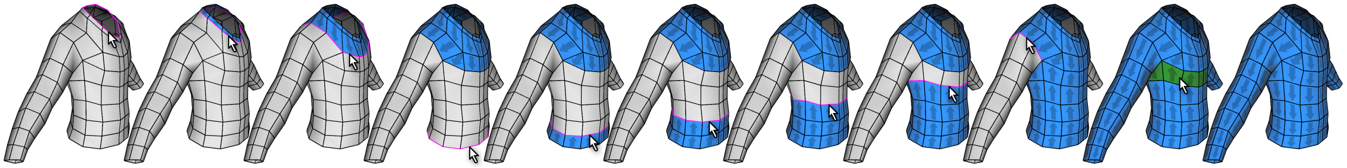

Figure 11: ***The user interface for labeling:*** *Starting with the input mesh on the left, the user selects a border, and the first row is automatically labeled. Subsequent operations label the remaining rows and finally the knitting direction of some rows are flipped as desired to avoid mismatched wale directions.*

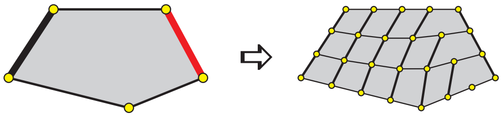

Figure 12: ***Generating the stitch mesh:*** *An example input mesh face (Left) is tessellated into stitch-mesh faces (Right). The red wale edge is the marked increase/decrease edge.*

the knitting pattern. Some patterns can be tiled entirely and others can be tiled partially, such that a number of stitch-mesh faces on either side of each row can be part of a border that is not tiled. To model a pattern we use the low-level editing operations explained above. A pattern can also be created using a text-based description: each line corresponds to a row of the stitch mesh, and lists the stitch types on that row. Figure 14b shows an example pattern modeled based on the knitting instructions in Figure 14a.

Prepared patterns can be used to replace a number of faces on any stitch mesh with the tiled pattern. In our implementation the user selects a face on a stitch mesh, and our system automatically finds the group of stitch mesh faces to be replaced. Then, all faces inside this region are removed from the stitch mesh and new faces are generated according to the pattern description.

6 Offline Relaxation

In the previous section we explained how to prepare a stitch mesh that defines the stitch topology of the knitted garment. When the yarn geometry is generated from this stitch mesh, it generally does not have a realistic appearance. This is because the realistic shape of a stitch can only be determined in the presence of other stitches around it, and knitted garments often include complicated 3D deformations that are defined by the knitting pattern. Therefore, we use a physically based yarn-level relaxation to compute the realistic rest shape of the yarn curves. Since we use yarn-level relaxation, the shapes of the stitch models used for generating the yarn level model from the stitch mesh are less important.

Furthermore, the stitch mesh prepared using the interactive modeling procedure may have faces with significantly different sizes, leading to varying stitch sizes over the model. This distortion is undesirable and often unintended, being caused by the fact that input-mesh edge lengths are not equal to their tessellation value times the average stitch size. Moreover, certain knitting patterns may significantly change the stitch layout, dictating significant amounts of physical deformation. To handle these large deformations and estimate more uniform stitch distributions over the model surface, we employ a mesh-based relaxation procedure that uses the stitch mesh itself *before* we generate the yarn curves. Next, after we generate the yarn curves, we use the yarn-level relaxation to compute the final rest shape of the garment model. Figure 14 shows the stages of offline relaxation on a small pattern.

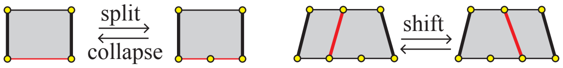

Figure 13: ***Split/collapse and shift*** *are low-level operations on course and wale edges, respectively.*

6.1 Mesh-based Relaxation of the Stitch Mesh

Mesh-based relaxation first moves the vertices of the stitch mesh onto a subdivision surface defined by the input mesh model. Instead of projecting the stitch-mesh vertices directly onto the subdivision surface, we tessellate the input mesh using the subdivision rule and project the vertices onto this surface to compute the barycentric coordinates of the projection, which are then used to compute the vertex positions on the subdivided input mesh. In our implementation we use Catmull-Clark subdivision [Catmull and Clark 1978].

During the simulation we restrict the vertices of the stitch mesh to remain on the subdivided surface, and vertices on stitch-mesh borders are restricted to remain on the borders of the subdivided surface. In order to compute the quasistatic rest state of the vertex positions, $\mathbf{x}$, we timestep a simple first-order dynamics model, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, where $\mathbf{f}$ are spring forces defined later. A backward Euler discretization yields a linear system to be solved for the timestep's change in position $\Delta\mathbf{x}$:

$$\left(\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{x} = h\mathbf{f}_0 \ , \qquad (1)$$

where h is the timestep size, and $\mathbf{f}_0 = \mathbf{f}(\mathbf{x})$ is the force at the beginning of the timestep. We solve this linear system using the modified Conjugate Gradients algorithm of Baraff and Witkin [1998]. We constrain $\Delta\mathbf{x}$ by discarding the component in the surface normal direction, and at the end of each step we move the vertices of the stitch mesh on a high resolution triangular mesh approximation of the subdivision surface in the direction of $\Delta\mathbf{x}$.

Before computing forces we estimate the rest lengths r for all wale and course edges using $N r_{\text{wale}} r_{\text{course}} = \alpha A$ and $r_{\text{wale}} = a\ r_{\text{course}}$, where a is the aspect ratio of a stitch (in our implementation $a = 1$), N is the number of faces, A is the total surface area of the subdivision surface, and α is a scaling factor ($0 < \alpha \leq 1$). We use α to make sure that we do not overestimate the stitch size around the parts of the model where the stitches are not stretched. While underestimating r_{wale} and r_{course} makes minor changes to the final relaxed state, over-estimating these values deforms the stitches due to strong stretch forces.

We use three different types of spring forces: *stretch*, *shear*, and a *wale strut*. The way we apply the stretch and shear forces assumes that the rest shape of each quad face is a rectangle with edge lengths r_{course} and r_{wale}. Stretch and shear forces on non-quad faces are constructed assuming that such faces are composed of multiple *sub-quads* as shown in Figure 15, with a quad face having a single sub-quad, and triangular faces having no sub-quads.

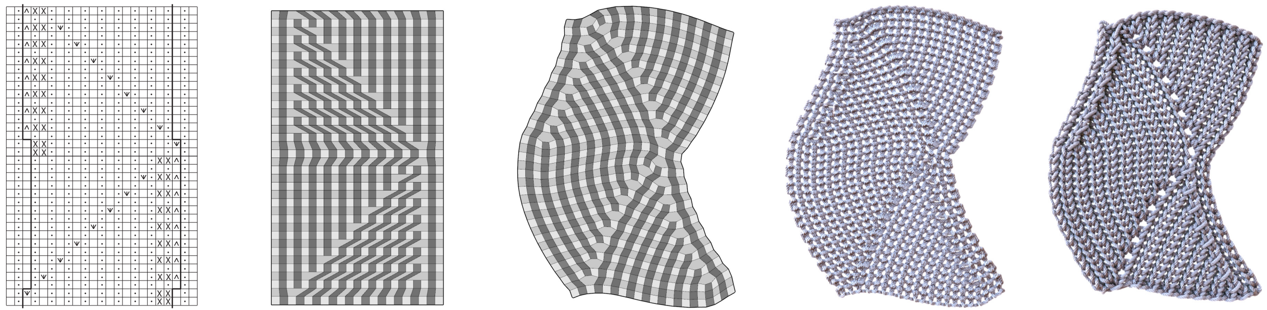
(a) knitting instructions (b) stitch mesh model (c) mesh-based relaxation (d) yarn curves (e) yarn-level relaxation

Figure 14: ***Preparation of the flame ribbing pattern:*** *(a) The real-world knitting instructions adapted from [Walker 2001] (⋀ is three way decrease, ⩛ is three-way increase,* **X** *is no stitch, • is purl, and empty cells are knit), (b) the stitch mesh model (□ k and ■ p) generated from these instructions has significant distortions evident, (c) the result of the mesh-based relaxation, (d) the initial yarn model generated from the relaxed stitch mesh, and (e) the final result after yarn-level relaxation.*

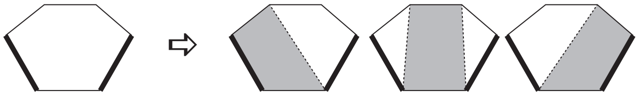

Figure 15: ***Sub-quads of a non-quad face*** *are used to define stretch and shear forces. Here a face representing a three-way increase has three sub-quads connecting course edges from different rows.*

We model the stretch force contribution to vertex i by an edge connected to vertex j using

$$\mathbf{f}_{ij}^{\text{stretch}} = \kappa_{\text{stretch}} \left(\frac{\ell_{ij}}{|\mathbf{x}_i - \mathbf{x}_j|} - 1 \right) \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|} \ , \qquad (2)$$

where κ_{stretch} is the stiffness, and ℓ_{ij} is the rest length, which is r_{course} or r_{wale} for course and wale edges respectively. Note that this is a nonlinear stretch force that goes to infinity as the edge length goes to zero. This nonlinearity prevents edges from shrinking too much even when other forces offer strong resistance. We also apply stretch forces between diagonal vertices of each sub-quad of each face, and use rest length $\ell = \sqrt{r_{\text{course}}^2 + r_{\text{wale}}^2}$.

Shear forces try to give a rectangular shape to each sub-quad. Given a corner vertex j of a sub-quad with neighboring vertices i and k, we apply a shear force to it using

$$\mathbf{f}_{ijk}^{\text{shear}} = -\kappa_{\text{shear}} \ (\mathbf{x}_i - \mathbf{x}_j)^{\text{T}} (\mathbf{x}_k - \mathbf{x}_j) \left(\mathbf{x}_j - \frac{\mathbf{x}_i + \mathbf{x}_k}{2} \right) \ , \qquad (3)$$

and the vertices i and k are both applied half of this force in the opposite direction.

Finally, we use a wale strut force that tries to stiffen wale edges on consecutive rows, which helps keep them aligned. Consider a wale edge that connects vertices i and j, and another wale edge on the next row that connects j and k. A wale strut force with stiffness κ_{wale} is applied to vertex i using

$$\mathbf{f}_{ijk}^{\text{wale}} = -\kappa_{\text{wale}} \left(\frac{|\mathbf{x}_i - \mathbf{x}_k|}{r_{ijk}} - \mathbf{1} \right) \frac{\mathbf{x}_i - \mathbf{x}_k}{|\mathbf{x}_i - \mathbf{x}_k|} \ , \qquad (4)$$

where $r_{ijk} = \max(r_{\text{wale}}, |\mathbf{x}_i - \mathbf{x}_j| + |\mathbf{x}_k - \mathbf{x}_j|)$, and an opposite force is applied to vertex k.

We use the same stiffness parameters to generate all the examples in this paper ($\kappa_{stretch} = 2$, $\kappa_{shear} = 0.2$, and $\kappa_{wale} = 2$) and we needed no parameter tuning for different examples.

Note that mesh-based relaxation merely uses the topology of the stitch mesh and it does not take the stitch types or cables into consideration. Therefore, while the yarn geometry generated after the mesh-based relaxation is closer to the final relaxed shape, we still need yarn-level relaxation to estimate the final position and shape of each stitch.

6.2 Yarn Generation

After the mesh-based relaxation is completed, we are ready to generate the actual yarn curves. We first convert the cable edges to cable faces (Section 4.5). We replace each face of the stitch mesh with the corresponding stitch model that is embedded on the face using mean value coordinates and the surface normal (Section 4). Finally, we handle the borders of the stitch mesh (Section 4.3). The generated yarn curves have the desired topology, but their shapes are not necessarily realistic. Therefore, we employ yarn-level relaxation to compute a realistic rest pose for the yarn curves.

6.3 Yarn-level Relaxation

Given the spline geometry generated by the modeling step, the cloth is relaxed using an implementation of adaptive contact linearization [Kaldor et al. 2010]. Length constraints are replaced with stiff springs with a user-defined rest length per yarn loop, the rod is treated as isotropic with a straight rest configuration, and bending plasticity is disabled. In addition, gravity is set to zero and large amounts of mass-proportional damping are used to stably relax the cloth. Because the initial geometry may be distorted, the contact set strategy proposed by Kaldor et al. [2010] may result in excessively large contact sets being created; to avoid this inefficiency, we define our contact sets as a pair of yarn segments in contact. We also introduce several modifications so that the overall shape of the cloth model is preserved while realistic small-scale deformations due to the stitch pattern are properly resolved.

Shape Preservation: To preserve the overall shape of the garment, we use moment-preserving constraints and forces as in [Bergou et al. 2007]. For each yarn loop S_j on the boundary, a hard constraint is inserted, such that

$$\mathbf{C}_j = \sum_{i \in S_j} \mathbf{q}_i - \sum_{i \in S_j} \bar{\mathbf{q}}_i \ , \qquad (5)$$

where $\mathbf{q}_i$ is control point i in yarn loop S_j, and $\bar{\mathbf{q}}_i$ is its initial position at the start of relaxation. This causes the boundary to closely match the original mesh, but does not constrain the interior of the cloth. Since it is expected that significant relaxation and movement will likely occur, hard constraints are inappropriate for the interior. Instead, patches P_k are defined by choosing yarn loops to be centers, selecting all yarn loops that are within a k-ring of faces on the

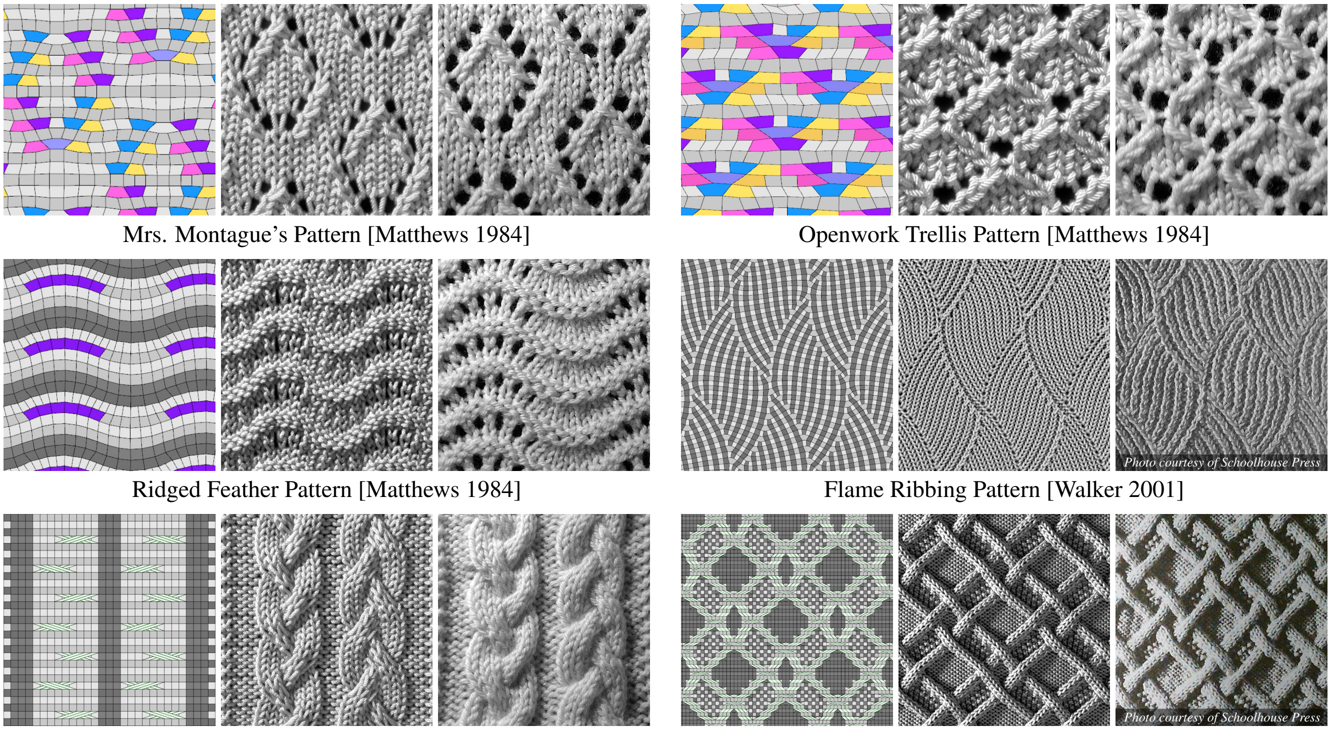
Photo courtesy of Schoolhouse Press
Photo courtesy of Schoolhouse Press

Mrs. Montague's Pattern [Matthews 1984]

Openwork Trellis Pattern [Matthews 1984]

Ridged Feather Pattern [Matthews 1984]

Flame Ribbing Pattern [Walker 2001]

Braid Cables Pattern [Allen et al. 2008]

Cable Work Pattern [Walker 2001]

Stitch Mesh color coding (odd rows are slightly darker): ⬜ *k* ⬛ *p* 🟦 *ky* 🟪 *yk* 🟪 *yky* 🟨 $d_{12}k$ 🟥 $d_{21}k$

Figure 16: ***Comparison to real knitted samples:*** *(Left images) Stitch meshes after mesh-based relaxation, (Middle images) knit patterns after yarn-level relaxation, and (Right images) photographs of real-world knitted samples using the same knitting patterns.*

stitch mesh. An anisotropic, biphasic force is then defined from an energy term as follows:

$$E_k^{\text{tracks}} = \frac{\kappa_{\text{N}}(\mathbf{n}_k^T\mathbf{v}_k)\ (\mathbf{n}_k^T\mathbf{v}_k)^2 + \kappa_{\text{T}}\,\mathbf{v}_k^T(\mathbf{I}-\mathbf{n}_k\mathbf{n}_k^T)\mathbf{v}_k}{\sum\limits_{S_j\in P_k}\sum\limits_{i\in S_j} w_j^k}, \quad (6)$$

$$\mathbf{v}_k = \sum_{S_j\in P_k}\sum_{i\in S_j} w_j^k\mathbf{q}_i - \sum_{S_j\in P_k}\sum_{i\in S_j} w_j^k\bar{\mathbf{q}}_i, \quad (7)$$

where $\kappa_{\text{N}}()$ is a linearly biphasic stiffness function in the normal direction which stiffens when $|\mathbf{n}_k^T\mathbf{v}_k|$ is greater than a defined threshold, κ_{T} is the constant stiffness in the tangential direction, $\mathbf{n}_k$ is the normal of the cloth surface at the center of the patch at the start of relaxation, and w_j^k is a tent function which falls off according to the topological distance of yarn loop S_j from the center of P_k. This encourages patches to stay close to their original starting positions (which are on the subdivision surface), while allowing them to move and change shape as necessary for relaxation.

Detection of Yarn Pull-through: Perhaps the most important improvement over prior yarn-level simulators, though, is a method to guarantee that the knit topology remains consistent through the entire process by detecting when a piece of yarn could pass through another, an event we call *yarn pull-through*, and preventing pull-through from occuring. To begin with, the simulator enforces a rate limit τ on the maximum movement of any point on the yarn curves per step, where τ is some fraction of the yarn radius. This reduces the problem of detecting yarn pull-through per step to only those pieces of yarn already in contact at the beginning of the step.

We represent the yarn curves using cubic Catmull-Rom splines, hence directly solving for the intersection of two cubic curves within a timestep forms a multivariate nonlinear equation which, in general, is challenging to solve both robustly and efficiently. Instead, we place a number of bounding spheres with regular parameter intervals along the two parametric curves, and check for the intersection of these spheres within the time step interval, which involves solving a quadratic equation. If an intersection is found, we replace each intersecting sphere with a number of smaller spheres that bound the same part of the curve, and repeat the intersection test until the sizes of the spheres are sufficiently small. If an intersection still exists at the finest bounding sphere level, we conclude that step size would cause a pull-through and reduce the step size to avoid it.

However, performing this for every pair of contacting segments at every timestep is still expensive. We can further accelerate this detection by computing safe bounds for each control point at the end of the timestep and using this bound to possibly avoid intersection test for the next timestep. Suppose that while detecting pull-through, we evaluate bounding spheres $\mathbf{f}_{[s_1,s_2]}(t)$ and $\mathbf{g}_{[s_3,s_4]}(t)$, which respectively bound the parametric intervals $[s_1, s_2]$ and $[s_3, s_4]$ on the two spline segments over the timestep $t \in [0, 1]$, and determine that they do not intersect at the end of the step, thus $\|\mathbf{f}_{[s_1,s_2]}(1) - \mathbf{g}_{[s_3,s_4]}(1)\|_2 = d > 0$. This implies that each point within the intervals $[s_1, s_2]$ and $[s_3, s_4]$ can move by up to $\frac{d}{2}$ without possibly causing pull-through. Let $s' \in [s_1, s_2]$ be arbitrary, $b_i(s)$ be the spline basis functions, and $\mathbf{e}_i$ (which we want to bound) be the allowable movement for the i-th control point before these computed bounding spheres will intersect. Then, we can write

$$\left\|\sum_{i=1}^{4} b_i(s')\,\mathbf{e}_i\right\|_2 \le \sum_{i=1}^{4} |b_i(s')|\ \|\mathbf{e}_i\|_2\ , \quad (8)$$

Introducing d_i as any partition of $\frac{d}{2}$ such that $\sum d_i = \frac{d}{2}$, we can

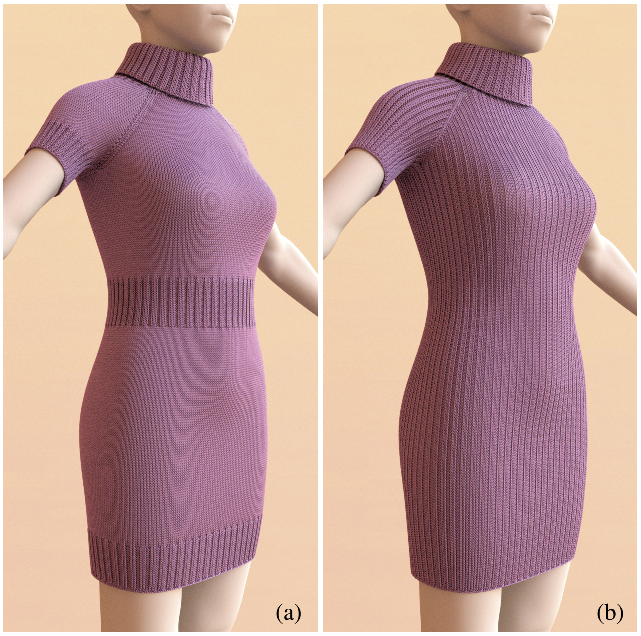

Figure 17: ***Knitted dresses*** *with two different stitch layouts and knitting patterns.*

simplify the equation above as

$$\|\mathbf{e}_i\|_2 \leq \frac{d_i}{|b_i(s')|} \ . \tag{9}$$

While the simplest partition is $d_i = \frac{d}{8}$, this doesn't account for the relative importance of the control points, and so a better choice is $d_i = \frac{|b_i(s')|d}{\sum |b_i(s')|}$ for $s' \in [s_1, s_2]$. We must take the maximum value to compute the bound for the whole interval, such that

$$\|\mathbf{e}_i\|_2 \leq \frac{d_i}{\sum \left(\max_{s' \in [s_1, s_2]} |b_i(s')|\right)} \ . \tag{10}$$

Taking the minimum allowed movement for each control point over all non-contacting spheres as we descend in the hierarchy gives us a bound on the allowable movement for each control point in which we are guaranteed not to have a pull-through for this segment pair. This allows us to bypass the hierarchical descent on future steps if the movement of the control points does not violate these bounds, providing more than an order of magnitude speedup in our tests.

Relaxation: We run the yarn-level relaxation on a model until it converges to a final shape. When the change in the garment shape per time step falls below a certain threshold, we conclude that the rest shape for the yarn-level model has been computed.

Note that a garment can be relaxed on a character with collision constraints. However, since we are not using gravity during the relaxation and we apply forces that preserve the overall garment shape, we may need to simulate the garment on the character with full gravity for a number of frames until wrinkles are formed wherever necessary and a natural garment shape has been reached.

7 Results

We have tested our approach on two fronts: (1) its ability to produce the small-scale details of challenging knitting patterns, and (2) its effectiveness at generating full-scale garments for virtual characters. All stages of our framework are implemented in C++ except for the yarn-level relaxation, which is written in Java.

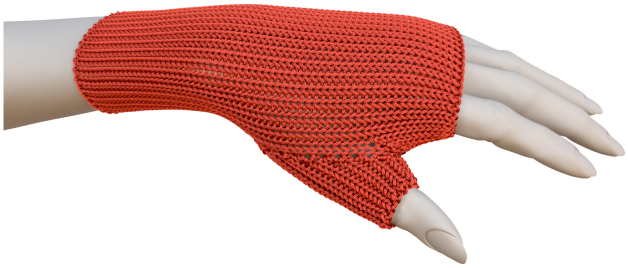

Figure 18: ***Knitted glove*** *with a Ribbing pattern formed by alternating knit and purl stitches.*

The procedure we follow for generating knitting patterns is shown in Figure 14. We begin with a written or visual pattern description for real-world knitting instructions (Figure 14a). Once we understand the pattern description, we can easily model a stitch mesh model in several minutes (Figure 14b). Then we use mesh-based relaxation with no constraints to see the natural shape of the pattern, which typically converges in less than 10 seconds (Figure 14c). Finally, we can generate the yarn curves (Figure 14d) and perform yarn-level relaxation (Figure 14e). The computation time for yarn-level relaxation heavily depends on the total deformation that the yarn curves need to undergo. For example, models with cables take significantly longer to converge to a relaxed state, because the cables introduce large deformations that are not handled by the mesh-based relaxation. The yarn-level relaxation for the model in Figure 14e took about 10 minutes.

We have picked a number of knitting patterns from knitting books with visually different characteristics to test how closely we could replicate them. Considering that the results of knitting a particular pattern depend on the yarn and needles used, as well as the style of the individual knitter, Figure 16 shows that we can qualitatively match the visual appearance of various patterns. It is very difficult to exactly match the results produced by a particular knitter, since various parameters affect the outcome both in the real world and in our yarn-level relaxation procedure. Note that preparing such patterns by manually modeling the yarn curves using standard modeling practices would be extremely difficult.

Figures 1 and 17 through 20 show full scale cloth models prepared using our framework. Some of the knitting patterns we prepared are tiled on these models (using Section 5.3), which shows how our framework can be used to model various knitted garments for virtual characters with realistic details. The computation times for the relaxation operations are provided in Table 2. Both relaxation operations generate relatively large deformations in the beginning of the simulation, and we terminate the relaxation as the magnitude of the deformations per timestep diminish. Notice that we had to use significantly more yarn-level relaxation steps for the model in Figure 21b merely because of the Braid Cables pattern.

8 Limitations and Future Directions

The modeling framework we propose in this paper makes it possible to generate realistic full-scale knitted garments with yarn-level detail. Yet, our framework has certain limitations that might be overcome with future research.

The high computational demand of the yarn-level relaxation stage is arguably the most obvious limitation. In addition, since our stitch models are not optimized for a particular knitting pattern, the effective curve sampling rates we use are higher than in earlier works [Kaldor 2011], leading to higher computational costs. Since yarn-level simulations are computationally expensive, we designed our system such that in a production environment the yarn-level re-

Table 2: ***Performance Results*** *for the Mesh-based and Yarn-level Relaxation Implementations*

		# Stitch Mesh Faces	# Control Points	Mesh-based Relaxation	Yarn-level Relaxation		
					Time/Step	# Steps	Total
Sweater	(Fig.1)	21,568	407,746	∼ 5 min.	∼ 8 min.	50	7 hours
Dress 1	(Fig.17a)	60,560	1,134,597	∼ 6 min.	∼ 10 min.	50	9 hours
Dress 2	(Fig.17b)	65,732	1,194,833	∼ 6 min.	∼ 13 min.	50	11 hours
Glove	(Fig.18)	5,373	88,196	∼ 1 min.	∼ 1 min.	50	1 hour
Tea Cozy	(Fig.19)	10,220	208,506	∼ 1 min.	∼ 2 min.	50	2 hours
Poncho	(Fig.20)	9,976	166,882	∼ 1 min.	∼ 1 min.	50	1 hour
Alien 1	(Fig.21a)	13,440	209,170	∼ 3 min.	∼ 2 min.	50	2 hours
Alien 2	(Fig.21b)	13,440	213,856	∼ 3 min.	∼ 4 min.	500	33 hours
Sheep	(Fig.22)	52,916	1,094,173	∼ 6 min.	∼ 13 min.	100	21 hours

The timings were generated using a dual Intel Xeon X5690 CPU @ 3.46 GHz with 48 GB RAM.

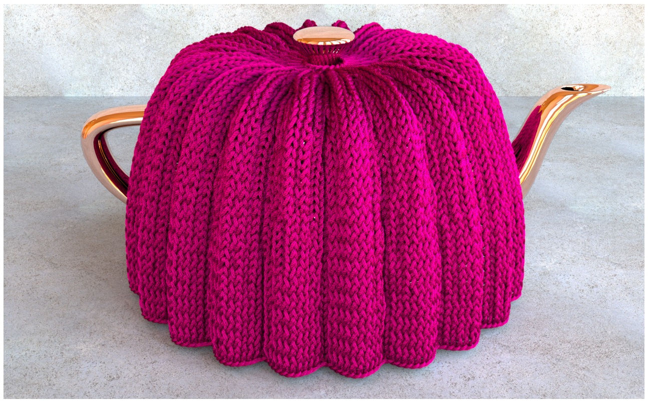

Figure 19: ***Knitted tea cozy*** *with a Stockinette pattern formed by repeated knit stitches. The model shape is defined by the input mesh.*

laxation procedure can be performed overnight after the model is prepared. By constraining the overall shape of the model during relaxation, we try to minimize the possibility of unexpected outcomes and the need for multiple relaxation operations per model. Nonetheless, investigating ways to speed up the yarn-level relaxation step would be a good direction for future research.

While developing our modeling framework, we decided to rely on the topology of the input mesh so that the user can precisely define the desired stitch layout and specify the tessellation value of each input mesh edge. On the other hand, this requires that the user have some understanding of the knitting process while preparing the input mesh, so that the topology and the geometry of the input mesh can be prepared considering the stitch layout and the knitting pattern. Thus, our framework cannot automatically convert any polygonal mesh to a realistic knitted cloth model. An interesting future direction would be decoupling the stitch mesh generation from the input mesh topology in a way that would still allow the user to precisely control the stitch layout over the surface.

The stitch mesh structure provides a very good abstraction for most knit topologies, but obviously it cannot represent everything that can be done with yarn. In our investigations we did not come across any knitting patterns using a single piece of yarn that we cannot represent with the stitch mesh structure [Matthews 1984; Walker 2001]. Even though we have not designed the stitch mesh structure to handle knitting patterns that use multiple yarns with different colors, in computer graphics we always have the option to modify the yarn color as needed to simulate such colored patterns. More importantly, while knitting a real garment, one can always pull loops through anywhere on the garment (not just the loops on the previous row). Such unusual operations are typically reserved for special cases, such as adding a pocket on a knitted garment or knitting pieces of a garment separately, disregarding the knitting direction at the seams, and our stitch mesh abstraction cannot currently handle these cases.

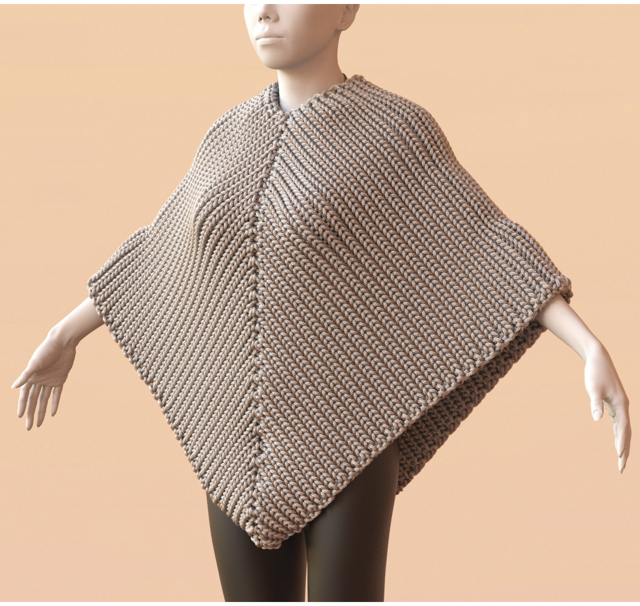

Figure 20: ***Knitted poncho*** *with a Ribbing pattern, simulated on a mannequin after yarn-level relaxation.*

Finally, our system requires the stitch layout over the surface to be carefully constructed so that enough stitches are generated everywhere on the surface to allow the final garment model to conform realistically to the desired surface. While our offline simulation preserves the shape of the garment, individual stitches may have to deform more than desired as a result of the surface constraints. In other words, the relaxation operations can force the knitted garment to take a shape that is not natural for the number of stitches specified. The top row of Figure 23 shows a dress model that does not have enough stitches under the armpit and has too many stitches around the arm, resulting a large hole under the armpit and unintended wrinkle formation around the arm. In our implementation, during and at the end of the mesh-based relaxation, we show the user the stretching of each face by color coding. This lets the user immediately identify undesired stretching or compression and modify the input mesh or the stitch mesh if needed, without having to wait for the yarn-level relaxer to see if a modification would be necessary. This is analogous to knitting in the real world, which often requires repeated measurements of the knitted portions of the garment to see if the number of stitches should be adjusted anywhere. The bottom row of Figure 23 shows the final model after

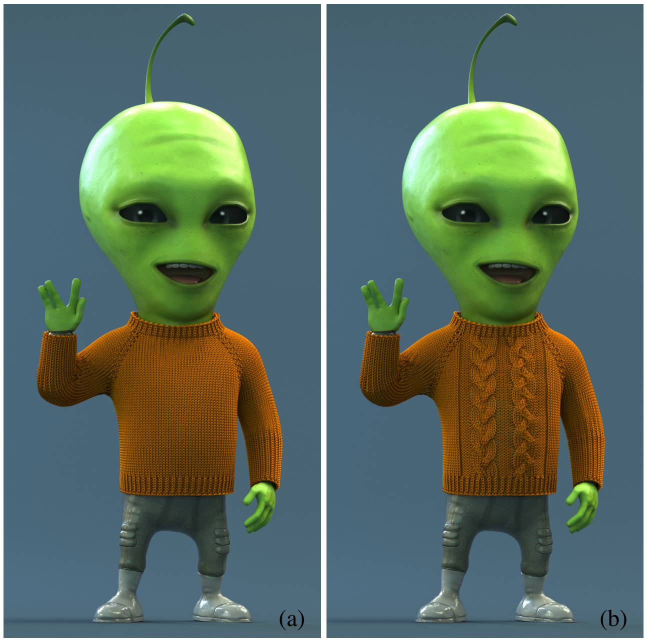

Figure 21: ***A knitted sweater*** *for an alien character and the same sweater model with the Braid Cables pattern.*

the input mesh and the stitch mesh are modified to avoid excessive stretching/compression. An interesting future direction would be semi-automatically modifying the stitch layout to better suit the desired garment shape.

9 Conclusion

In this paper we have presented the first modeler for realistic knitted clothing suitable for computer graphics. The interactive modeling tool provides the user precise control over the placement and the shape of each individual stitch with immediate feedback, and the final model is produced by a physically based simulation that relaxes local yarn shape, while preserving the global shape of the cloth model to ensure predictable results. With this tool it is now possible to create yarn-level models of unprecedented intricacy and structural detail, making fuller use of the potential of yarn-level simulation and rendering to create visually compelling garments of a quality that cannot be approached with sheet-based simulation.

Acknowledgements

We would like to thank **Alexander Tomchuk** (Fig. 1, 17-18) and **Rune Spaans** (Fig. 21-22) for the input models and their help, **Christer Sveen** for the alien character, **Deborah Dawson** for knitting the patterns in Figure 16, Manuel Vargas for the volume generation code, and anonymous reviewers for their feedback. All results are rendered using **Mitsuba**. This material is based upon work supported by the National Science Foundation (NSF) under Grant #1019343 to the Computing Research Association for the CIFellows Project. This work was supported in part by NSF grants HCC-0905506 and IIS-1011919, fellowships from the Alfred P. Sloan Foundation and the John Simon Guggenheim Memorial Foundation, Pixar, and software donations from Autodesk. This research was conducted in conjunction with the Intel Science and Tech. Center - Visual Computing. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or others.

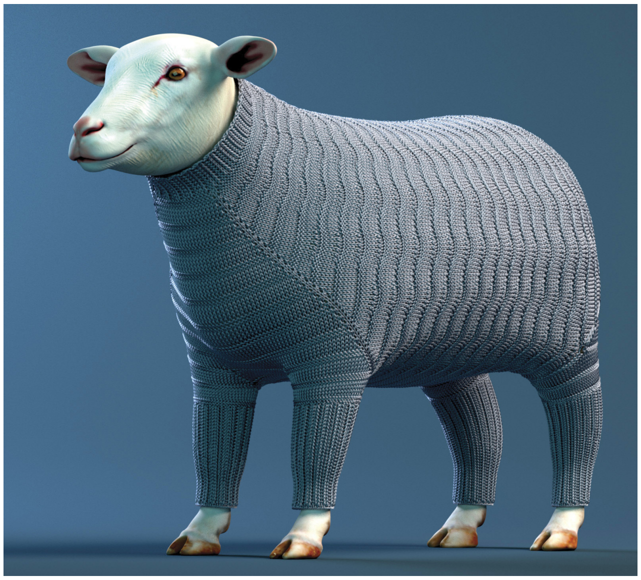

Figure 22: ***Knitted sheep sweater*** *with the Ridged Feather pattern.*

References

Akleman, E., Chen, J., Xing, Q., and Gross, J. L. 2009. Cyclic plain-weaving on polygonal mesh surfaces with graph rotation systems. *ACM T. Graph. (SIGGRAPH'09) 28*, 3, 78.

Allen, P., Barr, T., and Okey, S. 2008. *Knitting For Dummies*. Wiley Publishing.

Baraff, D., and Witkin, A. 1998. Large steps in cloth simulation. *ACM SIGGRAPH'98*, 43–54.

Bergou, M., Mathur, S., Wardetzky, M., and Grinspun, E. 2007. TRACKS: Toward Directable Thin Shells. *ACM T. Graph. (SIGGRAPH'07) 26*, 3, 50.

Bridson, R., Fedkiw, R., and john Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM T. Graph. (SIGGRAPH'02)*, 594–603.

Cabral, M., Lefebvre, S., Dachsbacher, C., and Drettakis, G. 2009. Structure-preserving reshape for textured architectural scenes. *CG Forum (Eurographics) 28*, 2, 469–480.

Carignan, M., Yang, Y., Thalmann, N. M., and Thalmann, D. 1992. Dressing animated synthetic actors with complex deformable clothes. *ACM SIGGRAPH'92*, 99–104.

Catmull, E., and Clark, J. 1978. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6, 350–355.

Choi, K., and Lo, T. 2003. An energy model of plain knitted fabric. *Textile Research Jour. 73*, 739–748.

Choi, K., and Lo, T. 2006. The shape and dimensions of plain knitted fabric: A fabric mechanical model. *Textile Research Jour. 76*, 10, 777–786.

Chu, L. 2005. *A Framework for Extracting Cloth Descriptors from the Underlying Yarn Structure*. PhD thesis, University of California, Berkeley.

Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A., and Cani, M.-P. 2006. Virtual garments: A fully geometric approach for clothing design. *CG Forum (Eurographics) 25*, 3, 625–634.

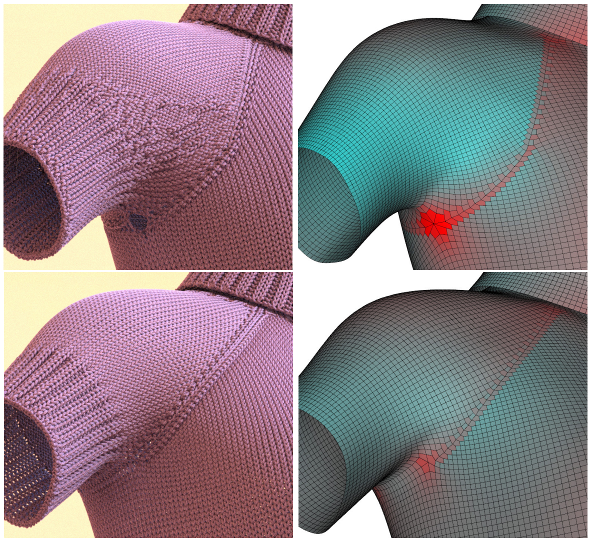

Figure 23: ***Excessive stretching and compression:*** *(Top row) A dress model after yarn-level relaxation with a large hole under the armpit and wrinkle formation on the arm, next to its stitch mesh; (Bottom row) the model generated from a different stitch mesh with less stretching and compression. (stretched* 🟥🟦 *compressed)*

Demiroz, A., and Dias, T. 2000. A study of the graphical representation of plain-knitted structures part I: Stitch model for the graphical representation of plain-knitted structures. *Journal of the Textile Institute 91*, 463–480.

Duhovic, M., and Bhattacharyya, D. 2006. Simulating the deformation mechanisms of knitted fabric composites. *Composites Part A: Applied Science and Manufactur. 37*, 11, 1897–1915.

Eberhardt, B., Meissner, M., and Strasser, W. 2000. Knit fabrics. In *Cloth Modeling and Animation*, D. House and D. Breen, Eds. A K Peters, ch. 5, 123–144.

Floater, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

Göktepe, O., and Harlock, S. C. 2002. Three-dimensional computer modeling of warp knitted structures. *Textile Research Jour. 72*, 266–272.

Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., and Grinspun, E. 2007. Efficient simulation of inextensible cloth. *ACM T. Graph. (SIGGRAPH'07) 26*, 3, 49.

Grinspun, E., Hirani, A., Desbrun, M., and Schröder, P. 2003. Discrete shells. *Symp. on Computer Animation*, 62–67.

Heeger, D. J., and Bergen, J. R. 1995. Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH'95*, 229–238.

Igarashi, T., and Mitani, J. 2010. Apparent layer operations for the manipulation of deformable objects. *ACM T. Graph. (SIGGRAPH'10) 29*, 4, 110.

Igarashi, Y., Igarashi, T., and Suzuki, H. 2008. Knitting a 3D Model. *CG Forum (Eurographics) 27*, 7, 1737–1743.

Igarashi, Y., Igarashi, T., and Suzuki, H. 2008. Knitty: 3D Modeling of Knitted Animals with a Production Assistant Interface. *Eurographics 2008 Annex to Conf. Proc.*, 187–190.

Kaldor, J. M., James, D. L., and Marschner, S. 2008. Simulating knitted cloth at the yarn level. *ACM T. Graph. (SIGGRAPH'08) 27*, 3, 65.

Kaldor, J. M., James, D. L., and Marschner, S. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM T. Graph. (SIGGRAPH'10) 29*, 4, 105.

Kaldor, J. 2011. *Simulating Yarn-Based Cloth*. PhD thesis, Cornell University.

Kurbak, A., and Alpyildiz, T. 2008. A geometrical model for the double lacoste knits. *Textile Research Jour. 78*, 3, 232–247.

Kurbak, A., and Soydan, A. S. 2009. Geometrical models for balanced rib knitted fabrics part III: 2x2, 3x3, 4x4, and 5x5 rib fabrics. *Textile Research Jour. 79*, 7, 618–625.

Kurbak, A. 2009. Geometrical models for balanced rib knitted fabrics part I: Conventionally knitted 1x1 rib fabrics. *Textile Research Jour. 79*, 5, 418–435.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM T. Graph. (SIGGRAPH'03) 22*, 3, 277–286.

Lai, Y.-K., Jin, M., Xie, X., He, Y., Palacios, J., Zhang, E., Hu, S.-M., and Gu, X. 2010. Metric-driven rosy field design and remeshing. *IEEE Trans. on Viz. Comp. Graph. 16*, 95–108.

Luo, Z. G., and Yuen, M. 2005. Reactive 2D/3D garment pattern design modification. *CAD 37*, 6, 623–630.

Matthews, A. 1984. *Vogue Dictionary of Knitting Stitches*. The Condé Nast Publications, Ltd., New York, NY.

Meissner, M., and Eberhardt, B. 1998. The art of knitted fabrics, realistic physically based modelling of knitted patterns. *CG Forum (Eurographics) 17*, 3, 355–362.

Mori, Y., and Igarashi, T. 2007. Plushie: an interactive design system for plush toys. *ACM T. Graph (SIGGRAPH'07) 26*, 3, 45.

Nocent, O., Nourrit, J.-M., and Remion, Y. 2001. Towards mechanical level of detail for knitwear simulation. In *WSCG 2001 Conference Proceedings*, 252–259.

Renkens, W., and Kyosev, Y. 2011. Geometry modelling of warp knitted fabrics with 3D form. *Textile Research Jour. 81*, 4, 437–443.

Robson, C., Maharik, R., Sheffer, A., and Carr, N. 2011. Context-aware garment modeling from sketches. *Computers and Graphics (SMI 2011) 35*, 3, 604–613.

Turquin, E., Wither, J., Boissieux, L., Cani, M.-P., and Hughes, J. 2007. A sketch-based interface for clothing virtual characters. *IEEE Comp. Graph. and Applications 27*, 1, 72–81.

Umetani, N., Kaufman, D. M., Igarashi, T., and Grinspun, E. 2011. Sensitive couture for interactive garment editing and modeling. *ACM T. Graph. (SIGGRAPH'11) 30*, 4, 90.

Volino, P., and Magnenat-Thalmann, N. 2000. *Virtual Clothing: Theory and Practice*. Springer.

Volino, P., and Magnenat-Thalmann, N. 2005. Accurate garment prototyping and simulation. *CAD & App. 2*, 5, 645–654.

Volino, P., Magnenat-Thalmann, N., and Faure, F. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM T. Graph. 28*, 4, 105.

Walker, B. G. 2001. *A Fourth Treasury of Knitting Patterns*. Schoolhouse Press, Pittsville, WI.

Zhou, K., Huang, X., Wang, X., Tong, Y., Desbrun, M., Guo, B., and Shum, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM T. Graph. (SIGGRAPH'06) 25*, 3, 690.