Realistic Rendering and Animation of Knitwear

Yanyun Chen Stephen Lin Hua Zhong
Ying-Qing Xu Baining Guo Heung-Yeung Shum

Contact information for all authors except Hua Zhong:

Microsoft Research, Asia
3F, Beijing Sigma Center
No. 49, Zhichun Road, Haidian District
Beijing 100080, P.R.C.
Telephone: 8610-6261-7711 (Ext. for corresponding author: 5422)
Fax: 8610-8809-7306
Email for corresponding author: bainguo@microsoft.com

Contact information for Hua Zhong, who was an intern at Microsoft Research, Asia, during this project:

Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Abstract

We present a framework for knitwear modeling and rendering that accounts for characteristics that are particular to knitted fabrics. We first describe a model for animation that considers knitwear features and their effects on knitwear shape and interaction. With the computed free-form knitwear configurations, we present an efficient procedure for realistic synthesis based on the observation that a single cross-section of yarn can serve as the basic primitive for modeling entire articles of knitwear. This primitive, called the *lumislice*, describes radiance from a yarn cross-section that accounts for fine-level interactions among yarn fibers. By representing yarn as a sequence of identical but rotated cross-sections, the lumislice can effectively propagate local microstructure over arbitrary stitch patterns and knitwear shapes. The lumislice accommodates varying levels of detail, allows for soft shadow generation, and capitalizes on hardware-assisted transparency blending. These modeling and rendering techniques together form a complete approach for generating realistic knitwear.

Index terms: knitwear, image-based rendering, photorealistic rendering, animation models, parametric surfaces, transparency blending

1 Introduction

The widespread presence of textiles and clothing in everyday life has motivated much computer graphics research on their modeling and rendering [32, 27, 3, 4, 5, 7, 9, 2, 37, 15]. In particular, knitwear poses a considerable challenge because of its special characteristics. Unlike woven fabrics that can be well-represented by specialized BRDF models [36, 31, 8], knitwear is characterized by a macroscopic stitch structure that requires an explicit model of the knitting pattern and geometry. Knitwear additionally differs from other fabrics in that its thickness requires volume modeling and significantly affects its interaction with its environment.

While cloth woven from thread is generally modelled as having zero thickness, the relatively large diameter of yarn and its 3D stitch patterns require consideration of thickness to reasonably exhibit the physical appearance of knitwear as shown in Figure 1. This feature moreover affects the interaction of knitwear articles with other objects. Since knitwear is a volume whose layers of thin fibers are penetrable, the forces counteracting penetration affect knitwear shape and friction.

A second knitwear attribute that we model arises from the yarn loops that comprise stitches. Although stretching and compression of knitwear have been addressed in previous works [9, 26, 20, 2, 29], the effect of gaps within yarn loops has yet to be considered. In our system for fabric simulation, we propose a model that incorporates the behavior associated with these gaps. The inclusion of these special knitwear characteristics in a physically-based model leads to more convincing appearance and animation.

Aside from modeling the behavior of knitwear, challenges exist in realistic synthesis due to the complicated microstructure of yarn. Close examination of yarn reveals countless thin fibers which give knitwear a fuzzy appearance. Improper rendering of this delicate structure can result in aliasing that causes a scene to look artificial. Given this fine-level complexity, rendering these fibers with existing techniques such as [1, 33, 28, 16, 10, 22] is cumbersome, if not impossible. Moreover, the appearance of fibers changes in detail with different viewing distances.

An additional difficulty in rendering knitwear is its variations in shape. The microstructure of flat knitwear has been efficiently displayed using volume-rendering techniques [12]; however, the only known techniques for rendering free-form knitwear with microstructure utilize Gouraud-

shaded triangles [37], which do not handle close-up views, and curved ray tracing [13], which is computationally expensive and does not effectively deal with multiple scattering that is especially evident in light-colored yarn.

Photorealistic rendering of knitwear must efficiently handle free-form surfaces, varying levels of detail from different viewing distances, and complex stitch geometry. We address these challenges with a primitive element called the *lumislice*, which we first introduced in [35]. The lumislice represents radiance from a yarn cross-section, and arises from the key observation that based on the repetitive structure of yarn, articles of knitwear may be composed entirely of lumislices. This suggests a two-level structural hierarchy that can be formed for yarn to facilitate efficient photorealistic rendering. Locally, fine-level interaction among yarn fibers (occlusion, shadowing, multiple scattering) is modelled in the lumislice for a representative yarn cross-section. Globally, the repetitive nature of knitted yarn and its known path along a skeleton allows knitwear to be represented as a collection of identical lumislices in various positions and orientations, where we model the fiber density distribution as being the same for all yarn cross-sections.

It is this detailed but flexible quality that gives the lumislice much utility. Local microstructure can be closely represented in the lumislice, yet it can easily be disseminated to all points of a knitwear. The lumislice furthermore does not introduce restrictions on the global structure of the knitted yarn. The advantages elicited by our lumislice-based method are listed as follows:

- Fine local interaction can be precomputed and used throughout a piece of knitwear.
- Different levels of detail for varying viewing distances are elegantly handled by multiresolution lumislices, in a manner similar to mip-maps.
- The lumislice can be applied to arbitrary free-form surfaces, facilitating physically-based animation and other applications.
- The lumislice can be used for rendering arbitrary stitch patterns with better visual effects and lower cost than raytracing. Existing volumetric representation methods have significant difficulties in dealing with complex stitches on a deformable surface [12, 13].
- It can easily be implemented by standard graphics APIs, such as OpenGL.

Additionally, enhanced realism can be achieved with a proposed shadow map for yarn that is integrated into our method for soft shadow generation. Inclusion of such shadows allow individual yarn fibers to be distinguished, as deep shadow maps have done for hair [19].

This method efficiently produces knitwear rendering results not previously achievable. Examples of lumislice rendering are displayed in Figure 2 for different viewing situations. The close-up views on the left side exhibit the delicate fiber characteristics of knitted yarn, and from a normal viewing distance shown on the right side, the rendering of variable macroscopic structure is demonstrated. The efficient and effective generation of this range of details distinguishes lumislice-based rendering. The lumislice may furthermore be extended to other materials consisting of subpixel elements such as hair, carpet or distant foliage. Applications of photorealistic knitwear rendering include web-based display of clothing products and fabric animation on moving characters. Automated generation of knitwear would particularly be desirable for motion pictures such as Stuart Little, where the fur and clothing required significant labor for generation [24].

The remainder of this paper is organized as follows. Section 2 provides an overview of related work on fabric and knitwear modeling and rendering. In Section 3, we describe our basic model for the knitwear skeleton and stitch patterns. We present in Section 4 our knitwear representation for animation that includes special knitwear features, and then discuss their effects on animation. The lumislice and knitwear rendering are described in Section 5, and simulation results of our system are exhibited in Section 6. Lastly, we present conclusions in Section 7.

2 Related Work

Realistic motion of cloth has been an area of interest in computer graphics, but these previous works have generally dealt with finely woven fabrics rather than knitwear. Works for woven fabrics include those based on finite element simulations [27] and on particle systems [2, 29], in which the fabric is represented by a set of particles that interact with each other. The aggregate motion of these particles produces the global behavior of the fabric. Breen *et al.* [4] developed a particle-based system for static cloth draping where local interactions are defined by energy functions approximated from real cloth draping data.

Particle-based systems were later applied to knitwear, where spring meshes are used for modeling knitwear deformations [9, 26, 20]. In our work, we consider additional knitwear characteristics such as knitwear thickness and gaps of space in the knitwear yarn loops. The modeling of these additional factors results in increased realism of knitwear appearance and motion.

For rendering, there have been several previous methods that deal with knitwear or other similar materials. An initial approach to rendering of complex repetitive patterns uses volumetric textures [16, 22]. The teddy bear rendered by Kajiya and Kay's original volumetric texture model [16], called the texel, remains one of the most successfully synthesized furry objects to date, and its associated ray-tracing algorithm continues to be one of the most general realistic rendering techniques for such scenes.

Other approaches for rendering furry objects include the so-called fake fur rendering that was introduced by Goldman [10] and the real-time fur rendering method introduced by Lengyel [17]. Goldman used a probabilistic lighting model for rendering a thin coat of fur on skin seen from normal distances. Lengyel used an alpha-blending technique to approximately render fur represented by volumetric texture.

Knitwear has unique features different from fur and other fabrics that should be considered when rendered. As previously mentioned, the microstructure of knitted yarn typically consists of a large number of thin fibers, and the size of yarn and stitches exceeds that of thread, thus precluding representation by BRDF models. Graphics researchers have successfully applied volume-rendering techniques to efficiently display yarn microstructure for knitwear on a flat surface [12], but for free-form knitwear, few techniques for rendering yarn microstructure exist [13, 37].

The curved ray-tracing method introduced in [13] has three major drawbacks. One is the high cost of rendering. Second, users must build different volumetric textures for different knitting patterns. This makes it difficult to model knitwear with advanced stitching patterns. Third, evident artifacts arise when the knitwear is severely deformed. Zhong *et al.* developed an effective model for rendering yarn microstructure using Gouraud-shaded triangles [37]. Their method works well for situations of normal viewing distances, but it does not address close-up views.

Because of the detailed structure of knitwear, past methods are burdened with high computational costs and/or large storage requirements for synthesis of photorealistic free-form knitwear

over multiple scales. We approach this problem by developing a reflectance primitive for yarn that concisely represents microstructure in a manner that allows for various levels of detail and hardware-assisted rendering.

To avoid the high cost of raytracing, volumetric objects can be rendered by hardware-aided texture mapping with transparency blending [7, 21, 17]. However, limitations arise from the use of graphics hardware. First of all, it cannot in general compute the shading or shadowing for each individual texture pixel [21], with the exception of GeForce3 which can do per-pixel lighting but cannot capture multiple scattering nor the attenuation effect of light passing through yarn. Secondly, if 2D alpha texture is employed, the volumetric texture should be split into slices parallel to the viewport and sorted from far to near for accurate blending computation [7].

While splitting and sorting can be solved, the rendering result would likely appear artificial if shading and shadowing cannot be accurately computed according to the lighting and viewing conditions of the volumetric texture. A way to achieve realistic shading effects is to compute the results of all possible lighting and viewing conditions offline, and save them in a volumetric texture data structure. However, a moderately complex scene would require a prohibitive amount of storage and computation resources.

To address this problem, a key observation is made. An examination of knitwear reveals that although patterns of knitted yarn can be complex, they are entirely composed of a single basic element. For a given type of yarn, the distribution of fibers for each cross-section is similar. To make our knitwear rendering goals realizable, we model a yarn strand by winding a fixed 2D fiber distribution along the path of the yarn. Hence, we can completely synthesize knitwear from a single volumetric yarn cross-section. To exhibit microscopic realism, this cross-section incorporates fine visual effects, such as self-occlusion, self-shadowing, and multiple scattering. For different levels of detail, the resolution of this slice can be varied with smooth visual transitions. It is compactly represented in 2D as a single thin volumetric layer and can be computed offline. In addition, this structure is amenable to transparency blending for efficient rendering.

3 Knitwear Model

In our work, the knitwear model begins with a free-form surface, a stitch pattern and optionally a color pattern. We take the global structure of knitwear given by a free-form surface s and parameterize it by indices (u, v) such that the (u, v) correspond to the yarn loops as exhibited in Figure 3. These loops are determined by the stitch pattern and have the color given by the color pattern.

As shown in Figure 3, each yarn loop is defined by six points $\{k_i = (u_i, v_i) \mid 0 \leq i \leq 5\}$ in the (u, v) parameter domain that map to six key points $\{\mathbf{s}(u_i, v_i) \mid 0 \leq i \leq 5\}$ on the 3D knitwear surface $s(u, v)$. For a given loop as exhibited in the figure, we define the points $\{k_i = (u_i, v_i) \mid 0 \leq i \leq 5\}$ by

$$k_0 = 0.35(n_3 - n_0) + n_3,\ k_1 = 0.375(n_2 - n_3) + n_3,$$

$$k_2 = 0.125(n_1 - n_0) + n_0,\ k_3 = 0.35(n_4 - n_0) + 0.5(n_0 + n_1),$$

$$k_4 = 0.875(n_1 - n_0) + n_0,\ k_5 = 0.625(n_2 - n_3) + n_3.$$

The corresponding 3D key points $\mathbf{s}(u_i, v_i)$ can be slightly offset to account for random loop variations, and then are interpolated using cubic cardinal splines to give the yarn path.

The use of key points allows us to easily model advanced stitch patterns by arbitrary placement of the points. As described in instructional knitting books such as [14], advanced stitches can be produced by combining basic stitches, as exhibited in Figure 4. The key points of the two basic stitches in (a) can be repositioned to give the stitch shown in (b). This alteration may introduce thread collisions, as seen in the red circles. To remedy this problem, additional key points are added as shown in (c) and are assigned appropriate offsets to avoid collision. In this case, the points are offset by the yarn diameter in the normal direction of the surface. Arbitrary stitches can be represented by key points and collision avoidance, and are easily incorporated into our system.

4 Knitwear Animation

To account for the special characteristics of knitwear, we enhance the basic knitwear model of the previous section to allow for more physically realistic shape appearance and motion. In our anima-

tion model, a simple mass-spring mesh is employed to handle basic fabric motion, and integrated into this basic framework are novel features that are particular to knitwear.

4.1 Animation Model

Characteristics of knitwear that are of relatively little significance in finely-woven fabrics include deformations of yarn loops and knitwear thickness. These features result in physical behaviors such as multiple stable configurations, volume and friction, which are essential for realistic animation of knitwear. In this subsection, we present our knitwear animation model and its resulting effects.

4.1.1 Yarn Loop Deformations

Yarn loops that comprise knitwear are represented in a mass-spring mesh like the one used in [29] such that the mass points correspond to the (u, v) indices in the parametric surface of the preceding section. This correspondence relates the yarn loops, which are the basic elements of knitwear, to the fundamental structures of the mass-spring system so that a knitwear loop reacts to external forces like a combination of springs.

The length of the mesh springs is related to four external forces: stretching, compression, bending and shearing. These forces affect the lengths l of their associated springs according to a set of basic formulas. For stretching

$$|F_{stretch}| = c_0 * e^{(l-\theta_1)/\theta_1 - 1}$$

when $l > \theta_1$, and for compression

$$|F_{compress}| = c_1 * e^{(\theta_2 - l)/\theta_2 - 1}$$

when $l < \theta_2$. θ_1, θ_2 each represent the balance length of the spring when there are no external forces, and c_0, c_1 are constants. Bending and shearing forces can be expressed proportionally to the angle between neighboring loops, with respective constants c_2 and c_3. In this paper, the mechanical simulation model we use is very simple, to more clearly highlight the novel features

we introduce for knitwear. The presented knitwear features could potentially be combined with a more sophisticated model such as [2, 29] used for general cloth simulation.

One attribute of knitwear is the free spaces between stitch loops. Because of these gaps, loops are loosely linked with one another, and consequently some change in loop shapes and positions should be possible without stretching or compressing the yarn. To account for this property, we represent the balance length in the spring stretching and compression equations not with a single point where $\theta_1 = \theta_2$, but with an interval where $\theta_1 > \theta_2$. This allows these springs to vary in length by a certain amount without the presence of stretching or compression forces. In our implementation, we modeled a 5% change in length corresponding to values of $\theta_1 = 0.95$ and $\theta_2 = 1.05$, but different values may be used depending on the desired tightness of the weave.

Because of the physical correspondence between the mass-spring mesh and yarn loops, this model simulates only the basic stitch structure. We did not specifically address complex stitch patterns, since a physical correspondence cannot easily be made with a 2D mass-spring mesh. Nevertheless, the behavior of complex stitches can be approximated with our elementary mesh by empirical adjustment of the model parameters. More precise modeling is an area of future work.

4.1.2 Thickness

The structure of springs and masses forms merely a 2-D surface that we represent with a triangle mesh, but knitwear requires thickness modeling for it to exhibit realistic motion and interaction with objects. Because of the fiber protrusions and the compressibility of yarn in its normal direction, pressure placed against the thickness of knitwear will be opposed by a repulsion force from the volume layer. This repulsion increases as an object more deeply penetrates the knitwear volume. We express this physical characteristic with a force field model.

At a distance d from the center of a knitwear with thickness $2T$, an object is affected by the force field according to the following empirical equations:

$$|F_{thickness}| = c_5(T-d)^2 \quad \text{for} \quad \tau \leq d < T \tag{1}$$

$$|F_{thickness}| = c_6 e^{\alpha/(d+\beta)} \quad \text{for} \quad 0 < d < \tau \tag{2}$$

where c_5, c_6, α, β are constants, and τ is a parameter less than T that denotes the thickness of the knitwear yarn structure excluding the fiber protrusions. When an object penetrates only the fiber protrusion layer, (1) is used, and when an object enters deeper into the fabric and penetrates the main yarn structure, (2) is utilized instead. The values of α, β must be selected to ensure continuity between the two equations.

For an object point, its distance to a given triangle in the knitwear mesh is denoted by d. If this distance is less than T and the angle of the distance vector from the triangle's normal is less than a specified value, which is 45 degrees in our implementation, then this point is considered to be affected by the force field of this triangle. The intensity of this repulsion force as computed in (1) or (2) is in the direction of the distance vector, and an object point can potentially be affected by more than one triangle.

4.1.3 Friction

The modeling of knitwear thickness with a force field leads to realistic interaction effects between a knitwear and its environment. Movement of a knitwear article can produce opposing forces when the knitwear is in contact with other objects or itself. We handle this friction as two components, static and dynamic, formulated from basic physics.

For a force to move knitwear, it must exceed a certain threshold τ_s defined by the static friction coefficient K_s and the pressure $|F_{thickness}|$ from the force field equations (1) and (2):

$$\tau_s = K_s |F_{thickness}|.$$

If the knitwear moving force does not exceed this value, then it is effectively nullified by an opposing force of the same magnitude.

If the static friction threshold is surpassed, then the knitwear will move under the resistance of the dynamic friction force F_d opposite to the movement direction:

$$|F_d| = K_d |F_{thickness}| \tag{3}$$

where K_d is the dynamic friction coefficient.

More complex is the case of self-friction, friction caused by knitwear self-collisions. In the previous section, we discussed the force field model of knitwear thickness and also the range of influence for each triangle in the polygonal mesh. When the ranges of influence for two non-adjacent triangles overlap, the triangles exert an equal and opposing force on each other. This force is translated into friction using the same dynamic friction equation, but the coefficients c_5, c_6 for $|F_{thickness}|$ in (1) and (2) are replaced by different values c_7, c_8. Static self-friction is ignored in our model, since it is difficult to formulate reasonably, and the result is nonetheless convincing without this factor.

The final friction force we consider is viscosity, or friction with the air. We model this simply with the equation

$$\vec{\nu} = -c_9\vec{v}$$

where c_9 is a constant and $\vec{v}$ is the velocity vector of a knitwear point. The friction forces described in this subsection together lead to more physically plausible interaction of knitwear articles with other objects and itself.

4.2 Implementation Details

In this subsection, we present our algorithm for realistic animation from the preceding formulations. The algorithm is followed by a smoothing stage that provides stability to the integration of all the mesh forces.

Before outlining the steps of our algorithm, we note that a self-collision in knitwear simulation is a "soft" collision, because of the force field surrounding the triangle mesh. These forces physically should prevent mesh triangles from penetrating each other, but because of calculations at discrete time intervals, triangle penetration becomes a possibility. Since this is not physically accurate, it is necessary to retreat in time and then proceed again in smaller time increments for more precise computation.

The algorithm begins with initializing the positions of the mesh points in the mass-spring system and of the object points in the environment. Then for each time step, which is set by default

according to 25 frames per second, the following steps are performed:

1. Clear the force accumulator for each mass point.
2. For each mass point, calculate the spring forces (stretching, compression, bending and shearing) and gravity, then add them to its force accumulator.
3. Perform collision detection, and add the resulting forces to the corresponding force accumulators.
4. Calculate friction and add it to the force accumulators.
5. Perform triangle penetration detection, and if penetration is detected, repeat this iteration but reduce the time step by half.
6. Apply the Courant-Friedrichs-Levy (CFL) condition to adjust the time step.
7. Using force accumulator values, update the positions, velocities and accelerations of the mass points with second-order Runge-Kutta integration.

For the time steps in the above algorithm, we begin each iteration with the default value, and at item 5, we check whether this time step would result in triangle collision. If a collision would occur, the iteration is repeated at half the current time step, until collision is avoided. Subsequently in item 6, we check whether this time step satisfies the CFL condition for integration stability. If it does not, then the time step is reduced to the maximum value that satisfies the CFL condition. Although beginning each iteration with a default time step might not be computationally optimal, it nevertheless yields effective results for offline processing.

Though we utilize an adaptive time step with the CFL test to facilitate integration stability, the integration result can nevertheless exhibit some jitter in parts of the knitwear due to fast fluctuations of the viscosity and friction forces. This jitter can be considered as a high frequency noise of the forces on the mass points. This problem could be handled by applying a lowpass filter to the force of each mass point, but we instead implement this by filtering the position result, since acceleration is proportional to force, and filtering the position is equivalent to filtering the acceleration. This is

simple to implement, and our simulations have shown that a simple averaging filter can effectively stabilize an otherwise jittery result.

The algorithm for collision detection is a standard $O(n^2)$ algorithm that performs well for detailed meshes such as our knitwear model with two triangles per knit loop. Each knitwear vertex is checked for penetration through any of the scene triangles including the knitwear itself, with some computational acceleration based on bounding boxes.

By consideration of fabric characteristics particular to knitwear, this method can produce animation results with realistic shape and feasible interaction with the environment.

5 Photorealistic Rendering

From our presented model for realistic animation, the remaining challenge is to render knitwear in the computed configurations. But similar to the differences in animation between knitwear and other fabrics, knitwear characteristics lead to unique obstacles in rendering as well. We handle these difficulties with our lumislice modeling primitive.

An overview of our rendering approach is illustrated in Figure 5. In Sections 2-4, we described the movement of knitwear control points and their corresponding placements of key points for the yarn path. This knitwear skeleton is divided into yarn segments that are further divided into cross-sectional volumetric slices. The reflectance characteristics of these yarn cross-sections are each represented by the lumislice, the basic structural element for the entire knitwear. In this section, we present the lumislice model and then describe lumislice-based rendering for knitwear.

5.1 Lumislice Model

The lumislice is the fundamental structure we introduce for knitwear rendering. It is this element that facilitates the rendering of knitwear at different viewing distances while maintaining appropriate microscopic and macroscopic features.

For a given illumination direction, the lumislice is essentially a computed Lumigraph or light field [11, 18] for a cross-sectional slice of yarn. The slice is composed of voxels, each with an associated four-dimensional array that contains its opacity and reflectance function. This reflectance

function represents the brightness of a voxel viewed from direction $V(\theta_v, \phi_v)$ when illuminated by a unit intensity light from direction $L(\theta_l, \phi_l)$, and it differs from the traditional notion of BRDF in that it accounts for the attenuation of incident light passing through the surrounding yarn. Using this *voxel reflectance function* (VRF) allows us to precompute the effects of the surrounding yarn.

In this subsection, we present the method for computing the lumislice for a given type of yarn, and then describe the calculation of viewed radiance from a lumislice voxel.

5.1.1 Lumislice Computation

As mentioned, each lumislice voxel has two associated quantities, the opacity of its collective yarn fibers and the VRF. The opacity value is derived from the fiber density distribution, illustrated in Figure 6, which is assumed to be identical for all cross-sections of a given yarn. Different types of yarn may have different density distributions.

Our formulation of the VRF is presented in terms of R_p, the outgoing radiance from a voxel p. Three primary physical elements factor into the reflectance at p: the fiber density ρ_p, the shading model Ψ of yarn, and the incident light distribution I_{msp} that results from multiple scattering among neighboring voxels N. The emission intensity from a voxel can in principle also affect a VRF, but this quantity for yarn is zero. These factors influence R_p according to the equation

$$R_p = \rho_p \Psi (I_p + I_L \sum_N I_{msp}) \tag{4}$$

where I_L is the light intensity and I_p is its attenuated intensity upon reaching p, as shown in Figure 7. The shading model Ψ of [16] for diffuse reflection is used, and the multiple scattering term is described in [35]. Based upon the emission-absorption model of volume illumination [6], I_p may be expressed as

$$I_p = I_L e^{-\gamma \sum_{r=p}^{P_{in}} \rho_r} \tag{5}$$

where P_{in} is the point of light entry into the yarn bounding box, γ is a light transmission factor in the emission-absorption model, and voxel dimensions are unit length. Then (5) can be substituted

into (4) to produce

$$R_p = \rho_p \Psi I_L (e^{-\gamma \sum_{r=p}^{P_{in}} \rho_r} + \sum_N I_{msp}). \tag{6}$$

R_p can then be restated in terms of the spherical angles in Figure 7:

$$R_p(\theta_l, \phi_l, \theta_v, \phi_v) = I_L C_p(\theta_l, \phi_l, \theta_v, \phi_v) \tag{7}$$

where

$$C_p(\theta_l, \phi_l, \theta_v, \phi_v) = \rho_p \Psi(\theta_l, \phi_l, \theta_v, \phi_v)[e^{-\gamma \sum_{r=p}^{P_{in}} \rho_r} + \textstyle\sum_{i \in N} I_{msp}(\theta_i, \phi_i)]. \tag{8}$$

Here, C_p is the VRF, which is a 4D function of θ_l, ϕ_l, θ_v, and ϕ_v. It can be approximately represented by a 4D color array after discretization of the four angles, where the discretization increments are determined by the accuracy required. For all lumislices used in this paper, the longitude angle $\theta \in [0, 2\pi]$ and altitude angle $\phi_l \in [-\pi/2, \pi/2]$ are discretized into 32×21 directions, such that the 2D lumislice is computed from a 3D swept rotation of yarn cross-sections. In normal situations, the distance between the knitwear and the light source greatly exceeds the radius of the knitted yarn, so we assume parallel illumination. The spacing of the lumislices along yarn segments is related to the resolution of the lumislice, which is presented in Section 5.2.2 on levels of detail.

For additional simplification, the specular component of the VRF is ignored in our implementation, resulting in a view-independent reflectance function. This allows C_p to become a 2D function, and consequently the computational costs and storage requirement are significantly reduced. All lumislices used in this paper consider only the diffuse effect.

The lumislice of a given yarn is precomputed and stored in tables, with the VRF in a 2D RGB color table indexed by (θ_L, ϕ_L), and the opacity in an alpha table.

5.1.2 Volumetric Yarn and Radiance Calculation

Since the same lumislice represents all cross-sections of a knitted yarn, it is rotated and translated along the yarn path to form the yarn volume, as shown in Figure 6. The radiance R of the knitwear

at a given image pixel from viewpoint V in Figure 8 can be computed as

$$R = \sum_{p=P_{near}}^{P_{far}} e^{-\gamma \sum_{s=P_{near}}^{p} \rho_s} \cdot R_p \tag{9}$$

where R_p is calculated from (4).

When the knitwear is rendered, all yarn segments are subdivided into a series of volumetric cross-sections, each represented by the lumislice. When a slice is drawn, its location, its local coordinate system, the light source position and the viewpoint together determine the reflection angles θ_L, ϕ_L, θ_V, and ϕ_V, as shown in Figure 7. These angles, specifically only θ_L and ϕ_L, are then used to index the 2D color table of the lumislice. This color texture and the opacity table of the slice together form the RGBA transparency texture, which is rendered by the texture mapping and transparency-blending functions of standard graphics APIs [21].

The path of the yarn, and likewise the arrangement of the lumislices, is given by the knitwear skeleton. An important feature of this representation by key points is that it becomes simple to model and edit advanced stitch patterns [14]. This free-form structure for knitwear complements our lumislice rendering model.

5.2 Rendering

For rendering, the lumislice model is first incorporated into the knitwear skeleton. The yarn along the skeleton is divided into a series of short straight segments with a twisting angle. These slices undergo depth sorting and slice partitioning, which is needed because the transparency-blending function supported by graphics hardware is computed along slices parallel to the viewport, while the lumislices can have various orientations. The details of our depth sorting and slice partitioning are presented in [35].

With these slices, we describe the rendering issues of shadow casting and levels of detail, and this is followed by a step-by-step description of our rendering algorithm.

5.2.1 Shadows

Shadows are an essential factor for realism [30, 25], and there exist many well-known techniques for shadow generation. The shadow map method introduced by Williams [34] is perhaps the most basic. This technique was improved by Reeves *et al.* [23] and was extended to the so-called deep shadow map by Lokovic and Veach [19]. Shadows are particularly important for realistic rendering of knitwear because they not only indicate the spatial relationship between knitwear and other geometric objects, but also visually reveal the knitting pattern of yarns. Figure 9 exemplifies the significant effects of shadows.

Deep shadow maps can deal with semi-transparent or sub-pixel objects such as hair, fur and smoke because they represent the fractional visibility through a pixel at all possible depths, rather than only a single depth value. However, this technique is incompatible with the lumislice, because part of the light attenuation has been already computed and stored in the lumislice to provide better effects of yarn microstructure. We therefore extend the traditional shadow map to handle yarn, an object with geometry not clearly defined. Experimental evidence suggests that while shadows are important in 3D rendering, they need not be exact [30, 25], so we develop a rough shadowing technique consisting of two steps.

Like traditional shadow map techniques, the first step of our method places a camera at the light source, making sure that all shadow generators and receivers are within the field of view. All objects except for knitwear are then projected to produce a shadow map containing a depth value. For knitwear, only the centerlines of yarns are projected to the shadow map. Thus the shadow map we build will contain depth information of geometric objects and yarn centers.

Next, the effect range R_y of yarn in the shadow map is computed according to the radius of yarn. This parameter is used to determine whether light is partially absorbed by the yarn before reaching more distant positions.

Finally, before a point p is rendered, it is projected onto the shadow map for shadow testing. If the projection result $p_s(x_s, y_s, z_s)$ indicates total occlusion by geometric surfaces, then it is in shadow. Otherwise, the yarn-occlusion test is performed by computing on the shadow map the closest distance d from (x_s, y_s) to the centerline of any yarn. If d is smaller than R_y, light is

partially absorbed by that yarn. Given the fiber density distribution ρ as shown in Figure 10, we compute the light transmission for distance d through voxels (d, i) of the lumislice as

$$T_y(d) = 1 - k_0 \frac{1 - \prod_{i=-R_y}^{R_y} e^{-\gamma\rho(d,i)}}{1 - \prod_{i=-R_y}^{R_y} e^{-\gamma\rho(0,i)}} \tag{10}$$

where we empirically take $k_0 = 0.8$ and the voxel dimensions to be unit length.

The lumislice-based method renders knitwear by a volumetric yarn slice using hardware-aided transparency blending. As described in [21], shadow cannot be calculated for every pixel on a texture if the alpha-texture is employed to render volume-represented objects. Thus the shadow test can be performed only at the slice vertices, and the result is blended to the lumislice by adjusting the texture brightness so that the shadow effect can be seen. If higher accuracy is required, such as for rendering close-up views, the slice could be divided into sub-slices so that shadow computation can be done at more points to improve the result quality.

5.2.2 Level of Detail Representation

To handle a wide range of viewing distances, a view-dependent level of detail representation of the lumislice is utilized to reduce the computational cost, to ensure smooth visual transitions, and to reduce rendering error. When the viewing distance changes, the projection size and visible features of a yarn slice on the screen should change accordingly. A lumislice of fixed resolution will cause error because there is no simple method for determining proper opacity values even though the color values of the lumislice can be obtained by using a mip-map. For this reason, we precompute lumislices of different resolutions and render each slice with the one most appropriate to its current viewing condition.

For decreasing resolution of the lumislice, its unit of thickness should increase by the same factor. For instance, we can form a low-resolution lumislice from a high-resolution slice by doubling its thickness, then grouping together 2x2 blocks. Locally, a voxel on a low-resolution slice can be computed from corresponding voxels in a high-resolution slice such that its low-resolution density ρ' and its color C' are

$$\rho' = \frac{\rho_1 + \rho_2 + \rho_3 + \rho_4}{4}$$

$$C' = \frac{C_1\rho_1 + C_2\rho_2 + C_3\rho_3 + C_4\rho_4}{\rho_1 + \rho_2 + \rho_3 + \rho_4} \tag{11}$$

where C_1, C_2, C_3 and C_4 are colors of the high-resolution slice, and ρ_1, ρ_2, ρ_3 and ρ_4 are densities of the high-resolution slice. The formulation of ρ' can be derived from the resultant transmission T' from the voxel grouping:

$$T' = \sqrt{T_1T_2T_3T_4} = \sqrt{e^{-\gamma\rho_1}e^{-\gamma\rho_2}e^{-\gamma\rho_3}e^{-\gamma\rho_4}} = e^{-2\gamma\frac{\rho_1+\rho_2+\rho_3+\rho_4}{4}} = (e^{-\gamma\rho'})^2$$

where the squaring factor results from thickness doubling. For more accurate results when the resolution is below 8×8, we form the low-resolution lumislice by sampling the high-resolution lumislice.

The resolution for rendering a slice depends upon the projection area of the slice on the screen. For a projection whose dimensions are at most m pixels, a lumislice with resolution $2^n \times 2^n$ is chosen such that n is an integer and $2^n \geq m$, to prevent aliasing. Although our implementation precomputes lumislices at intervals according to 2^n, finer intervals may be used for more accuracy.

The unit length corresponding to the determined resolution also describes the thickness of the lumislice. Lumislice densities along yarn are determined by dividing yarn segments by these computed resolution units. So if we were to instead consider lumislices as two-dimensional, then the resolution unit describes the distance between lumislices along yarn segments.

5.2.3 Rendering Algorithm

From the described processes, we outline the steps of the rendering algorithm, given models of the lumislice and the knitwear skeleton. The use of hardware-aided graphics APIs entails the masking or disabling of the z-buffer depth test for accurate transparency-blending computation. Consequently, scene rendering requires more than one step. Our rendering routine proceeds as follows:

1. Create the shadow map as described in Section 5.2.1.

2. Using the ID color method, draw the color and depth of geometric objects as seen from the viewport into Buffer1.

3. Draw yarn as cylindrical polygons on the result of step 2, then turn the resulting image into a bitmap called Image1. Pixels that are covered by yarn are set to 0, and others are set to 1.

4. Draw the geometric objects with the shadow test under the same projection condition and then save as Image2.

5. Sort all discrete yarn segments by distance from the viewpoint.

6. Disable the depth test, draw volumetric yarn on the result of step 4. Before each slice is rendered, project its vertices into Buffer1. If it is totally occluded by geometric objects, it is not rendered. Render all slices of the yarn segments from far to near. Save the result as Image3.

7. For every pixel in Image3, check its associated value in Image1. If the value is 1, then this pixel should display a geometric object, so the color of this pixel is replaced by the color of the corresponding pixel in Image2. This step is necessary to properly process slices that are partially occluded. The final result is obtained after all the pixels are checked.

6 Results

Several synthesis results of our method are displayed in this section. Figure 11 exhibits a range of complex stitching patterns that can be rendered with our technique. The soft shadows in the knitwear are evident and contribute significantly to the realism of the images.

Another set of renderings is shown in Figure 12. The left column displays real images of a scarf, while renderings from corresponding view distances are shown on the right. Comparison of the two columns demonstrates the similarity of our synthesis results to actual knitwear. Lastly, Figure 14 shows a sweater that displays deformations that can easily be made to fit the form of a given person.

These renderings were performed on a 864MHz Pentium III with 256M RAM and a Matrox Millenium G400 DualHead Max display adapter. For the images in Figure 11, the number of rendered slices are, clockwise from the upper left, 211k, 215k, 225k and 216k. The respective

rendering times in minutes using our unoptimized implementation are 14.30, 15.07, 15.33 and 15.08. The rendering time for the sweater in Figure 14, which consists of 368k slices, is 31.93 minutes.

The results of our knitwear animations, which utilize lumislice-based rendering, are exhibited in a few snapshots. The spring parameters used in our simulations are as follows: 1500.0 for stretching c_0 in the wale direction, 2000.0 for stretching c_0 in the course direction, 3000.0 for compression c_1, 100.0 for bending c_2 in the wale direction, 150.0 for bending c_2 in the course direction, and 1000.0 for shearing c_3, c_4. For an initial balance length between mass points of l, the balance length parameters are $\theta_1 = 0.95l$ and $\theta_2 = 1.05l$. The force field coefficients are $c_5, c_6 = 0.2$, $\tau = 0.1$ and $T = 0.9$. The friction parameters are $c_7, c_8 = 0.3$, $K_d = 0.5$, $K_s = 1$ and $c_9 = 0.6$.

Figure 1 displays two still photographs of a semi-transparent ball rebounding off a folded scarf, where light refraction through the ball is disregarded to clearly show the scarf deformation. The effect of the thickness force field is evident on the scarf, which would otherwise appear flat. Additionally, the force of the ball produces a realistic deformation of the scarf that includes effects from both springs and force fields.

Frames from a falling scarf animation, shown in the top row of Figure 13, demonstrate both self-collision and self-friction, which are challenging to animate realistically. The scarf consists of approximately 15000 triangles, and a 40-second sequence at 25 fps was rendered on a 864MHz Pentium III in approximately 30 minutes. The bottom row of Figure 13 shows images of a scarf being stretched then released to snap back into a stable configuration. Because of friction from the underlying surface and because of the gaps of space within the knitwear that allow for various balance lengths of the yarn loops, the ending stable configuration is not a straight scarf. This animation required about 11000 triangles for the scarf and took about 2 minutes to render a 3-second sequence. These realistic results are achievable by consideration of the aforementioned knitwear properties.

All the presented images utilize a single light source. Multiple light sources can potentially be handled using multi-texturing.

7 Conclusion

In this paper, we presented models for realistic animation and efficient photorealistic rendering of free-form knitwear. These models are designed to incorporate fabric characteristics that are particular to knitwear. For animation modeling, knitwear thickness and gaps within knitwear loops are represented, and their effects are manifested in simulations involving knitwear layering, friction forces and knitwear deformations.

With this animation model for global shape control, efficient rendering is performed by taking advantage of the repetitive structure of yarn. We introduced a reflectance element called the lumislice that represents the reflected radiance of light, as influenced by the yarn microstructure. It provides an effective means for propagating local microstructure features over a knitwear article while allowing for arbitrary global characteristics, such as the stitch pattern and overall knitwear shape. In conjunction with a proposed shadow technique for yarn, the lumislice can provide convincing visual results.

Although all examples in this paper are knitwear, the lumislice can potentially be applied to other fabric objects consisting of repetitive macroscopic structures, such as carpet and coarse weaves. Additional possibilities for future work include development of a smooth transition between levels of detail at closer viewing distances and BRDFs from farther away.

Acknowledgements

We thank David Salesin and Xin Tong for useful discussions, and the anonymous reviewers for their constructive critiques on the lumislice.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A Simple Method for Extracting the Natural Beauty of Hair. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):111–120, July 1992.

[2] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[3] D. E. Breen, D. H. House, and P. H. Getto. A Physically-Based Particle Model of Woven Cloth. *The Visual Computer*, 8(5-6):264–277, June 1992.

[4] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the Drape of Woven Cloth Using Interacting Particles. *Proceedings of SIGGRAPH 94*, pages 365–372, July 1994.

[5] D. E. Breen, D. H. House, and M. J. Wozny. A particle-based model for simulating the draping behavior of woven cloth. *Textile research joural*, 64(11):663–685, November 1994.

[6] S. Chandrasekar. *Radiative Transfer*. Dover Publications, New York, 1960.

[7] M. Courshesnes, P. Volino, and N. M. Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. *Proceedings of SIGGRAPH 95*, pages 137–144, August 1995.

[8] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and Texture of Real-world Surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[9] B. Eberhardt, A. Weber, and W. Straßer. A Fast, Flexible, Particle-System Model for Cloth Draping. *IEEE Computer Graphics & Applications*, 16(5):52–60, September 1996.

[10] D. B. Goldman. Fake Fur Rendering. *Proceedings of SIGGRAPH 97*, pages 127–134, August 1997.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. *Proceedings of SIGGRAPH 96*, pages 43–54, August 1996.

[12] E. Gröller, R. T. Rau, and W. Straßer. Modeling and Visualization of Knitwear. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):302–310, December 1995.

[13] E. Gröller, R. T. Rau, and W. Straßer. Modeling Textiles as Three Dimensional Textures. *Eurographics Rendering Workshop 1996*, pages 205–214, June 1996.

[14] Harmony. *The Harmony Guide to Knitting Techniques*. Collins & Brown, 1992.

[15] D. H. House and D. E. Breen. *Cloth Modeling and Animation*. A K Peters, Natick Massachusetts, 2000.

[16] J. T. Kajiya and T. L. Kay. Rendering Fur with Three Dimensional Textures. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):271–280, July 1989.

[17] J. Lengyel. Real-Time Fur. *Eurographics Rendering Workshop 2000*, pages 243–256, June 2000.

[18] M. Levoy and P. Hanrahan. Light Field Rendering. *Proceedings of SIGGRAPH 96*, pages 31–42, August 1996.

[19] T. Lokovic and E. Veach. Deep Shadow Map. *Proceedings of SIGGRAPH 2000*, pages 385–392, July 2000.

[20] M. Meißner and B. Eberhardt. The Art of Knitted Fabrics, Realistic & Physically Based Modeling Of Knitted Patterns. *Computer Graphics Forum*, 17(3):355–362, 1998.

[21] A. Meyer and F. Neyret. Interactive Volumetric Textures. *Eurographics Rendering Workshop 1998*, pages 157–168, June 1998.

[22] F. Neyret. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55–70, January-March 1998.

[23] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering Antialiased Shadows with Depth Maps. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):283–291, July 1987.

[24] B. Robertson. Building a better mouse. *Computer Graphics World*, 22(12), December 1999.

[25] C. Soler and F. X. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. *Proceedings of SIGGRAPH 98*, pages 321–332, July 1998.

[26] W. Straßer and B. Eberhardt. Representation of Knit Fabrics. *SIGGRAPH Course Notes: Cloth and Clothing in Computer Graphics*, pages F–1–18, 1998.

[27] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[28] N. M. Thalmann, S. Carion, M. Courchesne, P. Volino, and Y. Wu. Virtual Clothes, Hair and Skin for Beautiful Top Models . *Computer Graphics International*, 1996.

[29] P. Volino and N. M. Thalmann. Implementing Fast Cloth Simulation with Collision Response. *Computer Graphics International*, 2000.

[30] L. R. Wanger, J. A. Ferwerda, and D. P. Greenberg. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Computer Graphics and Application*, 12(3):44–58, May 1992.

[31] S. H. Watson, J. R. Arvo, and K. E. Torrance. Predicting Reflectance Functions From Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):255–264, July 1992.

[32] J. Weil. The Synthesis of Cloth Objects. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):49–54, August 1986.

[33] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting Reflectance Functions From Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):255–264, July 1992.

[34] L. Williams. Casting Curved Shadows on Curved Surfaces. *Computer Graphics (Proceedings of SIGGRAPH 78)*, 21(3):270–274, August 1978.

[35] Y. Q. Xu, Y. Chen, S. Lin, H. Zhong, E. Wu, B. Guo, and H. Y. Shum. Photo-Realistic Rendering of Knitwear Using the Lumislice. *Proceedings of SIGGRAPH 2001*, pages 391–398, August 2001.

[36] T. Yasuda, S. Yokoi, and J. Toriwaki. A Shading Model for Cloth Objects. *IEEE Computer Graphics & Applications*, 12(6):15–24, November 1992.

[37] H. Zhong, Y. Xu, B. Guo, and H. Shum. Realistic and Efficient Rendering of Free-Form Knitwear. *Journal of Visualization and Computer Animation, Special Issue on Cloth Simulation*, 12(1), 2001.

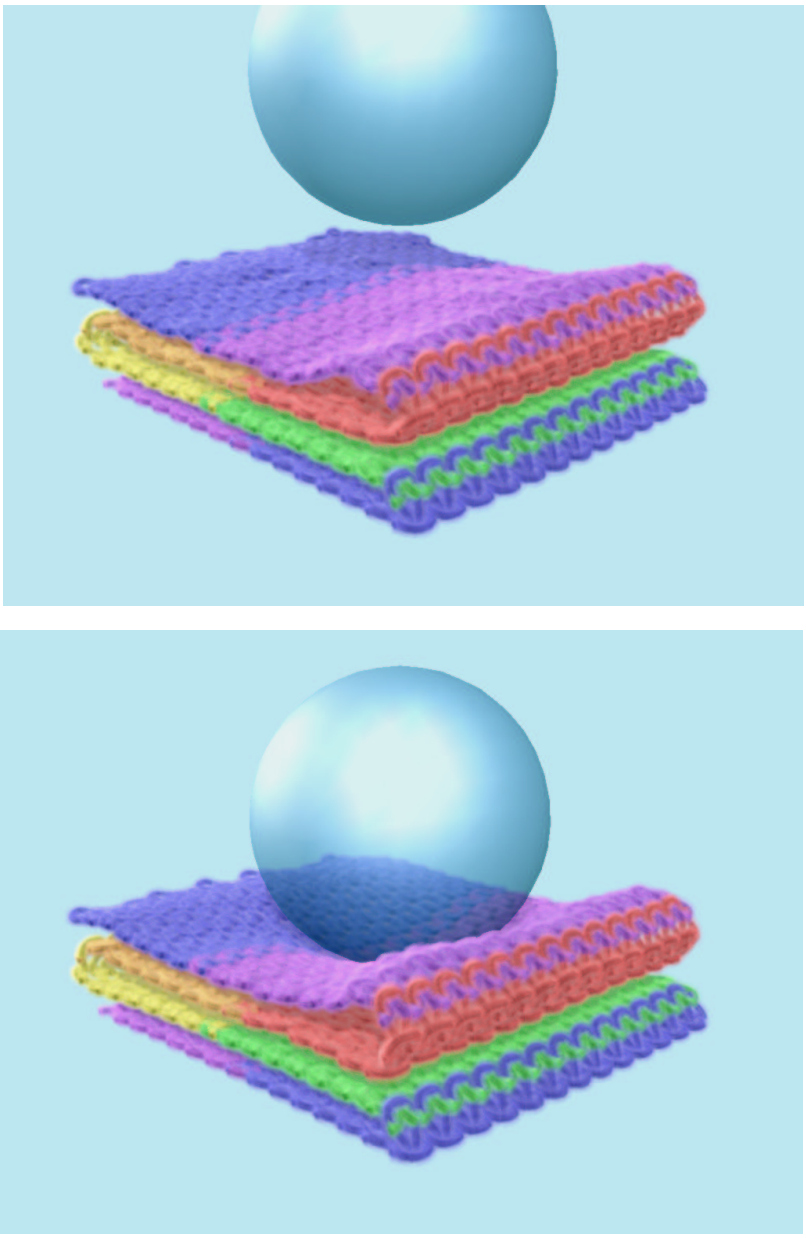

Figure 1: Rendered images of a folded scarf. Unlike animation models for other fabrics, our knitwear method accounts for thickness and its effect on shape. The lower figure exhibits a realistic deformation caused by the weight of a ball.

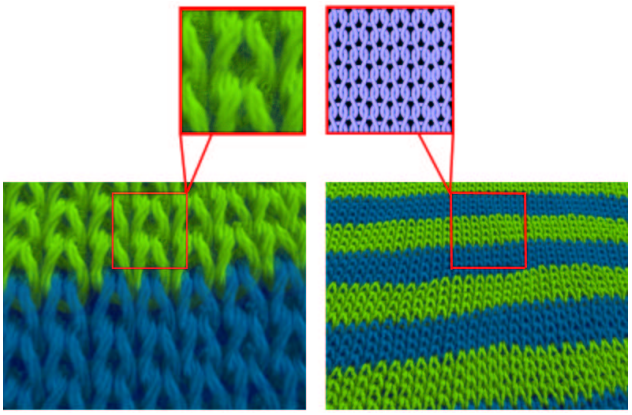

Figure 2: Knitwear synthesized using the lumislice and including soft shadows. Left side: close-up views of knitwear microstructure. Right side: stitch pattern and irregular macroscopic structure.

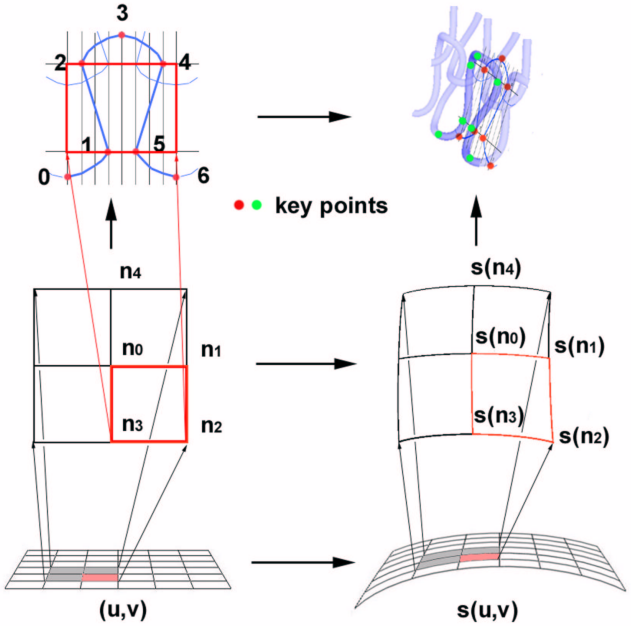

Figure 3: Construction of the knitwear skeleton. A loop in the parameter space is shown as a red rectangle on the left and in 3D on the right. For $0 \leq i \leq 5$, the key point k_i is marked by i on the loop in the top left. In 3D, the key points before offsetting are marked by red dots and are denoted after offsetting by green dots.

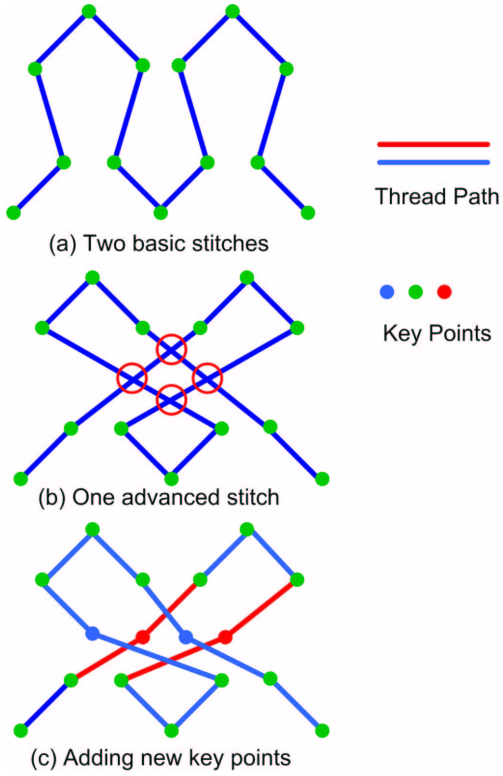

Figure 4: An advanced stitch form by combining two simple stitches.

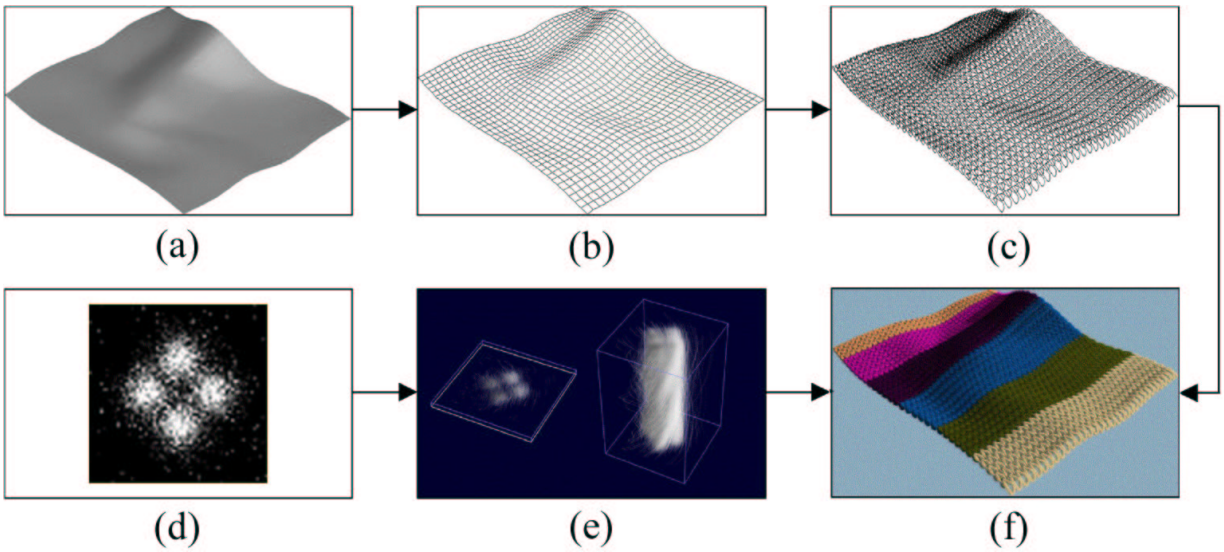

Figure 5: An overview of lumislice-based rendering. Pattern stitching: (a) free-form surface, (b) placement of control points determined by physical animation model, (c) knitwear skeleton with stitches. Lumislice modeling: (d) density distribution of yarn fibers, (e) lumislice computation and winding over a yarn segment. With (c) and (e), a piece of knitwear can be rendered as (f).

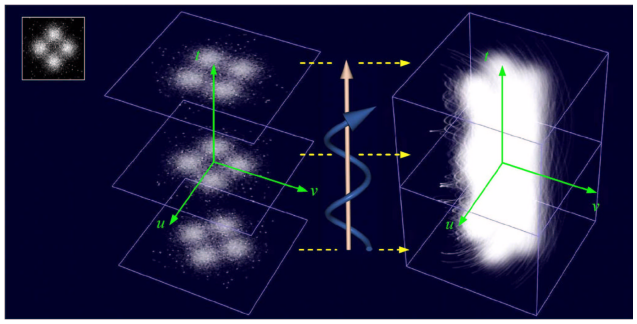

Figure 6: Generation of a volumetric yarn segment. The fiber density distribution in the upper-left corner is rotated along the path of the yarn to yield a yarn segment.

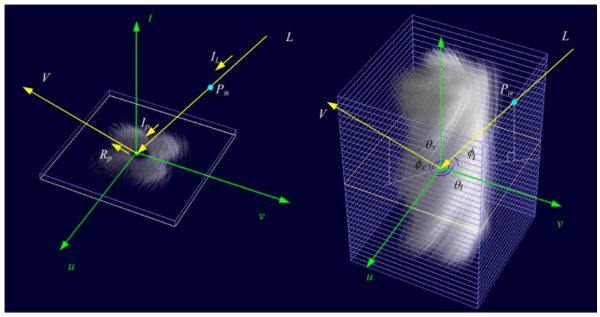

Figure 7: Notation for lumislice calculation. Left: Light of intensity I_L enters the yarn segment at voxel P_{in} and is attenuated to intensity I_p by the yarn upon reaching voxel p; the reflected light intensity upon exiting p towards the viewing direction is R_p. Right: Spherical angles at voxel p of the light direction L and the viewing direction V.

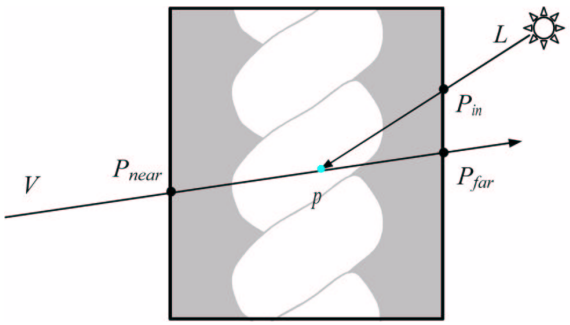

Figure 8: Radiance calculation through a yarn segment. An emission-absorption model is used to compute the contributions to viewed radiance at V from voxels between P_{near} and P_{far}.

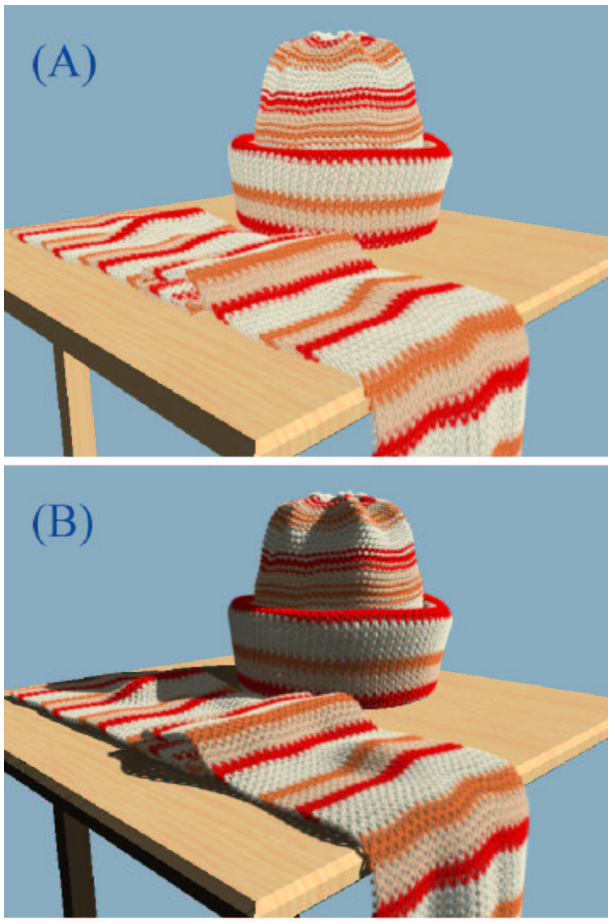

Figure 9: Effects of shadows in rendered knitwear. (A) Synthesized image without shadows. (B) Enhanced realism resulting from our shadow technique.

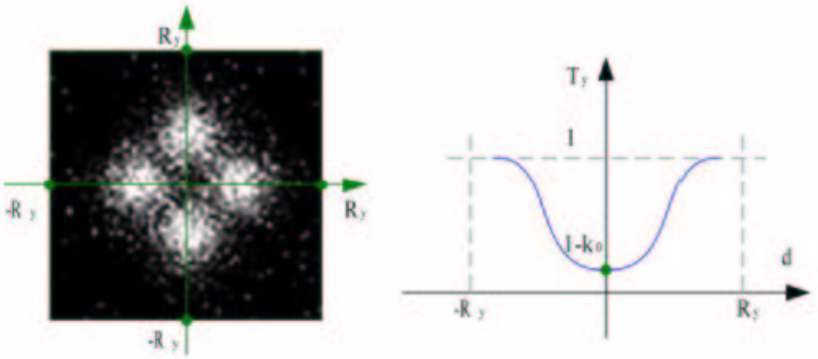

Figure 10: Soft shadows from knitwear. Transmission factor T_y of light passing a distance $d \leq R_y$ from a yarn center.

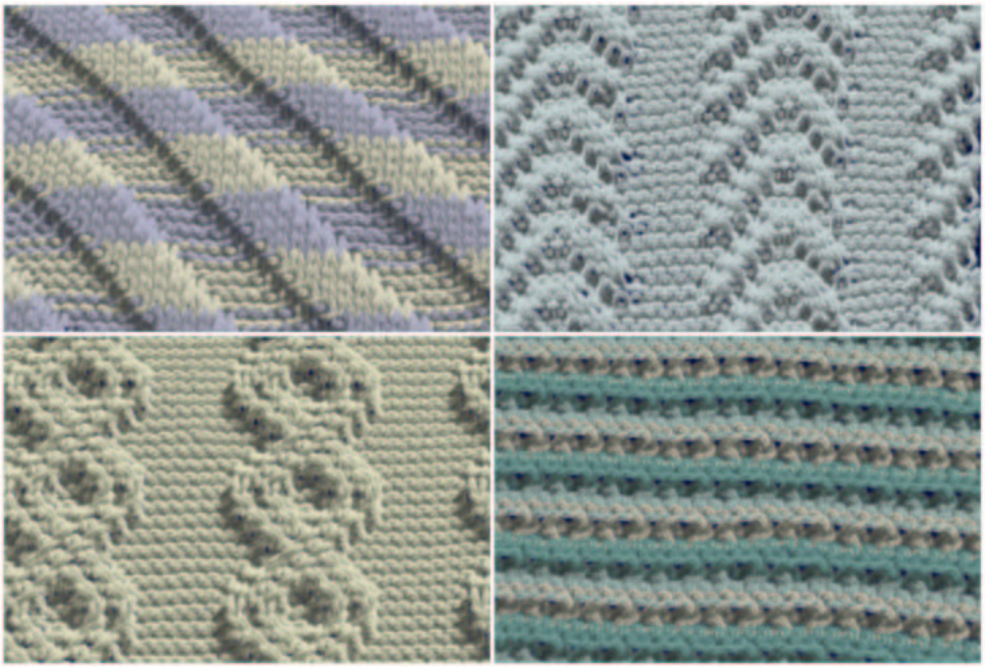

Figure 11: Rendered stitching patterns that depict various knitting styles which can be synthesized by lumislice-based rendering. The desirable effects of soft shadows are evident.

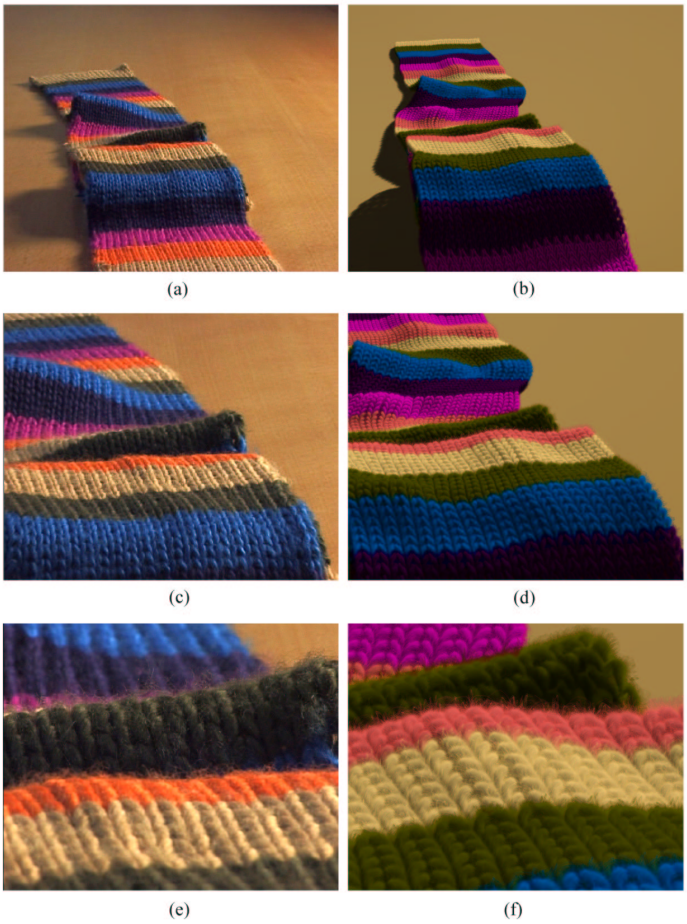

Figure 12: Comparison of lumislice-based rendering to actual images. (a)(c)(e) display real images of a scarf captured from various viewing distances. (b)(d)(f) show corresponding renderings by our method.

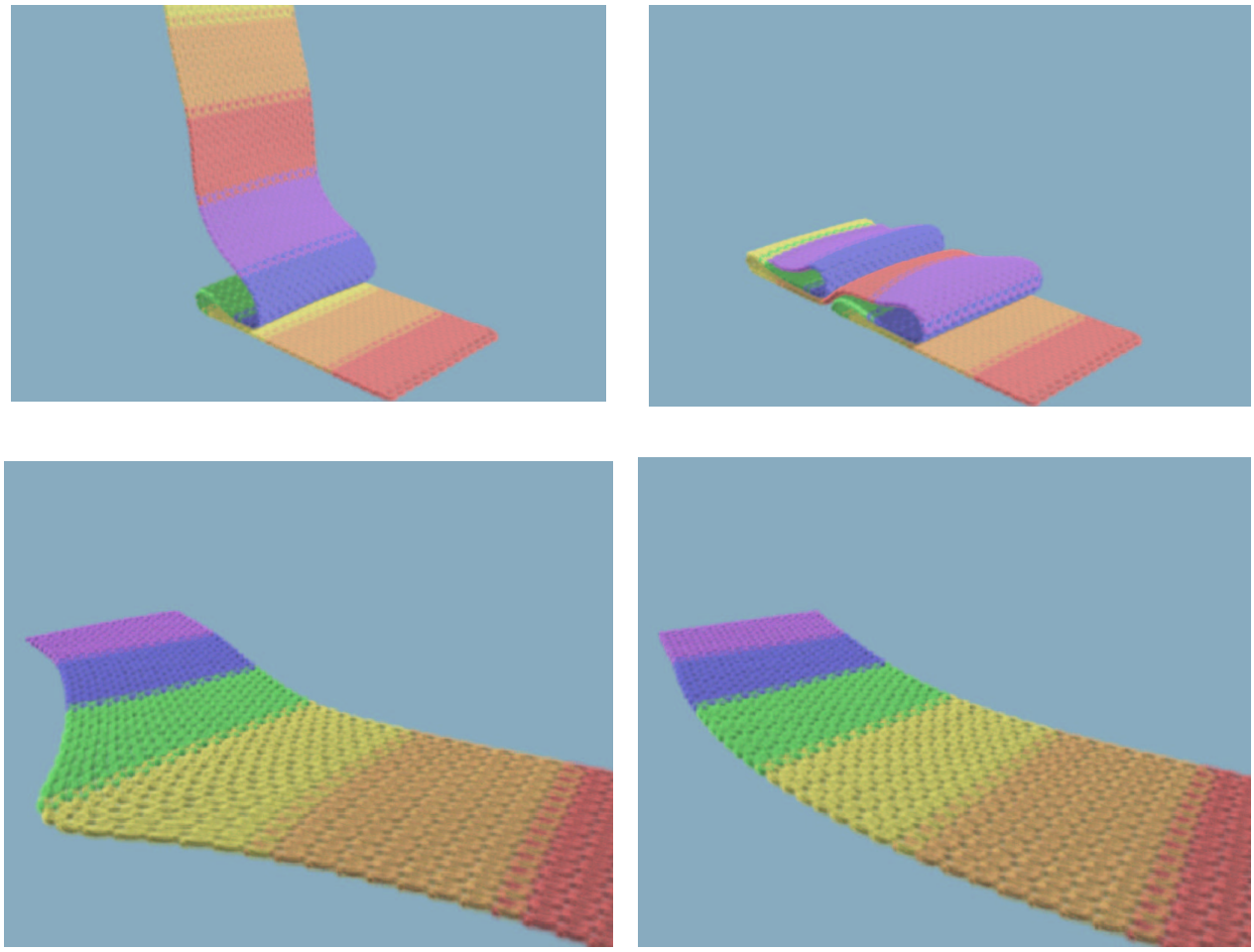

Figure 13: Frames from animation sequences. Top row: Knitwear falling. Bottom row: Knitwear stretching, where the left image exhibits the deformation of stretching, and the right image shows the scarf configuration after release of the stretching force.

Figure 14: Sweater worn on a person.