COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2012; **23**:253–265
Published online 24 May 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1456

SPECIAL ISSUE PAPER

Photorealistic cloth in real-time applications

Young-Min Kang[1]* and Chang-Sik Cho[2]

[1] Tongmyong University, Busan, Korea
[2] ETRI, Daejeon, Korea

ABSTRACT

In this paper, we propose an efficient and stable animation and rendering method for complex mass-spring-based cloth model for interactive applications. Although the mass-spring model can be easily constructed and simulated, it is well known that the model suffers from the instability problem. The method proposed in this paper employs the harmonic oscillation model and analytically integrates the force for better stability and accuracy while keeping the integration scheme still explicit. As the integration scheme is explicit, the method can be easily parallelized, and the performance improvement can be easily obtained by exploiting the parallelism in the graphics processing unit. For the realistic representation of virtual clothes, we also proposed an efficient and effective rendering method for woven surfaces on the basis of alternating anisotropic reflectance and microfacet distribution function rotation. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS
real-time cloth animation; geometrically complex mass-spring; harmonic motion; cloth rendering

***Correspondence**
Young-Min Kang, Tongmyong University, Busan, Korea.
E-mail: ymkang@tu.ac.kr

1. INTRODUCTION

Real-time animation of photorealistic cloth object in interactive application requires two difficult problems to be solved: the stable integration scheme for real-time simulation and photorealisitc rendering of the complex woven surfaces.

For the physics simulation, we employed the commonly used mass-spring model. Although the model is simple and intuitive, the simulation of the model with a large amount of mass-points and springs is not simple at all because of the well-known instability problem. The implicit integration schemes are usually employed to stabilize the system and to make it possible to use large time steps. However, the implicit method is, in essence, a linear system solving and cannot be easily parallelized. As the performances of graphics hardwares are rapidly being improved, the importance of the parallelism in simulation and rendering is also increasing. In this paper, we propose more stable and more accurate explicit integration method on the basis of the harmonic oscillation model. The method produces realistic motion of the geometrically complex mass-spring model by taking advantage of the parallel processing ability of current graphics hardwares.

In this paper, we also propose a rendering method that expresses the light scattering on the woven surface. The proposed method procedurally genererates the bumpy illusion on the woven surface and effectively renders the light reflection of the woven structure. Figure 1 shows the snapshot of a real-time animation of realistic cloth in an interactive application implemented with our method.

2. RELATED WORK

Since Terzopoulos *et al.* characterized soft object animation as a deformable surface problem [1], various physically based approaches have been proposed for soft object animation.

Pentland and Williams employed modal analysis to efficiently animate non-rigid models [2]. However, the method was not devised for accurate animation of a deformable object with complex geometry.

Thalmann's group has proposed many techniques for dressed virtual characters [3]. However, their major interest was the realism of the virtual cloth instead of computational efficiency.

Various techniques have been proposed for efficient animation of soft objects, and one of the most important advances is the use of implicit integration for stable animation [4]. However, the implicit integration involves a large linear system. Therefore, some efficient approximate approaches have been proposed [5,6].

A method to generate seams and wrinkles for virtual cloth was also proposed [7]. However, such methods can

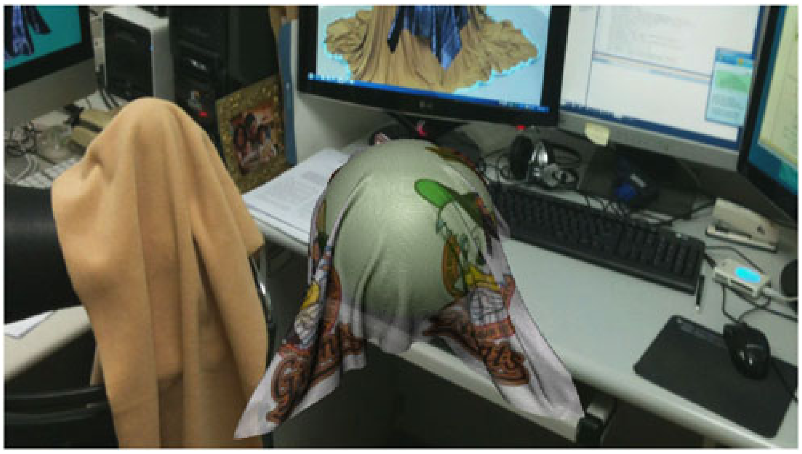


Figure 1. Photorealistic cloth animation result.

be regarded approximate methods because the geometric details were not generated on the basis of physics.

In this paper, we employed extremely complex geometric structure and animated the model without any geometric approximation.

For realistic rendering, we employed microfacet-based model. The microfacet-based rendering was introduced by Torrance and Sparrow [8]. The reflectance property of this surface is determined by microfacet distribution function (MDF). The microfacet-based rendering model has been continuously improved to represent various materials [9–11].

There have been also various techniques based on microfacet model for representing the anisotropic reflectance surfaces [12–17]. In our observation, however, the normal perturbation on anisotropic reflectance surface usually introduces unrealistic reflection artifacts. In order to avoid such artifacts and improve the rendering quality of normal mapped anisotropic surface, we propose a method that rotates the MDF of the surface in accordance with the perturbation of the normal vectors.

There have been various research efforts to model and render the knitted fabric [18–23]. However, those methods cannot be directly employed for rendering woven fabrics where the yarns are far thinner than those of the knitted fabrics.

Yasuda *et al*. proposed a shading model for woven fabric by applying anisotropic reflectance according to the yarn direction [24]. However, the method is not capable of rendering the close-up scene where weave patterns are visible.

Adabala *et al*. proposed a woven fabric rendering method that can be applied to both distant and close-up observations of woven surface [25,26]. In this method, however, realistic light reflection was not the major concern.

Some researchers tried to capture the spatially varying bidirectional reflectance distribution function (SVBRDF) for realistic representation of the fabric material [27–29]. However, capturing the SVBRDF requires expensive devices and huge amount of storage. Wang *et al*. proposed a SVBRDF measuring technique using data captured from a single view [30,31]. However, the method still requires huge amount of storage for SVBRDF. Moreover, one needs to measure all kinds of fabric which will be possibly used in rendering.

Sattler *et al*. employed bidirectional texture function proposed by Dana *et al*. [27] to render photorealistic woven fabric [32]. However, this method also suffers from the common disadvantages of example-based approaches.

Zinke and Weber proposed bidirectional fiber scattering distribution function (BFSDF) for physically plausible rendering of dielectric filaments [33]. However, the BFSDF is an 8D function which cannot be easily dealt.

3. ANIMATION

In this section, described are the animation techniques that can be easily parallelized in order to exploit the current graphics hardwares.

3.1. Problem Formulation

The simplest numerical integration of force can be described in explicit Euler scheme as follows:

$$\frac{\mathbf{v}(t+h)-\mathbf{v}(t)}{h} \simeq \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}(t) = \frac{\mathbf{f}(t)}{m} \tag{1}$$

where h denotes the integration time step.

It is well known that Equation (1) has the serious problem of instability when it is applied to stiff equations. The instability arises as the $\mathbf{f}(t)h$ is not the accurate integration during the time interval. The implicit integration schemes use $\mathbf{f}(t+h)$ instead of $\mathbf{f}(t)h$, and the result simulation is unconditionally stable. However, the force at the future cannot be computed with the current situation. In order to approximate or predict the future force, the implicit integration exploits the Taylor series as follows:

$$\mathbf{f}(t+h) \simeq \mathbf{f}(t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x} \tag{2}$$

The approximation shown in Equation (2) involves a matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x}$ with $n \times n$ elements, where n is the number of mass-points.

We propose a new animation method that can be easily parallelized because of its explicitness and shows an improved stability. The basic idea of our animation method is to improve the inaccurate integration approximated as $\mathbf{f}(t)h$. An accurate integration should be computed as follows:

$$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + \frac{1}{m} \int_{t_0}^{t_0+h} \mathbf{f}(t) \mathrm{d}t \tag{3}$$

However, it is common to use Hooke's law when a mass-spring model is used. The problem of Hooke's law is that the force is not a function of time. Therefore, the changing force during the integration scheme cannot be taken into account. In our model, the force is modeled on the basis of harmonic oscillation. The harmonic oscillation model can be analytically integrated, and the change of the force during the time interval can be easily predicted.

Suppose a particle i and another particle j are linked with a spring (i, j). The locations of the particles are denoted as $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. The velocities are also denoted similarly as $\mathbf{v}_i$ and $\mathbf{v}_j$. The masses of the particles are m_i and m_j, respectively. Let us denote $\mathbf{x}_j - \mathbf{x}_i$ to be $\mathbf{x}_{ji}$ and similarly $\mathbf{v}_j - \mathbf{v}_i$ to be $\mathbf{v}_{ji}$. When $\mathbf{z}$ is a vector, its normalized vector is denoted as $\hat{\mathbf{z}}$. Because the location and velocity are functions of time, the location and velocity of a particle i at time t are denoted as $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t)$, respectively. The rest length of a spring is l_0, and the length of the spring at time t is $l(t)$ and is equivalent with $|\mathbf{x}_{ji}(t)|$. The stiffness of the spring is denoted to be κ.

3.2. Harmonic Oscillation Model

A simple example of harmonic oscillation is the motion of a mass-point, and a static object is linked with a spring. In this 1D case, the location of the mass-point can be represented as a scalar value y, and the oscillation can be expressed as $y = A \sin \omega t$, where A is the amplitude of the oscillation, m is the mass, and ω is $\sqrt{\kappa/m}$. When two particles are linked with each other, the deformation of the spring can be expressed as follows:

$$\delta(t) = A \sin \omega t = A \sin \left(t \sqrt{\frac{\kappa}{m_i} + \frac{\kappa}{m_j}} \right) \tag{4}$$

As the amplitude A is not known, we must compute the amplitude on the basis of the energy conservation law. The total energy of the system is $\frac{1}{2}\kappa A^2$, and it should be equal to the sum of the kinetic energy and the potential energy. The potential energy can be computed with the deformation as $\frac{1}{2}\kappa\delta^2$. The kinetic energy can be computed as follows:

$$\begin{aligned} &\frac{1}{2} m_i \left(\frac{m_j}{m_i + m_j} \right) \dot{\delta}^2 + \frac{1}{2} m_j \left(\frac{m_i}{m_i + m_j} \right) \dot{\delta}^2 \\ &= \left(\frac{m_i m_j}{m_i + m_j} \right) \dot{\delta}^2 \end{aligned} \tag{5}$$

Therefore, we can relate the total energy and the sum of the kinetic and potential energies as follows:

$$\frac{1}{2}\kappa A^2 = \frac{1}{2}\kappa\delta^2 + \left(\frac{m_i m_j}{m_i + m_j} \right) \dot{\delta}^2 \tag{6}$$

The amplitude of the oscillation can then be computed with the current state variables as follows:

$$A = \sqrt{\delta^2 + 2\frac{m_i m_j}{\kappa(m_i + m_j)} \dot{\delta}^2} \tag{7}$$

In order to compute the amplitude with Equation (7), we must know the time derivative of the deformation $\dot{\delta}$, and this can be computed as follows:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\delta(t) &= \frac{\mathrm{d}}{\mathrm{d}t}(l(t) - l_0) = \frac{\mathrm{d}}{\mathrm{d}t} l(t) \\ &= \frac{\mathrm{d}}{\mathrm{d}t} \sqrt{\mathbf{x}(t)^T \mathbf{x}(t)} \\ &= \frac{1}{2} \left(\mathbf{x}(t)^T \mathbf{x}(t) \right)^{-\frac{1}{2}} \left(2\mathbf{x}(t)^T \mathbf{v}(t) \right) \\ &= \frac{\mathbf{x}(t)^T \mathbf{v}(t)}{\sqrt{\mathbf{x}(t)^T \mathbf{x}(t)}} \\ &= \hat{\mathbf{x}(t)}^T \mathbf{v}(t) \end{aligned} \tag{8}$$

The harmonic oscillation is periodic and can be expressed as $\delta(t) = A \sin(\omega T)$, where t is the time in simulation world, whereas T is the time defined within the oscillation period ranging from 0 to $2\pi/\omega$. In order to accurately integrate the force, we must convert t into T. Let us denote T corresponding to the current time t to be T_0. Then it can be computed as follows:

$$T_0 = \frac{1}{\omega} \sin^{-1} (\delta(t)/A) \tag{9}$$

Now we can describe the spring force $\kappa\delta(t)$ as $\kappa A \sin(\omega T)$ on the basis of the harmonic oscillation model and analytically integrate for more accurate simulation of springs as follows:

$$\begin{aligned} \int_{t_0}^{t_0+h} \mathbf{f}(t) \mathrm{d}t &= \int_{t_0}^{t_0+h} \kappa\delta(t) \mathrm{d}t \\ &= \int_{T_0}^{T_0+h} \kappa A \sin(\omega T) \mathrm{d}T \\ &= -\frac{\kappa A}{\omega} \cos \omega T |_{T_0}^{T_0+h} \end{aligned} \tag{10}$$

The integration of the force shown in Equation (10) makes it possible for the system to predict the velocity and the location of a particle more accurately. Because the period of the cosine function in Equation (10) is $2\pi/\omega$, the time interval h should be also adjusted to range from 0 to $2\pi/\omega$. Let us denote the adjusted interval to be H, and it can be easily computed as follows:

$$H = h - \left\lfloor \frac{h\omega}{2\pi} \right\rfloor \cdot \frac{2\pi}{\omega} \tag{11}$$

Now we can compute the velocity change on the basis of the integration of the force. Let us denote the magnitude of the velocity change for each particle to be ϕ. The amount can then be computed as follows:

$$\begin{aligned}\phi &= \frac{1}{2}\frac{1}{h}\int_{t_0}^{t_0+h} \mathbf{f}(t)\mathrm{d}t \qquad (12)\\ &= \frac{\kappa A}{2\omega H}(\cos\omega(T_0 + H) - \cos\omega T_0)\end{aligned}$$

Let us denote the specific ϕ value for the spring that links particle i and j to be ϕ_{ij}. Then the simulation is reduced to the computation of ϕ_{ij} for every spring (i, j), and the velocity change of each particle i is updated as $\Delta\mathbf{v}_i = \sum_{(i,j)\in E} \phi\hat{\mathbf{x}_{ji}}$. The rest of the simulation is exactly the same as the usual explicit Euler method.

4. RENDERING

In this chapter, we propose a rendering method that manipulates the distribution function of the microfacets in order to express the peculiar reflectance of woven fabric. The rendering method is based on our previous alternating anisotropy approaches [34,35], and the accuracy and efficiency were significantly improved.

4.1. Alternating Anisotropy

Woven fabric has weft and warp yarns. Because the yarns are oriented in different directions, the reflectance anisotropy is alternating according to the yarn direction. Therefore, we employed alternating anisotropy for woven fabric, and the anisotropy is determined by the underlying weave patterns.

The reflectance property of microfacet-based surface model is determined by the MDF $D(\mathbf{h})$, which gives the probability that a microfacet is oriented to the direction $\mathbf{h}$.

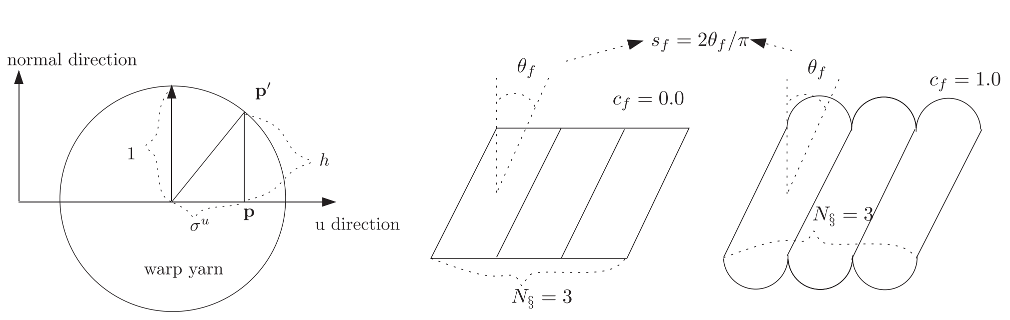
4.2. Procedural Normal Perturbation

Although the alternating anisotropy can represent the yarn directions, it cannot represent the bumpy surface of woven fabric.

In order to produce realistic bumpy surface caused by a woven structure, we have to perturb the normal vectors on the fabric according to the weave patterns as proposed in [34]. We first define *weave elements* which compose the woven fabric. There are two types of weave elements, weft element and warp element. Each weave element is a building block of woven fabric.

The weave element type of the sampled point can be determined only when the weave pattern is defined. In our method, the weave pattern is determined by three parameters, n_w, n_π, and n_σ. The parameter n_w is the number of continuous weft elements in a single weave row, and n_π is similarly the number of continuous warp elements after the continuous weft elements. n_σ is the number of element shifts in the next weave row. Therefore, the normal weave, 1/2 twill weave, and satin weave can be described with parameter sets such as (1,1,1), (1,2,1), and (5,1,2) for n_w, n_π, and n_σ. Once the weave pattern and the yarn thickness are given, the weave element at the sampled point (u, v) can be easily determined.

Figure 2(a) shows the cross section of a perfectly circular warp yarn. As shown in the figure, the sampled point $\mathbf{p}$ should be rendered as if it is located at the displaced point $\mathbf{p}'$. We can easily notice that the x component of the perturbed normal is proportional to the offset ratio σ^u, which ranges from -1 to 1. The original normal vector in tangent space is always $(0, 0, 0)^T$. Therefore, the perturbed normal at the sampled point on a warp yarn can then be expressed as $\left(c_y\sigma^u, 0, \sqrt{1-(c_y\sigma^u)^2}\right)^T$, where c_y denotes the yarn curvature control parameter ranging



(a) Circular yarn cross section (b) Twisted fiber parameters

Figure 2. Bumpy cloth model and reflectance deformation.

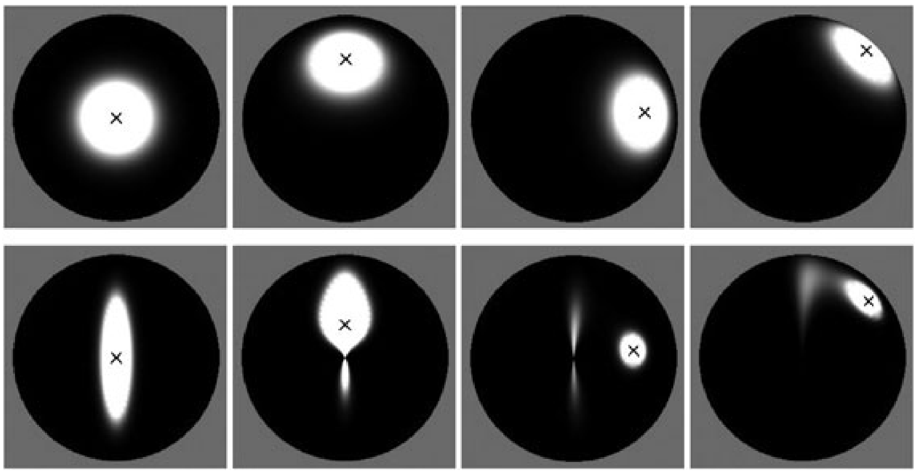


Figure 3. Ashikhmin MDF with perturbed normal vectors: isotropic (top row) and anisotropic MDF (bottom row).

from 0 to 1. We can similarly perturb the normal at the sampled point on a weft yarn as $\left(0, c_y\sigma^v, \sqrt{1-(c_y\sigma^v)^2}\right)^T$. As the z component of the normal vector is dependent on x and y components, the perturbed normal can be expressed only with x and y components.

A single yarn is usually a collection of twisted fibers. Figure 2(b) shows the parameters for twisted yarn model. θ_f is the slope angle of the twist, and c_f is the curvature control parameter for twisted fiber. For simplicity, we denoted $2\theta_f/\pi$ as s_f to control the slope, and then s_f ranges from 0 to 1. As shown in the figure, the twisted fiber is flat when c_f is zero, and curvature increases as c_f increases. The parameter $N_\S$ denotes the number of twists within a weave element space. The perturbed normal for a sampled point on the warp yarn can be expressed as $(0, c_f(2\mathbf{fract}(N_\S(v_w - s_f\sigma^v)) - 1)^T$, where the function $\mathbf{fract}(x)$ returns $x - \lfloor x \rfloor$. The perturbed normal for a weft yarn can also be similarly computed. Thus, the perturbed normals $\tilde{\mathbf{n}} = (\tilde{\mathbf{n}}_x, \tilde{\mathbf{n}}_y)^T$ can be described as follows:

$$\begin{aligned} \text{weft}: &\ \left(c_f\left(2\mathbf{fract}\left(N_\S\left(u_w - s_f\sigma^u\right)\right) - 1, c_y\sigma^v\right)^T \\ \text{warp}: &\ \left(c_y\sigma^u, c_f\left(2\mathbf{fract}\left(N_\S\left(v_w - s_f\sigma^v\right)\right) - 1\right)^T \end{aligned} \tag{13}$$

The normal perturbation based on Equation (13) procedurally produce bumpy woven illusion.

4.3. Microfacet Distribution Function Rotation for Perturbed Normal

There have been continuous efforts to represent higher geometric complexity with simple mesh by perturbing the normal vectors [36–38]. However, the normal mapping on anisotropic reflection surface, unfortunately, cannot reproduce the original anisotropic reflectance on the distorted surface because the normal mapping or other normal vector perturbation methods only change the normal vector **n**.

Figure 3 shows the MDF computed with Ashikhmin–Shirley model and perturbed normal vectors. The cross mark in the figure indicates the perturbed normal. The top row of Figure 3 shows isotropic MDF when the normal vector is perturbed. As shown in the figure, the simple perturbation of the normal produces reasonable deformed MDF for the isotropic MDF. However, the simple perturbation is not successful with anisotropic MDFs. The bottom row of Figure 3 shows the results when we employed an anisotropic MDF. The results show that simple normal perturbation approach is hopelessly unsuccessful to preserve the original reflection property.

In order to overcome the limitation of the simple normal mapping on anisotropic reflection surface, the MDF should be properly deformed with the original anisotropic property maintained. Figure 4 shows the MDF rotation concept. The original MDF is defined in tangent space with three axes, **u**, **v**, and **w**. In this original frame, **u** is $(1,0,0)^T$, **v** is $(0,1,0)^T$, and **w** is $(0,0,1)^T$. The normal vector is coincident with **w**. When we apply normal mapping, the normal should be perturbed and will not be

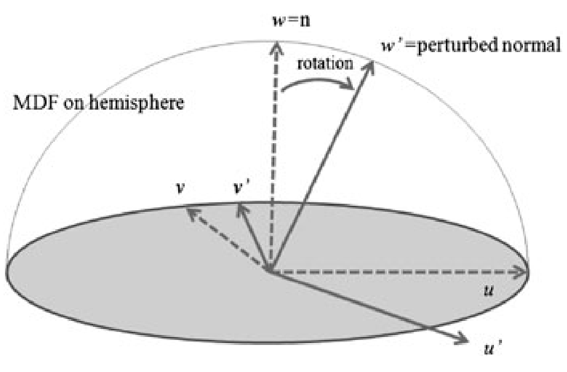


Figure 4. MDF rotation concept.

$(0, 0, 1)^T$ any more. Suppose that the perturbed normal $\tilde{\mathbf{n}}$ is $\left(\tilde{\mathbf{n}}_x, \tilde{\mathbf{n}}_y, \sqrt{1-\tilde{\mathbf{n}}_x^2-\tilde{\mathbf{n}}_y^2}\right)^T$. Let us denote the deformed MDF as $D'(\mathbf{h})$. We can easily derive $D'(\mathbf{h})$ with the MDF rotation concept shown in Figure 4. We simply rotate the original MDF defined on the hemisphere to the new coordinate system with axes $\mathbf{u}'$, $\mathbf{v}'$, and $\mathbf{w}'$.

As it is obvious that $\mathbf{w}'$ is the perturbed normal, we can easily determine the $\mathbf{w}'$, obtain orthogonal axes $\mathbf{u}'$ and $\mathbf{v}'$, and the rotation matrix as follows:

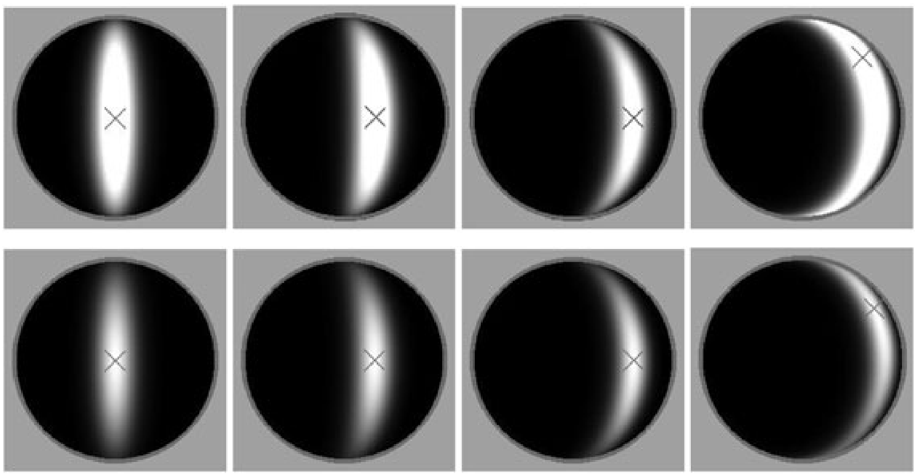


Figure 5. Rotation of Ashikhmin (top row) and Ward BRDF (bottom row).

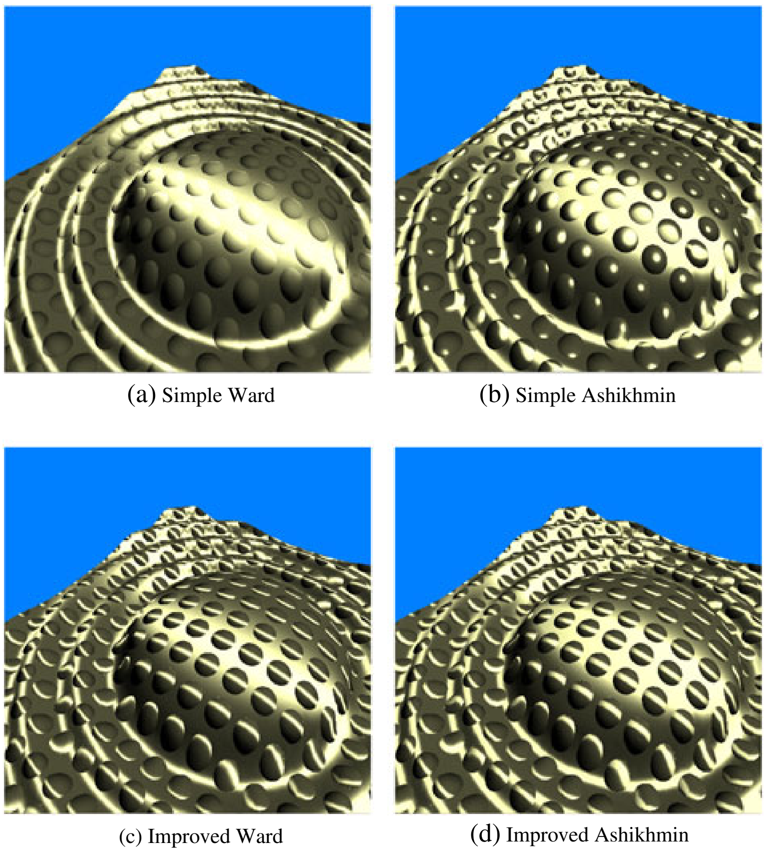


Figure 6. Effect of MDF deformation: simple normal mapping on (a) Ward and (b) Ashikhmin; MDF rotation on (c) Ward and (d) Ashihkmin.

$$\mathbf{R} = [\mathbf{u}', \mathbf{v}', \mathbf{w}'] \tag{14}$$

The rotation matrix **R** transforms the original coordinate frame in accordance with the perturbed normal as shown in Figure 4. Let us denote the transformation of a vector **p** according to the perturbed normal as follows:

$$\mathcal{T}(\mathbf{p}, \tilde{\mathbf{n}}) = \mathbf{R}\mathbf{p}. \tag{15}$$

Figure 5 shows the result when the original MDF is rotated with the transformation **R**, and we denote this MDF as $D'(\mathbf{h})$. However, it is obvious that computing the rotated MDF at each sampling point on the surface is extremely inefficient. Explicit deformation of the MDF is only a conceptual process. In the actual rendering process, we never compute $D'(\mathbf{h})$. Only the original MDF $D(\mathbf{h})$ is used with the inverse transformation $\mathcal{T}^{-1}(\mathbf{p}', \tilde{\mathbf{n}})$. In other words, we conceptually employ $D'(\mathbf{h})$ for the normal mapped surface but actually use $D(\mathcal{T}^{-1}(\mathbf{h}, \tilde{\mathbf{n}}))$, which has the equivalent value.

The inverse transformation of Equation (15) can be easily obtained as follows:

$$\mathcal{T}^{-1}(\mathbf{p}', \tilde{\mathbf{n}}) = \mathbf{R}^T \mathbf{p} \tag{16}$$

Now we can simply calculate $D(\mathcal{T}^{-1}(\mathbf{h}, \tilde{\mathbf{n}}))$ to compute the MDF at the point where the normal vector is perturbed as $\tilde{\mathbf{n}}$.

Figure 6 shows the effect of the MDF rotation by comparing the specular reflections on the illusory bumps. The bumpy illusion on the surface shown in Figure 6(a and b) are generated only with normal mapping method on Ward and Ashikhmin surfaces, respectively. Figure 6(c and d) are generated with MDF rotation techniques on the same surfaces.

As shown in the figure, simple normal mapping with the original MDF cannot produce anisotropic reflectance on the bumpy illusion. Even worse, the shapes of the specular reflection areas are weirdly distorted on some bumps. The deformed MDF removes such disadvantages; the anisotropic reflectance is well preserved on each illusory bump, and no weird shapes are found.

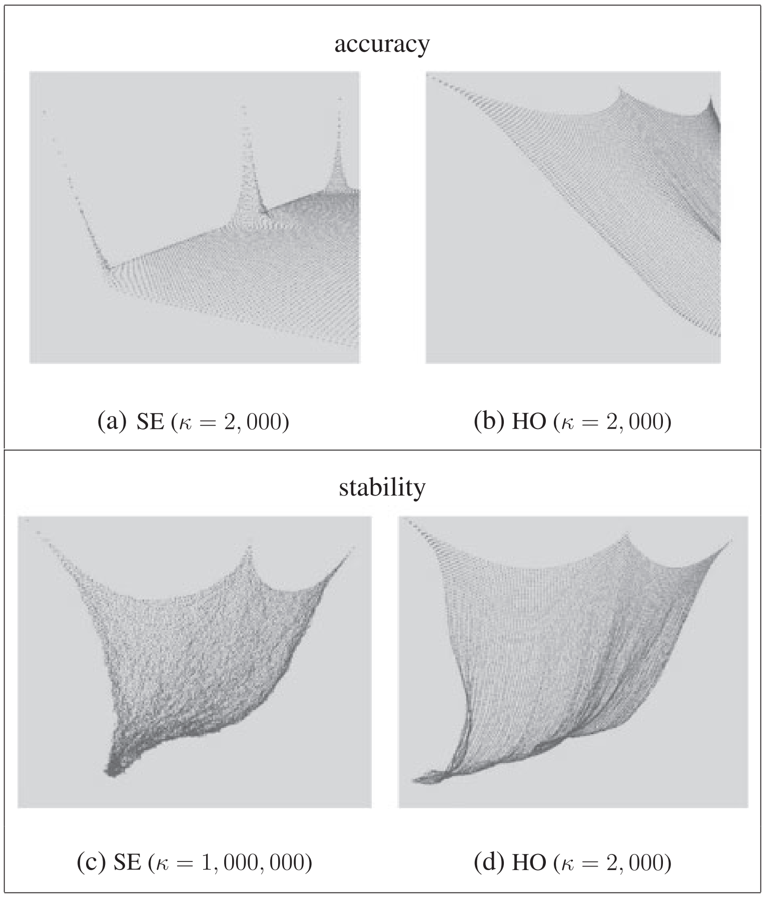
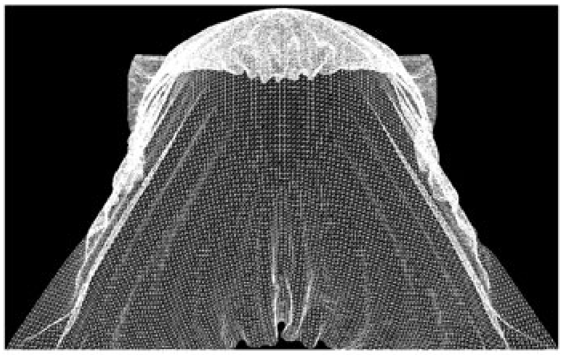
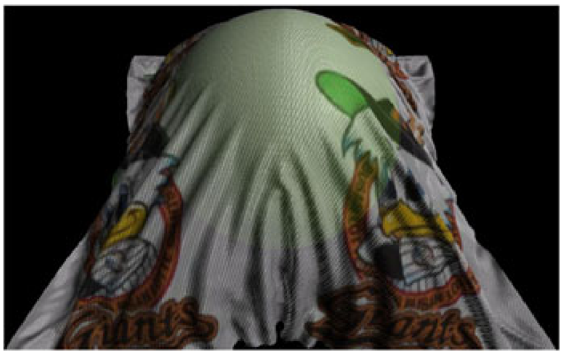

accuracy
(a) SE ($\kappa = 2,000$)
(b) HO ($\kappa = 2,000$)
stability
(c) SE ($\kappa = 1,000,000$)
(d) HO ($\kappa = 2,000$)

Figure 7. Comparison between the simple Euler (SE) method and the proposed harmonic oscillation (HO)-based animation method: (a) and (b) compare the accuracy with $\kappa = 2000$, and (c) and (d) compare the stability with $\kappa = 1,000,000$ for the simple method and $\kappa = 2000$ for the proposed method.

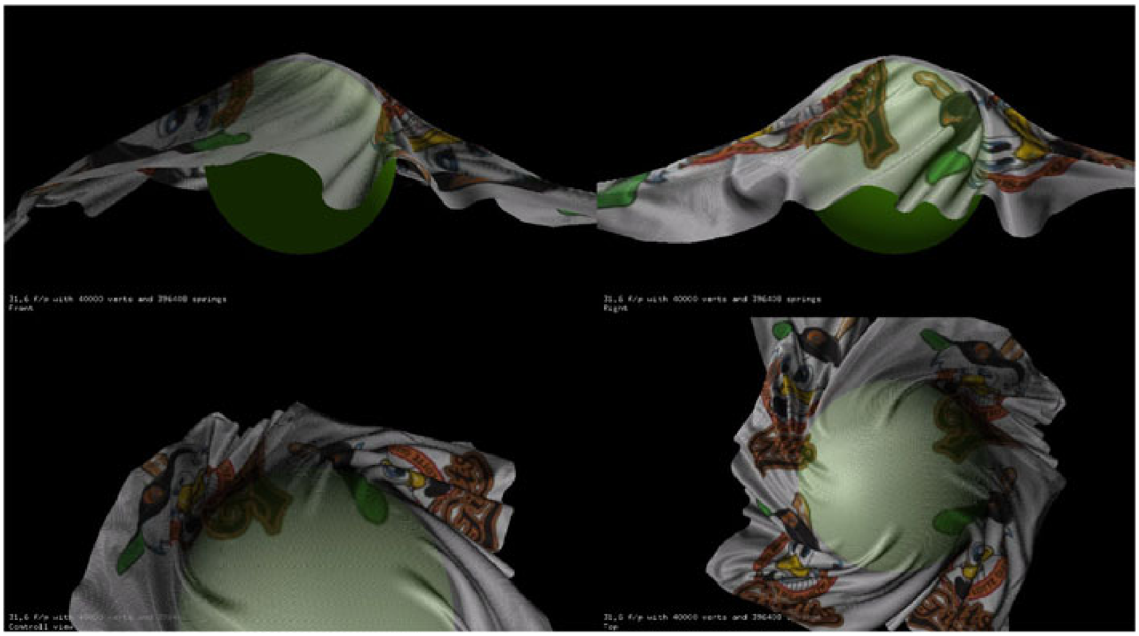


(a) wire frame image



(b) rendered image

Figure 8. Rendering result produced by the proposed alternating anisotropy with rotated MDF.



```
31,6 f/s with 40000 verts and 396408 springs
Front
```

```
31,6 f/s with 40000 verts and 396408 springs
Right
```

```
31,6 f/s with [illegible]
Controll view
```

```
31,6 f/s with 40000 verts and 396408 [illegible]
Top
```

Figure 9. Draped cloth on a twisting ball.

In order to alleviate the computational burden, we used an approximate rotation matrix $\tilde{\mathbf{R}}^T$ as follows:

$$\begin{aligned}
\mathbf{w}' &= \tilde{\mathbf{n}} = (\tilde{\mathbf{n}}_x, \tilde{\mathbf{n}}_y, \tilde{\mathbf{n}}_z)^T \qquad (17)\\
\mathbf{u}' &= (0,1,0)^T \times \mathbf{w}' = (\tilde{\mathbf{n}}_z, 0, -\tilde{\mathbf{n}}_x)^T\\
\mathbf{v}' &= \mathbf{w}' \times \mathbf{u}' = (-\tilde{\mathbf{n}}_x\tilde{\mathbf{n}}_y, 1-\tilde{\mathbf{n}}_y^2, -\tilde{\mathbf{n}}_y\tilde{\mathbf{n}}_z)^T\\
\tilde{\mathbf{R}}^T &= \begin{bmatrix} \tilde{\mathbf{n}}_z & 0 & -\tilde{\mathbf{n}}_x \\ -\tilde{\mathbf{n}}_x\tilde{\mathbf{n}}_y & 1-\tilde{\mathbf{n}}_y^2 & -\tilde{\mathbf{n}}_y\tilde{\mathbf{n}}_z \\ \tilde{\mathbf{n}}_x & \tilde{\mathbf{n}}_y & \tilde{\mathbf{n}}_z \end{bmatrix}
\end{aligned}$$

Let us denote $\tilde{\mathbf{h}}$ as $\mathcal{T}^{-1}(\mathbf{h}, \tilde{\mathbf{n}})$. The rotated MDF $D'_W(\mathbf{h}, \tilde{\mathbf{n}})$ is now simply computed with the original MDF as follows:

$$D'(\mathbf{h}, \tilde{\mathbf{n}}) = D(\tilde{\mathbf{h}}, \mathbf{n}) = D(\tilde{\mathbf{R}}^T\mathbf{h}, \mathbf{n}) \qquad (18)$$

We can easily compute $\tilde{\mathbf{h}}$ without actual matrix operations as follows:

$$\tilde{\mathbf{h}} = \begin{bmatrix} \tilde{\mathbf{n}}_z\mathbf{h}_x - \tilde{\mathbf{n}}_x\mathbf{h}_z \\ -\tilde{\mathbf{n}}_x\tilde{\mathbf{n}}_y\mathbf{h}_x + (1-\tilde{\mathbf{n}}_y^2)\mathbf{h}_y - \tilde{\mathbf{n}}_y\tilde{\mathbf{n}}_z\mathbf{h}_z \\ \tilde{\mathbf{n}}_x\mathbf{h}_x + \tilde{\mathbf{n}}_y\mathbf{h}_y + \tilde{\mathbf{n}}_z\mathbf{h}_z \end{bmatrix} \qquad (19)$$

Now we can obtain normal mapping or normal perturbation effect by simply applying the perturbed half vector shown in Equation (19), and the computational cost can be significantly reduced.

5. EXPERIMENTS

We implemented the techniques on a system with Intel 3.47 GHz i7 CPU, 24 GB RAM, and GTX 590 running on Windows 7 OS. Although the system has 12 CPU cores, we did not exploit CPU parallelism. The implemented software utilized the graphics processing unit (GPU) parallelism instead.

Figure 7 compares our method with the Euler method in the aspect of accuracy and stability. The result shown in Figure 7(a) is implemented with the Euler method, and Figure 7(b) is the result produced with our method. For both simulations, the stiffness κ is set to be 2000. As shown in the figure, the shape restoration property of Euler method is not satisfactory, whereas our method shows more accurate restoration behaviors. Figure 7(c) shows a snapshot of the simple Euler method when the stiffness is high enough for reasonable restoration behavior. As shown in the figure, the simple Euler method tends to be unstable

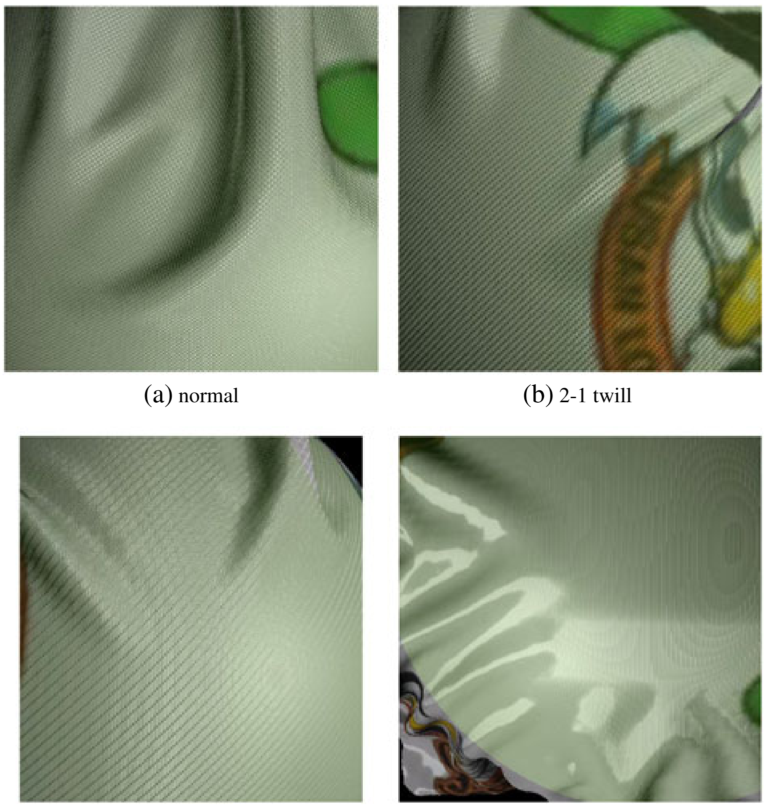

(a) normal (b) 2-1 twill (c) 4-1 twill (d) silk

Figure 10. Different look in accordance with the weave patterns.

with the strong springs. However, our method produces stable and reasonable behavior with far lower stiffness as shown in Figure 7(d).

The proposed method can be easily parallelized, and GPU parallelism can be successfully utilized. We implemented our method in Compute Unified Device Architecture (CUDA) environments, and the model with 16,384 particles and 161,544 springs shown in Figure 7 were simulated with the performance of more than 68 frames per second.

Figure 8(a) shows the wire frame structure of a virtual cloth, and Figure 8(b) shows the rendered result after our rendering method is applied. The animation method proposed in this paper plausibly generated the cloth behavior as shown in Figure 9. The cloth model shown in the figure was draped on a rotating ball.

The reflectance on the woven surface is dependent on the weave patterns as shown in Figure 10. In the figure, (a) shows the reflectance on the surface with normal weave, that is, weave pattern parameters are 1, 1, and 1 for n_w, n_π, and n_σ, whereas (b) and (c) show the reflectance when the surface is woven with 2/1 twill and 4/1 twill weaves, respectively. The weave parameters for (b) are (2,1,1), and those for (c) are (4,1,1). The figure shown in (d) is the result rendered with parameters (0,100,1), and the surface seems like silk.

The reflectance on the surface can also be controlled with the fiber twist parameters as shown in Figure 11. The rendered image shown in Figure 11(a) has the fiber twist parameter of 0 (i.e., filament yarns). As shown in the figure, the filament yarns produce two orthogonal anisotropic specular reflection areas. As we increase the fiber twist parameters, the fiber becomes a spun yarn and scatters the light as shown in (b) and (c). Figure 11(b) shows the rendering result with the fiber twist parameter of 1, whereas (c) shows the rendering result with the parameter of 3.

Figure 12 shows the frame rate in accordance with the geometric complexity. The performance graph shown in Figure 12(a) demonstrates the frame rate change in accordance with the number of vertices, and Figure 12(b) demonstrates those in accordance with the number of springs. The method remains interactive even when the number of vertices is more than 100,000 and that of springs is more than 1,000,000. The accurate frame rates computed in our experimental system are shown in Table I.

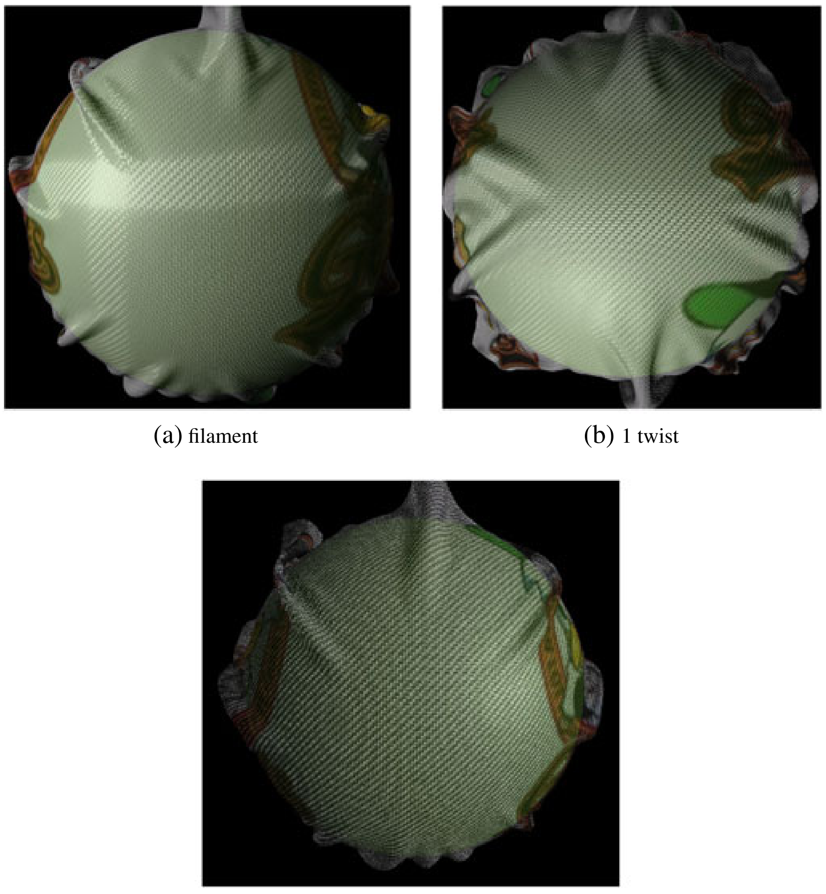
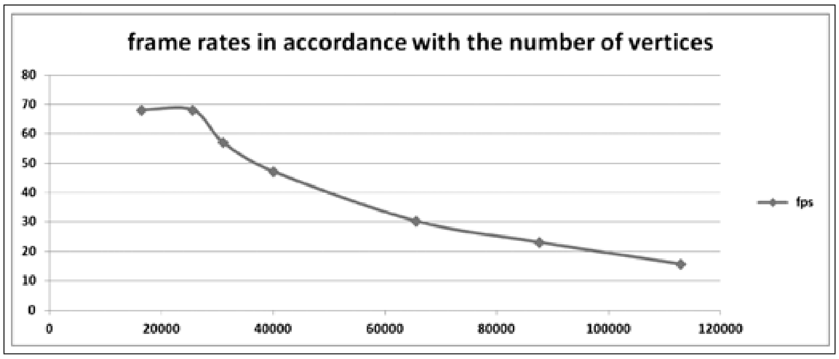
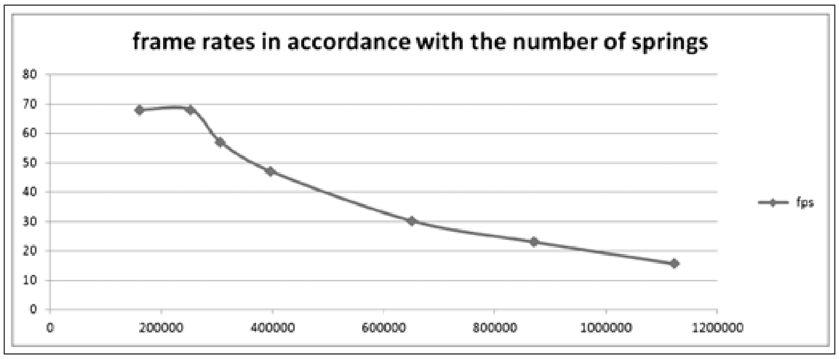

(a) filament (b) 1 twist (c) 3 twist

Figure 11. Rendering with different fiber twist parameters.



(a) frame rate to # vertices



(b) frame rate to # springs

Figure 12. Performance in accordance with the number of (a) vertices and (b) springs.

Table I. Frame rates in accordance with the geometric complexity

# Vertices	# Springs	fps
25,600	253,128	68.1
30,976	306,600	57.1
40,000	396,400	47.2
65,536	650,760	30.3
87,616	870,840	23.1
112,896	1,233,920	15.7

6. CONCLUSION

In this paper, we proposed an efficient and stable method for animating and rendering mass-spring-based complex cloth models for interactive applications. The proposed method is based on the harmonic oscillation model and simulates the mass-spring model with improved stability and accuracy while keeping the integration scheme still explicit. Because the integration scheme is explicit, the method can be easily parallelized, and the performance improvement by exploiting the parallelism in GPU can be easily obtained.

We also proposed a rendering method for the realistic result of cloth animation. The method efficiently and effectively expresses the light scattering on the woven surface. The rendering method is based on alternating anisotropy and improved normal mapping with rotated MDF on the anisotropic reflectance surfaces.

The proposed method can be applied to real-time application for efficient animation and rendering of realistic virtual cloth.

ACKNOWLEDGEMENTS

This work was supported in part by the SW computing R&D program of MKE/KEIT [10035184], "Game Service Technology Based on Realtime Streaming", and also in part by the Ministry of Knowledge Economy (MKE), Korea, under the Information Technology Research Center (ITRC) support program supervised by the National IT Industry Promotion Agency (NIPA) (NIPA-2011-(C-1090-1021-0006)).

REFERENCES

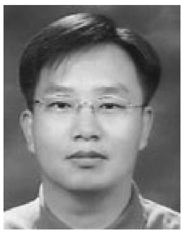
1. Terzopoulos D, Platt J, Barr A, Fleischer K. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)* 1987; **21**(4): 205–214.

2. Pentland A, Williams J. Good vibrations: model dynamics for graphics and animation, In *SIGGRAPH' 1989*, Los Angeles, CA, 1989; 215–222.
3. Volino P, Courshesnes M, Magnenat-Thalmann N. Versatile and efficient techniques for simulating cloth and other deformable objects, In *Proceedings of SIGGRAPH 95*, Los Angeles, CA, 1995; 137–144.
4. Baraff D, Witkin A. Large steps in cloth simulation, In *Proceedings of SIGGRAPH 98*, Orlando, FL, 1998; 43–54.
5. Desbrun M, Schröder P, Barr A. Interactive animation of structured deformable objects, In *Graphics Interface '99*, Los Angeles, CA, 1999; 1–8.
6. Meyer M, Debunne G, Desbrun M, Bar AH. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visulalization and Computer Animation* 2001; **12**: 1–12.
7. Ma L, Baciu JHG. Generating seams and wrinkles for virtual clothing, In *Proceedings of ACM International Conference on Virtual Reality Continuum and Its Applications*, Hong Kong, 2006; 14–17.
8. Torrance KE, Sparrow EM. Theory for off-specular reflection from roughened surfaces. *Journal of Optical Society of America* 1967; **57**(9): 1105–1112.
9. Blinn J, Newell M. Texture and reflection in computer generated images. *Communication of ACM* 1976; **19**(10): 542–547.
10. Blinn J. Models of light reflection for computer synthesized pictures, In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, San Jose, CA, 1977; 192–198.
11. Cook RL, Torrance KE. A reflectance model for computer graphics. *Computer Graphics (ACM SIGGRAPH' 81 Conference Proceedings)* 1981; **15**(3): 307–316.
12. Poulin M, Founier A. A model for anisotropic reflection. *Computer Graphics (ACM SIGGRAPH '90 Conference Proceedings)* 1990; **23**(4): 273–282.
13. Ward G. Measuring and modeling anisotropic reflection. *Computer Graphics (ACM SIGGRAPH '92 Conference Proceedings)* 1992; **26**(2): 265–272.
14. Schilick C. A customizable reflectance model for everyday rendering, In *Proceedings of the 4th Eurographics Workshop on Rendering*, Paris, France, 1993; 73–84.
15. Szirmay-Kalos L, Umenhoffer T, Patow G, Szecsi L, Sbert M. Specular effects on the gpu: State of the art. *Computer Graphics Forum* 2009; **28**(6): 1586–1617.
16. Ashikhmin M, Premoze S, Shirley P. A microfacet-based BRDF generator, In *Proceedings of ACM SIGGRAPH 2000*, New Orleans, LA, 2000; 65–74.
17. Ashikhmin M, Shirley P. An anisotropic phong BRDF model. *Journal of Graphics Tools* 2002; **5**(2): 25–32.
18. Daubert K, Lensch HPA, Heindrich W, Seidel H-P. Efficient cloth modeling and rendering, In *Rendering Techniques 2001: Proceedings of the 12th Eurographics Workshop on Rendering*, London, England, 2001; 63–70.
19. Daubert K, Seidel H-P. Hardwarebased volumetric knitwear. *Computer Graphics Forum (Eurographics 2002 Proceedings)* 2002; **21**: 575–584.
20. Gröller E, Rau R, Strasser W. Modeling and visualization of knitwear. *IEEE Transaction on Visualization and Computer Graphics* 1995; **1**(4): 302–310.
21. Gröller E, Rau R, Strasser W. Modeling textile as three dimensional texture, In *Rendering Techniques 1996: Proceedings of the 7th Eurographics Workshop on Rendering*, Porto, Portugal, 1996; 205–214.
22. Meissner M, Eberhardt B. The art of knitted fabrics, realistic and physically based modeling of knitted patterns. *Computer Graphics Forum (Eurographics 1998 Proceedings)* 1998; **17**(3): 355–362.
23. Xu Y, Lin YCS, Zhong H, Wu E, Guo B, Shum H. Photorealistic rendering of knitwear using the lumislice, In *Proceedings of SIGGRAPH 2001*, Los Angeles, CA, 2001; 391–398.
24. Yasuda T, Yokoi S, Toriwaki J, Inagaki K. A shading model for cloth objects. *IEEE Computer Graphics and Applications* 1992; **12**(6): 15–24.
25. Adabala N, Magnenat-Thalmann N, Fei G. Visualization of woven cloth, In *Proceedings of the 14th Eurographics Workshop on Rendering (ACM International Conference Proceeding Series),* Vol. 44, Leuven, Belgium, 2003; 178–185.
26. Adabala N, Magnenat-Thalmann N, Fei G. Real-time rendering of woven clothes, In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, Osaka, Japan, 2003; 41–47.
27. Dana KJ, Nayar SK, Van Ginneken B, Koenderink JJ. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics* 1999; **18**(1): 1–34.
28. Lawrence J, Ben-Artzi A, Decoro C, Matusik W, Pfister, H, Ramamoorthi R, Rusinkiewicz S. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics* 2006; **25**(3): 735–745.
29. McAllister DK, Lastra AA, Heidrich W. Efficient rendering of spatial bi-directional reflectance distribution functions, In *Proceedings of the 17th Eurographics/SIGGRAPH Workshop on Graphics Hardware (EGGH-02)*, Saarbrueken, Germany, 2002; 79–88.
30. Wang J, Zhao S, Tong X, Synder J, Guo B. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Transactions on Graphics (SIGGRAPH 2008)* 2008; **27**(3): 41:1–41:9.

31. Wang J, Ren P, Gong M, Snyder J, Guo B. All-frequency rendering of dynamic, spatially-varying reflectance, In *Proceedings of ACM SIGGRAPH Asia 2009*, Yokohama, Japan, 2009; 1–10.
32. Sattler M, Sarlette R, Klein R. Efficient and realistic visualization of cloth, In *EGRW '03: Proceedings of the 14th Eurographics Workshop on Rendering*, Leuven, Belgium, 2003; 167–177.
33. Zinke A, Weber A. Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics* 2007; **13**(2): 342–356.
34. Kang Y-M. Realtime rendering of realistic fabric with alternation of deformed anisotropy. In *Proceedings of Motion in Games 2010*, Vol. 6459, Lecture Notes in Computer Science. Springer, Berlin, 2010; 301–312.
35. Kang Y-M, Cho H-G, Kim S-S. Plausible and realtime rendering of scratched metal by deforming MDF of normal mapped anisotropic surface. *Journal of WSCG* 2011; **19**(3): 101–110.
36. Rushmeier H, Taubin G, Gueziec A. Applying shapes from lighting variation to bump map capture, In *Proceedings of Eurographics Rendering Workshop '97*, Saint-Etiemme, France, 1997; 35–44.
37. Heidrich W, Seidel H-P. Realistic, hardware-accelerated shading and lighting, In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, 1999; 171–178.
38. Pharr M, Humphreys G. *Physically Based Rendering*. Elsevier (Morgan Kaufman Publishers), Burlington, MA, 2004.

AUTHORS' BIOGRAPHIES

Young-Min Kang is an associate professor at the Department of Game Engineering, Tongmyong University, Busan, Korea. He received his BS, MSc, and PhD from Pusan National University, Korea. He worked as a researcher for the next-generation game technology project in ETRI, Daejeon, Korea, and temporarily worked at MIRALab, Geneve, Switzerland. His major research interests involve interactive physics-based animation, real-time rendering, and game contents.



Chang-Sik Cho received his BS and MS degrees from Kyungpook National University, Korea, in 1993 and 1995. In February 1995, he joined the Real-Time Multimedia Research Team at the Electronics and Telecommunications Research Institute (ETRI), Korea, where he is currently a team leader and senior researcher. His research interests are real-time multimedia communication and game streaming.