Young-Min Kang
Digital Contents Research Division
Electronics and Telecommunication
Research Institute
Daejeon, 305-350 South Korea
ymkang@etri.re.kr

Hwan-Gue Cho
Department of Computer
Engineering
Pusan National University
South Korea

Real-Time Animation of Complex Virtual Cloth with Physical Plausibility and Numerical Stability

Abstract

Numerical instability has been a major obstacle to real-time cloth animation. Although the implicit method can make the simulation stable, it is still impossible to generate interactive animation when the geometric model of the virtual cloth is complex enough to represent realistic details. In this paper, we propose efficient animation techniques for the real-time animation of complex deformable objects. The proposed method exploits the stability of the implicit integration in order to use sufficiently large time steps for real-time environments, and obtains the approximate solution with optimized computation and storage. Unlike previous methods for real-time cloth animation, any severe simplification of the problem itself has been avoided in order to produce realistic motions of complex models. In addition, the proposed method deals with instability caused by a force derivative. This problem has been avoided in the previous work with excessive approximation or additional manipulation. However, the technique in this paper avoids instability with a minimal approximation, and does not increase any computational or geometric complexity. The method can be successfully integrated in various real-time environments, such as PC games and VR systems.

1 Introduction

The animation of a nonrigid object is much more time-consuming than that of a rigid object because it incurs a special numerical problem known as the instability problem. In this paper, efficient animation techniques are proposed for a realistic cloth model, as shown in Figure 1. In the early stages of cloth modeling, geometric approaches were widely employed because physically based correct simulation required too heavy a computational cost for the hardware environments at that time (Ng & Grimsdale, 1996).

Feynman's (1986) model for generating the appearance of cloth is one of the earliest works that represents the cloth model on a physical basis. However, the method was not devised to generate the dynamic motion of cloth. Breen, House, and Wozny (1994) proposed a method based on the particle system, which was also devised to achieve the restricted goal of generating the final equilibrium state. Their method was extended by Eberhardt, Weber, and Strasser (1996).

Presence, *Vol. 13, No. 6, December 2004,* 668–680

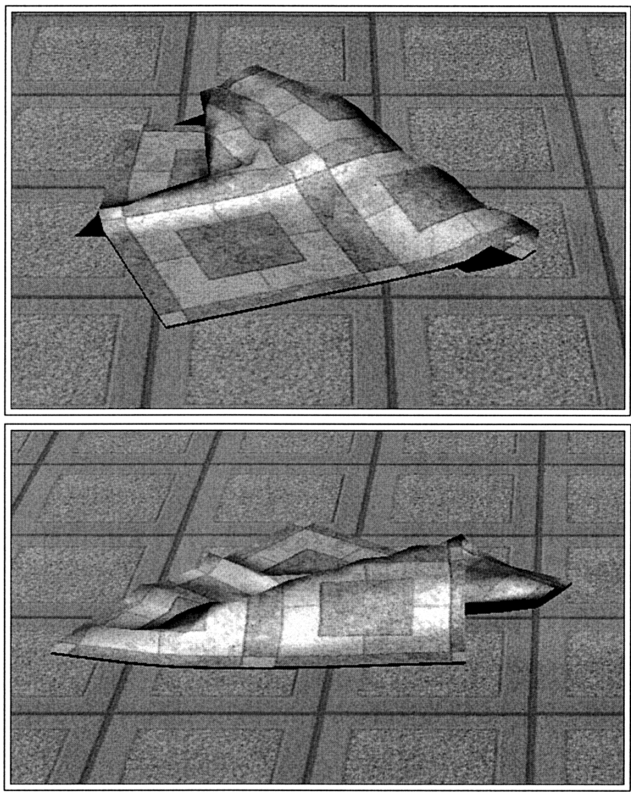


Figure 1. *Virtual cloth models to be used in our real-time environments.*

Terzopoulos, Platt, Barr, and Fleischer (1987) characterized cloth simulation as a deformable surface problem. Their work provided a fundamental basis for physically based modeling for the animation of deformable objects, and their idea has been followed by various research groups. Carignan, Yang, Thalmann, and Thalmann (1992) have proposed various techniques for dressed virtual characters. However, computational efficiency was not the major concern of their methods. Many researchers have endeavored to devise efficient methods for cloth simulation, and one of the most important advances in recent years is the use of implicit integration (Baraff & Witkin, 1998). This method could use arbitrarily large time steps during the simulation. However, implicit integration requires the solution of a linear system, which involves a large sparse matrix (Baraff & Witkin; Desbrun, Schröder, & Barr, 1999).

Some researchers have tried to devise efficient methods that can interactively manipulate deformable objects in virtual environments, and a few methods have been proposed (Desbrun et al., 1999; Meyer, Debunne, Desbrun, & Bar, 2001). Desbrun et al. proposed an efficient technique that approximates the matrix involved in the linear system as a constant matrix and precomputes the inverse matrix. Although this method is more efficient than the original implicit method, it also involves an $O(n^2)$-sized matrix, and the precomputed inverse matrix is usually a dense matrix. Therefore, their method can be used only for moderately complex mesh structures. A stable method that approximates the solution with a direct update formula was also introduced (Kang, Choi, Cho, & Park, 2001). However, the stability of the method was obtained by introducing excessive damping.

The heavy computational cost of cloth simulation led researchers to devise various hybrid methods. Oshita and Makinouchi (2001) proposed a geometric approach that generates wrinkly details on cloth models composed of sparse particles. Texture-mapping techniques have also been used for producing a realistic appearance of the cloth model (Hadap, Bangarter, Volino, & Magnenat-Thalmann, 1999). However, the geometric or texture-based approaches hardly produce realistic cloth animation.

Recently, Cordier and Magnenat-Thalmann (2002) proposed real-time animation techniques for a fully dressed virtual human. The fundamental idea of their method is the employment of different methods for different parts of the dress. However, the method uses conventional simulation methods for floating cloth, such as a long skirt, and the animation of floating cloth is the bottleneck of the performance.

2 Problem Formulation

In this paper, a mass-spring-based cloth model is employed. The spring force can be easily computed with Hooke's law, and the movement of a particle can be obtained by numerically integrating the accumulated force on the particle. Let $\mathbf{f}_j^i$ denote the force on the

mass-point i caused by the spring between the mass-points i and j. The internal force $\mathbf{f}_i$ on the ith mass-point can then be calculated as follows:

$$\mathbf{f}_i = \sum_{(i,j)\in E} \mathbf{f}_i^j = \kappa_{ij} \sum_{(i,j)\in E} (|\mathbf{X}_j - \mathbf{X}_i| - l_{ij}^0) \frac{(\mathbf{X}_j - \mathbf{X}_i)}{|\mathbf{X}_j - \mathbf{X}_i|} \quad (1)$$

where κ_{ij} and l_{ij}^0 denote the spring constant and the rest length of the spring between the ith and the jth mass-points respectively. The 3D vector x_i denotes the location of the ith mass-point, and E is a set of edges that correspond to springs between mass-points.

Instability is a common problem in any kind of explicit integration schemes. Therefore, any explicit integration method can hardly be selected as a numerical integrator for real-time animation of deformable objects, especially for stiff objects such as cloth (Witkin & Baraff, 1994; Kass, 1995). We can stably animate mass-spring-based objects by integrating the spring forces with the following backward Euler method (Desbrun et al., 1999; Kass):

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix} \quad (2)$$

where h denotes the time interval, and $\mathbf{v}$ denotes the vector composed of the velocity vectors of the mass-points. Similarly, $\mathbf{f}$ and $\mathbf{x}$ are the vectors of force and location, respectively. The matrix $\mathbf{M}$ is the mass matrix. The superscripts t and $t + h$ denote time, and refer to the current state and the next state respectively.

The objective of computation is to find $\mathbf{x}^{t+h}$ (i.e., the new positions of the mass-points at the next step), and the vector $\mathbf{x}^{t+h}$ can be easily determined when we know $\mathbf{v}^{t+h}$. The next velocity can be computed if we find the velocity change $\Delta\mathbf{v}^{t+h}$ that denotes $\mathbf{v}^{t+h} - \mathbf{v}^t$. Therefore, cloth animation with Equation 2 is reduced to finding $\Delta\mathbf{v}^{t+h}$ as follows:

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t+h} \quad (3)$$

The difficulty arises from the unknown force at time $t + h$. Because Hooke's law can compute only $\mathbf{f}^t$, the force at the next time step should be approximated. Because the force at the current state can be easily computed, the force at the next time step can be approximated by applying the Taylor expansion as follows:

$$\begin{aligned} \mathbf{f}^{t+h} &= \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta\mathbf{x}^{t+h} = \mathbf{f}^t + \mathbf{J}\Delta\mathbf{x}^{t+h} \\ &= \mathbf{f}^t + \mathbf{J}(\mathbf{v}^t + \Delta\mathbf{v}^{t+h})h \end{aligned} \quad (4)$$

where $\mathbf{J}$ denotes the Jacobian matrix of the force vector with respect to the position vector, and $\Delta\mathbf{x}^{t+h}$ is the positional change from time t to $t + h$. The size of the matrix $\mathbf{J}$ is $n \times n$, and each component of $\mathbf{J}$ is a 3×3 submatrix. Since the force does not depend on the velocity, a partial derivative with respect to the velocity does not need to be considered.

By substituting $\mathbf{f}^{t+h}$ in Equation 3 with the approximation shown in Equation 4, the stable animation of the mass-spring model is finally reduced to a linear system, solving as follows:

$$(\mathbf{M} - h^2\mathbf{J})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t \quad (5)$$

3 Real-Time Animation with Approximate Solution

In order to produce plausible cloth animation in real-time environments, the solution of the linear system shown in Equation 5 should be obtained in an efficient way. In this section, an efficient method that obtains the approximate solution of Equation 5 is proposed.

For simplicity, let us define a matrix $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{M} - h^2\mathbf{J} \quad (6)$$

The matrix $\mathbf{W}$ is sparse because $\mathbf{M}$ is a diagonal matrix, and a component of the Jacobian matrix $\mathbf{J}_{ij}$ has a non-zero matrix only when the mass-points i and j are linked with a spring. The components of the matrix $\mathbf{W}$ are 3×3 submatrices, which can be easily computed. The component in the ith row and jth column ($i \neq j$), $\mathbf{W}_{ij}$, is $-h^2\mathbf{J}_{ij}$, and the diagonal components, $\mathbf{W}_{ii}$, can be computed as $\mathbf{M}_i - h^2\mathbf{J}_{ii}$. As $\mathbf{J}_{ii}$ is the negated sum of the off-diagonal components $\mathbf{J}_{ij}$ in the same row, the diago-

nal components can be expressed as $\mathbf{M}_i + h^2 \Sigma_{(i,j)\in E} \mathbf{J}_{ij}$, where E is the set of springs.

The matrix in the linear system can be regarded as a block matrix composed of 3×3 submatrices. Previous real-time techniques (Desbrun et al., 1999; Kang et al., 2001) approximated these submatrices as scalar values. Although the result is stable enough to use arbitrarily large time steps, such excessive approximation introduces unnecessary damping in every direction. In this paper, these submatrices are not approximated into scalar values in order to minimize the factitious slowdown.

The vector $(h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t)$ in the right side of Equation 5 can be described as $h(\mathbf{f}^t + h\mathbf{J}\mathbf{v}^t)$ and the additional forces $h\mathbf{J}\mathbf{v}^t$ can be considered as viscosity forces (Desbrun et al., 1999). Because of the sparseness of the Jacobian matrix, we can efficiently compute the vector $h\mathbf{J}\mathbf{v}^t$. If we denote the sum of spring forces and viscosity forces $(\mathbf{f}^t + h\mathbf{J}\mathbf{v}^t)$ as internal forces $\tilde{\mathbf{f}}$, each component (3D vector) of internal force vector can be calculated with the consideration of linked mass-points as follows:

$$\begin{aligned}\tilde{\mathbf{f}}_i^t &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}\mathbf{v}_j^t + h\mathbf{J}_{ii}\mathbf{v}_i^t \\ &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}(\mathbf{v}_j^t - \mathbf{v}_i^t)\end{aligned} \tag{7}$$

Then, the linear system in Equation 5 can be rewritten as follows:

$$\mathbf{W}\Delta\mathbf{v}^{t+h} = h\tilde{\mathbf{f}}^t \tag{8}$$

The linear system shown in Equation 8 is the actual problem that should be solved for stable cloth animation. In the formal problem, the matrix $\mathbf{W}$ and the vector $\tilde{\mathbf{f}}^t$ can be efficiently computed without any difficulties. The problem is how to find a reasonable approximation of $\Delta\mathbf{v}^{t+h}$ that satisfies the linear system.

The linear system of Equation 8 can be decomposed into a similar equations, and each equation can be described as follows:

$$\Delta\mathbf{v}_i^{t+h} = \mathbf{W}_{ii}^{-1}\left(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h}\right) \tag{9}$$

Each equation, unfortunately, cannot be computed independently because it depends on the result of some other equations ($\Delta\mathbf{v}_j^{t+h}$). Note that Equation 9 is the Jacobi iteration scheme if we introduce another superscript k as the number of iterations. Let $\Delta\mathbf{v}_j^{t+h^{(k)}}$ be the result after the k iterations, and suppose that the initial guess of velocity change of each mass-point is simply $\Delta\mathbf{v}_i^{t+h} = \mathbf{W}_{ii}^{-1}h\tilde{\mathbf{f}}_i^t$ by ignoring the velocity changes of linked mass-points. The iterative scheme can then be described as follows:

$$\begin{aligned}\Delta\mathbf{v}_i^{t+h^{(0)}} &= \mathbf{W}_{ii}^{-1}h\tilde{\mathbf{f}}_i^t \\ \Delta\mathbf{v}_i^{t+h^{(k+1)}} &= \mathbf{W}_{ii}^{-1}\left(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h^{(k)}}\right)\end{aligned} \tag{10}$$

The diagonal components of the matrix in Equation 8 are usually more dominant than the off-diagonal components, so that the initial guess of the iterative scheme is already a good approximation of the solution. Therefore, the satisfactory approximation of the solution can be obtained with a small number of iterations. In most cases, the approximate solution after only one iterative update was stable and plausible enough to be used in a real-time system. Let us define $\mathbf{p}_i$ as the initial guess for $\Delta\mathbf{v}_i^{t+h^{(0)}}$. The velocity change of each mass-point can then be approximated as follows:

$$\begin{aligned}\mathbf{p}_i &= h\mathbf{W}_{ii}^{-1}\tilde{\mathbf{f}}_i \\ \Delta\mathbf{v}_i^{t+h} \simeq \Delta\mathbf{v}_i^{t+h^{(1)}} &= \mathbf{p}_i + h^2\mathbf{W}_{ii}^{-1} \sum_{(i,j)\in E} \mathbf{J}_{ij}\mathbf{p}_j\end{aligned} \tag{11}$$

The fast approximation of the velocity change of each mass-point with Equation 12 enables real-time cloth animation. The proposed method is efficient because it computes the velocity change of each mass-point by considering only the linked mass-points. The method requires the system to compute the inverse of the diagonal submatrices $\mathbf{W}_i$. Compared to the whole linear system solving, the burden of calculating the 3×3 inverse matrix is trivial. In other words, the method can update the state as efficiently as explicit methods. We found that the approximate solution in Equation 12 produces sufficiently stable results for the most reasonable situation in real-time cloth animation.

Although the proposed approximate method shows a good stability, the motion generated with Equation 12

is produced without any consideration of damping. Actual movement of any object in the real world reaches static equilibrium by a damping effect that dissipates the energy. A simple damping model can be described as follows:

$$\mathbf{f}_d = -C_d \sum_{(i,j)\in E} (\mathbf{v}_i - \mathbf{v}_j) \tag{12}$$

The sparseness of the matrix $\partial \mathbf{f}_d / \partial \mathbf{v}$ is exactly the same as the sparseness of $\mathbf{J}$ because the 3×3 submatrix $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ij}$ is a nonzero matrix only when the mass-points i and j are linked by a spring edge. $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ij}$ is $C_d \mathbf{I}_3$ only if the two mass-points i and j are linked. The diagonal submatrix $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ii}$ can also be simply computed as $-n_i C_d \mathbf{I}_3$ where n_i is the number of mass-points linked to the mass-point i. Note that this damping model affects only the diagonal components of the nonzero submatrices. Therefore the velocity change in Equation 12 can be modified to include damping without introducing significant additional computation. In order to integrate the damping effect into the velocity-change computation in Equation 12, let us define $\mathbf{W}'_{ii}$, $\mathbf{J}'_{ij}$, and $\mathbf{p}'_i$ as follows:

$$\mathbf{W}'_{ii} = (\mathbf{W}_{ii} + n_i h C_{\mathrm{d}} \mathbf{I}_3)^{-1} \tag{13}$$

$$\mathbf{J}'_{ij} = \mathbf{J}_{ij} + \frac{C_d}{h} \mathbf{I}_3$$

$$\mathbf{p}'_i = h \mathbf{W'}_{ii}^{-1} \tilde{\mathbf{f}}_i$$

The velocity change with respect to damping can then be obtained as follows:

$$\Delta \mathbf{v}_i^{t+h} \simeq \mathbf{p}'_i + h^2 \mathbf{W}'^{-1}_i \sum \mathbf{J}'_{ij} \mathbf{p}'_j \tag{14}$$

Equation 14 enables the system to successfully deal with the damping force without instability. The damping force, which depends on the velocity of the model, is also numerically integrated with the implicit integration scheme in order to guarantee the stability of the system, and the derivative of the damping force is efficiently computed and included in the system.

4 Stabilized Force Derivative for Plausible Real-Time Animation

In the implicit integration scheme, the force at the next time step was approximated with the first derivative as shown in Equation 4. However, the force derivative leads to another problem. The element at the ith row and the jth column of the Jacobian matrix is the derivative of the force on the mass-point i with respect to the location of the mass-point j, and can be expressed as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathrm{x}_j} = \kappa_{ij} \frac{|\mathrm{x}_j - \mathrm{x}_i| - l^0_{ij}}{|\mathrm{x}_j - \mathrm{x}_i|} \mathbf{I}_3 + \kappa_{ij} \frac{l^0_{ij}}{|\mathrm{x}_j - \mathrm{x}_i|} \left(\frac{(\mathrm{x}_j - \mathrm{x}_i)(\mathrm{x}_j - \mathrm{x}_i)^{\mathrm{T}}}{|\mathrm{x}_j - \mathrm{x}_i|^2} \right) \tag{15}$$

where $\mathbf{I}_3$ denotes the 3×3 identity matrix.

The force derivative shown in Equation 15 does not incur any problem when the spring is stretched. However, it gives serious problems if the distance between the mass-points i and j is smaller than l^0_{ij}. When the distance between two mass-points approaches 0, the derivative of the force becomes an arbitrarily large negative value, and makes the system fail (Choi & Ko, 2002).

In order to avoid the undesirable instability caused by compressed springs, several approaches have already been proposed. The previous methods can be categorized into three classes:

- **Scalar approximation:** The simplest approach approximates the force derivative matrix in Equation 15 to a positive scalar value κ_{ij}. Some techniques for real-time cloth animation have used this approach because of its simplicity and efficiency (Desbrun et al., 1999; Meyer et al., 2001; Kang et al., 2001). However, this simple approximation introduces excessive damping, and consequently leads to unrealistic motion. Therefore, this approach can be applied only to cloth models with simple geometry.
- **Matrix approximation:** Some methods use a simplified force derivative matrix. In these methods, $\kappa_{ij}((\mathrm{x}_j - \mathrm{x}_i)(\mathrm{x}_j - \mathrm{x}_i)^{\mathrm{T}})/(|\mathrm{x}_j - \mathrm{x}_i|^2)$ was used as ap-

proximate force derivative (Volino & Magnenat-Thalmann, 2000). The problem in this method is that it performs unnecessary approximation when springs are stretched.

- **Postbuckling manipulation:** In order to avoid the instability caused by compressed springs, Choi and Ko (2002) proposed a postbuckling manipulation method. They assumed that the spring force and its derivative are 0 when the spring is compressed. Instead of using the unstable force derivative, they employed a distinct particle interaction method that manipulates the buckling situation. Although their method avoids instability and produces a realistic appearance of cloth, it has a few problems. It requires special links between particles, and the mass-points linked with the special links are not simulated with a mass-spring model. Their method performs additional simulation for the special links and this additional simulation phase is not based on the mass-spring model. Therefore, the special links and additional simulation phase increase the computational and geometric complexity.

In order to obtain a stable and plausible result, we approximated the force derivative only when the spring is compressed. When the spring e_{ij} is compressed, the $(|\mathrm{x}_j - \mathrm{x}_i| - l_{ij}^0)/|\mathrm{x}_j - \mathrm{x}_i|$ portion of the first term in Equation 15 becomes negative, and may incur the instability problem. To avoid this undesirable situation, we should keep this factor positive. We approximate only the first term by assuming l_{ij}^0 has the same value as $|\mathrm{x}_j - \mathrm{x}_i|$ when the distance between two linked mass-points i and j is smaller than the rest length l_{ij}^0. Even when the spring is compressed, the second term in Equation 15 is not approximated with this assumption. This method avoids unnecessary approximation when springs are stretched, and enables the system to avoid failure caused by compressed springs. In other words, the accurate force derivative is used when the spring is stretched and the stabilized approximate force derivative is used when the spring is compressed. Therefore, the force derivative used for the proposed method can be expressed as follows:

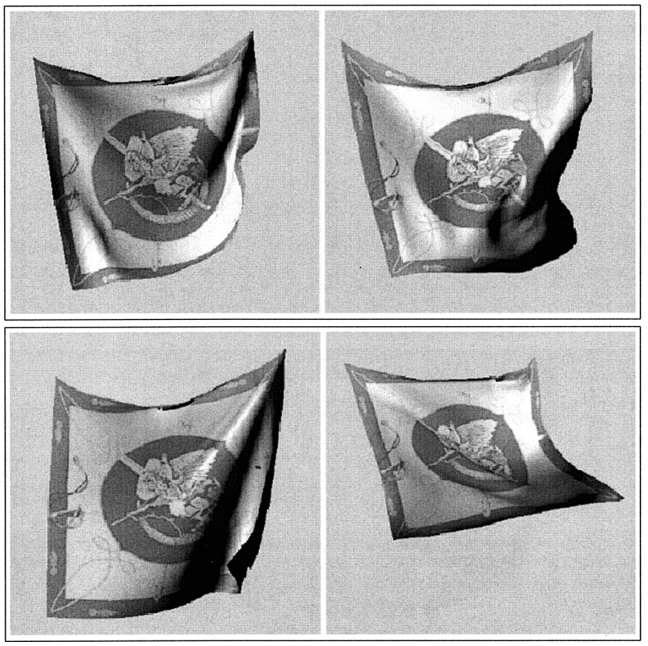


Figure 2. *Real-time animation produced with the proposed method.*

$$if\ l_{ij}^0 < |\mathrm{x}_j - \mathrm{x}_i|$$

$$\frac{\partial \mathbf{f}_i}{\partial \mathrm{x}_j} = \kappa_{ij} \frac{|\mathrm{x}_j - \mathrm{x}_i| - l_{ij}^0}{|\mathrm{x}_j - \mathrm{x}_i|} \mathbf{I}_3 + \kappa_{ij} \cdot \frac{l_{ij}^0}{|\mathrm{x}_j - \mathrm{x}_i|} \left(\frac{(\mathrm{x}_j - \mathrm{x}_i)(\mathrm{x}_j - \mathrm{x}_i)^{\mathrm{T}}}{|\mathrm{x}_j - \mathrm{x}_i|^2} \right)$$

$$if\ l_{ij}^0 \geq |\mathrm{x}_j - \mathrm{x}_i|$$

$$\frac{\partial \mathbf{f}_i}{\partial \mathrm{x}_j} = \kappa_{ij} \frac{l_{ij}^0}{|\mathrm{x}_j - \mathrm{x}_i|} \left(\frac{(\mathrm{x}_j - \mathrm{x}_i)(\mathrm{x}_j - \mathrm{x}_i)^{\mathrm{T}}}{|\mathrm{x}_j - \mathrm{x}_i|^2} \right) \quad (16)$$

The approximate force derivative shown in Equation 16 gives a reasonable solution for real-time environments. The proposed force derivative minimizes the use of approximation in order to produce plausible motion, and does not increase any computational or geometric complexity. Figure 2 shows the real-time animation result produced with the proposed method.

5 External Forces and Collision Handling

An actual cloth in the real world experiences various external forces caused by gravity or interaction with

air. Collision should be also considered for realistic cloth animation.

5.1 External Forces for Plausible Animation

In order to generate a realistic animation of a thin flexible object, two kinds of forces, drag force and lift force, must be considered. The magnitudes of drag force $|\mathbf{f}^{drag}|$ and lift force $|\mathbf{f}^{lift}|$ are well known, as follows (Brancazio, 1988):

$$|\mathbf{f}^{drag}| = \frac{1}{2} C_D \rho |\nu|^2 S \sin\theta$$
$$|\mathbf{f}^{lift}| = \frac{1}{2} C_L \rho |\nu|^2 S \cos\theta \qquad (17)$$

where C_D and C_L are the drag force coefficient and the lift force coefficient respectively, ρ is the density of a fluid, ν is the velocity of an object relative to the fluid ($\mathbf{v}^{object} - \mathbf{v}^{fluid}$), S is the area of the object's surface, and θ is the angle between ν and the surface. The direction of the drag force is opposite to ν, and the direction of the lift force is perpendicular to ν.

In order to implement a simple and efficient air-interaction model, let us define $\hat{\nu}_i$ as the relative velocity of the ith mass-point to the air, and $\hat{\mathrm{n}}_i$ as the unit normal of the ith mass-point. Similarly, $\hat{\nu}_i$ denotes $\nu_i/|\nu_i|$. The angle between $\hat{\mathbf{n}}_i$ and $\hat{\nu}_i$ is then $\pi/2 - \theta$. Thus, $\hat{\mathbf{n}}_i \cdot \hat{\nu}_i$ is $\sin\theta$ ($= \cos(\pi/2 - \theta)$). Therefore, the magnitude of the drag force is proportional to $|\hat{\mathbf{n}}_i \cdot \hat{\nu}_i|$. Because the direction of the drag force is the opposite direction of the velocity, the magnitude of the drag force can be implemented as follows:

$$\mathbf{f}_i^{drag} = -K_D |\hat{\mathbf{n}}_i \cdot \hat{\nu}_i| |\nu_i|^2 \hat{\nu}_i \qquad (18)$$

where K_D is the control parameter for the drag force.

The direction of the lift force should be perpendicular to the direction of ν_i. We define $\mathbf{U}_i$, the direction of the lift force on the ith mass-point, as follows:

$$\mathrm{U}_i = (\hat{\mathbf{n}}_i \times \hat{\mathbf{v}}_i) \times \hat{\nu}_i, \; if\, \hat{\mathbf{n}}_i \cdot \hat{\nu}_i > 0$$
$$(-\hat{\mathbf{n}}_i \times \hat{\nu}_i) \times \hat{\nu}_i, \; otherwise \qquad (19)$$

In order to avoid the *cosine* function, we simply substitute $1 - \hat{\mathbf{n}}_i \cdot \hat{\nu}_i$ for $\cos\theta$. The lift force was implemented as follows:

$$\mathbf{f}_i^{lift} = (K_L(1 - \hat{\mathbf{n}}_i \cdot \hat{\nu}_i)|\nu_i|^2)\mathbf{U}_i \qquad (20)$$

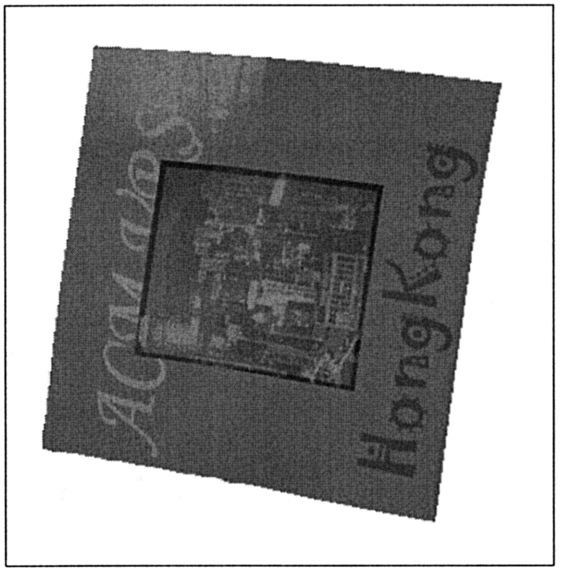
where K_L is the control parameter for the lift force. Although the implemented lift-force effect is actually proportional to $1 - |\sin\theta|$, we found this implementation gives plausible lift-force effects. The magnitude of the lift force is maximal when the drag force is minimal, and vice versa.

Figure 3 shows the effect of the drag force and lift force. Figure 3 (a) shows a snapshot of cloth animation when the drag and lift forces are not taken into account, while (b) shows the movement of the cloth affected by both forces. As shown in the figures, the drag and lift forces increase the realism.

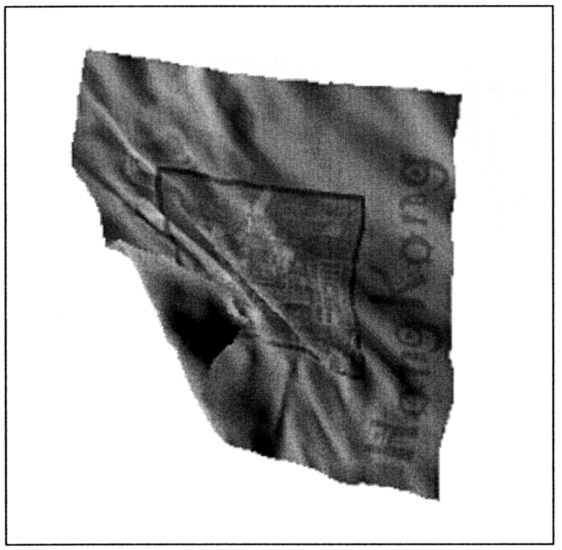
5.2 Collision Handling

Collision handling is essential for realistic animation, and it has been one of the major subjects in computer graphics literature. Self-collision handling has especially been a computational bottleneck in cloth animation. The most prominent work on self-collision handling was proposed by Volino and Magnenat-Thalmann (1994); Provot (1997) improved the method.

In this paper, we use an efficient self-collision handling method in order to animate cloth models in real-time environments. This method uses a hash function to store n mass-points of a cloth model into n different buckets. This hashing-based method constructs the bounding box of the cloth model, and the bounding box is divided into cubic cells along the coordinate axis. A mass-point will then be located in one cell that can be described with three integers (i_x, i_y, i_z). Since we must alleviate the computational and storage overhead, we transform the 3D cell index (i_x, i_y, i_z) into 1D space with a hash function $h(i_x, i_y, i_z)$. The hash function transforms the cell index into a non-negative integer k ($0 \leq k < n$). The value k can then be used as the bucket index, and each mass-point located in the cell (i_x, i_y, i_z) is



(a) vacuum motion



(b) motion in air

Figure 3. *Free falling of cloth model: (a) Falling cloth without consideration of air interaction, (b) Falling cloth with consideration of air interaction.*

stored in the bucket k. The mass-points in an identical bucket are then considered collision candidates. A serious problem of this method is that two mass-points can approach each other without being located in an identical cell. In order to avoid such an undesirable situation, the method employs two different cell-division methods with different cell sizes. This enables each mass-point to keep two different bucket indexes, and to be stored in two different buckets. Another problem is that adjacent mass-points are usually located in the same cell. We avoid this problem by exploiting previous self-collision techniques that divide the cloth surface into collision-free subsurfaces (Volino & Magnenat-Thalmann, 1994; Provot, 1997). After dividing the surface into m collision-free subsurfaces, we prepared m bucket sets. Each of the bucket sets has n buckets. If a mass-point i is included in a subsurface $s(0 \leq s < m)$ and its hashed index is k, the mass-point will be stored in the kth bucket in the sth bucket set. If we do not consider the mass-points in the same bucket set, the adjacent mass-points are easily excluded from the collision candidates.

6 Experimental Results

The proposed method has been successfully implemented for an interactive cloth-animation system. The cloth-animation system has been developed on PC environments. It is obvious that the time complexity of the proposed method is linearly proportional to the number of mass-points and spring edges because the update process for each mass-point considers only the neighboring mass-points linked by springs.

Table 1 shows the time required to compute the next state in normal PC environments (3 GHz Pentium 4 CPU with 1 Gb main memory). The fourth and fifth columns show the time required for numerical integration and self-collision handling, respectively. As shown in the table, the proposed method can generate real-time animation of virtual cloth with thousands of polygons. Note that the computational complexity is linearly proportional to the geometric complexity.

Figure 4 shows the real-time animation of a cloth model composed of 2048 polygons. As shown in the figure, the proposed method generated a plausible appearance of the cloth model. The size of the time step was 1/150 s when the result shown in the figure was generated. The geometric complexity of the model was

Table 1. *Measured Performance of the Proposed Method in PC Environments*

Vertices	Links	Faces	Frame update (s)	Self-collision handling (s)
289	800	512	0.0005	0.00092
1089	3136	2048	0.00174	0.00354

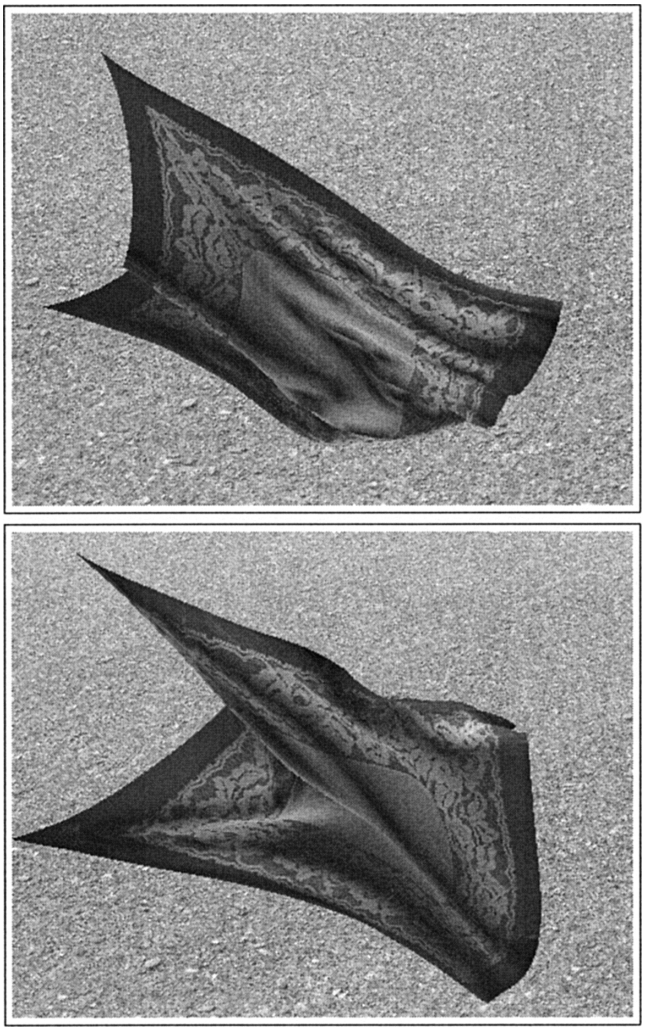


Figure 4. *Realistic cloth model used for real-time animation.*

great enough to reproduce realistic motion of virtual cloth in real-time. Figure 5 shows the animation result with the self-collision handling technique. The computational efficiency and the physical plausibility of the

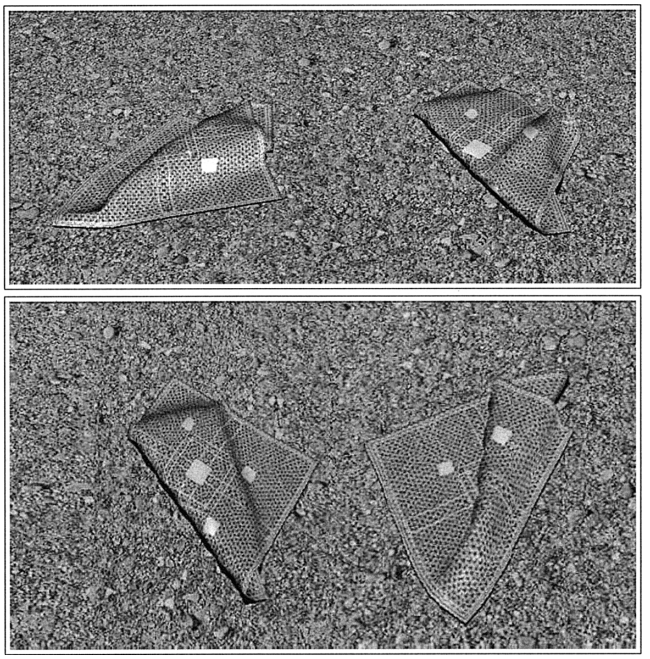
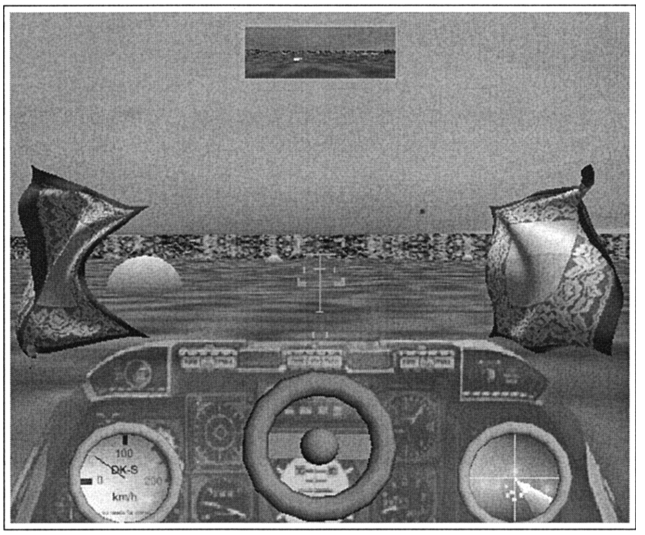


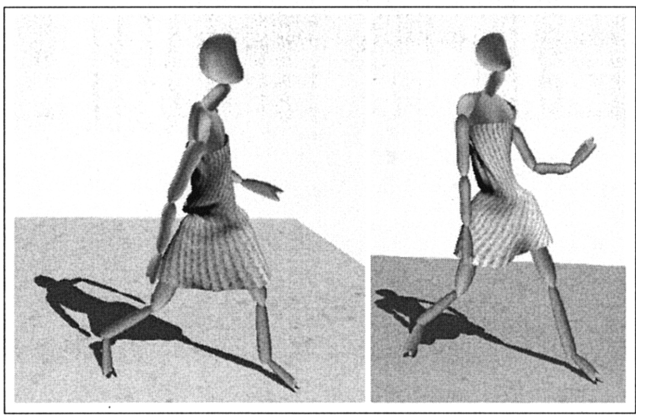
Figure 5. *Real-time cloth animation with efficient self-collision handling.*

proposed techniques enable interactive systems to integrate a complex virtual cloth model. Figure 6(a) shows the cloth-animation result in an actual game running on a PC, and Figure 6(b) shows the interactive animation of a dressed virtual human implemented with the proposed method. The method efficiently generated a plausible and interactive animation of the dressed human.

Although the proposed method is stable and efficient, another important criterion should be taken into account: physical plausibility. If the resulting animation is not plausible, the minimized computational cost is meaningless. Because the proposed method does not use any severe approximation, it is expected that the method will produce plausible animation.



(a) Cloth in game environments



(b) Dressed virtual character

Figure 6. *Real-time cloth-animation results.*

In order to investigate the plausibility of the proposed method, two kinds of evaluation functions were devised: positional error ε and directional similarity S. Let us denote Δx^{ref} the reference motion from the previous checkpoint to the current checkpoint. The reference motion is assumed to be physically accurate. Similarly, Δx^{exp} denotes the experimental movement generated by the proposed real-time method with a large time step. The positional error of the experimental motion of the mass-point i with respect to the reference motion is represented as $\varepsilon_i(\Delta x^{ref}, \Delta x^{exp})$, and it is defined as follows:

$$\varepsilon_i(\Delta x^{ref}, \Delta x^{exp}) = \frac{|\Delta x_i^{ref} - \Delta x_i^{exp}|^2}{|\Delta x_i^{ref}|^2}$$

where Δx_i^{ref} denotes the motion of the mass-point i in the reference motion, and Δx_i^{exp} is, similarly, the motion of the mass-point i in the experimental motion.

The total error function is the arithmetic average of the specific error values of all the mass-points, and can be described as follows:

$$\varepsilon(\Delta x^{ref}, \Delta x^{exp}) = \frac{\sum_{i=1}^{n} \varepsilon_i(\Delta x^{ref}, \Delta x^{exp})}{n}$$

Note that the smaller the $\varepsilon(\Delta x^{ref}, \Delta x^{exp})$ values are, the more plausible the experimental motion is. However, accurate reference motion cannot be easily obtained. In this paper, the reference motion was assumed to be a result of explicit numerical integration without any approximation. The actual reference motion in this section was generated by explicit Euler method with an extremely small time step (1/30000 s). Although the reference motion and the experimental motion are generated with different simulation time steps, the checkpoints should be synchronized.

The function S represents the directional similarity between the reference motion and the experimental motion. The definition of $S(\Delta x^{ref}, \Delta x^{exp})$ can be expressed as follows:

$$S(\Delta x^{ref}, \Delta x^{exp}) = \frac{\sum_{i=1}^{n} (\Delta \hat{x}_i^{ref} \cdot \Delta \hat{x}_i^{exp})^2}{n}$$

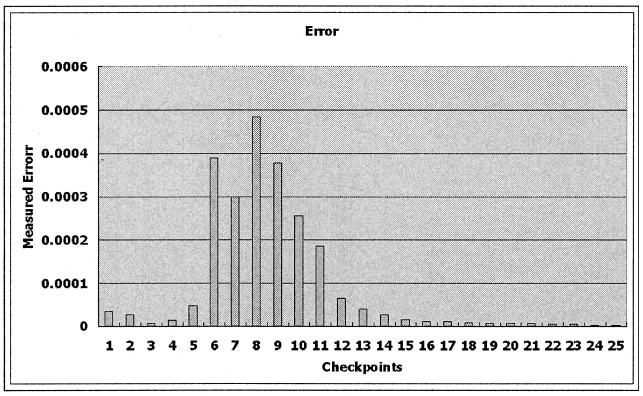
where $\Delta \hat{x}_i^{ref}$ denotes the normalized unit vector along the direction of the reference motion Δx_i^{ref}, and $\Delta \hat{x}_i^{exp}$ is similarly the unit vector along the experimental motion. The range of the function S is between 0 and 1, and the value 1 means that the direction of experimental motion is the same as the reference motion.

Table 2 shows the error ε and similarity S measured at the first 14 checkpoints. Figure 7 shows the same result measured at 25 checkpoints. In order to measure the values, a 1 m × 1 m square cloth model with 512 polygons was used. The mass of the cloth model is 0.05 kg, and the stiffness constant of each spring e_{ij} is 10. The reference motion was generated with a time step size of

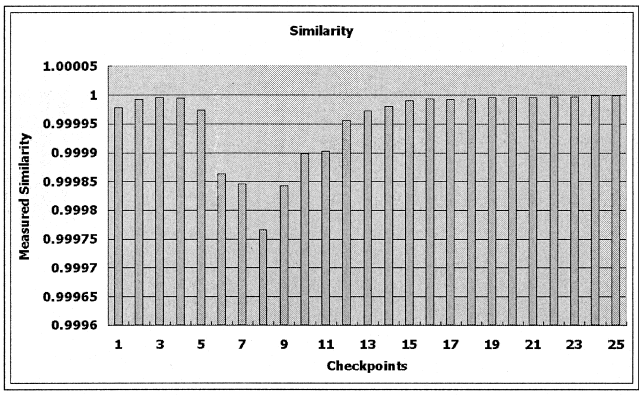
Table 2. *Measured Error and Similarity*

Checkpoint	1	2	3	4	5	6	7
ε	0.000034	0.000028	0.000007	0.000014	0.000047	0.00039	0.000299
S	0.999978	0.999993	0.999996	0.999995	0.999974	0.999863	0.999846
Checkpoint	8	9	10	11	12	13	14
ε	0.000485	0.000378	0.000256	0.000186	0.000065	0.000041	0.000028
S	0.999766	0.999843	0.999899	0.999903	0.999956	0.999973	0.999981

Reference motion: Explicit Euler (time step = 1/30000 s). Cloth model: 512 polygons, 1 m × 1 m square, total mass = 0.1 kg, and $k_{ij} = 10$.



(a) Measured Error



(b) Measured Similarity

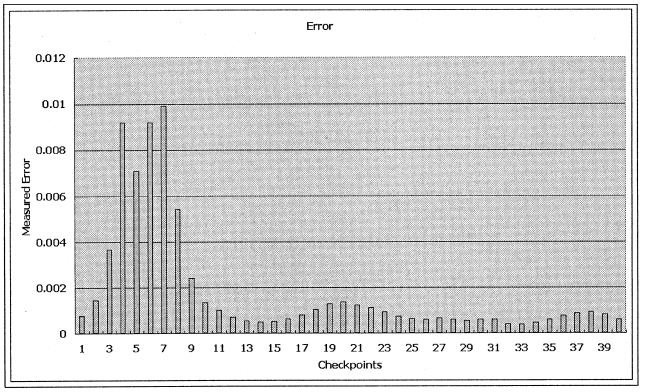
Figure 7. *Measured error and similarity. Reference motion: explicit Euler (time step = 1/30000 s). Cloth model: 512 polygons, 1 m × 1 m square, total mass = 0.1 kg, and $\kappa_{ij} = 10$: (a) Measured error (b) Measured similarity.*

1/30000 s and the experimental motion was generated with 1/150 s. The time interval between the checkpoints was fixed to 1/30 s. As shown in the figure, the motion produced by the proposed method has small ε and large S values even with the large time steps of 1/150 s. Figure 8 shows the result when a more complex model was tested. The result was obtained in the same manner as those in Figure 7, except that 2048 polygons were used. The experimental results show our method can be successfully used for generating plausible motion in real-time environments, such as PC games and VR systems.

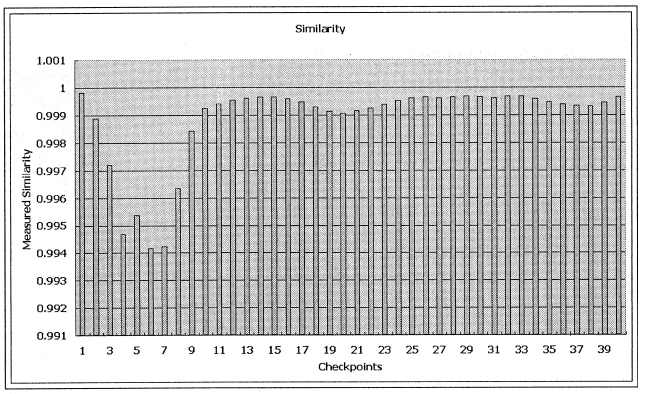
7 Conclusion

In this paper, we proposed an efficient method to animate virtual cloth based on mass-spring models. Cloth animation has been one of the major problems in computer graphics for more than a decade, and a large number of researchers and research groups have endeavored to find an efficient solution to the problem.

One of the major difficulties in cloth animation lies in the stability of the system. The instability of a numerical integration method affects the overall performance of the cloth-animation system because the unstable method requires extremely small time-steps. Therefore, real-time animation of a complex cloth model is not feasible with unstable methods. Stability is thus the first



(a) Measured error



(b) Measured similarity

Figure 8. *Measured error and similarity. Reference motion: explicit Euler (time step = 1/30000 s). Cloth model: 2048 polygons, 1 m × 1 m square, total mass = 0.1 kg, and* κ_{ij} *= 10: (a) Measured error (b) Measured similarity.*

essential property of animation techniques in real-time environments, such as VR systems.

The method proposed in this paper provides sufficient stability because the approximate solution of the method is based on implicit integration. The stability of the method has been obtained by employing an implicit integration scheme. Therefore, the method can produce cloth animation with large time-steps. In order to alleviate the computational burden of solving a linear system in an implicit method, the proposed method efficiently approximates the solution by taking advantage of various properties of the problem. The proposed method independently computes the next state of each mass-point by considering linked mass-points only. The system can also become unstable because of the force derivative used in the implicit integration scheme. Although previous methods used excessive approximation or additional manipulations, the technique in this paper avoids instability with minimal approximation.

Apart from real-time performance, the proposed method has additional important advantages. It is intuitive and easy to implement, and does not require developers to struggle with optimization of storage and the performance of the linear system solver.

The major benefits of the proposed method can be listed as follows:

- It produces stable cloth animation in optimized computation time
- It avoids instability caused by force derivative with minimal approximation
- It can be easily implemented without any special data structures or optimization techniques

References

Baraff, D., & Witkin, A. (1998). Large steps in cloth simulation. *SIGGRAPH '98 Conference Proceedings,* 43–54.

Brancazio, P. J. (1988). Physics and sports: The aerodynamics of projectiles. In D. Halliday, & R. Resnick (Eds.), *Fundamentals of physics,* E6:1–E6:8. New York: Wiley.

Breen, D. E., House, D. H., & Wozny, M. J. (1994). Predicting the drape of woven cloth using interacting particles. *SIGGRAPH'94 Conference Proceedings,* 365–372.

Carignan, M., Yang, Y., Thalmann, N. M., & Thalmann, D. (1992). Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics, SIGGRAPH 92, Conference Proceedings 26*(2), 99–104.

Choi, K.-J., & Ko, H.-S. (2002). Stable but responsive cloth. *ACM Transactions on Graphics: SIGGRAPH 2002 Conference Proceedings,* 604–611.

Cordier, F., & Magnenat-Thalmann, N. (2002). Real-time animation of dressed virtual humans. *Computer Graphics Forum, Proceedings of Eurographics 2002, 21*(3), 327–336.

Desbrun, M., Schröder, P., & Barr, A. (1999). Interactive animation of structured deformable objects. *Graphics Interface '99,* 1–8.

Eberhardt, B., Weber, A., & Strasser, W. (1996). A fast, flexible particle-system model for cloth draping. *IEEE Computer Graphics & Applications, 16*(5), 52–59.

Feynman, C. (1986). *Modeling the appearance of cloth.* Unpublished master's thesis, Massachusetts Institute of Technology, Cambridge.

Hadap, S., Bangarter, E., Volino, P., & Magnenat-Thalmann, N. (1999). Animating wrinkles on clothes. *IEEE Visualization '99,* 175–182.

Kang, Y.-M., Choi, J.-H., Cho, H.-G., & Park, C.-J. (2001). An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer, 17*(3), 147–157.

Kass, M. (1995). An introduction to continuum dynamics for computer graphics. In *SIGGRAPH course note: Physically-based modeling.* ACM SIGGRAPH.

Meyer, M., Debunne, G., Desbrun, M., & Bar, A. H. (2001). Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation, 12,* 1–12.

Ng, H. N., & Grimsdale, R. L. (1996). Computer graphics techniques for modeling cloth. *IEEE Computer Graphics & Applications, 16*(5), 28–41.

Oshita, M., & Makinouchi, A. (2001). Real-time cloth simulation with sparse particles and curved faces. *Proceedings of Computer Animation 2001,* 220–227.

Provot, X. (1997). Collision and self-collision handling in cloth model dedicated to design. *Computer Animation and Simulation '97,* 177–190.

Terzopoulos, D., Platt, J., Barr, A., & Fleischer, K. (1987). Elastically deformable models. *Computer Graphics, SIGGRAPH '87 Conference Proceedings, 21*(4), 205–214.

Volino, P., & Magnenat-Thalmann, N. (1994). Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum, 13*(3), 155–166.

Volino, P., & Magnenat-Thalmann, N. (2000). Implementing fast cloth simulation with collision response. *Proceedings of the Conference on Computer Graphics International 2000,* 257–268.

Witkin, A., & Baraff, D. (1994). Differential equation basics. In *SIGGRAPH course note: Physically-based modeling.* ACM SIGGRAPH.