The Visual Computer Journal, 2001. Springer-Verlag. (to appear)

The Visual Computer

An efficient animation of wrinkled cloth with approximate implicit integration

Young-Min Kang[1],
Jeong-Hyeon Choi[1],
Hwan-Gue Cho[1],
Do-Hoon Lee[2]

[1] Department of Computer Science, Pusan National University, Kumjeong-gu, Pusan, 609–735, South Korea
e-mail: {ymkang, jhchoi, hgcho}@pearl.cs.pusan.ac.kr
[2] Department of Computer Engineering, Miryang National University, Korea
e-mail: dhlee@arang.miryang.ac.kr

This paper presents an efficient method for creating the animation of flexible objects. The mass-spring model was used to represent flexible objects. The easiest approach to creating animation with the mass-spring model is the explicit Euler method, but the method has a serious weakness in that it suffers from an instability problem. The implicit integration method is a possible solution, but a critical flaw of the implicit method is that it involves a large linear system. This paper presents an approximate implicit method for the mass-spring model. The proposed technique updates with stability the state of n mass points in $O(n)$ time when the number of total springs is $O(n)$. In order to increase the efficiency of simulation or reduce the numerical errors of the proposed approximate implicit method, the number of mass points must be as small as possible. However, coarse discretization with a small number of mass points generates an unrealistic appearance for a cloth model. By introducing a wrinkled cubic spline curve, we propose a new technique that generates realistic details of the cloth model, even though a small number of mass points are used for simulation.

Key words: Cloth animation – Mass spring model – Implicit method – Realistic detail – Wrinkled curve

Correspondence to: Y.-M. Kang

1 Introduction

For physically based modeling, the mass-spring model is a simple and powerful approach for representing flexible objects such as cloth. There are many techniques for simulating flexible objects (Carignan et al. 1992; Provot 1995; Terzopoulos et al. 1987; Volino et al. 1995), and they use various models such as the finite element model (Celniker and Gossard 1991), and the deformable surface model (Wang et al. 1998). There are trade-offs between realism and efficiency. Some researchers have studied the realistic representation of flexible objects (Breen et al. 1994; Eberhardt et al. 1996), and others have tried to efficiently generate the motion of flexible objects (Baraff and Witkin 1998; Desbrun et al. 1999). Among these models, the mass-spring model is the easiest and the most intuitive.

Various techniques have been introduced to use the mass-spring model to simulate or animate flexible objects (Chen et al. 1998; Desbrun et al. 1999; Provot 1995). An animation technique based on the mass-spring model can be formulated as a simple ordinary differential equation. It is easy to calculate the force on each mass point, and we can animate the mass point by numerically integrating the force. The explicit Euler integration is the simplest approach to the numerical integration procedure, but it requires very small time steps for simulation or animation because this approach suffers severely from the instability problem (Kass 1995).

The implicit method is a well-known solution to the instability problem in numerical analysis (Nakamura 1991). Recently, Baraff and Witkin (1998) have exploit the implicit integration method to take large time steps during cloth simulation. Because this method can significantly reduce the simulation time, the implicit method is regarded as the best choice for the interactive animation of mass-spring-based objects. However, the implicit method also has a problem in that a large linear system must be solved. Although the implicit method elegantly insures that the spring forces will be stably integrated, the large linear system involved in the method is a major obstruction to real-time animation. Desbrun et al. (1999) propose an efficient method of cloth animation with a precomputed inverse matrix (i.e., a precomputed filter). However, the method also suffers from intensive computation because the inverse matrix may not be sparse.

This paper presents a fast and stable animation technique that updates with stability the state of n mass points in $O(n)$ time for the real-time animation of

The Visual Computer (2001) 17:???–???

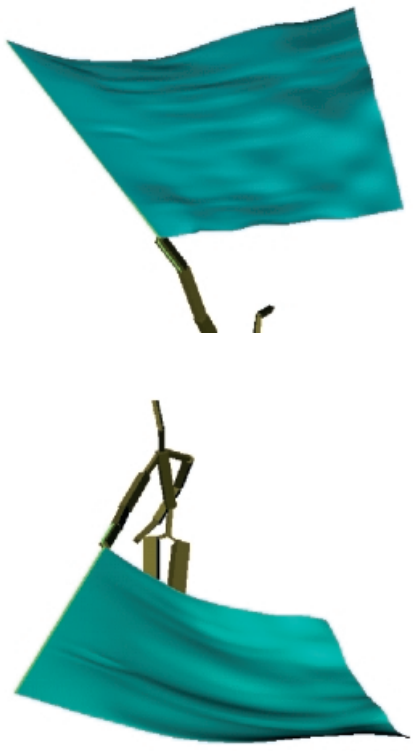


Fig. 1. Snapshots of cloth animation

flexible objects as shown in Fig. 1. The adaptive time step strategy and the dynamic modification of animation parameters, such as mass or stiffness, are not restricted in our model. In addition, we considered the wrinkles for the sake of a realistic appearance, and also implemented interaction between flexible objects and air in order to generate plausible motion of the cloth.

2 The mass-spring model and stable integration

The mass-spring model is a simple technique for representing flexible objects. The model represents an object as a set of mass points that are linked to each other by springs. The forces caused by the springs induce the movement of mass points, and the springs can be arbitrarily constructed. This is a very intuitive approach to the representation of cloth.

Mass points are easily animated with the explicit Euler integration. However, due to the instability problem, the simple explicit Euler integration cannot be used unless the time step h is very small. Thus, the explicit method requires too much time to animate the mass-spring model (Desbrun et al. 1999; Kass 1995; Nakamura 1991).

2.1 The implicit method for stable animation

The implicit method is a solution to the instability problem. By using the implicit Euler method, we can stably update the state of each mass point as follows:

$$\begin{aligned} \boldsymbol{v}_i^{t+h} &= \boldsymbol{v}_i^t + \boldsymbol{F}_i^{t+h}\frac{h}{m_i} \\ \boldsymbol{x}_i^{t+h} &= \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+h}h, \end{aligned} \tag{1}$$

where $\boldsymbol{v}_i^t$ denotes the velocity of the ith mass point at time t, and $\boldsymbol{F}_i^{t+h}$ is the force acting on the mass point at time $t+h$ (i.e., the force at the next time step). Similarly, $\boldsymbol{x}_i^t$ denotes the location of the ith mass point at time t, m_i is the mass of the ith mass point, and h denotes the interval between the animation steps. Because this integration method stably updates the state of the mass-spring model, a large time step can be taken for animation. Therefore, the implicit method is the best choice for real-time or interactive animation systems.

However, the implicit Euler method involves $\boldsymbol{F}_i^{t+h}$, which can be approximated with a first-order derivative as follows:

$$\boldsymbol{F}^{t+h} = \boldsymbol{F}^t + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}\Delta \boldsymbol{x}^{t+h}, \tag{2}$$

where $\boldsymbol{F}^t$ denotes the internal forces consisting of all the internal forces $\boldsymbol{F}_i^t$ on the ith mass point (i.e., $\boldsymbol{F}^t = [\boldsymbol{F}_1^t, \boldsymbol{F}_2^t, \cdots, \boldsymbol{F}_n^t]^{\mathrm{T}}$), and similarly, $\Delta \boldsymbol{x}^t = [\Delta \boldsymbol{x}_1^t, \Delta \boldsymbol{x}_2^t, \cdots, \Delta \boldsymbol{x}_n^t]^{\mathrm{T}}$. Because $\partial \boldsymbol{F}/\partial \boldsymbol{x}$ is a negated Hessian matrix of the system (Desbrun et al. 1999), we will henceforth denote $\partial \boldsymbol{F}/\partial \boldsymbol{x}$ as $\boldsymbol{H}$.

Since $\Delta \boldsymbol{x}^{t+h} = \boldsymbol{x}^{t+h} - \boldsymbol{x}^t = (\boldsymbol{v}^t + \Delta \boldsymbol{v}^{t+h})h$, the first equation of implicit update in (1) can be rewritten as:

$$\left(\boldsymbol{I} - \frac{h^2}{m}\boldsymbol{H}\right)\Delta \boldsymbol{v}^{t+h} = (\boldsymbol{F}^t + h\boldsymbol{H}\boldsymbol{v}^t)\frac{h}{m} \tag{3}$$

where $\Delta \boldsymbol{v}^{t+h}$ is $\boldsymbol{v}^{t+h} - \boldsymbol{v}^t$. If $\Delta \boldsymbol{v}^{t+h}$ can be calculated, the velocity and location of each mass point at the next step can easily be updated with (1). The

animation of flexible objects is finally reduced to finding the value of $\Delta \boldsymbol{v}^{t+h}$. In (3), $h\boldsymbol{H}\boldsymbol{v}^t$ represents additional forces. As mentioned by Desbrun et al. (1999), these additional forces are viscosity forces and can easily be calculated as

$$(h\boldsymbol{H}\boldsymbol{v}^t)_i = h \sum_{(i,j)\in E} k_{ij}(\boldsymbol{v}_j^t - \boldsymbol{v}_i^t), \tag{4}$$

where E is the set of spring edges between mass points.

Despite the stability of the implicit method, the method has a critical weakness in that (3) involves a $\boldsymbol{I} - (h^2/m)\boldsymbol{H}$, which is an $O(n \times n)$-sized matrix. Because of this matrix, a large linear system must be solved in order to update the state of the model. A modified conjugate-gradient method has been used to alleviate the computation, but it is still far from the interactive animation of a mass-spring model (Baraff and Witkin 1998).

In order to alleviate the computational burden of the implicit method, Desbrun et al. (1998) propose an efficient method that approximates the Hessian matrix $\boldsymbol{H}$. In the method, $\boldsymbol{H_{ij}}$, the entry of the Hessian matrix at the ith row and the jth column, was approximated as $\boldsymbol{H_{ij}} = k_{ij}$, and $\boldsymbol{H}_{ii} = -\sum_{j\neq i} k_{ij}$, where k_{ij} denotes the stiffness constant of the spring between the ith and the jth mass points. k_{ij} is 0 when the ith and the jth mass springs are not linked. Then, the matrix $(\boldsymbol{I} - (h^2/m)\boldsymbol{H})^{-1}$ remains constant during animation. The inverse matrix of $\boldsymbol{I} - (h^2/m)\boldsymbol{H}$ was precomputed and used as a force filter for the cloth animation. This technique uses simple calculations and produces stable results. Their method can be expressed as

$$\Delta \boldsymbol{v}^{t+h} = \left(\boldsymbol{I} - \frac{h^2}{m}\boldsymbol{H}\right)^{-1} \frac{\tilde{\boldsymbol{F}}^t h}{m}, \tag{5}$$

where $\tilde{\boldsymbol{F}}$ is the sum of the spring forces and the viscosity forces (i.e., $\tilde{\boldsymbol{F}}^t = \boldsymbol{F}^t + h\boldsymbol{H}\boldsymbol{v}^t$).

The precomputed filter method is faster than the general implicit method. However, the inverse matrix of $\boldsymbol{I} - (h^2/m)\boldsymbol{H}$ is not necessarily a sparse matrix, even though $\boldsymbol{I} - (h^2/m)\boldsymbol{H}$ is sparse. Moreover, the adaptive time-step strategy cannot be applied to the precomputed filter method, and the mass or stiffness cannot be dynamically changed because the calculation of the inverse matrix requires much time. This is why we did not use the precomputed filter.

2.2 An approximation method for interactive animation

We can update the velocity change of the ith mass point by considering only the linked mass points because $\boldsymbol{H}_{ij}$ is 0 when the ith and the jth mass points are not linked with a spring. Therefore, the implicit update scheme (3) can be rewritten as

$$\left(1 - \frac{h^2\boldsymbol{H}_{ii}}{m_i}\right)\Delta \boldsymbol{v}_i - \frac{h^2}{m_i}\sum_{(i,j)\in E}(\boldsymbol{H}_{ij}\Delta \boldsymbol{v}_j) = \frac{\tilde{\boldsymbol{F}}_i^t h}{m_i}. \tag{6}$$

We adopted the approximate Hessian described by Desbrun et al. for the simplicity. If the uniform spring constant k is assumed for all the spring links, and n_i denotes the number of neighboring mass points that are linked to the ith mass point, the Hessian matrix can be rewritten as $\boldsymbol{H}_{ij} = k$ and $\boldsymbol{H}_{ii} = -kn_i$. The update formula can then be rewritten as

$$\frac{m_i + h^2 k n_i}{m_i}\Delta \boldsymbol{v}_i^{t+h} = \frac{\tilde{\boldsymbol{F}}_i^t h}{m_i} + \frac{h^2 k \sum_{(i,j)\in E}\Delta \boldsymbol{v}_j^{t+h}}{m_i}.$$

Therefore, $\Delta \boldsymbol{v}_i^{t+h}$ can be expressed as follows:

$$\Delta \boldsymbol{v}_i^{t+h} = \frac{\tilde{\boldsymbol{F}}_i^t h + kh^2 \sum_{(i,j)\in E}\Delta \boldsymbol{v}_j^{t+h}}{m_i + kh^2 n_i}. \tag{7}$$

However, $\Delta \boldsymbol{v}_i^{t+h}$ cannot be calculated directly by (7) because the equation contains an unknown $\Delta \boldsymbol{v}_j^{t+h}$, the velocity changes of the jth mass points that are linked to the ith mass point in the next state. In order to calculate $\Delta \boldsymbol{v}_i^{t+h}$, the velocity change of the ith mass point at the next step, the velocity change of the jth mass point at the next step is approximated.

$\Delta \boldsymbol{v}_j^{t+h}$ can be expressed as

$$\Delta \boldsymbol{v}_j^{t+h} = \frac{\tilde{\boldsymbol{F}}_j^t h + h^2 \sum_{(j,l)\in E} k_{jl}\Delta \boldsymbol{v}_l^{t+h}}{m_j + h^2 \sum_{(j,l)\in E} k_{jl}}. \tag{8}$$

When the term, $h^2 \sum_{(j,l)\in E} k_{jl}\Delta \boldsymbol{v}_l^{t+h}$, is dropped, an approximation of $\Delta \boldsymbol{v}_j^{t+h}$ remains:

$$\Delta \boldsymbol{v}_j^{t+h} \simeq \frac{\tilde{\boldsymbol{F}}_j^t h}{m_j + h^2 \sum_{(j,l)\in E} k_{jl}}. \tag{9}$$

By using this approximation and assuming the uniform stiffness k, we can rewrite the update formula for $\Delta \boldsymbol{v}_i^{t+h}$ as

$$\Delta \boldsymbol{v}_i^{t+h} = \frac{\tilde{\boldsymbol{F}}_i^t h + h^2 k \sum_{(i,j)\in E} \tilde{\boldsymbol{F}}_j^t h/(m_j + h^2 k n_j)}{m_i + h^2 k n_i}, \tag{10}$$

where n_i is the number of mass points that are linked to the ith mass point by springs, and n_j is the number of mass points linked to the jth mass point.

Since $\Delta \tilde{\boldsymbol{F}}_j^t$ is already known, the velocity change of the ith mass point at the next step can be calculated directly. This means that we can generate the approximate motion of flexible objects without solving the linear system that was a major obstruction to interactive animation. Since we update the velocity changes of a mass point by considering only a small number of linked mass points, it is obvious that our model works in $O(n)$ time and is faster than the precomputed inverse matrix method or any general implicit integration. Moreover, the mass, time step, or stiffness can easily be changed during animation without any additional computations. These dynamic changes of parameters cannot be achieved when the precomputed inverse matrix is used. It is easy to modify (10) for general cases where the stiffness values of springs differ from each other. We have tested the approximate motion of flexible objects and found that our model generates stable animation results.

2.3 Stability of the proposed method

To verify the stability of our method, let us consider a simple example where only two mass points (i and j) of the same mass m are linked with a spring. This is not a proof, but it illustrates why our method is stable. Suppose that the rest length of the spring is 0, and the stiffness of the spring is k. For the sake of simplicity, let us assume that the current velocity of each mass point is $[0, 0, 0]^{\mathrm{T}}$. Then, no viscosity forces are exerted on the mass points, and the force acting on the mass points i and j can be calculated as

$$\begin{aligned} \tilde{\boldsymbol{F}}_i^t &= -k(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t) \\ \tilde{\boldsymbol{F}}_j^t &= -k(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) = -\tilde{\boldsymbol{F}}_i^t. \end{aligned} \tag{11}$$

It is clear that the location of the mass points does not diverge when $|\boldsymbol{x}_i^{t+h} - \boldsymbol{x}_j^{t+h}| \le |\boldsymbol{x}_i^t - \boldsymbol{x}_j^t|$. Since $n_i = n_j = 1$, $\tilde{\boldsymbol{F}}_j = -\tilde{\boldsymbol{F}}_i$, and $m_i = m_j = m$, the location of each mass point at the next step, ($\boldsymbol{x}_i^{t+h}$ and $\boldsymbol{x}_j^{t+h}$), can be calculated as

$$\begin{aligned} \boldsymbol{x}_i^{t+h} &= \boldsymbol{x}_i^t + \frac{\tilde{\boldsymbol{F}}_i^t h - k h^2 \tilde{\boldsymbol{F}}_i^t h/(m + kh^2)}{m + kh^2} h \\ \boldsymbol{x}_j^{t+h} &= \boldsymbol{x}_j^t - \frac{\tilde{\boldsymbol{F}}_i^t h - k h^2 \tilde{\boldsymbol{F}}_i^t h/(m + kh^2)}{m + kh^2} h. \end{aligned} \tag{12}$$

Let $\boldsymbol{u}$ denote $\boldsymbol{x}_i^t - \boldsymbol{x}_j^t$. Then, $\tilde{\boldsymbol{F}}_i^t$ can be expressed as $-k\boldsymbol{u}$, and the difference of the locations of the mass points at the next step can be expressed as

$$\boldsymbol{x}_i^{t+h} - \boldsymbol{x}_j^{t+h} = \boldsymbol{u}\left(1 - 2 \cdot \frac{mkh^2}{(m + kh^2)^2}\right). \tag{13}$$

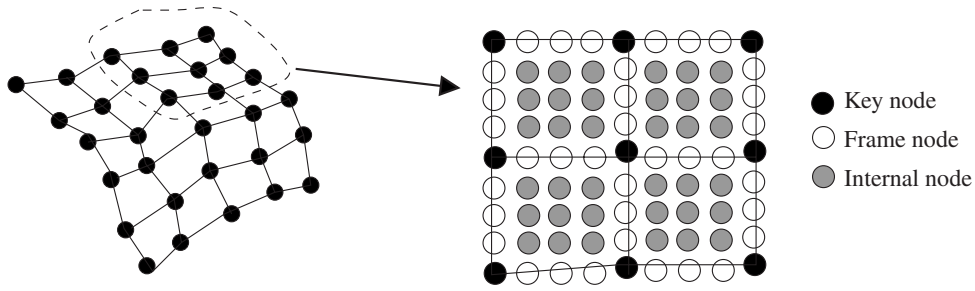
It is clear that $|\boldsymbol{x}_i^{t+h} - \boldsymbol{x}_j^{t+h}|$ is less than $|\boldsymbol{x}_i^t - \boldsymbol{x}_j^t|$ when $0 \le mkh^2/(m + kh^2)^2 \le 1$. It is trivial to show that $mkh^2/(m + kh^2)^2$ is larger than 0. Now, it is only necessary to show that $mkh^2/(m + kh^2)^2 \le 1$. Because $(m + kh^2)^2$ is $m^2 + k^2h^4 + 2mkh^2$, it can easily be found that $(m + kh^2)^2 - mkh^2$ is $m^2 + k^2h^4 + mkh^2$ and larger than 0. Therefore, $mkh^2/(m + kh^2)^2 \le 1$.

3 Realistic details

This section presents a new technique for generating realistic details of cloth. Because the approximate implicit method proposed in this paper determines the next position of a mass point by considering the positions of a limited number of linked neighbors, undesirable results can be generated when a large number of mass points are used to represent a cloth model. In order to overcome the limitations of our animation technique, we introduce a wrinkle generation method based on a cubic spline curve.

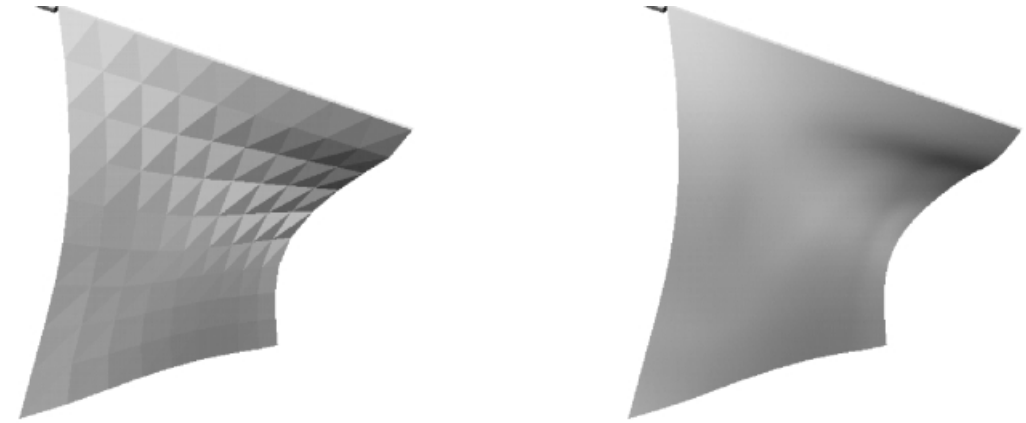
3.1 Generating a smooth surface with cubic spline curves

As previously mentioned, our animation technique suffers from numerical errors when cloth is discretized into too many mass points. Moreover, other animation methods such as implicit integration or precomputed filtering cannot efficiently generate the motion of cloth when it is composed of too many



Mass-Spring Structure (key nodes) Frame nodes and internal nodes for wrinkle representation

2



3a 3b

Fig. 2. Three types of nodes used for representing our cloth model. The black spots are the key nodes, the white spots are the frame nodes, and the gray spots are the internal nodes

Fig. 3a,b. Generating a smooth surface by a cubic spline: **a** a rendered cloth with the key nodes used during the simulation; **b** a rendered cloth with the frame nodes and the internal nodes calculated with spline curves

mass points. We believe that the best way to animate cloth efficiently, regardless of the simulation techniques, is a hybrid method that generates the motion of a small number of important mass points through a simulation technique and calculates the locations of many other mass points by an interpolation method such as that of the cubic spline curve.

We define three types of nodes that are used in our cloth model: key nodes, frame nodes, and internal nodes. The key nodes are those the locations of which are determined by an animation technique, for example, the approximate implicit method. The frame nodes are those that are located between two adjacent key nodes. The locations of the frame nodes are not simulated during the simulation phase, but simply calculated with the cubic spline curves. The control points of the curves are the key nodes. The third type of nodes are internal nodes, and their locations are also calculated with cubic spline curves; the frame nodes are used as the control points. Figure 2 shows how the various types of nodes are linked to each other.

By using the cubic spline curves, we can generate a smooth surface, even though only a small number of mass points are used in the simulation phase. This enables the motion of cloth to be generated efficiently, and reduces the numerical errors in our ap-

proximate method during the simulation. Figure 3 shows the effects of generating the frame and the internal nodes with cubic spline curves. We used only 100 mass points in the simulation phase (Fig. 3a) and generated 2500 nodes to represent the cloth (Fig. 3b). However, the cubic spline generates an excessively smooth surface so that the result is far from a realistic representation of the cloth. The following subsection presents a technique that generates a plausible cloth representation when only a small number of nodes are given.

3.2 Generating realistic details through wrinkled cubic spline curves

A cubic spline curve can be used to generate a smooth surface of cloth by calculating the frame nodes and the internal nodes, but the results do not closely resemble real cloth. The undesirable result is due to the incapability of the cubic spline curve to express detailed wrinkles. In order to represent realistic cloth wrinkles, we modified our method for generating the frame and the internal nodes by adding wrinkles to the cubic spline. The results of the wrinkled cubic spline curves are much better than those of the simple cubic spline curves. Figure 4 shows a result of the wrinkled cubic spline curve.

Figure 5a shows the cubic spline curve, and Fig. 5b, the wrinkled cubic spline curve. The wrinkled cubic spline curve in Fig. 5b generally follows the curve shown in Fig. 5a, and wrinkles are generated in segments where two adjacent control points are closer than a given threshold. The closer the points are, the more wrinkles are generated.

Wrinkles are generated according to the length of each spline segment. If the distance between two control points that generate one spline segment is shorter than the rest length of a spring, the segment must have some wrinkles in order to preserve the length. Since the focus of this study is not on the exact simulation of cloth, but on the efficient generation of a plausible motion of cloth, the strict preservation of length is not important. Therefore, we implemented a simple wrinkle model that can generate plausible wrinkles. The wrinkle model creates wrinkles when the distance between two adjacent control points is shorter than the rest length of the virtual spring that links the points, assuming that the frequency of wrinkles is proportional to the difference between the rest length and the actual distance.

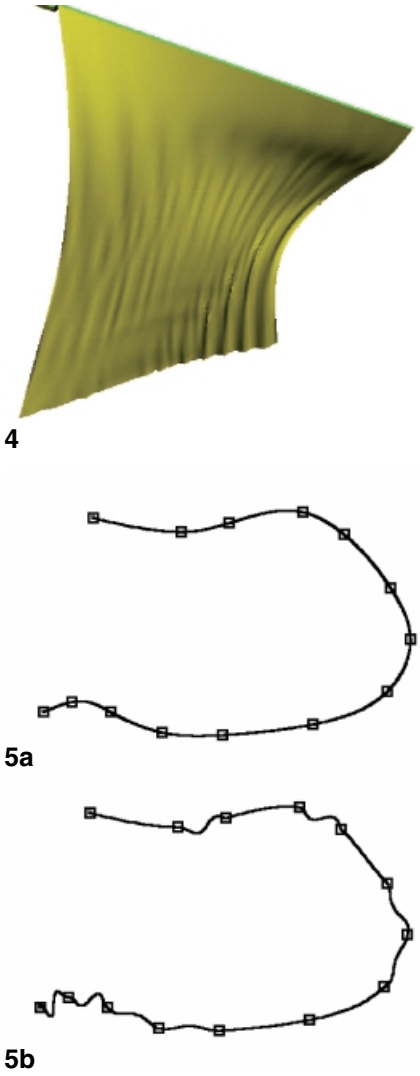


Fig. 4. The rendering result of cloth with wrinkled cubic spline curves. The number of nodes used here is the same as that in Fig. 3

Fig. 5a,b. A wrinkled cubic spline curve: **a** cubic spline; **b** wrinkled cubic spline

Let $Q(t)$ be a cubic spline curve segment between two control points, P_1 and P_2, with $0 \leq t \leq 1$, and let $Q'(t)$ be the wrinkled spline curve segment between the control points. $Q'(t)$ is exactly the same curve as $Q(t)$ when the distance between the control points is longer than the rest length. If the distance is shorter

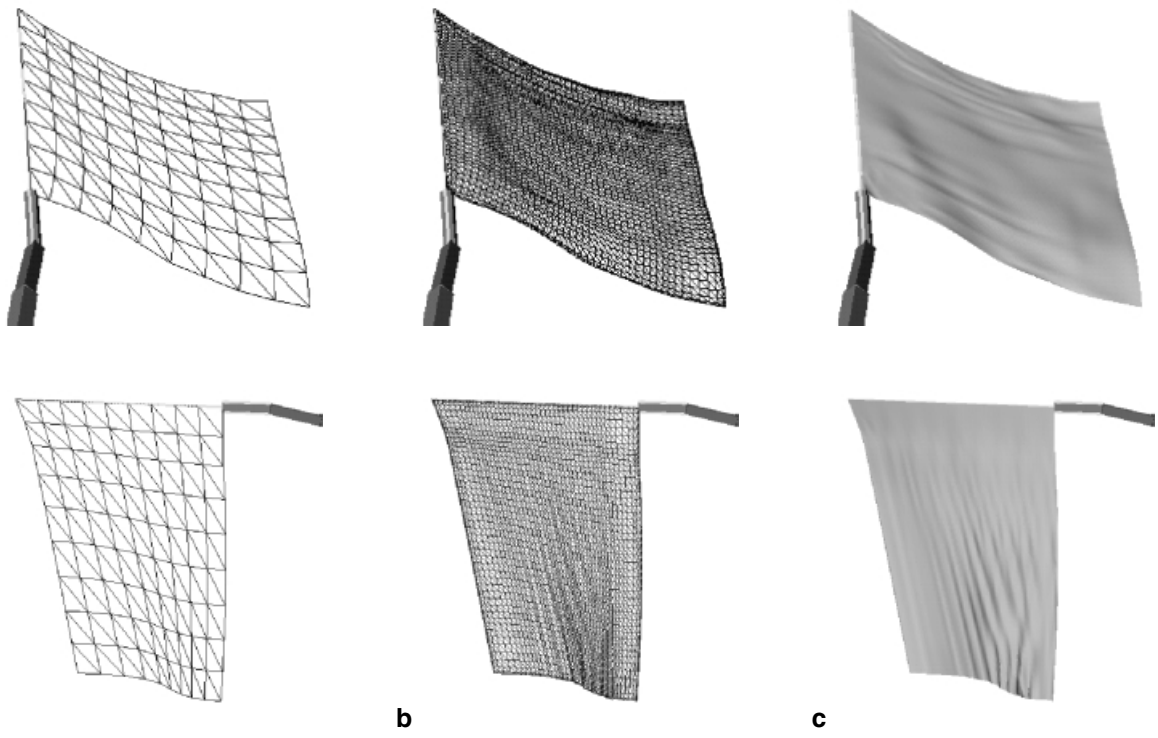


Fig. 6a–c. Effects of wrinkles: **a** mass-spring structure; **b** wrinkle generation; **c** rendering result

than the length, $Q'(t)$ can be expressed as

$$\begin{aligned} Q'(t) &= Q(t) + w(t)A\ \sin(ft)\boldsymbol{N}(\boldsymbol{t}) \\ w(t) &= 1/2 - 2(t-1/2)^2 \\ A &= (l_0 - |P_1 - P_2|)/2 \\ f &= c(l_0 - |P_1 - P_2|)/l_0, \end{aligned} \tag{14}$$

where $\boldsymbol{N}(\boldsymbol{t})$ is the normal vector of the curve at t, l_0 is the rest length of the spring that links the points, and c is the control parameter for the frequency of wrinkles. When this method is applied to the cloth model, $\boldsymbol{N}(\boldsymbol{t})$ can be the normal vector of the cloth surface.
Figure 6 shows the effects of our wrinkle model. Figure 6a shows the wireframe images of the cloth when only 100 (10×10) nodes are used for simulation; Fig. 6b shows the results with 2500 nodes, the locations of which are calculated only with wrinkled cubic spline curves; and Fig. 6c shows the final rendering results based on the 2500 nodes.

4 An air-interaction model for plausible animation

In order to generate realistic animation of thin flexible objects, two kinds of forces, drag force and lift force, must be considered. The magnitude of drag force is known to be

$$|F_D| = \frac{1}{2} C_D \varrho |V|^2 S \sin\theta, \tag{15}$$

where $|F_D|$ denotes the magnitude of the drag force, C_D is the drag force coefficient, ϱ is the density of a fluid, V is the velocity of an object relative to the fluid, S is the area of the object surface, and θ is the angle between V and the surface. The direction of the drag force is opposite to that of the velocity.
Similarly, the magnitude of the lift force can be expressed as

$$|F_L| = \frac{1}{2} C_L \varrho |V|^2 S \cos\theta, \tag{16}$$

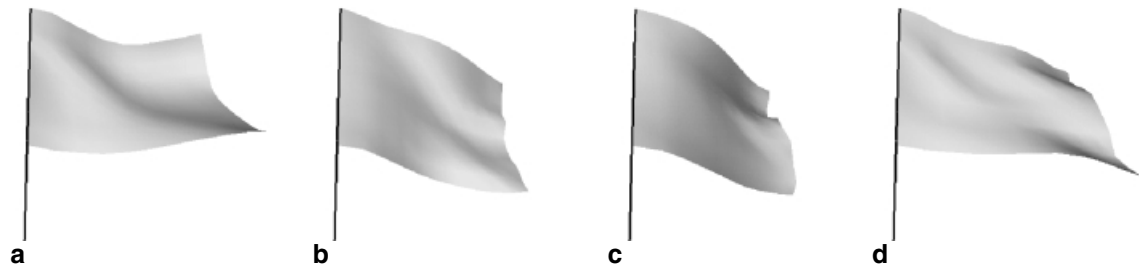


Fig. 7a–d. Flag in wind: a 50 cm × 50 cm flag is tied to a pole, and the wind is blowing from left to right. The velocity of the wind is 40 m/s, K_D is 0.01, and K_L is 0.01. Our air-interaction model generated very plausible animation of a thin object: **a** $t = 0$; **b** $t = 1$; **c** $t = 2$; **d** $t = 3$

where $|F_L|$ denotes the magnitude of the lift force, and C_L is the lift force coefficient. The direction of the lift force is perpendicular to the direction of the velocity.

When $\hat{\boldsymbol{N}}_i$ denotes the unit normal of the ith mass point, and $\hat{\boldsymbol{v}}_i$ denotes $\boldsymbol{v}_i/|\boldsymbol{v}_i|$, the angle between $\hat{\boldsymbol{N}}_i$ and $\hat{\boldsymbol{v}}_i$ is $\pi/2-\theta$. Thus, $|\hat{\boldsymbol{N}}\cdot\hat{\boldsymbol{v}}_i|$ is $\sin\theta$ $(=\cos(\pi/2-\theta))$. Therefore, the drag force is proportional to $|\hat{\boldsymbol{N}}\cdot\hat{\boldsymbol{v}}_i|$. Because the direction of the drag force is opposite to the direction of the velocity, the drag force was implemented as

$$\boldsymbol{F}_{Di} = -K_D|\hat{\boldsymbol{N}}_i\cdot\hat{\boldsymbol{v}}_i||\boldsymbol{v}_i|^2\hat{\boldsymbol{v}}_i, \tag{17}$$

where K_D is the control parameter for the drag force. To implement the lift force, its direction must first be determined. Because the direction of the lift force is perpendicular to the direction of the velocity $\boldsymbol{U}_i$, the direction of the lift force on the ith mass point, was determined as

$$\begin{aligned}\tilde{\boldsymbol{N}}_i &= \hat{\boldsymbol{N}}_i, \quad \textit{if } \hat{\boldsymbol{N}}_i\cdot\hat{\boldsymbol{v}}_i > 0\\ &\quad -\hat{\boldsymbol{N}}_i, \quad \textit{otherwise}\\ \boldsymbol{U}_i &= (\tilde{\boldsymbol{N}}_i\times\hat{\boldsymbol{v}}_i)\times\hat{\boldsymbol{v}}_i.\end{aligned} \tag{18}$$

Since the direction of the lift force has already been determined, only the magnitude of the lift force needs to be determined. The lift force $\boldsymbol{F}_{Li}$ on the ith mass point was implemented as

$$\boldsymbol{F}_{Li} = (K_L\cos\theta|\boldsymbol{v}_i|^2)\boldsymbol{U}_i, \tag{19}$$

where K_L is the control parameter for the lift force.

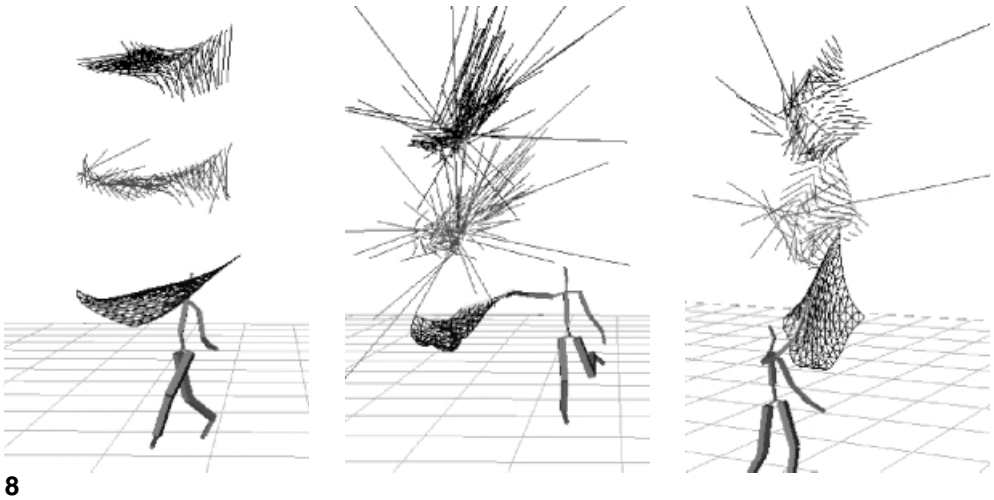
The effects of drag and lift forces caused by wind can easily be taken into account by calculating the relative velocity of each mass point with respect to air. Figure 7 shows the animation results when the drag force and the lift force are considered. The velocity of the wind is 40 m/s, K_D is 0.01, and K_L is 0.01. As seen in the figure, our model generated a very plausible scene.

5 Experimental results

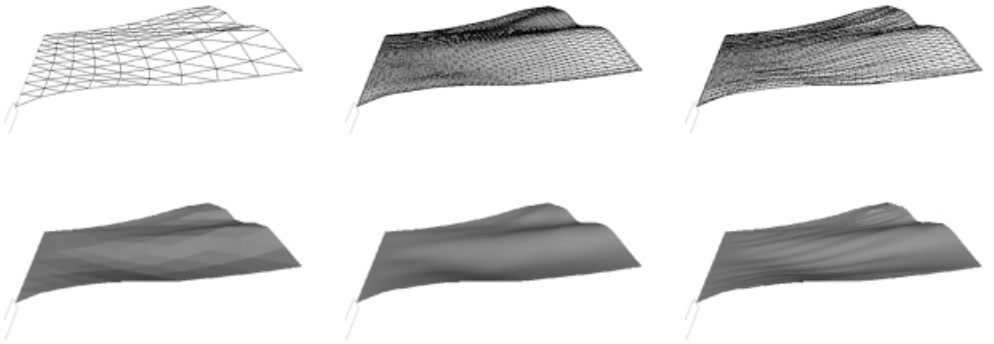
We implemented an animation system for thin flexible objects by using the *OpenGL* and the *Open Inventor* library on *SGI Indigo*^{2} and O_2 machines with $R10000$ processors. The system was implemented with the techniques proposed in this paper. However, the mass-spring model has limitations in representing flexible objects because it shows superelastic effects. The inverse dynamics process for adjusting the superelongated springs should be considered when the mass-spring model is used. We adopted the techniques proposed by Provot (1995).

Our technique generated real-time animation of flexible objects with hundreds of mass points at the frame rate of 30 Hz or 60 Hz. We were also able to generate interactive animation with more mass points (i.e., thousands of mass points).

We compared the results of precomputed filtering proposed by Desbrun et al. with those of our approximation. Figure 8 shows that the velocity changes calculated by our model is very similar to those calculated by the precomputed filter. In Fig. 8, a virtual human moves a flag and the velocity change of each mass point is drawn as a line segment. Two clusters of line segments represent the velocity changes calculated with the



8



9a 9b 9c

Fig. 8. A comparison between the precomputed filter method and the proposed approximation: a character moves a flag and the velocity change of each mass point is shown as a line segment. Two clusters of line segments represent velocity changes calculated with precomputed filtering (*top*) and those approximated by the proposed technique (*middle*)

Fig. 9a–c. The results of cubic spline curves and wrinkled cubic spline curves: **a** an original mass-spring structure; **b** the result generated by cubic spline curves; **c** the result generated by wrinkled cubic spline curves

precomputed filter (top) and those approximated by the proposed technique (middle). The velocity changes calculated by both methods are similar in magnitude and direction. Thus, our model generates the stable motion of the mass-spring model, similar to the case of the precomputed filter method.

Figure 9 shows the rendering results of different details. The results shown in column **a** are rendered with only 100 key nodes; those in column **b** are rendered with the key nodes, the frame nodes, and the internal nodes with cubic spline curves; and those in column **c** are rendered with wrinkled cubic spline curves.

Figure 10 shows the animation results when the proposed techniques are used. The motion of the character was captured by a motion capture system, and the motion of the flag shown in the figure was generated by applying our techniques. One hundred nodes were used to generate the important movements of the flags, and the locations of 2400 frame nodes and internal nodes were calculated with the wrinkled cu-

Fig. 10. Animation results: 100 key nodes are used as mass points for the proposed approximate implicit method, and the frame nodes and the internal nodes are generated with the wrinkled cubic spline curves. The total number of nodes for the scenes is 2500

bic spline for creating a plausible appearance of the flags.

6 Conclusion

We have proposed an approximation method of implicit integration for the stable and rapid animation of a mass-spring model, and we have also shown the results produced with an animation system implemented with the proposed techniques. Physical correctness was not a major concern of our work; we were only interested in rapid and plausible animation of flexible objects. The experimental results show that our technique produces very plausible animation of flexible objects.

Our technique is very stable because we exploit the filtering property of the implicit method. Moreover, our method is as fast as the explicit method in the calculation of the next state because ours does not involve linear system solving, which is a bottleneck in the implicit method. Another important advantage of our technique is that it is very intuitive and easy to implement because it calculates the next state with a direct update formula. Moreover, our technique allows adaptive time steps, and dynamic modifications of physical parameters such as mass and stiffness.

The biggest flaw of our method is that undesirable results can be generated when a large number of mass points are used for simulation because the proposed method computes the location of each mass point by considering only its linked neighbors. In order to solve this problem, we introduced a new technique that generates realistic details of the cloth model when a small number of mass points are given. The technique employs the wrinkled cubic spline curve, which generates a curve that is similar to the cubic spline curve, but involves wrinkles according to the distance between two adjacent control points. By using the wrinkled cubic spline curve, we could efficiently generate cloth motion with a small number of mass points and render realistic details of cloth with the wrinkled curve. Our technique can be applied to various animation systems that require the interactive animation of flexible objects.

Acknowledgements. This work was supported in part by the Ministry of Information and Communication of Korea: The Support Project for University Foundation Research 2000, supervised by CE^C(IITA).

References

1. Baraff D, Witkin A (1998) Large steps in cloth simulation. (SIGGRAPH '98) Comput Graph CE^d:43–52
2. Breen D, House D, Wozny M (1994) Predicting the drape of woven cloth using interacting particles. (SIGGRAPH '94) Comput Graph CE^d:365–372
3. Carignan M, Yang Y, Thalmann N, Thalmann D (1992) Dressing animated synthetic actors with complex deformable clothes. (SIGGRAPH '92) Comput Graph CE^d:99–104
4. Celniker G, Gossard D (1991) Deformable curve and surface finite-elements for free-form shape design. (SIGGRAPH '91) Comput Graph CE^d:257–266

5. Chen Y, Zhu Q, Kaufman A (1998) Physically based animation of volumetric objects. Proceedings of Computer Animation '98, CE[e], pp 154–160
6. Desbrun M, Schröder P, Barr A (1999) Interactive animation of structured deformable objects. Proceedings of Graphics Interface '99 CE[f]
7. Eberhardt B, Weber A, Strasser W (1996) A fast, flexible, particle-system model for cloth draping. IEEE Comput Graph Appl 16:52–59
8. Kass M (1995) An introduction to continuum dynamics for computer graphics. SIGGRAPH Course Notes vol CE[g]. ACM SIGGRAPH
9. Nakamura S (1991) Initial value problems of ordinary differential equations. In: Applied numerical methods with software. Prentice-Hall, pp 289–350 CE[h]
10. Provot X (1995) Deformation constraints in a mass-spring model to describe rigid cloth behavior. Graphics Interface CE[d]:147–154
11. Terzopoulos D, Platt J, Barr A (1987) Elastically deformable models. (SIGGRAPH '87) Comput Graph CE[d]: 205–214
12. Volino P, Courchesne M, Thalmann N (1995) Versatile and efficient techniques for simulating cloth and other deformable objects. (SIGGRAPH '95) Comput Graph CE[d]:137–144
13. Wang B, Wu Z, Sun Q, Yuen M (1998) A deformation model of thin flexible surfaces. Proceedings of WSCG '98, pp 440–446 CE[1]

YOUNG-MIN KANG is a PhD student in the Department of Computer Science at Pusan National University, Pusan, Korea. He received his MSc and BSc degrees in Computer Science from Pusan National University in 1999 and 1996, respectively. His research interests include computer animation and simulation. He is currently working on the simulation and animation of flexible objects and is also interested in the efficient control of nonautonomous objects.

JEONG-HYEON CHOI received his BSc degree in Physics from Pusan National University in 1995. He received his MSc degree in Computer Science from Pusan National University in 2000, and is currently a PhD student in the Department of Computer Science at Pusan National University. His current research interests include computer animation and bioinformatics.

HWAN-GUE CHO received his BSc degree in Computer Science and Statistics from Seoul National University, Seoul, South Korea, in 1984. He also received his MSc and PhD degrees in Computer Science from the Korea Advanced Institute of Science and Technology (KAIST) in 1986 and 1990, respectively. Since 1990, he has been a faculty member of the Department of Computer Science at Pusan National University. In 1994, he was a visiting scholar at the Max Planck Institute for Informatics in Saarbrücken, Germany. His main research areas are algorithm design and analysis, and computational geometry. Currently, he is conducting research on flexible object animation and bioinformatics, especially phylogenetics.

DO-HOON LEE received his BSc degree in Computer Science and Statistics from Pusan National University, Pusan, Korea, in 1986. He also received his MSc and PhD degrees in Computer Science from Pusan National University, in 1992 and 1998, respectively. Since 1995, he has been a faculty member of the Department of Computer Engineering, Miryang National University, Korea. He is also the Director of Advanced Technology in Information and Communication Institute at Miryang National University. His research interests include virtual reality, human animation, and compression.