Fast and Stable Animation of Cloth with an Approximated Implicit Method

Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho
Graphics Application Lab
Department of Computer Science
Pusan National University
{ymkang, jhchoi, hgcho}@pearl.cs.pusan.ac.kr

Chan-Jong Park
Virtual Reality Center
Electronics and Telecommunication Research Institute
cjpark@etri.re.kr

Abstract

Realistic animation of soft objects such as cloth is essential for plausible character animation. Many techniques have been proposed for the simulation of soft objects, and most of them are based on numerical integration. Among the techniques, the implicit integration method is the most likely technique for real-time environments, since it allows large time steps for cloth simulation by ensuring the stability of systems. However, the most critical flaw of the implicit method is that it involves a large linear system. This paper presents a fast animation technique for animating soft objects based on mass-spring model with an approximated implicit method which does not involve linear system solving. The proposed technique stably updates the state of n mass-points in $O(n)$ time when the number of total springs are $O(n)$. Because the mass-spring model shows a super-elastic effect, the excessively deformed springs(i.e., super-elongated springs) should be adjusted for reality. This paper presents an efficient inverse dynamics process to adjust the super-elongated springs.

Keywords: cloth animation, implicit method, stability, interactive animation.

1. Introduction

Plausible character animation of various soft objects, such as cloth and hair, cannot be achieved without realistic simulation techniques for such objects.

Since Terzopoulos[10] characterized the simulation of cloth as a problem in deformable surface, many research groups have tried to give efficient solution to the cloth simulation, and various techniques have been proposed[1, 4, 2, 3, 11, 12].

Most of the techniques are based on physically-based modeling, and the problem is formulated as an ordinary differential equation[1]. Explicit Euler integration is the simplest approach to this problem. This approach calculates the internal forces at the current state and derives the next state by a simple method. However, this approach has a well-known flaw that it severely suffers from the instability[7], thus, it requires very small time steps for simulation.

Carignan modeled cloth simulation with rectangular discretization[3]. Their work has been successfully applied to the garments on moving characters. However, their method takes very long time for simulating a cloth, since they use very small time steps for explicit integration[1].

Breen focused on generating realistic images of draped shapes of cloth[2]. Their method utilized the properties of real cloth for generating static images, but simulation speed was not a major concern in this work. Eberhardt refined Breen's method to reduce the computational cost with higher-order explicit integration methods[6].

Recently, Baraff and Witkin have exploited implicit integration method to take large steps during the simulation[1]. Since this method can reduce the simulation time significantly by taking large steps, the implicit method is regarded as the best choice for the interactive animation of clothes. However, the implicit method also has a problem, in that a large linear system must be solved in order to calculate the next state of each mass-point. Desbrun alleviated the computational burden of the implicit method by precomputed

0-7695-0643-7/00 $10.00 © 2000 IEEE

filter[5].

1.1. Motivation and contributions

A Physically-based mass-spring model is a straight-forward approach to cloth simulation. This model represents a soft object as a set of mass-points and spring links between the mass-points. The spring links generate force on each mass-point, and the simulation system numerically integrates the force to calculate the position of each mass-point. The Explicit Euler integration is the easiest choice for numerical integration. However, the explicit method is not stable when the stiffness values of the spring links or the time steps of the simulation are very large. The Implicit method is clearly the best solution for this instability problem[1, 5].

Although the implicit method elegantly enforces an animation process to be a stable integration, it also suffers from computational overhead because it must solve a large linear system. The implicit method is considered to be the best way for a fast simulation of soft objects, but the computational overhead due to the linear system is a major obstruction to creating real-time animation.

Desbrun proposed a method to solve the linear system with a precomputed inverse matrix (i.e., precomputed filter) by considering only the linear part of each force. However, this method also suffers from heavy computation because the inverse matrix may not be sparse. Moreover, the precomputed filter method does not allow for dynamic changes of parameters such as mass, time step, or stiffness of cloth, since the calculation of the precomputed filter would require too much time when a large number of mass-points are used to represent a cloth-like object.

This paper presents a fast and stable animation technique that does not solve the large linear system. The proposed technique stably updates the state of n mass points in $O(n)$ time when the number of total springs are $O(n)$.

Because the mass-spring model is likely to be super-elongated, we must adjust the super-elongated springs by applying the inverse dynamics process. We present an efficient method for determining the order in which the springs are to be adjusted.

2. Instability and the implicit method

The simplest numerical integration method for cloth animation is to use Euler's explicit method. However, this explicit method causes a serious problem in cloth animation: This problem is known as "instability problem."

The Implicit method has been introduced in numerical analysis literature[8] in order to successfully overcome this instability problem, and Baraff used the implicit method for cloth animation with success[1].

2.1. Straight-forward simulation

A simple straight-forward approach to cloth simulation is to use the explicit Euler integration:

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

where v_i^t denotes the velocity of the i-th mass-point at time t, and F_i^t is the force acting on the mass-point at time t. Similarly, x_i^t denotes the location of the i-th mass-point at time t, and h denotes the time interval between simulation steps. By this simple method, we can easily calculate x_i^{t+h}, the location of the i-th mass-point at the next time step $t+h$, with current state values, x_i^t, v_i^t, and F_i^t.

The mass-spring model can be used for calculating F_i. Any i-th mass-point can be considered to have several neighbor mass-points that are linked with the i-th mass-point by springs, and the internal force F_i on the i-th mass-point can be easily calculated as follows:

$$\mathrm{F}_i = \sum_{\forall j | (i,j) \in E} k_{ij} (|\mathrm{x}_j - \mathrm{x}_i| - l_{ij}^0) \frac{(\mathrm{x}_j - \mathrm{x}_i)}{|\mathrm{x}_j - \mathrm{x}_i|}$$

where E is a set of spring edges between mass-points, k_{ij} denotes the spring constant of the link between the i-th and the j-th mass-points, x_i is the location of i-th mass-point, and l_{ij}^0 is the rest length of the spring between the i-th and the j-th mass-points.

However, this simple integration scheme (i.e., explicit method) cannot be applied to cloth simulations unless the time step h is very small, because the explicit method easily becomes unstable when the stiffness is increased. The generic problem of instability in the explicit method is inevitable, because it assumes that forces do not change during the time interval between the current step and the next step. Therefore, the explicit method is not an appropriate integration model for the real-time or interactive cloth animation systems that require very large time steps to achieve fast animation. It is generally regarded that the implicit method is a much better choice for the fast simulation of cloth.

2.2. The Implicit method

In order to solve a differential equation $dY/dt = f(Y)$ numerically, the general implicit integration assumes that the derivative is not simply $f(Y^t)$ throughout the time interval, but the derivative is a weighted average of the derivative $f(Y^t)$ at the beginning of the interval and $f(Y^{t+h})$ at the end of the interval[7]. Then, the update formula can be written as follows:

$$Y^{t+h} = Y^t + h[(1-\lambda) f(Y^t) + \lambda f(Y^{t+h})]$$

0-7695-0643-7/00 $10.00 © 2000 IEEE

where λ is a constant between 0 and 1. When λ is 0, the update turns to the explicit Euler method. If λ is 1, the update formula is called the implicit Euler method or the backward Euler method. With the implicit Euler method, the update formula can now be rewritten as follows:

$$\begin{aligned} \mathrm{v}_i^{t+h} &= \mathrm{v}_i^t + \mathrm{F}_i^{t+h}\frac{h}{m_i} \\ \mathrm{x}_i^{t+h} &= \mathrm{x}_i^t + \mathrm{v}_i^{t+h}h \end{aligned} \quad (2)$$

The only change is that F_i^t is replaced by F_i^{t+h}, but it has been proven that this simple change enables unconditionally stable integration[7, 8]. Thus, we can take large steps for cloth animation.

2.3. The weakness of the implicit method

The implicit method involves F_i^{t+h}, which cannot be calculated at the current step. However, F_i^{t+h} can be approximated by a first-order derivative:

$$\mathrm{F}^{t+h} = \mathrm{F}^t + \frac{\partial \mathrm{F}}{\partial x}\Delta \mathrm{x}^{t+h}$$

where F^t denotes the internal forces consisting of all the internal forces F_i^t on the i-th mass-point (i.e., $\mathrm{F}^t = [\mathrm{F}_1^t, \mathrm{F}_2^t, \cdots, \mathrm{F}_n^t]^T$), and similarly, $\Delta \mathrm{x}^t = [\Delta \mathrm{x}_1^t, \Delta \mathrm{x}_2^t, \cdots, \Delta \mathrm{x}_n^t]^T$.

$\partial \mathrm{F}/\partial \mathrm{x}$ is the negated Hessian matrix of the system[5], and will be denoted as H henceforth.

Because $\Delta \mathrm{x}^{t+h} = \mathrm{x}^{t+h} - \mathrm{x}^t = (\mathrm{v}^t + \Delta \mathrm{v}^{t+h})h$, the first equation of Implicit update in Eq.3 can be rewritten as follows:

$$(\mathrm{I} - \frac{h^2}{m}\mathrm{H})\Delta \mathrm{v}^{t+h} = (\mathrm{F}^t + h\mathrm{H}\mathrm{v}^t)\frac{h}{m} \quad (3)$$

where $\Delta \mathrm{v}^{t+h}$ is $\mathrm{v}^{t+h} - \mathrm{v}^t$ (i.e., $\mathrm{F}^{t+h}h/m$). If $\Delta \mathrm{v}^{t+h}$ can be calculated, velocity and location of each mass-point at the next step can easily be updated with Eq.3. Finally, cloth animation can be reduced to finding the value of $\Delta \mathrm{v}^{t+h}$. In Eq.3, $h\mathrm{H}\mathrm{v}^t$ represents additional forces. As mentioned by Desbrun[5], these additional forces are viscosity forces and can be easily calculated as follows:

$$(h\mathrm{H}\mathrm{v}^t)_i = h \sum_{\forall j | (i,j) \in E} k_{ij}(\mathrm{v}_j^t - \mathrm{v}_i^t)$$

However, the critical problem is that Eq.3 involves $\mathrm{I} - (h^2/m)\mathrm{H}$ which is an $O(n \times n)$ matrix. Because of this matrix, a large linear system must be solved to update the state of cloth. Modified conjugate gradient method has been used to alleviate the computation, but it also requires iteration until the error converges to be smaller than a user-defined tolerance value[1].

Since the guaranteed stability of the implicit method allows large steps, the animation of soft objects can be efficiently generated. However, the implicit method also suffers from heavy computation because it involves solving a large linear system, and its implementation is not easy.

3. Approximation method for fast and stable cloth animation

The idea of our paper is that the effects of the implicit method could be approximated with a few simple calculations. The main difference between the explicit method and the implicit method is that the implicit scheme involves a matrix $\mathrm{I} - (h^2/m)\mathrm{H}$ and an additional force $h\mathrm{H}\mathrm{v}$. The additional force (i.e., viscosity force) can be easily calculated, but the matrix $\mathrm{I} - (h^2/m)\mathrm{H}$ makes it difficult to efficiently animate cloth efficiently.

3.1. Precomputed filtering

This section briefly explains the cloth animation method with precomputed filtering proposed by Desbrun $et\ al.$[5]. Cloth simulation by the implicit method calculates the velocity change during time steps with Eq.3. Let F_s^t be the spring forces. Because $h\mathrm{H}\mathrm{v}^t$ is viscosity forces, we can rewrite $\mathrm{F}_s^t + h\mathrm{H}\mathrm{v}^t$ as F^t, the total internal forces at time t. Then, Eq.3 can be rewritten as follows:

$$(\mathrm{I} - \frac{h^2}{m}\mathrm{H})\Delta \mathrm{v}^{t+h} = \frac{\mathrm{F}^t h}{m} \quad (4)$$

Since $\mathrm{I} - (h^2/m)\mathrm{H}$ is an $O(n^2)$-sized matrix, the calculation of $\Delta \mathrm{v}$ requires solving a large linear system with $O(n)$ unknowns ($\Delta \mathrm{v}_i$, $i = 1, 2, ..., n$) and $O(n)$ equations. Eq.4 can be rewritten with the inverse of $\mathrm{I} - (h^2/m)\mathrm{H}$ as follows:

$$\Delta \mathrm{v}^{t+h} = (\mathrm{I} - \frac{h^2}{m}\mathrm{H})^{-1}\frac{\mathrm{F}^t h}{m}$$

Desbrun approximated the Hessian matrix by splitting the spring force into two parts and considering only the linear part[5]. The approximated Hessian matrix is as follows:

$$\left\{ \begin{array}{ll} \mathrm{H}_{ij} = k_{ij} & \text{if } i \neq j \\ \mathrm{H}_{ii} = -\sum_{j \neq i} k_{ij} & \end{array} \right\}$$

where H_{ij} denotes the entry of H at the i-th row and the j-th column. Then the matrix $(\mathrm{I} - (h^2/m)\mathrm{H})^{-1}$ remains constant during simulation if spring constant of each spring, the time step, and the mass of each mass-point are constant. Desbrun precomputed the inverse matrix of $\mathrm{I} - (h^2/m)\mathrm{H}$, and used the inverse matrix as a force filter for the simulation of cloth. Thus, the precomputed inverse matrix of $\mathrm{I} - (h^2/m)\mathrm{H}$ can be considered as a precomputed filter for

0-7695-0643-7/00 $10.00 © 2000 IEEE

forces. This technique produces stable result with simple calculations.

However, the inverse matrix of $\mathrm{I}-(h^2/m)\mathrm{H}$ is not necessarily a sparse matrix, even though $\mathrm{I}-(h^2/m)\mathrm{H}$ is sparse. If the inverse matrix has almost $O(n^2)$ entries, the state update calculation would require $O(n^2)$ time. Moreover, the adaptive time step strategy cannot be applied to the precomputed filter method, and the mass or stiffness of cloth cannot be dynamically changed, because the calculation of the inverse matrix requires much time. Because of these reasons, we did not employ the precomputed inverse matrix but a fast and stable calculation of $\Delta \mathrm{v}_i^{t+h}$ without solving linear system. The following section will explain how to calculate the approximated $\Delta \mathrm{v}_i^{t+h}$, and show how stable the results are by comparing them with those obtained by the precomputed filter method.

3.2. Approximated implicit method for a fast and stable integration

The velocity change of the i-th mass-point can be updated by considering only the linked mass-points, because H_{ij} is 0 when the i-th and the j-th mass-points are not linked with a spring. Therefore, the implicit update scheme can be rewritten as follows:

$$(1-\frac{h^2\mathrm{H}_{ii}}{m_i})\Delta \mathrm{v}_i-\frac{h^2}{m_i}\sum_{\forall j|(i,j)\in E}(\mathrm{H}_{ij}\Delta \mathrm{v}_j)=\frac{\mathrm{F}_i^t h}{m_i}$$

For the sake of simplicity, we adopted the approximated Hessian of Desbrun $et\ al.$ If the uniform spring constant k for all the spring-links is assumed, and n_i denotes the number of mass-points that are linked to the i-th mass-point, the approximated Hessian matrix can be rewritten as $\mathrm{H}_{ij}=k$ and $\mathrm{H}_{ii}=-kn_i$. The update formula can then be rewritten as follows:

$$\Delta \mathrm{v}_i^{t+h}=\frac{\mathrm{F}_i^t h+kh^2\sum_{(i,j)\ inE}\Delta \mathrm{v}_j^{t+h}}{m_i+kh^2n_i} \tag{5}$$

However, $\Delta \mathrm{v}_i^{t+h}$ cannot be calculated directly by Eq.5 because it contains some unknown values of $\Delta \mathrm{v}_j^{t+h}$, the velocity change of the j-th mass-points, that are linked to the i-th mass-point, at the next state. In order to directly calculate the $\Delta \mathrm{v}_i^{t+h}$, the velocity change of the i-th mass-point at the next step, we approximate the velocity change of the j-th mass-point at the next step.

$\Delta \mathrm{v}_j^{t+h}$ can be expressed as follows:

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

When we drop the term, $h^2\sum_{(j,l)\in E}k_{jl}\Delta \mathrm{v}_l^{t+h}$, we have an approximation of $\Delta \mathrm{v}_j^{t+h}$:

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

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

$$\Delta \mathrm{v}_i^{t+h}=\frac{\mathrm{F}_i^t h+h^2k\sum_{(i,j)\in E}\mathrm{F}_j^t h/(m_j+h^2kn_j)}{m_i+h^2kn_i} \tag{6}$$

where n_i is the number of mass-points which are linked to i by springs, and n_j is the number of mass-points linked to j.

Since $\Delta \mathrm{F}_j^t$ is already known, we can directly calculate the velocity change of the i-th mass-point at the next step. This means that we can generate approximated motion of flexible objects without solving the linear system which was a major obstruction to interactive animation. Since we update the velocity change 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 precomputed inverse matrix (precomputed filter) method or any general approaches to the implicit integration.

Moreover, in this model, the mass, time step, or stiffness during animation can be easily modified without any additional computations. Such dynamic changes in parameters cannot be achieved when the precomputed inverse matrix is used.

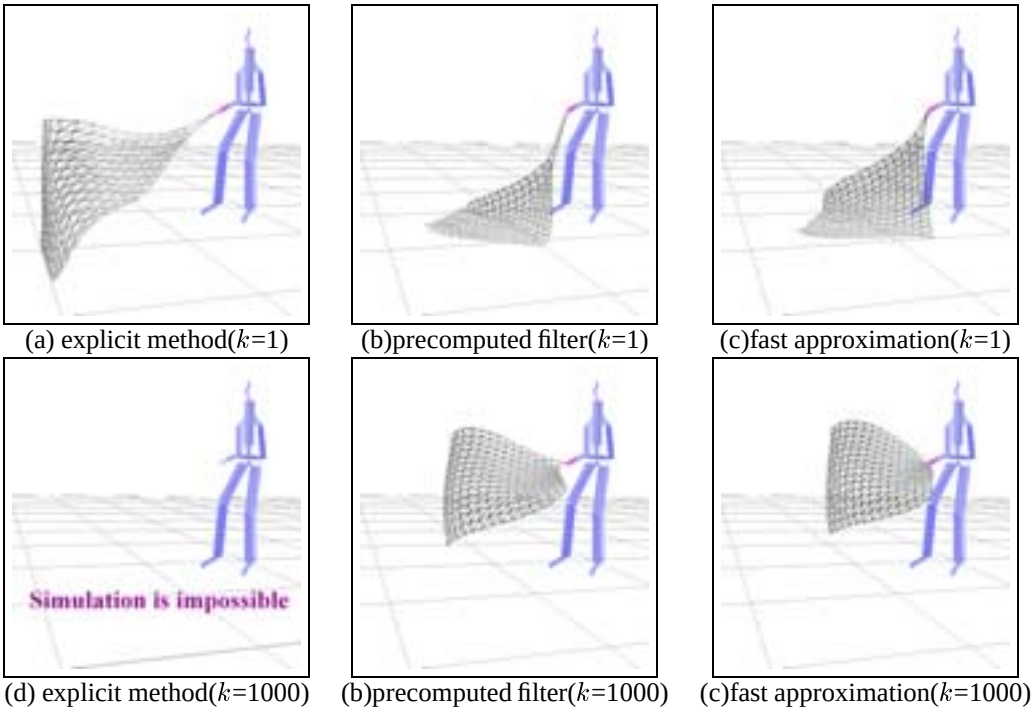
Fig.1 compares the results of our model with those of precomputed filter method. (a) and (d) are the results when the explicit method is used. When the stiffness k is increased to 1000, the explicit method fails. (b) and (e) are generated by calculating the inverse matrix of $\mathrm{I}-(h^2/m)\mathrm{H}$. (c) and (f) are generated by using our model. For (a), (b), and (c), the parameters are k=1, h=1/30sec, m=0.1kg, size = $1m\times1m$, and n= 225. For (d), (e) and (f), the parameters are k=1000, h=1/30sec, m=0.1kg, size = $1m\times1m$, and n= 225, where k denotes the spring constant of a link, m is the mass of each mass-point, and n denotes the number of mass-points. Although no inverse dynamics process was used to generate the results shown in Fig.1, our model generated very stable results.

Although the results of our model appear stiffer than those generated by the precomputed filter, we found, by experiments, the proposed model works stably in any condition. The stiffness was increased up to 10^6, and no instability problems were found in our model.

4. Stable damping

The damping forces must be considered to generate more realistic motion of cloth, but damping forces should be care-

0-7695-0643-7/00 $10.00 © 2000 IEEE



(a) explicit method(k=1) (b)precomputed filter(k=1) (c)fast approximation(k=1)

(d) explicit method(k=1000) (b)precomputed filter(k=1000) (c)fast approximation(k=1000)

Figure 1. Result comparison - (a) and (d) are the results when explicit method is used. (b) and (e) are the results when precomputed filter is used. (c) and (f) are generated by using our fast approximation model. When the stiffness k is 1000, the explicit method fails to simulate the cloth while precomputed method and our model stably work: (a), (b) and (c) are generated with parameters of k=1, h=1/30sec, m=0.1kg, n=225, and size=1$m\times$1m, (d), (e) and (f) with k=1000, h=1/30sec, m=0.1kg, n=225, and size=1$m\times$1m.

fully tuned. The simple damping force on the i-th mass-point can be implemented as follows:

$$\mathrm{F}_{di} = -k_d \mathrm{v}_i \quad (k_d > 0)$$

where F_{di} denotes the damping force on the i-th mass-point, k_d is the damping factor, and v_i is the current velocity of the i-th mass-point.

However, the mass-point would move in the opposite direction of its current velocity when the damping factor k_d is greater than a certain threshold. Moreover, such a situation can cause the animation to become unstable. We have implemented a stable damping method which does not create any of the undesirable situations described above.

4.1. Velocity change caused by damping

The simple damping force model blindly generates the damping force, and the damping force can caused a mass-point to move in the opposite direction of its velocity. However, it is impossible for the natural damping force to cause a mass-point move in the opposite direction of its current velocity. Therefore, it is obvious that the maximum damping force enforces a mass-point to stop at its current position. In other words, the velocity change caused by the maximum damping force on the i-th mass-point cannot exceed $-(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})$. When the velocity change caused by damping force is $-(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})$, the mass-point stops moving. Therefore, the velocity change caused by damping force is:

$$\Delta\mathrm{v}_{di} = -\lambda(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h}) \qquad (0 \le \lambda \le 1)$$

where $\Delta\mathrm{v}_{di}$ denotes the velocity change of the i-th mass-point caused by damping force.

Since damping force increases when velocity increases, we modeled the velocity change caused by damping as follows:

$$\Delta\mathrm{v}_{di} = -(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})(1 - \frac{1}{1 + k_d|\mathrm{v}_i^t|})$$

where k_d denotes the damping factor.

This damping model does not correspond to natural damping force. However, this study not interested in physical correctness. The primary purpose of this study is just

0-7695-0643-7/00 $10.00 © 2000 IEEE

to create plausible and stable damping that does not disturb the stability of animation. Since $-(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})(1 - 1/(1 + k_d|\mathrm{v}_i^t|))$ does not exceed $-(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})$, it is clear that our damping model is stable.

4.2. Considering air-interaction

Two kinds of forces, drag force and lift force, are exerted on an object moving in a fluid. The magnitude of drag force is known as follows:

$$|F_D| = \frac{1}{2} C_D \rho |V|^2 S \sin\theta$$

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 object surface, and θ is the angle between V and the surface. The direction of the drag force is opposite to the velocity.

It can be easily seen that the drag force is proportional to $\sin\theta$ and the square of the magnitude of velocity if the drag force coefficient and density are constant. When $\hat{\mathrm{N}}_i$ denotes the unit normal of the i-th mass-point, and $\hat{\mathrm{v}}_i$ denotes $\mathrm{v}_i/|\mathrm{v}_i|$, the angle between $\hat{\mathrm{N}}_i$ and $\hat{\mathrm{v}}_i$ is $\pi/2 - \theta$. Thus, $|\hat{\mathrm{N}} \cdot \hat{\mathrm{v}}_i|$ is $\sin\theta$ $(= \cos(\pi/2 - \theta))$, and the drag force is proportional to $|\hat{\mathrm{N}} \cdot \hat{\mathrm{v}}_i|$. Therefore, we modified the our damping model as follows:

$$\Delta\mathrm{v}_{di} = -|\hat{\mathrm{N}}_i \cdot \tfrac{\mathrm{v}_i}{|\mathrm{v}_i|}|(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})(1 - \tfrac{1}{1+k_d|\mathrm{v}_i^t|^2})$$

5. Summary of proposed model

The proposed model, which stably calculates the next state of each mass-point of cloth based on the mass-spring model, can be summarized as follows:

$$\begin{aligned}
\mathrm{F}_{si}^{\ t} &= \textstyle\sum_{\forall j|(i,j)\in E} k_{ij}(|\mathrm{x}_j - \mathrm{x}_i| - l_{ij}^0)\frac{(\mathrm{x}_j - \mathrm{x}_i)}{|\mathrm{x}_j - \mathrm{x}_i|} \\
\mathrm{F}_{vi}^{\ t} &= \textstyle\sum_{\forall j|(i,j)\in E} k_{ij} h(\mathrm{v}_j^t - \mathrm{v}_i^t) \\
\mathrm{F}_i^t &= \mathrm{F}_{si}^{\ t} + \mathrm{F}_{vi}^{\ t} \\
\Delta\mathrm{v}_i^{t+h} &= \frac{\mathrm{F}_i^t h + h^2 k \sum_{(i,j)\in E} \mathrm{F}_j^t h/(m_j + h^2 k n_j)}{m_i + h^2 k n_i} \\
\Delta\mathrm{v}_{di}^{\ t} &= -|\hat{\mathrm{N}}_i \cdot \tfrac{\mathrm{v}_i^t}{|\mathrm{v}_i^t|}|(\mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h})(1 - \tfrac{1}{1+k_d|\mathrm{v}_i^t|^2}) \\
\mathrm{v}_i^{t+h} &= \mathrm{v}_i^t + \Delta\mathrm{v}_i^{t+h} + \Delta\mathrm{v}_{di}^{\ t} \\
\mathrm{x}_i^{t+h} &= \mathrm{x}_i^t + \mathrm{v}_i^{t+h} h
\end{aligned}$$

It is important that F_i^t, the total internal force on the i-th mass-point at time t, involves the viscosity force, $\mathrm{F}_{vi}^{\ t}$. When this viscosity term is dropped, we get an unstable result. The damping term, $\Delta\mathrm{v}_{di}$, does not affect the stability of our model.

6. Adjustment for the excessive deformation

The mass-spring model is an intuitive and simple approach to cloth simulation, but a serious problem occurs when cloth is simulated with the mass-spring model. Because the mass-spring model represents the links between mass-points by springs, super-elongated links cannot be avoided.

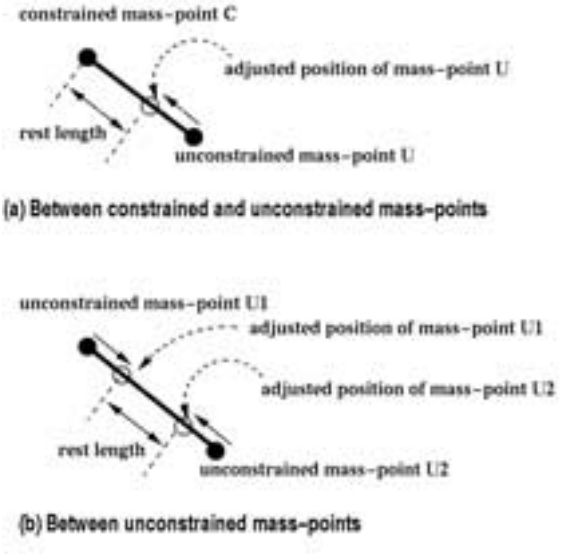


Figure 2. Position adjustment for inverse dynamics process proposed by Provot $et\ al.$

6.1. Our strategy

Provot $et\ al.$ proposed a simple and effective inverse dynamics process[9]. Their method is shown in Fig.2. When a super-elongated spring is linked with a constrained mass-point and an unconstrained mass-point, as shown in Fig.2 (a), the position of the unconstrained mass-point is adjusted to make the length between the mass-points become the rest length of the spring. If the mass-points are both unconstrained, the positions of both mass-points are adjusted, as shown in Fig.2 (b).

However, Provot did not present how to determine the order in which the edges are to be adjusted. The result of the inverse dynamics process is highly dependent on this order.

When a mass-point is constrained, a link between the constrained mass-point and unconstrained mass-point is

likely to be super-elongated. Thus, it may be possible to determine the order of adjustment in advance, if the constrained mass-point is known before the start of the simulation or animation. However, the constrained mass-point cannot be known when the interactive animation or simulation is performed. Therefore, we employed a dynamic ordering strategy for the inverse dynamics process.

6.2. Bucket sorting for ordering

One possible approach to the dynamic ordering may be implemented by sorting the springs according to their elongated length. Various sorting algorithms can be used. However, sorting algorithms may cause a problem in computational time because the time complexity of general sorting algorithms is $O(n \log n)$. Even worse, some sorting algorithms, such as $quick\ sort$, may work in $O(n^2)$ time in the worst case. This means that dynamic ordering may be a critical bottleneck in interactive or real-time animation.

We used a bucket sorting technique to determine the adjustment order in $O(n)$ time. The ordering is performed by a 2-phase process. First, every spring is scanned to find the maximum elongation length and the minimum elongation length. After the maximum and minimum elongation lengths of springs are found, buckets are created. If there are m buckets, the interval [minimum,maximum] is divided into m subintervals. Each bucket is assigned one subinterval. In the second phase, the locational assignment of each spring is determined. The order in a bucket was not considered. It is obvious that this ordering can be performed in $O(n)$ time. After all the springs are classified into m buckets, each spring is adjusted in accordance with the order of buckets.

7. Experimental results

Our cloth animation technique is implemented by using C++ with $OpenGL$ and $Open\ Inventor$ libraries on $SGI\ Indigo^2$ and O_2 machines with $R10000$ processors. Our technique generated real-time cloth animation with hundreds of mass-points at $30Hz$ or $60Hz$ frame rate. We were also able to generate interactive animation with more mass-points (i.e., thousands of mass-points).

Fig.3 shows the results of our method when it is applied to cloth animation. In Fig.3, One mass-point was constrained to move in accordance with an external input, and the others were calculated using our model.

Fig.6 shows various results from our animation model. Fig.6 (a) shows the result when one mass-point is constrained to be fixed. Fig.6 (b) and (c) show the results when two mass-points are constrained to stick to a wand which moves in accordance with input data. Fig.6 (d) shows the result when multiple mass-points are constrained to stick to a wand. Under those various cases, our model generated plausible scenes of moving clothes.

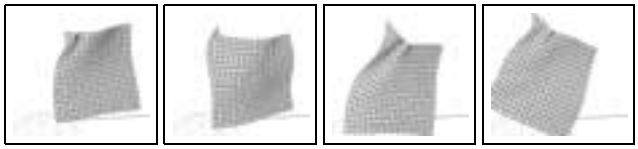


Figure 3. Sequence of cloth animation when one mass-point is constrained to move in accordance with external input data

Fig.7 shows the results according to different stiffness values. The results shown in Fig.7 (a), (b), and (c) were generated under the same parameters except for the stiffness value. The stiffness of k=1 was used to generate the result shown in Fig.7 (a); k=65 for the result shown in Fig.7 (b); and k=500 for Fig.7 (c). As the stiffness value increases, the cloth becomes more rigid. However, the motion of each mass-point is always stable

We have animated a ribbon dance, and the results are shown in Fig.4. The motion of the dancer was captured from a real human, and a virtual ribbon was attached to the character. The ribbon does not have a square shape, and the vertical and the horizontal links have diffrent rest lengths.

In order to observe how realistic our animation technique is, we generated clothes draped on a table-like object. Fig.5 shows the results. Fig.5 (a) and (b) were generated with the stiffness of 100 and 225, respectively. We also used 1/60 sec time steps. The result shows that our technique generates very realistic scenes of cloth movement. the number of mass-points were increased up to 2500 for reality of the scene.

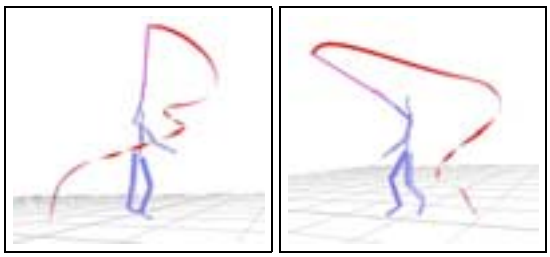
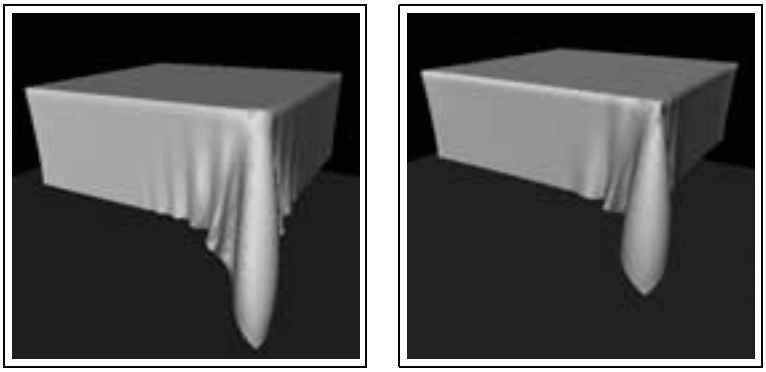


Figure 4. Snapshots of ribbon dance animation

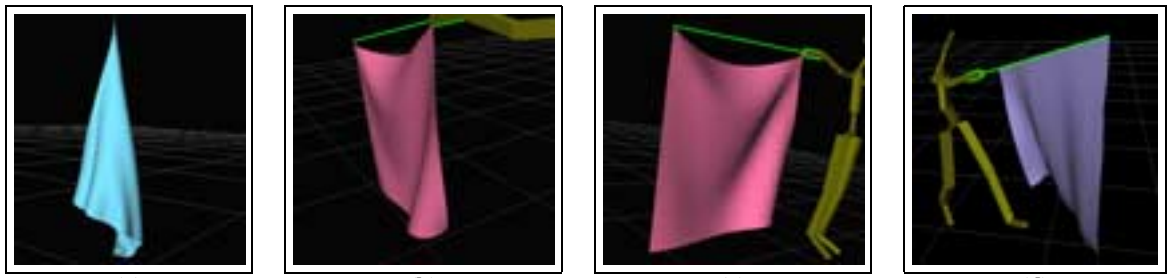
Our method can be applied to three dimensional objects. We have implemented a simple three dimensional object shown in Fig.8, and successfully animated the object.

0-7695-0643-7/00 $10.00 © 2000 IEEE



(a) (b)

Figure 5. A cloth drape on table (a) with sitffness 100 (b) with stiffness 225



(a) (b) (c) (d)

Figure 6. Generated clothes with our model (a) one constrained node (b) two constrained nodes (c) two constrained nodes (d) multiple constrained nodes

8. Conclusion

We have presented a fast cloth simulation technique that approximates the filtering effects of the implicit method in order to alleviate the heavy computation needed for solving a large linear system. Our experiments show that approximated method produces plausible animation results in a real-time environment.

Our technique generates cloth animation very stably. Moreover, our technique is as fast as the explicit method in the calculation of the next state, because our method does not involve solving linear system which is a bottleneck of the implicit method.

Another advantage of our technique is that it is very intuitive and easy to implement, since our technique calculates the next state of cloth with a direct update formula.

We were not concerned about the physical correctness of cloth simulation; we were only interested in fast and plausible animation of cloth. The results of the experiments show that our technique produces very plausible animation of cloth with large steps (i.e., fast and realistic).

Various systems that require fast animation of cloth, like VR systems or interactive systems, might be successfully implemented with our technique.

Acknowledegment

This research was supported in part by ETRI(Electronics and Telecommunication Research Institute) research contract (1999).

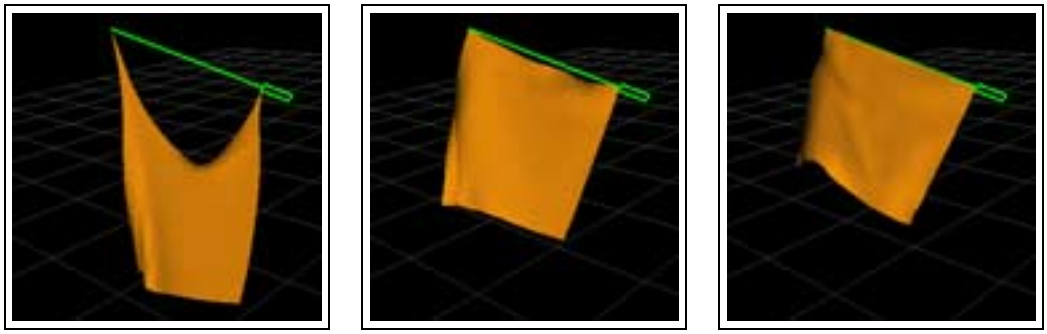
References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. *Computer Graphics (Proc. of SIGGRAPH '98)*, pages 43–52, 1998.

[2] D. Breen, D.House, and M. Wozny. Predicting the drape of woven cloth using interacting particles. *Computer Graphics (Proc. of SIGGRAPH '94)*, pages 365–372, 1994.

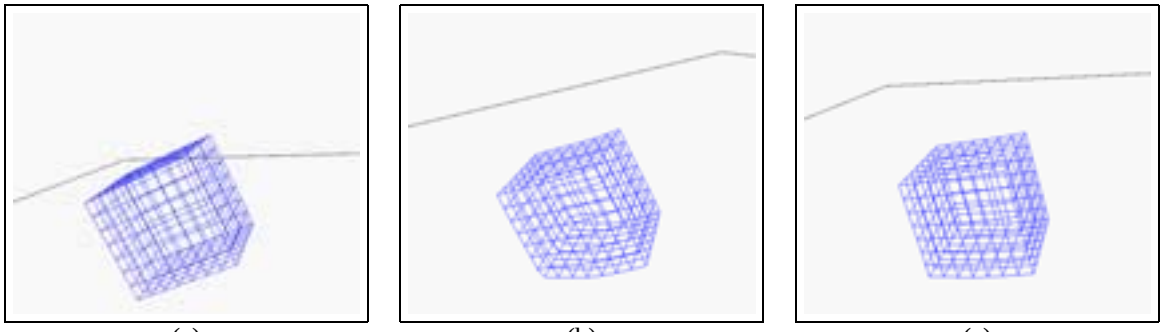
[3] M. Carignan, Y. Yang, N. Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics (Proc. of SIGGRAPH '92)*, pages 99–104, 1992.

[4] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics(Proc. of SIGGRAPH '91)*, pages 257–266, 1991.



(a) (b) (c)

Figure 7. Results with different stiffness values ((a) k=1 (b) k=65 (c) k=500). The size of every cloth is 1m×1m with 15×15 mass-points, and the damping parameter k_d is 0.22



(a) (b) (c)

Figure 8. 3D object animation with the proposed technique

[5] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. *Proc. of Graphics Interface '99*, 1999.

[6] B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16:52–59, 1996.

[7] M. Kass. An introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Note*. ACM SIGGRAPH, 1994.

[8] S. Nakamura. Initial value problems of ordinary differential equations. In *Applied Numerical Methods with Software*, pages 289–350. Prentice-Hall, 1991.

[9] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Graphics Interface*, pages 147–154, 1995.

[10] D. Terzopoulos, J. Platt, and A. Barr. Elastically deformable models. *Computer Graphics (Proc. of SIGGRAPH '87)*, pages 205–214, 1987.

[11] P. Volino, M. Courchesne, and N. Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Computer Graphics(Proc. of SIGGRAPH '95)*, pages 137–144, 1995.

[12] P. Volino, N. Thalmann, S. Jianhua, and D. Thalmann. An evolving system for simulating clothes on virtual actors. *IEEE Computer Graphics and Application*, 16:42–51, 1996.

0-7695-0643-7/00 $10.00 © 2000 IEEE