Real-time Animation Technique for Flexible and Thin Objects

Young-Min Kang†, Jeong-Hyeon Choi†, Hwan-Gue Cho†

Do-Hoon Lee††

Chan-Jong Park†††

†Department of Computer Science, Pusan National University
{ymkang,jhchoi,hgcho}@pearl.cs.pusan.ac.kr

††Department of Computer Engineering, Miryang National University
dhlee@arang.miryang.ac.kr

†††Virtual Reality Center, Electronics and Telecommunication Research Institute
cjpark@etri.re.kr

ABSTRACT

In this paper, we propose an efficient technique for the animation of flexible thin objects. Mass-spring model was employed to represent the flexible objects. Many techniques have used the mass-spring model to generate plausible animation of soft objects. The easiest approach to animation with mass-spring model is explicit Euler method, but the explicit Euler method has serious disadvantage that it suffers from 'instability problem'. The implicit integration method is a possible solution to overcome the instability problem. 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 mass-spring model with approximated implicit method. The proposed technique stably updates the state of n mass-points in $O(n)$ time when the number of total springs are $O(n)$. We also consider the interaction of the flexible object and air in order to generate plausible results.

Keywords: flexible object, mass-spring model, implicit method, stability

1 INTRODUCTION

For the physically-based modeling, the mass-spring model is a simple and powerful approach to representing flexible objects such as cloth. There have been many techniques for simulation of flexible objects [Carig92, Volin95, Provo95], and the techniques use various models such as finite element model, particle-system, deformable surface model [Wang98]. Among those models, the mass-spring model is the easiest and most intuitive.

Various techniques have been introduced to use the mass-spring model to simulate or animate the flexible objects [Provo95, Desbr99, Chen98]. Animation techniques based on the mass-spring model can be formulated as a simple ordinary differential equation. We can easily calculate the force on each mass-point, and we can animate the mass-point by numerical integration of the force.

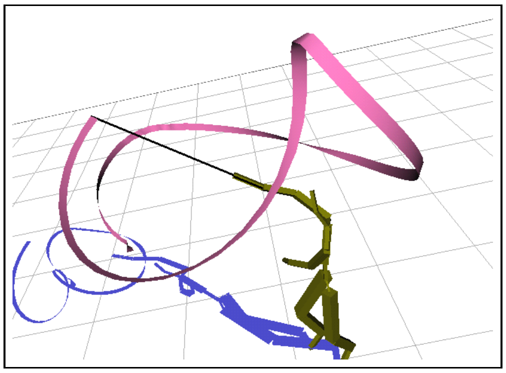


Figure 1: Snapshots of Ribbon Dance Animation

Explicit Euler integration is the simplest approach to this integration procedure. However, it requires very small time steps for simulation or animation since this approach severely suffers from the instability problem [Kass94]. Implicit method is well-known solution to this instability problem in numerical analysis literature [Nakam91]. Recently, Baraff and Witkin have exploited implicit integration method to take large steps during the simulation of cloth [Baraf98]. Since this method can significantly reduce the simulation time by taking large steps, 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 that a large linear system should be solved. Although the implicit method elegantly enforces an animation process to be a stable integration, the large linear system involved in the implicit method is major obstruction to the real-time animation.

Desbrun proposed an efficient method to solve the linear system with the precomputed inverse matrix (i.e., precomputed filter). However, it also suffers from heavy computation because the inverse matrix may not be sparse. Moreover, dynamic control of parameters such as mass, time step, or stiffness of flexible object is impossible.

This paper presents a fast and stable animation technique which stably updates the state of n mass points in $O(n)$ time for the real-time animation of flexible objects as shown in Fig. 1. The adaptive step size strategy or dynamic modification of animation parameters such as mass or stiffness are not restricted in our model. We also considered the interaction between the flexible object and the air for the plausible result.

2 MASS-SPRING MODEL AND STABLE INTEGRATION

Mass-spring model is a simple technique for representing flexible objects. Mass-spring model represents an object with mass-points and springs. The forces caused by the springs make the mass-points move. The springs can be arbitrarily constructed. This mass-spring model is very intuitive approach to the representation of soft objects such as cloth.

By using explicit Euler integration, the mass-points are easily animated. However, due to instability problem, the simple explicit Euler integration cannot be used unless the time step h is very small. Thus, the explicit method takes too much time for producing an animation result of mass-spring model [Nakam91, Kass94, Desbr99].

2.1 Implicit method for stable animation

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:

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

where $\mathbf{v}_i^t$ denotes the velocity of the i-th mass-point at time t, and $\mathbf{F}_i^t$ is the force acting on the mass-point at time t. Similarly, $\mathbf{x}_i^t$ denotes the location of the i-th mass-point at time t, m_i is the mass of the i-th mass-point, and h denotes time interval between animation steps. Since this integration method stably updates the next state of mass-spring model, we can take a large step for animation. Therefore, the implicit method is the best choice for real-time or interactive animation system.

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

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

where $\mathbf{F}^t$ denotes the internal forces consisting of all the internal forces $\mathbf{F}_i^t$ on the i-th mass-point (i.e., $\mathbf{F}^t = [\mathbf{F}_1^t, \mathbf{F}_2^t, \cdots, \mathbf{F}_n^t]^{\mathrm{T}}$), and similarly $\Delta\mathbf{x}^t = [\Delta\mathbf{x}_1^t, \Delta\mathbf{x}_2^t, \cdots, \Delta\mathbf{x}_n^t]^{\mathrm{T}}$. Because $\partial\mathbf{F}/\partial\mathbf{x}$ is negated Hessian matrix of the system [Desbr99], we will denote $\partial\mathbf{F}/\partial\mathbf{x}$ as $\mathbf{H}$ henceforth.

Since $\Delta\mathbf{x}^{t+h} = \mathbf{x}^{t+h} - \mathbf{x}^t = (\mathbf{v}^t + \Delta\mathbf{v}^{t+h})h$, we can rewrite the first equation of implicit update in Eq. 1 as follows:

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

where $\Delta\mathbf{v}^{t+h}$ is $\mathbf{v}^{t+h} - \mathbf{v}^t$. If we can calculate $\Delta\mathbf{v}^{t+h}$, we can easily update the velocity and location of each mass-point at the next step with Eq. 1. The animation of flexible objects is finally reduced to finding the value of $\Delta\mathbf{v}^{t+h}$. In Eq. 2, $h\mathbf{H}\mathbf{v}^t$ represents additional forces. As mentioned by Desbrun [Desbr99], these additional forces are viscosity forces and can be easily calculated as follows:

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

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

The implicit method has a critical weakness in that Eq. 2 involves $\mathbf{I} - (h^2/m)\mathbf{H}$ which is $O(n \times n)$-sized matrix. Due to this matrix, we must solve a large linear system to update the state of model. Modified conjugate gradient method has been used to alleviate the computation, but it is still far from interactive animation of mass-spring model [Baraf98].

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

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

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

Their method is faster than the general techniques which exploits the implicit method. However, the inverse matrix of $\mathbf{I} - (h^2/m)\mathbf{H}$ is not necessarily a sparse matrix, even though $\mathbf{I} - (h^2/m)\mathbf{H}$ is sparse. Moreover, the adaptive time step strategy cannot be applied to the precomputed filter method, and we cannot dynamically change the mass or stiffness, because the calculation of the inverse matrix requires much time. Due to these reasons, we did not employed the precomputed filter.

2.2 Approximation method for interactive animation

We can update the the velocity change of the i-th mass-point by considering only the linked mass-points, because $\mathbf{H}_{ij}$ is 0 when the i-th and the j-th mass-points are not linked with spring. Therefore, we can rewrite the implicit update scheme (Eq. 2) as follows:

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

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

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

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

$$\Delta \mathbf{v}_i^{t+h} = \frac{\tilde{\mathbf{F}}_i^t h + k h^2 \sum_{(i,j) \in E} \Delta \mathbf{v}_j^{t+h}}{m_i + k h^2 n_i} \tag{3}$$

However, we cannot calculate $\Delta \mathbf{v}_i^{t+h}$ directly by Eq. 3 because it contains unknown $\Delta \mathbf{v}_j^{t+h}$, the velocity changes of the j-th mass-points which are linked to the i-th mass-point at the next state. In order to calculate $\Delta \mathbf{v}_i^{t+h}$, 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.

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

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

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

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

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

$$\Delta \mathbf{v}_i^{t+h} = \frac{\tilde{\mathbf{F}}_i^t h + h^2 k \sum_{(i,j) \in E} \tilde{\mathbf{F}}_j^t h / (m_j + h^2 k n_j)}{m_i + h^2 k n_i} \tag{4}$$

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 $\Delta \tilde{\mathbf{F}}_j^t$ 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, we can easily modify the mass, time step, or stiffness during animation without any additional computations. These dynamic change of parameters cannot be achieved when precomputed inverse matrix is used. It is easy to modify this equation for general cases where the stiffness values of springs are different from each other. We have tested approximated motion of flexible objects, and found that our model generates stable animation results.

2.3 Stability of the proposed method

For the verification of 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 will be helpful for understanding 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 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-point i and j can be calculated as follows:

$$\begin{array}{rcl} \tilde{\mathbf{F}}_i^t &=& -k(\mathbf{x}_i^t - \mathbf{x}_j)^t \\ \tilde{\mathbf{F}}_j^t &=& -k(\mathbf{x}_j^t - \mathbf{x}_i)^t \quad = \quad -\tilde{\mathbf{F}}_i^t \end{array}$$

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

$$\begin{array}{rcl} \mathbf{x}_i^{t+h} &=& \mathbf{x}_i^t + \frac{\tilde{\mathbf{F}}_i^t h - kh^2 \tilde{\mathbf{F}}_i^t h/(m+kh^2)}{m+kh^2} h \\ \mathbf{x}_j^{t+h} &=& \mathbf{x}_j^t - \frac{\tilde{\mathbf{F}}_i^t h - kh^2 \tilde{\mathbf{F}}_i^t h/(m+kh^2)}{m+kh^2} h \end{array}$$

Let $\mathbf{u}$ denotes $\mathbf{x}_i^t - \mathbf{x}_j^t$. Then, $\tilde{\mathbf{F}}_i^t$ can be expressed as -$k\mathbf{u}$, we can rewrite the difference of the locations as follows:

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

It is obvious that $|\mathbf{x}_i^{t+h} - \mathbf{x}_j^{t+h}|$ is less than $|\mathbf{x}_i^t - \mathbf{x}_j^t|$ when $0 \leq mkh^2/(m+kh^2)^2 \leq 1$. It is trivial to show that $mkh^2/(m+kh^2)^2$ is larger than 0. Now, we have only to show that $mkh^2/(m+kh^2)^2 \leq 1$. Because $(m+kh^2)^2$ is $m^2 + k^2h^4 + 2mkh^2$, we can easily find that $(m+kh^2)^2 - mkh^2$ is $m^2 + k^2h^4 + mkh^2$ and larger than 0. Therefore, $mkh^2/(m+kh^2)^2 \leq 1$.

3 AIR-INTERACTION MODEL FOR PLAUSIBLE ANIMATION

In order to generate a realistic animation of flexible thin object, two kinds of forces, drag force and lift force, must be considered. 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. The magnitude of the lift force can be expressed similarly as follows:

$$|F_L| = \frac{1}{2} C_L \rho |V|^2 S \cos\theta$$

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 velocity.

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

$$\mathbf{F}_{Di} = -K_D |\hat{\mathbf{N}}_i \cdot \hat{\mathbf{v}}_i| |\mathbf{v}_i|^2 \hat{\mathbf{v}}_i$$

where K_D is the control parameter for the drag force.

For the implementation of the lift force, we must determine the direction of the lift force. Since the direction of the lift force is perpendicular to the direction of the velocity, we determined $\mathbf{U}_i$, the direction of the lift force on the i-th mass-point as follows:

$$\begin{array}{rcl} \tilde{\mathbf{N}}_i &=& \hat{\mathbf{N}}_i \ , \ \ if \ \ \hat{\mathbf{N}}_i \cdot \hat{\mathbf{v}}_i > 0 \\ && -\hat{\mathbf{N}}_i \ , \ \ otherwise \\ \mathbf{U}_i &=& (\tilde{\mathbf{N}}_i \times \hat{\mathbf{v}}_i) \times \hat{\mathbf{v}}_i \end{array}$$

Since the direction of the lift force was determined, we have only to determine the magnitude

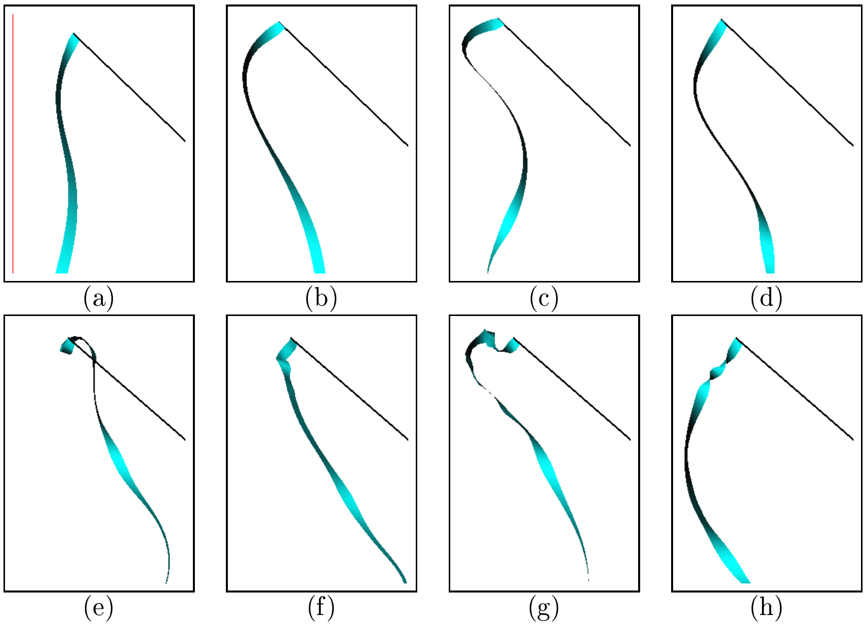


Figure 2: Movement of ribbon object in air : the figures from (a) to (d) show the sequence of ribbon animation when the drag and the lift forces are not taken into account, and the figures from (e) to (f) show the sequence of movement of the ribbon which is affected by the drag and the lift forces

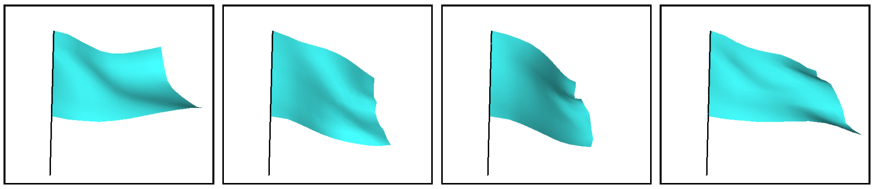


Figure 3: Flag in wind : $50cm \times 50cm$ size flag is tied to a pole, and wind is blowing from left to right. The velocity of the wind is 40 m/s, K_D is 0.01, and K_L is 0.01. Our air-interaction model generated very plausible animation of thin object

of the lift force. We implemented the lift force $\mathbf{F}_{Li}$ on the i-th mass-point as follows:

$$\mathbf{F}_{Li} = (K_L \cos\theta |\mathbf{v}_i|^2)\mathbf{U}_i$$

where K_L is the control parameter for the lift force.

Fig. 2 shows the effect of the drag force and lift force. The figures from (a) to (d) in Fig. 2 show the sequence of ribbon animation when the drag and the lift forces are not taken into account, while the figures from (e) to (f) show the sequence of movement of the ribbon which is affected by the drag and the lift forces

The drag and lift effect caused by wind can be easily taken into account by calculating the relative velocity of each mass-point respect to the air. Fig. 3 is the animation results when the drag force and 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 drag and lift force model generated very plausible scene.

4 COLLISION

Collision detection and response are also essential for realistic animation. We adopted general collision detection and response techniques [Volin94, Provo97]. We have found some problems when we implemented those general techniques. In this section, the problems and our solutions are described.

4.1 Collision detection

The point-triangle collision is the collision between a point of one triangle and the face of another triangle. Let $\mathbf{P}$ be a point and $\mathbf{V}_p$ the velocity of the point, and t be the elapsed time since t_0, then $\mathbf{P}(t) = \mathbf{P}(t_0) + t\,\mathbf{V}_P$. Similarly, let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be points of another triangle, then $\mathbf{A}(t) = \mathbf{A}(t_0) + t\,\mathbf{V}_A$, $\mathbf{B}(t) = \mathbf{B}(t_0) + t\,\mathbf{V}_B$, $\mathbf{C}(t) = \mathbf{C}(t_0) + t\,\mathbf{V}_C$. If there exists collision, then the collision time can

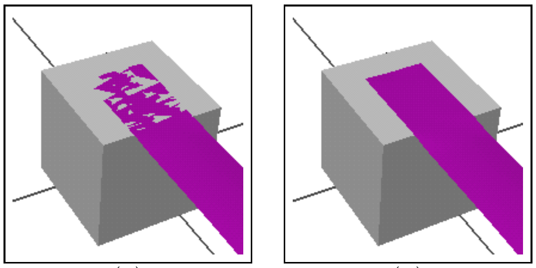

(a) (b)

Figure 4: (a) is the snapshot when the threshold distance is not considered, and (b) is the snapshot when the threshold distance is considered.

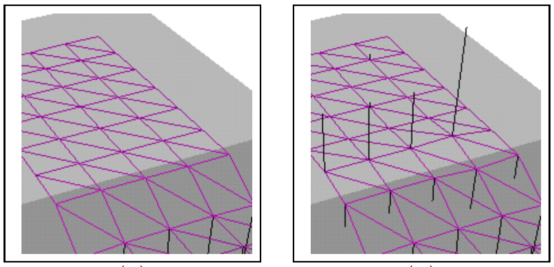

(a) (b)

Figure 5: (a) is the snapshot when the rotation is not considered, and (b) is the snapshot when the rotation is considered. The line segments attached to mass-points are velocities of corresponding mass-points

be computed by the following equations[Provo97]:

$$\left\{\begin{array}{l} t \in [t_0, t_0+h] \\ u, v \in [0,1],\ u+v \leq 1 \\ \mathbf{AP}(t) = u\ \mathbf{AB}(t) + v\ \mathbf{AC}(t) \\ \mathbf{AP}(t) \cdot (\mathbf{AB}(t) \times \mathbf{AC}(t)) = 0 \end{array}\right. \quad (5)$$

The edge-edge collision is the collision between an edge of one triangle and an edge of another triangle. Let $\mathbf{AB}$ be an edge and $\mathbf{V}_{AB}$ the velocity of the edge, then $\mathbf{AB}(t) = \mathbf{AB}(t_0) + t\ \mathbf{V}_{AB}$, and let $\mathbf{PQ}$ be another edge, then $\mathbf{PQ}(t) = \mathbf{PQ}(t_0) + t\ \mathbf{V}_{PQ}$. If there exists collision, then the collision time can be computed by the following equations[Provo97]:

$$\left\{\begin{array}{l} t \in [t_0, t_0+h] \\ u, v \in [0,1] \\ \mathbf{A}(t) + u\mathbf{AB}(t) = \mathbf{P}(t) + v\mathbf{PQ}(t) \\ \mathbf{AP}(t) \cdot (\mathbf{AB}(t) \times \mathbf{PQ}(t)) = 0 \end{array}\right. \quad (6)$$

Since the collision time calculated by Eq. 5 or Eq. 6 is the actual colliding time, and the computation may have floating point error, we adjust it to the time before collision. Let d_t be a threshold distance. When the distance between a flexible object and other objects is nearer than d_t, collision is detected. The collision time can be adjusted by the following equation:

$$t' = t - \frac{d_t}{\mathbf{V}}$$

where t' is the adjusted collision time, and t is the collision time computed by Eq. 5 or Eq. 6. Fig. 4 shows the effect of the threshold distance. As seen in Fig. 4(b), the flexible object does not penetrate other object when the threshold distance is considered.

4.2 Collision response

Objects undergo the friction and dissipation when colliding. Let $\hat{\mathbf{N}}$ be a unit normal of the colliding surface, then the normal component of velocity is $\mathbf{V}_N = (\mathbf{V} \cdot \hat{\mathbf{N}})\hat{\mathbf{N}}$, and the tangential component of velocity is $\mathbf{V}_T = \mathbf{V} - \mathbf{V}_N$. The velocity after collision can be calculated as follows[Provo97]:

$$\left\{\begin{array}{l} if|\mathbf{V}_T| \geq k_f|\mathbf{V}_N|, \\ \quad \mathbf{V}' = \mathbf{V}_T - k_f|\mathbf{V}_N|\frac{\mathbf{V}_T}{|\mathbf{V}_T|} - k_d\mathbf{V}_N \\ if|\mathbf{V}_T| < k_f|\mathbf{V}_N|, \\ \quad \mathbf{V}' = -k_d\mathbf{V}_N \end{array}\right. \quad (7)$$

where k_f is a friction coefficient and k_d dissipation coefficient.

When the direction of the velocity is almost the same as that of the normal $\mathbf{N}$, the velocity after collision is almost zero and the movement is stopped. This is because Eq. 7 does not consider the rotation. For edge-edge collision, colliding edge undergoes the rotation. Let $\mathbf{P}, \mathbf{Q}$ be the points of the edge, $\mathbf{V}_P$,$\mathbf{V}_Q$ the velocities of the points, and $\mathbf{O}$ the colliding point, then torque is expressed as follows:

$$\tau = m\,\frac{\mathbf{PO}\times\mathbf{V_P}+\mathbf{QO}\times\mathbf{V}_Q}{h}$$

by using $\tau = I\frac{\omega}{h}$, the angular velocity ω can be calculated as follows:

$$\omega = \frac{\mathbf{PO}\times\mathbf{V}_P+\mathbf{QO}\times\mathbf{V}_Q}{\mathbf{PO}^2+\mathbf{QO}^2}$$

Therefore, the velocities by the rotation can be calculated as follows:

$$\begin{cases} \mathbf{V}'_P = \omega\times\mathbf{PO} \\ \mathbf{V}'_Q = \omega\times\mathbf{QO} \end{cases} \tag{8}$$

Similarly the velocity is calculated for point-triangle collision. Fig. 5 shows the effect of the rotation. As seen in Fig. 5(b), the flexible object rotates when the rotation is considered. By Eq. 7 and 8, the velocity after collision can be computed as follows:

$$\begin{cases} if|\mathbf{V}_T| \geq k_f|\mathbf{V}_N|, \\ \quad \mathbf{V}' = \mathbf{V}_T - k_f|\mathbf{V}_N|\frac{\mathbf{V}_T}{|\mathbf{V}_T|} - k_d\mathbf{V}_N + \omega\times\mathbf{r} \\ if|\mathbf{V}_T| < k_f|\mathbf{V}_N|, \\ \quad \mathbf{V}' = -k_d\mathbf{V}_N + \omega\times\mathbf{r} \end{cases}$$

where $\mathbf{r}$ is the displacement from the colliding point to the mass-point. After collision, the position is changed by altered velocity.
Self-collision is the collision of the flexible object with itself. For self-collision problem, we used Provot's method which is based on the Volino's work[Volin94, Provo97].

5 EXPERIMENTAL RESULTS

We implemented animation system for flexible and thin objects with *OpenGL* and *Open Inventor* library on *SGI Indigo*^{2} and O_2 machines with $R10000$ processor. The system was implemented with the techniques proposed in this paper. However, mass-spring has a limitation in representing the flexible object, because mass-spring model shows super-elastic effects. Inverse dynamics process for adjustment of the super-elongated spring should be considered when the mass-spring model is used. We adopted the techniques proposed by Provot [Provo95].

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

We compared the results of precomputed filtering which was proposed by Desbrun *et al*. and those of our approximation. Fig. 6 shows that the velocity changes calculated by our model is very similar to those calculated by precomputed filter. In the Fig. 6, 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 by using precomputed filter (top) and those approximated with proposed technique (middle). The velocity changes calculated by the both method seems similar in magnitude and direction. Thus, our model generates the stable motion of mass-spring model as the precomputed filter method does. Fig. 7 shows the snapshots of ribbon dance animation which was generated by using our technique.

6 CONCLUSION

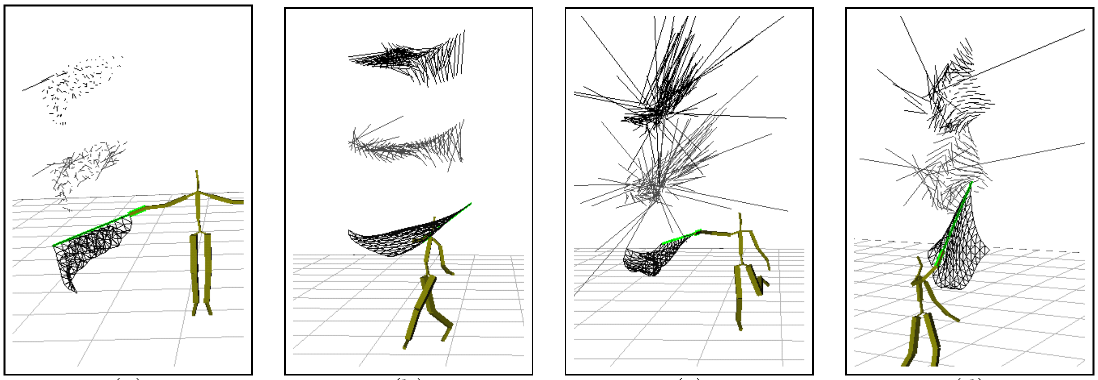
We have proposed an approximation method of implicit integration for the stable and fast animation of mass-spring model, and showed the results produced with an animation system which was implemented by using the proposed techniques. The physical correctness was not major concern of our work, but we were only interested in fast and plausible animation of flexible objects. The experimental results show that our technique produces very plausible animation of flexible objects with large steps (i.e., fast and realistic). The results show that our method can be used for interactive or real-time animation system which requires large time step.

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

Our technique can be applied to various animation system that requires interactive animation of flexible objects.

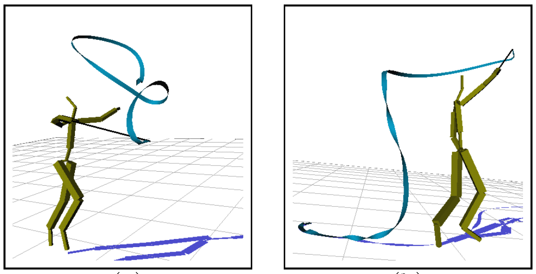
7 ACKNOWLEDGEMENT

This research was supported in part by ETRI research contract (1999).



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

Figure 6: Result comparison - Comparison between precomputed filter and 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 by using precomputed filtering (top) and those approximated with proposed technique (middle)



(a) (b)

Figure 7: Result - Two snapshots of ribbon dance animation

REFERENCES

[Baraf98] Baraff, D. Witkin, A.: *Large steps in cloth simulation, Computer Graphics(Proc. of SIGGRAPH '99),* pp. 43–52, 1998

[Carig92] Carignan, M., Yang, Y., Magnenat-Thalmann, N.: *Dressing animated synthetic actors with complex deformable clothes, Computer Graphics (Proc. of SIGGRAPH),* pp. 99–104, 1992

[Chen98] Chen, Y., Zhu, Q., Kaufman, A.: *Physically-based animation of volumetric objects, Proc. of Computer Animation '98,* pp. 154–160, 1998

[Desbr99] Desbrun, M., Schröder, P., Barr, A.: *Interactive animation of structured deformable objects, Proc. of Graphics Interface '99,* 1999

[Kass94] Kass, M.: *An Introduction to continuum dynamics for computer graphics, In SIGGRAPH Course Note, ACM SIGGRAPH,* 1994

[Nakam91] Nakamura, S.: *Initial value problems of ordinary differential equations, In Applied Numerical Methods with Software, Prentice-Hall,* pp. 289–350, 1991

[Provo95] Provot, X.: *Deformation constraints in a mass-spring model to describe rigid cloth behavior, Proc. of Graphics Interface '95,* pp. 147–154, 1995

[Provo97] Provot, X.: *Collision and self-collision handling in cloth model dedicated to design garments, Proc. of Graphics Interface '97,* pp. 177–189, 1997

[Volin94] Volino P., Magnenat-Thalmann N.: *Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity, Computer Graphics Forum (Proc. of Eurographics),* Vol. 13(3), pp. 155–166, 1994

[Volin95] Volino, P., Courchesne, M., Magnenat-Thalmann, N.: *Versatile and efficient techniques for simulating cloth and other deformable objects, Computer Graphics (Proc. of SIGGRAPH),* pp. 137–144, 1995

[Wang98] Wang, B., Wu, Z., Sun, Q., Yuen, M.: *A deformation model of thin flexible surfaces, Proc. of WSCG'98,* pp. 440–446, 1998