Bilayered Approximate Integration for Rapid and Plausible Animation of Virtual Cloth with Realistic Wrinkles

Young-Min Kang and Hwan-Gue Cho
Department of Computer Science
Pusan National University, Pusan, KOREA
{ymkang,hgcho}@pearl.cs.pusan.ac.kr

Abstract

We present an efficient method for rapid animation of mass-spring model with a large number of mass-points. Realistic cloth simulation requires large amount of time in general. This is not only because the calculation for one step requires much time but also because the cloth simulation easily tends to become unstable. Although the implicit method can make the simulation stable[1], it is still impossible to generate interactive animation when the number of mass-points is sufficiently large enough to represent realistic wrinkles. An efficient animation method proposed by Desbrun[3] also involves $O(n^2)$-sized matrix so that it cannot be applied to models with a large number of mass-points. A stable approximate method without matrix operations has been introduced[6], but its physical correctness is significantly impaired as the stiffness increases or the time-step becomes large. In this paper, we propose a bilayered approach for efficient cloth animation with a large number of mass-points. The proposed method uses two mass-spring meshes. One of them is a rough mesh for representing global motion, and the other is a fine mesh for realistic wrinkles. The experimental results show that the method can be successfully used for real-time animation of plausible cloth models.

1. Introduction

For more than a decade, cloth animation has been one of the major research topics in computer graphics. There are two main goals in generating the motion of a cloth model. One of them is the generation of realistic cloth behavior[2, 4, 10, 11], and the other is the efficiency of the computation[3, 6]. Collision resolution is also an important issue, and the collision problem in cloth animation involves special collision cases where some parts of one cloth model collide with each other. A few efficient methods have been proposed for handling the self-collision[9, 12].

Owing to the intensive research works on cloth simulation, we can generate realistic reproduction of the appearance and the movement of the cloth on the computer displays with various techniques. However, the realistic simulation of cloth requires extremely large amount of time. This is not only because the calculation of the next state requires much time but also because the numerical integration of cloth motion easily tends to become unstable[14]. In other words, we must use sufficiently small time-steps during the simulation in order to avoid the instability of the numerical integration. Therefore, real-time cloth animation with plausible appearance has been an extremely difficult problem.

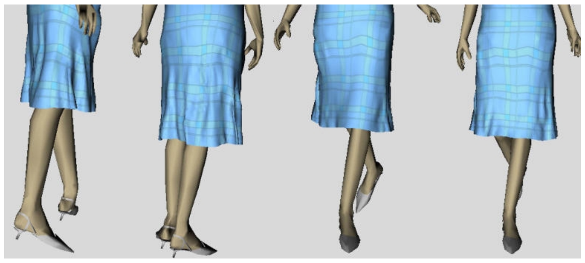

Figure 1. Interactive and Realistic Animation with the Proposed Method

Many researchers have endeavored to devise efficient methods for cloth simulation or animation, and one of the most important advances in recent years is the utilization of implicit integration with large time-steps[1]. The implicit integration is unconditionally stable during the simulation. Although the accuracy of the produced result decreases as the size of time-step increases, the stability of the method makes it possible to efficiently generate the cloth animation by using large time-steps. However, the method must solve a linear system that involves a large matrix of which size is proportional to the square of the number of mass-points. Therefore, it is impossible to generate real-time animation of cloth model even with the implicit method when the number of mass-points is sufficiently large enough to represent realistic wrinkles. In order to overcome the limitation of the implicit method, a few techniques have been proposed for more efficient animation[3, 6]. The techniques focus on the plausible animation rather than the accuracy of the motion. Although these techniques can efficiently generate cloth animation, it is still impossible to produce real-time animation of mass-spring model with sufficient number of mass-points for realistic appearance.

Desbrun proposed an efficient cloth animation technique that exploits the stability of the implicit method and improves the efficiency of the computation by approximating the matrix involved in the linear system and pre-computing the inverse matrix[3]. They then apply this approximate inverse matrix as force filter in every step in order to calculate the states of the cloth model in interactive rate. This method is more efficient than the original implicit method, but it also involves matrix with size of $O(n^2)$ where n is the number of mass-points of the mass-spring model. Therefore, it is impossible to generate real-time cloth animation when the number of mass-points is large enough to represent the appearance of the real cloth.

A stable method with direct update formula has been also introduced. The method does not involve any matrix operations so that it can calculate the next state in $O(n)$ time with arbitrary time-step[6]. However, the accuracy of this method drastically decreases when the stiffness or size of the time-step increases because it damps the motion excessively in order to maintain the stability and efficiency of the computation. Therefore, the method generates slower motion than the real motion of cloth when high stiffness or large time-steps are used[13].

Geometric approaches have been also introduced. Oshita have proposed a method that generates real-time cloth animation with sparse particles by using explicit Runge-Kutta method. The method geometrically produces the details of the cloth model[8]. However, the geometric approach cannot produce the realistic wrinkle patterns because the appearance of the cloth model is severely affected by the initial coarse mesh.

Texture mapping techniques have been also used for producing the realistic appearance of the cloth model[5]. However, the wrinkle patterns generated by this method are limited by the initial textures given by users. Therefore, it cannot produce natural wrinkles on the cloth model that undergoes various conditions.

In this paper, we propose a new animation technique that exploits the stability and efficiency of the approximate method with direct update formula and generates plausible cloth animation with realistic appearance in real-time. Fig.1 shows a real-time animation result of our method.

2. Overview of the Technique

The proposed method uses bilayered mesh structure as shown in Fig.2. The bilayered mesh structure is composed of two meshes of different numbers of mass-points. Fig.2 (a) shows the sparse mesh that represents the global feature of the cloth motion and (b) shows the dense mesh for realistic appearance of the cloth model. The sparse mesh is used for generating the global motion by considering both internal and external forces while the dense mesh simulation considers the internal forces only. Therefore, the external forces such as gravity do not influence the simulation of dense mesh.

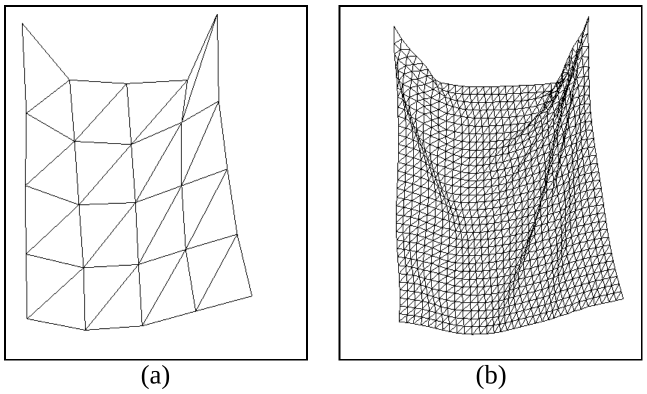

Figure 2. Bilayered structure: (a) global motion layer (b) realistic appearance layer

The method simply computes the velocity change of each mass-point of the sparse mesh with stable approximate method by considering both internal and external forces, and also computes the velocity change of each mass-point of the dense mesh with internal forces. The velocity changes of the mass-points of the sparse mesh are linearly interpolated in order to be applied to the dense mesh. By averaging the velocity change values computed in two layers of the sparse and the dense mesh, we can compute the velocity change values that will be actually applied to the dense mesh. We update the state of the dense mesh with the actual velocity change values to compute the location and velocity of each mass-point of the dense mesh. We then adjust the locations and the velocity values of the mass-points of the sparse mesh in accordance with the dense mesh.

The mass-spring model generates internal forces on mass-points when the spring edges are stretched or contracted. However, the distance between two mass-points in real cloth can be more easily decreased than a spring model, and the decreased distance generates wrinkles. In order to represent the realistic wrinkles, we enforce the coarse mesh to generate less force on mass-points when the springs are contracted in accordance with contraction resistance parameter. By specifying the parameter, users can control the contraction resistance of each layer. When the contraction resistance parameter is specified as 1, the edge between two mass-points works exactly the same as spring. If the parameter is 0, the edge does not generate any forces when it is contracted. By using this contraction resistance parameter, the proposed method can easily generate realistic appearance of the cloth model.

3. Stable and Efficient Animation

The movement of a particle $\mathbf{x}_i$ caused by a specific force $\mathbf{F}_i$ can be described as a simple motion equation, $\ddot{\mathbf{x}}_i = \mathbf{F}_i/m_i$. The mass-spring model generates forces according to the deformation of springs in the mesh. This motion equation can be numerically integrated with simple explicit integration methods such as explicit Euler method. Although the explicit method is very intuitive and easy to implement, it has a well-known instability problem. The instability can be avoided with implicit method. Therefore, we can stably animate the mass-spring based mesh structure by integrating the spring forces with the following backward Euler method[3, 7]:

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

where $\mathbf{v}_i^t$ denotes the velocity of a mass-point i, $\mathbf{F}_i^t$ is the force exerting on the mass-point i, $\mathbf{x}_i^t$ denotes the location of the mass-point i at time t, and m_i denotes the mass of the mass-points i, and h is the time step for the simulation.

For the simplicity, let us denote vector of forces, locations, and velocities ($\mathbf{F}$, $\mathbf{x}$, $\mathbf{v}$), and mass matrix $\mathbf{M}$ as follows:

$$\mathbf{F}^t = \begin{bmatrix} \mathbf{F}_1^t \\ \mathbf{F}_2^t \\ \vdots \\ \mathbf{F}_n^t \end{bmatrix} \quad \mathbf{x}^t = \begin{bmatrix} \mathbf{x}_1^t \\ \mathbf{x}_2^t \\ \vdots \\ \mathbf{x}_n^t \end{bmatrix} \quad \mathbf{v}^t = \begin{bmatrix} \mathbf{v}_1^t \\ \mathbf{v}_2^t \\ \vdots \\ \mathbf{v}_n^t \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix}$$

The update formulas shown in Eq.1 are the same as those of the explicit Euler method except that $\mathbf{F}_i^{t+h}$ is used instead of $\mathbf{F}_i^t$. This simple modification guarantees the stability of the numerical integration. However, we cannot calculate the force at the next time step so that we approximate the force at the next time-step with the first order derivative as follows:

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

where $\Delta\mathbf{x}^{t+h}$ is the location change during the time step, i.e., $\Delta\mathbf{x}^{t+h} = \mathbf{x}^{t+h} - \mathbf{x}^t$. The matrix $\partial\mathbf{F}/\partial\mathbf{x}$ is the negated Hessian matrix of the system as mentioned in [3], and will be denoted as $\mathbf{H}$ henceforth. If we denote the velocity change as $\Delta\mathbf{v}^{t+h} = \mathbf{v}^{t+h} - \mathbf{v}^t$, the location change $(\Delta\mathbf{x}^{t+h})$ can be easily rewritten as $(\mathbf{v}^t + \Delta\mathbf{v}^{t+h})h$. Therefore, the first update formula in Eq.1 can be rewritten as follows:

$$\begin{aligned} \Delta\mathbf{v}^{t+h} &= h\mathbf{M}^{-1}\mathbf{F}^{t+h} \\ &= h\mathbf{M}^{-1}(\mathbf{F}^t + \mathbf{H}\Delta\mathbf{x}^{t+h}) \\ &= h\mathbf{M}^{-1}(\mathbf{F}^t + h\mathbf{H}(\mathbf{v}^t + \Delta\mathbf{v}^{t+h})) \\ &= h\mathbf{M}^{-1}(\mathbf{F}^t + h\mathbf{H}\mathbf{v}^t) + h^2\mathbf{M}^{-1}\mathbf{H}\Delta\mathbf{v}^{t+h} \end{aligned}$$

If $\Delta\mathbf{v}^{t+h}$ is calculated, the velocity and location of each mass-point can be easily integrated with Eq.1. Therefore, cloth animation can be reduced to finding the velocity changes of mass-points, $\Delta\mathbf{v}^{t+h}$, by solving the following linear system:

$$(\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{H})\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}(\mathbf{F}^t + h\mathbf{H}\mathbf{v}^t) \tag{3}$$

In Eq.3, $h\mathbf{H}\mathbf{v}^t$ appears as an additional force. This additional force is the viscosity force as mentioned by Desbrun, and can be calculated as $(h\mathbf{H}\mathbf{v}^t)_i = h\sum_{(i,j)\in E}\kappa_{ij}(\mathbf{v}_j^t - \mathbf{v}_i^t)$ where E is the set of spring edges that links two mass-points, and κ_{ij} is the spring constant of the edge between the mass-point i and j[3]. The internal force on the mass-point i can then be described as the sum of the spring force $\mathbf{F}_i$ and the viscosity force. We will denote the sum of the spring force and the viscosity force as $\tilde{\mathbf{F}}_i$. Finally, the implicit method is reduced to linear system solving as follows:

$$(\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{H})\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\tilde{\mathbf{F}}^t \tag{4}$$

By solving the linear system in Eq.4, we can stably animate the motion of virtual cloth objects. Therefore, the simulation can be efficiently performed because the implicit method does not require a sufficiently small time interval which is the major obstacle to the real-time animation of complex cloth model.

3.1. Approximate Method for Efficiency

Although the guaranteed stability of the implicit method allows large time steps, the implicit method still suffers from heavy computational burden because the implicit method described as Eq.4 involves $O(n^2)$-sized matrix, $\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{H}$, and we must solve a large linear system when the number of mass-points increases.

For the efficient animation of dense meshes, an approximate method that updates the next state of the meshes with direct update formula is required. In order to obtain such an efficient approximate method, we adopted the Hessian approximation used in the pre-computed filter method proposed by Desbrun[3]. In the approximation of the pre-computer filter method, the i-th row and the j-th column element of the Hessian matrix $\mathbf{H}$ (i.e., H_{ij}) is considered as the stiffness of the spring that links the mass-point i and the mass-point j (κ_{ij}), and H_{ii} is the negated sum of the stiffness values of all the springs that are linked to the mass-point i ($-\sum_{j \neq i} \kappa_{ij}$). H_{ij} is 0 if the mass-point i and the mass-point j are not linked with each other[3].

By using this approximation of Hessian matrix, we have devised a simple approximate method for finding the velocity change of each mass-point. When $\mathbf{x}$, $\mathbf{v}$, and $\mathbf{F}$ are n-dimensional vectors (i.e., n mass-points are animated), and we adopt the approximation of the Hessian matrix, the linear system of Eq.4 can then be described as finding the n unknown $\Delta\mathbf{v}_i^{t+h}$ that satisfies the following n equations:

$$\frac{m_1 + h^2}{m_1} \sum_{(1,j)\in E} \kappa_{1j} \Delta\mathbf{v}_1^{t+h} + \frac{h^2}{m_1} \sum_{(1,j)\in E} \kappa_{1j} \Delta\mathbf{v}_j^{t+h} = \frac{\tilde{\mathbf{F}}_1^t h}{m_1}$$

$$\frac{m_2 + h^2}{m_2} \sum_{(2,j)\in E} \kappa_{2j} \Delta\mathbf{v}_2^{t+h} + \frac{h^2}{m_2} \sum_{(2,j)\in E} \kappa_{2j} \Delta\mathbf{v}_j^{t+h} = \frac{\tilde{\mathbf{F}}_2^t h}{m_2}$$

$$\vdots$$

$$\frac{m_n + h^2}{m_n} \sum_{(n,j)\in E} \kappa_{nj} \Delta\mathbf{v}_n^{t+h} + \frac{h^2}{m_n} \sum_{(n,j)\in E} \kappa_{nj} \Delta\mathbf{v}_j^{t+h} = \frac{\tilde{\mathbf{F}}_n^t h}{m_n}$$

The i-th equation can be simply rewritten to highlight the velocity change of a specific mass-point i as follows:

$$\Delta\mathbf{v}_i^{t+h} = \frac{\tilde{\mathbf{F}}_i^t h + h^2 \sum_{(i,j)\in E} \kappa_{ij} \Delta\mathbf{v}_j^{t+h}}{m_i + h^2 \sum_{(i,j)\in E} \kappa_{ij}} \tag{5}$$

We can notice that the velocity change of mass-point i ($\Delta\mathbf{v}_i^{t+h}$) is dependent on the unknown velocity change values of linked mass-points as they appear in the term $h^2 \sum_{(i,j)\in E} \kappa_{ij} \Delta\mathbf{v}_j^{t+h}$ in Eq.5. In order to obtain an direct update formula for the velocity change of mass-point i, we approximated the velocity change of each mass-point j linked to the mass-point i by eliminating the linked mass-points term in the update formula of $\Delta\mathbf{v}_j^{t+h}$ (i.e., $\Delta\mathbf{v}_j^{t+h} \simeq \tilde{\mathbf{F}}_j^t h / (m_j + h^2 \sum_{(j,l)\in E} \kappa_{jl})$). We can then rewrite Eq.5 in direct update form as follows:

$$\Delta\mathbf{v}_i^{t+h} = \frac{\tilde{\mathbf{F}}_i^t h + \sum_{(i,j)\in E} \frac{\kappa_{ij} \cdot \tilde{\mathbf{F}}_j^t \cdot h^3}{m_j + h^2 \sum_{(j,l)\in E} \kappa_{jl}}}{m_i + h^2 \sum_{(i,j)\in E} \kappa_{ij}} \tag{6}$$

Since $\tilde{\mathbf{F}}_j^t$ can be easily calculated by spring model, and κ_{ij}, κ_{jl}, $\sum_{(i,j)\in E} \kappa_{ij}$, and $\sum_{(j,l)\in E} \kappa_{jl}$ are constant throughout the simulation, we can efficiently calculate the velocity change of the mass-point i without linear system solving or matrix operations. The approximate method is stable enough to enable us to use very large time step h even when the stiffness of the cloth model is high. Because of the stability, it can be applied to various environments that require efficient animation of flexible objects.

4. Bilayered Model for Cloth Animation

The direct update formula in Eq.6 enables us to efficiently generate the motion of mass-spring model. However, this method over-damps the motion so that the result is not accurate enough to generate plausible motion when the stiffness is too high or the size of the time-step is too large.

Note that the direct update formula shown in Eq.6 involves $h^2 \sum_{(i,j)\in E} \kappa_{ij}$ in the denominator. The stability of this method is based on the damping. In other words, the method damps the velocity changes in order to maintain the stability as the stiffness κ_{ij} or time interval h increase, Although the damping effect caused by the term guarantees the stability, the computational results are significantly impaired when high stiffness values or large time-steps are used for simulation. Therefore, the method can generate plausible motion only for cloth model with a small number of mass-points because model with a large number of mass-points requires high stiffness values for avoiding excessive stretch. The motion of a mesh with high stiffness becomes unrealistic because the movement of each mass-point in the mesh is over-damped. The method is also incapable of generating realistic wrinkles. Therefore, the approximate method cannot be applied to the animation system where plausible motion or appearance of the cloth model is important. In this section, we propose a bilayered approach in order to improve the reality of the cloth animation.

4.1. Rapid and Plausible Cloth Animation

Bilayered integration method was applied in order to resolve the problem that the approximate method cannot rep-

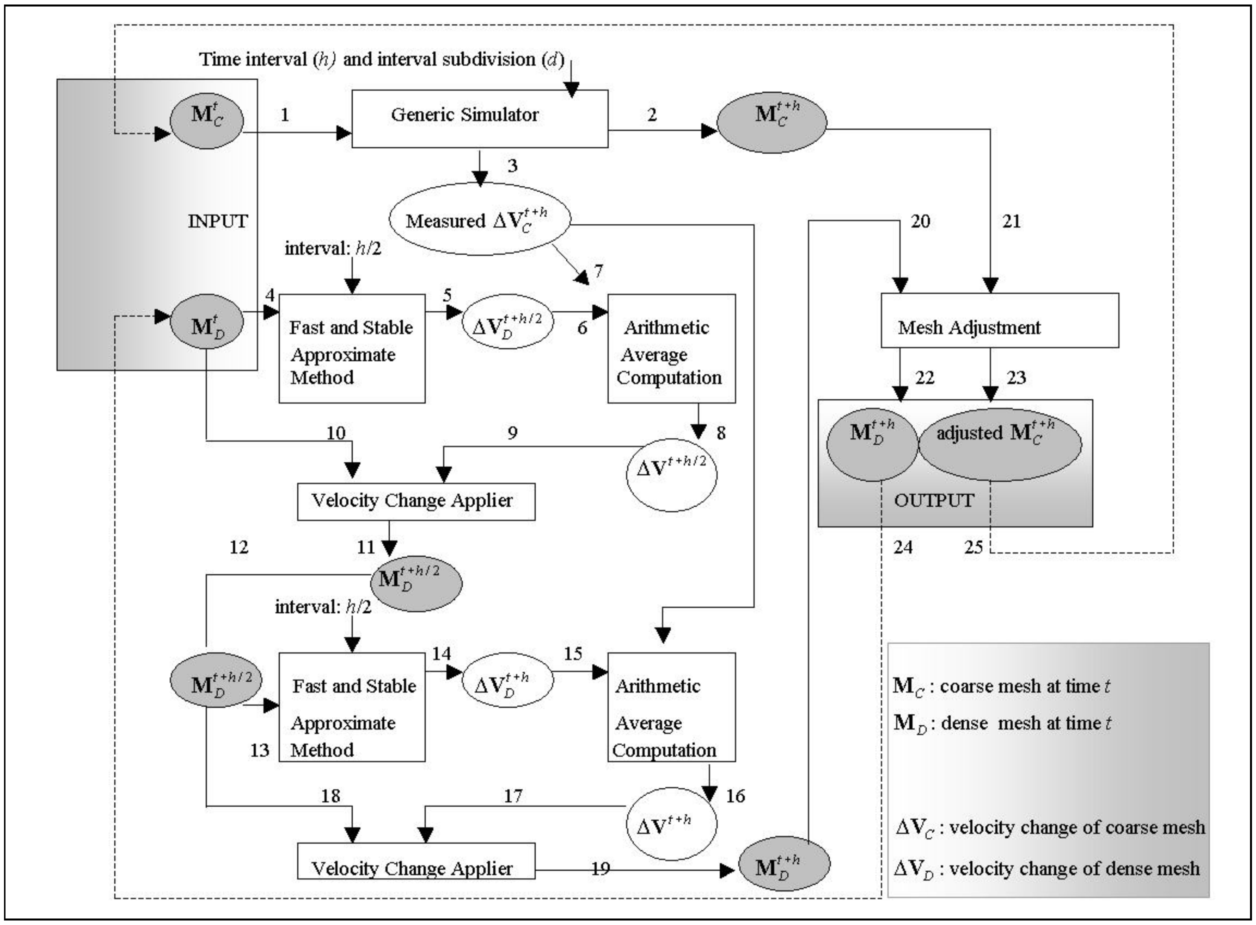
Time interval (h) and interval subdivision (d)
$\mathbf{M}_C^t$
1
Generic Simulator
2
$\mathbf{M}_C^{t+h}$
3
Measured $\Delta\mathbf{V}_C^{t+h}$
INPUT
interval: $h/2$
7
20
21
4
$\mathbf{M}_D^t$
Fast and Stable Approximate Method
5
$\Delta\mathbf{V}_D^{t+h/2}$
6
Arithmetic Average Computation
Mesh Adjustment
22
23
8
10
9
$\Delta\mathbf{V}^{t+h/2}$
$\mathbf{M}_D^{t+h}$
adjusted $\mathbf{M}_C^{t+h}$
Velocity Change Applier
OUTPUT
12
11
$\mathbf{M}_D^{t+h/2}$
24
25
interval: $h/2$
$\mathbf{M}_D^{t+h/2}$
Fast and Stable Approximate Method
14
$\Delta\mathbf{V}_D^{t+h}$
15
Arithmetic Average Computation
13
$\mathbf{M}_C$: coarse mesh at time t
$\mathbf{M}_D$: dense mesh at time t
18
17
16
$\Delta\mathbf{V}^{t+h}$
Velocity Change Applier
19
$\mathbf{M}_D^{t+h}$
$\Delta\mathbf{V}_C$: velocity change of coarse mesh
$\Delta\mathbf{V}_D$: velocity change of dense mesh

Figure 3. Flow of state-update module for bilayered animation

resent the realistic motion and appearance of a finely discretized dense mesh. The method generates the cloth animation by constructing a bilayered structure that is composed of two levels of meshes, a coarse mesh with a small number of mass-points and a dense mesh with a large number of mass-points for detailed wrinkles. By enforcing the motions of the dense mesh to follow the motion of the coarse mesh, the finely discretized mesh can be plausibly animated because the numerical errors of the coarse mesh animation with the approximate method are small enough to represent plausible motion.

Fig.3 shows the overall flow of the state-update module for the bilayered animation. As shown in the flow diagram, the coarse mesh ($\mathbf{M}_C$) is simulated with the consideration of the gravity and internal spring forces. The coarse mesh can be animated with Eq.6 or any other simulation methods.

After the computation of the motion of the coarse mesh, The stable and efficient approximate method is applied to the dense mesh ($\mathbf{M}_D$) in order to compute the velocity change during the first half of the frame interval. For the dense mesh animation, we consider internal forces only. Therefore, external forces such as gravity are not considered in this process. We can easily measure the velocity change of the coarse mesh during the frame interval, and we interpolate the velocity change data in order to make it applicable to the dense mesh. The velocity change data of the coarse mesh and the dense mesh are averaged to produce an actual velocity change data for the dense mesh. By using this velocity change data, we generate the intermediate state of the dense mesh. By applying the same process again to the dense mesh for the remaining half of the frame interval, we can produce the final state of the dense mesh at the end of the frame. After these processes, the geometric and kinetic information of the coarse mesh and the dense mesh do not correspond to each other. Therefore, we adjust the state of the coarse mesh according to the state of the dense mesh. After this post processing, the final state of the coarse and the dense meshes are obtained.

The state-update process can be described as shown in the Table 1. The pseudo-code shown in the table is the state-update module for computing the next state of the mesh. The module takes the current state of the coarse mesh ($\mathbf{M}_C^t$)

and the dense mesh ($\mathbf{M}_D^t$) of bilayered structure as input arguments. The output arguments of the module are the state of the meshes at the next step of the simulation, ($\mathbf{M}_C^{t+h}$ and $\mathbf{M}_D^{t+h}$).

```
Module: NEXTSTATE
Input : $\mathbf{M}_C^t$, $\mathbf{M}_D^t$: current mesh
Input : $h$: frame interval, $t$: current time
Output: $\mathbf{M}_C^{t+h}$, $\mathbf{M}_D^{t+h}$

$\mathbf{V}_t \leftarrow$ get velocities of $\mathbf{M}_C$;
// frame interval is divided into $d$ subintervals
for( $i$=0 to $d$-1) [
    $t_{start} \leftarrow t + i \cdot (h/d)$;
    $t_{end} \leftarrow t_{start} + h/d$;
    simulate coarse mesh from $t_{start}$ to $t_{end}$ with $\mathbf{F}_{int}$+$\mathbf{F}_{ext}$
]
$\mathbf{V}_{t+h} \leftarrow$ get average velocity of $\mathbf{M}_C$ from $t$ to $t+h$
$\Delta\mathbf{V}_C \leftarrow \mathbf{V}_{t+h} - \mathbf{V}_t$;

$\mathbf{F}_{int}^t \leftarrow$ compute internal force ($\mathbf{M}_D^t$)
$\Delta\mathbf{V}_D \leftarrow$velocity change of dense mesh ($t$, $t+\frac{h}{2}$, $\mathbf{F}_{int}^t$)
$\Delta\mathbf{V} \leftarrow$ average of ($\Delta\mathbf{V}_C$,$\Delta\mathbf{V}_D$);
$\mathbf{M}_D^{t+h/2} \leftarrow$ Apply $\Delta\mathbf{V}$ to $\mathbf{M}_D^t$;

$\mathbf{F}_{int}^{t+h/2} \leftarrow$ compute internal force ($\mathbf{M}_D^{t+h/2}$)
$\Delta\mathbf{V}_D \leftarrow$velocity change of dense mesh ($t+\frac{h}{2}$, $t+h$, $\mathbf{F}_{int}^{t+\frac{h}{2}}$)
$\Delta\mathbf{V} \leftarrow$ average of ($\Delta\mathbf{V}_C$,$\Delta\mathbf{V}_D$);
$\mathbf{M}_D^{t+h} \leftarrow$ Apply $\Delta\mathbf{V}$ to $\mathbf{M}_D^{t+h/2}$;

Adjust $\mathbf{M}_C^{t+h}$ in accordance with $\mathbf{M}_D^{t+h}$;
```

Table 1. Pseudo-code of state-update module for bilayered animation

The module computes the velocity change of the coarse mesh with internal and external forces. When we compute the motion of the coarse mesh, we can generate more accurate motion by dividing one frame into a number of small intervals because simulating the coarse mesh does not require much time. After we compute the motion of the coarse mesh, the method applies the fast and stable integration method to the dense mesh, and generates the intermediate state and the final state as described before. After this update process of the dense mesh, the adjustment of the coarse mesh finishes the state-update module.

Because the proposed method exploits the stability of the implicit method by damping the velocity change when the internal forces increase, we can generate the cloth animation efficiently by using large time-steps. For the real-time cloth animation, we used the time-step of which interval is 1/30 seconds, and found no instability problems even with extremely high stiffness values. Moreover, we can employ explicit methods for simulating the coarse mesh because the coarse mesh can be animated in real-time even with the explicit methods such as Runge-Kutta method.

Fig.4 shows the comparison between the results of the bilayered approach and dense mesh simulation. Fig.4 (a) is the result of the bilayered method, and (b) shows the result when the dense mesh is animated only with the approximate implicit method. As seen in the figures, the approximate implicit method cannot produce the realistic motion and appearance when it is applied to the dense mesh. On the other hand, the bilayered approach generates plausible animation because the dense mesh follows the global motion of the coarse mesh.

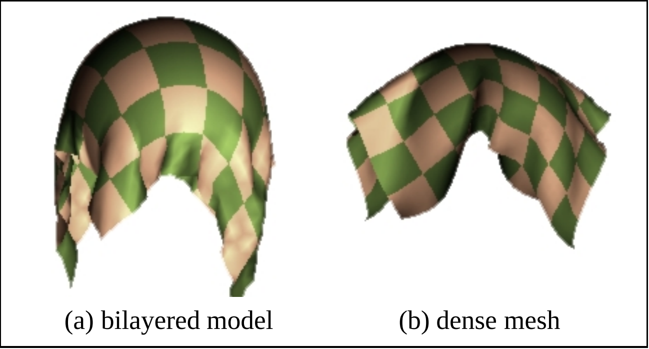

Figure 4. Comparison between bilayered animation and dense mesh animation

4.2. Wrinkle Control

The generation of the realistic wrinkles and the efficient control over the wrinkles are desirable features of the cloth animation method. When two mass-points of the cloth mesh get closer, the real cloth produces wrinkles between the mass-points. The proposed method represents the wrinkle generation mechanism by simply applying different contraction resistance parameters to the sparse mesh and the dense mesh. Moreover, these contraction resistance parameters enable us to control the wrinkle pattern efficiently.

In order to generate realistic cloth appearance, the dense mesh must produce the curved wrinkles between the two mass-points of the coarse mesh as the distance between the mass-points is decreased. By considering this fact, we can notice that the mass-points in the coarse mesh must be allowed to easily approach to other linked mass-points and the linked mass-points in the dense mesh must be kept to maintain the original spring length. For the realistic wrinkle generation, we do not use the spring force formula when the spring is contracted, but the spring forces are multiplied by the contraction resistance parameter. As a consequence, the edge works as normal spring if the resistance parameter is 1, and the edge exerts no force to the mass-points when it is contracted and the resistance parameter is specified as 0.

Therefore, the spring force $\mathbf{F}_i^{(i,j)}$ on the mass-point i exerted by the spring between the mass-point i and j can be represented as follows:

$$\mathbf{F}_i^{(i,j)} = C \cdot \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|}$$

$$C = 1 \quad if\ |\mathbf{x}_j - \mathbf{x}_i| \geq l_{ij}^0$$

$$C = c \quad if\ |\mathbf{x}_j - \mathbf{x}_i| < l_{ij}^0$$

where $\mathbf{x}_i$ denotes the location of the mass-point i, l_{ij}^0 denotes the rest length of the spring edge that links the mass-point i and the mass-point j, κ_{ij} is the stiffness of the spring, and c is user-defined contraction resistance parameter between 0 and 1.

It is obvious that a small contraction resistance must be used for the sparse mesh and a large value for the dense mesh. By applying different parameter values to the different layers, we can obtain the realistic appearance of cloth. If we want to generate very wrinkly cloth model, we can simply set the small contraction resistance value for the coarse mesh, and large value for the dense mesh. The small contraction resistance for coarse mesh will enable the linked mass-points of the mesh to easily get closer, and the large resistance for the dense mesh enforce the mesh generate the wrinkles between the mass-points of the coarse mesh. We can also generate a rubbery model by setting large contraction resistance values for both layers. Therefore, we can efficiently control the wrinkles of the cloth model.

5. Experimental Results

We implemented our cloth animation technique on SGI Octane machine with R12000 CPU and 512 Mb main memory, and PC environments were also used for testing the proposed method. Regardless of the computing environments, the proposed method generated real-time animation of cloth model with thousands of triangles. The proposed method was also implemented as a library for increasing the reality of existing cloth animation system. The experimental results show that the method can be successfully integrated with existing systems which employs various simulation methods.

The proposed method was stable enough to use large time-steps, and the results were plausible because the method avoids over-damping effects by employing the bilayered structure. The size of the time-step did not affect the stability of the animation, and the increased number of the mass-points decreased the performance of the system linearly. Therefore, we could apply the proposed method to realistic cloth animation in real-time environments by using time-step of which size is 1/30 sec.

Fig.5 shows the results when various contraction resistance parameters are applied to the dense and the sparse meshes in order to control the wrinkle patterns on the cloth model. Fig.5 (a) shows the result when the contraction resistance parameter of 0.8 is applied to the sparse mesh and 1.0 for the dense mesh, (b) with 0.4 for the sparse mesh and 1.0 for the dense mesh, and (c) with 0.1 for the sparse mesh and 1.0 for the dense mesh. As shown in the results, we can efficiently control the wrinkle patterns with the contraction resistance parameters in order to generate rubbery model like Fig.5 (a), model with moderate wrinkles as shown in (b), and wrinkly model like (c).

Although the quality of the result is affected by the size of the time-step used in the simulation, the proposed method can produce very plausible animation with relatively large time-step as shown in Fig.6. The result shown in Fig.6 was generated in interactive rates, and the animation of realistic cloth model with a large number of mass-points was efficiently generated in interactive rate. 4,225 mass-points and 12,416 spring edges are used for the result shown in Fig.6.

Fig.7 also shows the ability of the proposed method in reproducing the realistic appearance of the cloth model with small amount of computations. The cloth models shown in Fig.7 were enforced to drape on a hemisphere and a box. Fig.7 (a) is the result when two points are fixed, and (b) is the result with no mass-point fixed. As shown in the figures, the proposed method generates very plausible animation.

Fig.8 shows the animation results when the proposed method is applied to the interactive animation with dressed models. The results show that the proposed method can be successfully applied to interactive cloth animation for VR environments or computer game applications. Moreover, we have used heterogeneous simulation methods for each layers in the proposed method in order to generate the results shown in Fig.8. The normal implicit Euler method was used for the rough mesh animation, and the stable and efficient approximate method was used for the fine mesh which describes the detailed appearance of the cloth model. Therefore, the results shown in Fig.8 demonstrates that the proposed bilayered method can be combined to various cloth animation system in order to efficiently increase the reality of the result.

6. Conclusion

We proposed an efficient and stable animation method for mass-spring model with a large number of mass-points

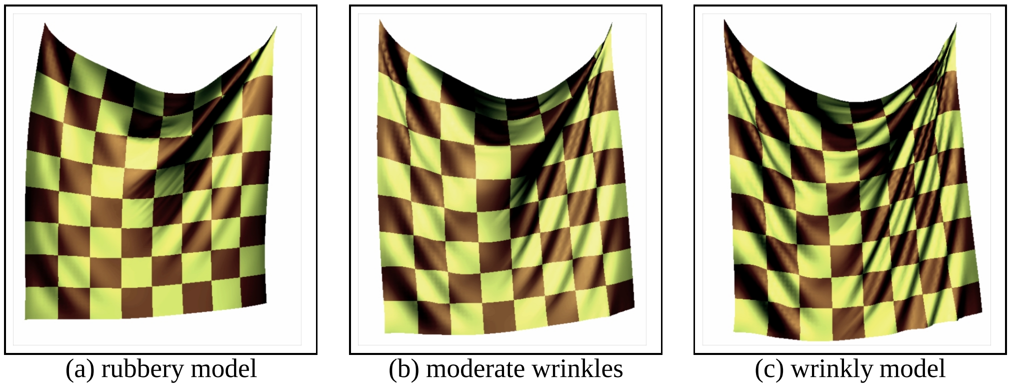

Figure 5. Wrinkle generation with different contraction resistance parameters, C_a for the sparse mesh and C_b for the dense mesh: (a) C_a=0.8, C_b=1.0 (b) C_a=0.4, C_b=1.0 (c) C_a=0.1, C_b=1.0

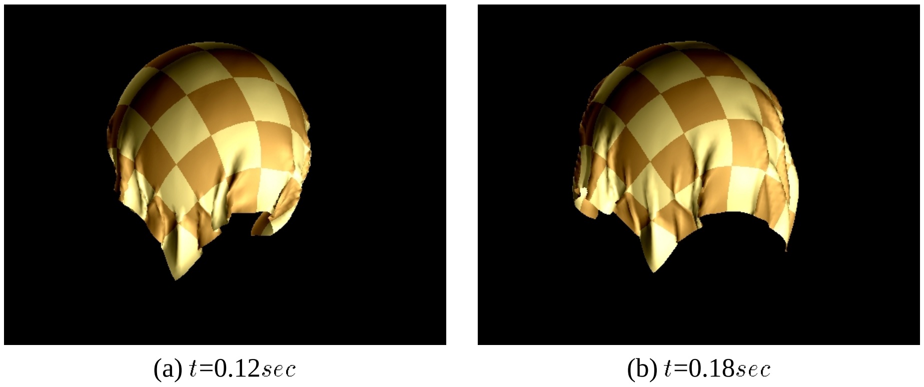

Figure 6. Drape animation result with 4225 mass-points and 12416 spring edges

for rapid and realistic cloth animation. In order to obtain the efficiency, the proposed method approximates the implicit method with direct update formula that maintains the stability of the implicit method and requires less computation. However, the approximate method produces over-damped motion of mass-spring model in order to keep the system stable when the stiffness values of the spring edges or the sizes of the time-steps increase. Therefore, the method generates unrealistic motion when it is applied to the stiff mass-spring model with a large number of mass-points in real-time environments.

In order to overcome this problem, the proposed method employs two layers for the animation. One of the layers represents the global feature of the motion of the cloth by considering both internal and external forces, and the other layer is composed of numerous mass-points to represent realistic appearance, and is animated with the consideration of the internal forces only. By using this bilayered cloth model, we could efficiently generate plausible cloth animation with realistic wrinkles.

The proposed method is suitable for the real-time animation system where many mass-points are used for representing realistic cloth appearance. The method can be applied to any cloth animation environment for efficient generation of plausible animation. Because the proposed method works with arbitrary time interval and generates realistic appearance, it can be used for implementing robust systems for interactive cloth animation such as VR systems and interactive games. Moreover, the proposed method can be used not only for the real-time system but also for the off-line simulation by using sufficiently small time-steps.

7 Acknowledgment

This research was supported in part by the Ministry of Information and Communication of Korea: The Support Project for University Foundation Research 2002, supervised by Institute of Information Technology Assessment (IITA). Authors also appreciate Mr. Frédéric Cordier in MIRALab for his help on testing the proposed method with dressed virtual models.

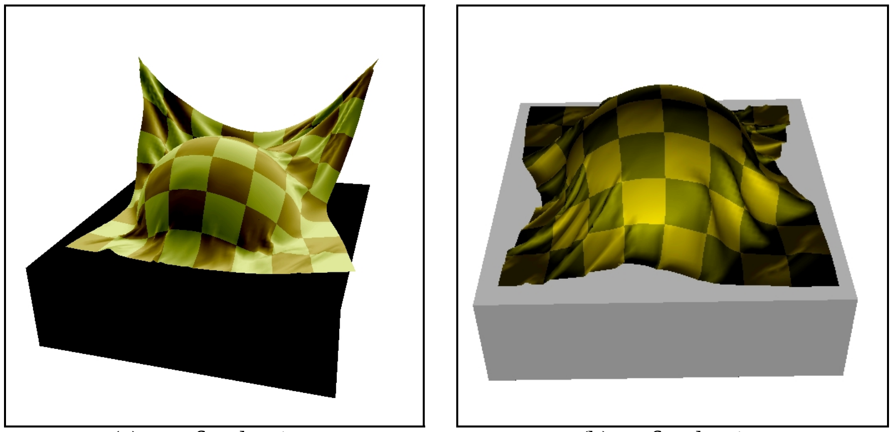
(a) two fixed points (b) no fixed point

Figure 7. Drape on hemisphere and box, 4225 mass-points, and 12416 spring edges

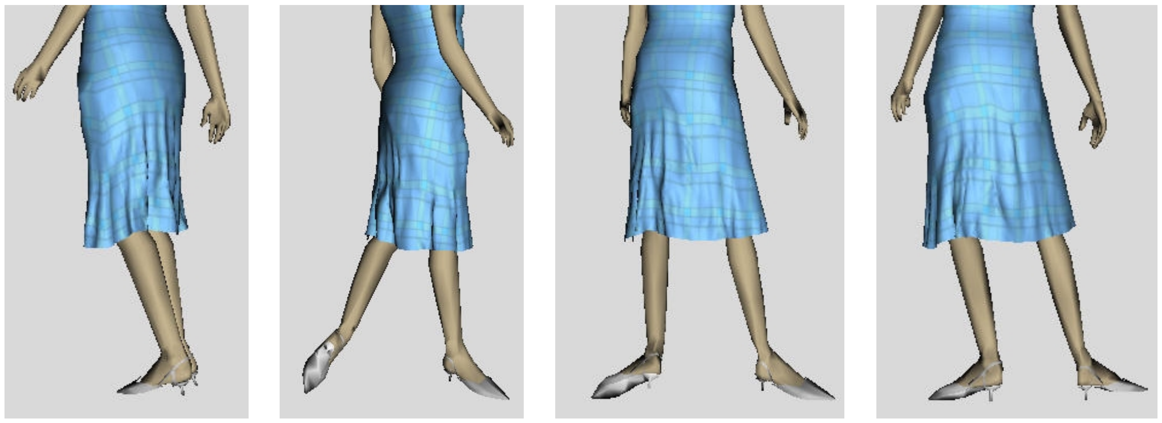

Figure 8. Interactive animation with dressed character model

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[2] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. *Proceedings of SIGGRAPH '94*, pages 365–372, July 1994.

[3] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pages 1–8, June 1999.

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

[5] S. Hadap, E. Bangarter, P. Volino, and N. Magnenat-Thalmann. Animating wrinkles on clothes. *IEEE Visualization '99*, pages 175–182, October 1999.

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

[7] M. Kass. An introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Note: Physically-based Modelling*. ACM SIGGRAPH, 1995.

[8] M. Oshita and A. Makinouchi. Real-time cloth simulation with sparse particles and curved faces. *Proc. of Computer Animation 2001*, pages 220–227, November 2001.

[9] X. Provot. Collision and self-collision handling in cloth model dedicated to design. *Computer Animation and Simulation '97*, pages 177–190, September 1997.

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

[11] P. Volino, M. Courshesnes, and N. M. Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Proceedings of SIGGRAPH 95*, pages 137–144, August 1995.

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

[13] P. Volino and N. M. Thalmann. Comparing efficiency of integration methods for cloth simulation. *Proc. of Computer Graphics International 2001*, pages 265–272, June 2001.

[14] A. Witkin and D. Baraff. Differential equation basics. In *SIGGRAPH Course Note: Physically-based Modelling*. ACM SIGGRAPH, 1994.