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

Abstract

People are familiar with the actual movement of real objects in everyday life so that a plausible computer animation requires high degree of physical correctness. Therefore, physically simulation should be employed when the generated animation is supposed to reproduce the real world in virtual environments. However, the simulation of a non-rigid object is much more time-consuming than that of a rigid object because it incurs a special numerical problem known as instability problem. Although the implicit method can make the simulation stable, it is still impossible to generate interactive animation when the geometric model of the virtual cloth is complex enough to represent realistic details. In this paper, efficient animation techniques are proposed for the real-time animation of complex deformable objects. The proposed method exploits the stability of the implicit integration in order to use sufficiently large time steps for real-time environments, and efficiently computes the next states of the cloth model. Dissimilar to the previous methods for real-time cloth animation, any severe simplification on the problem itself has been avoided in order to produce realistic motions of complex models. The method can be successfully integrated into virtual reality systems in order to increase the realism of virtual environments without violating the interactiveness of the system.

1 Introduction

The objects in computer graphics literature can be classified into two major categories: rigid objects and non-rigid objects. Regardless of the type of an object, physical simulation generates the motion of the object based on the dynamics. However, the animation of a non-rigid object is much more time-consuming than that of a rigid object because it incurs a special numerical problem known as instability problem. The instability problem requires the system to decrease the size of the simulation time step. As a consequence, it is still impossible to generate interactive animation when the geometric model of the virtual cloth is complex enough to represent realistic details. In this paper, efficient animation techniques are proposed for the real-time animation of complex deformable objects as shown in Fig. 1.

Cloth is a vital example of the non-rigid deformable objects in the real world. Since one of the most important goals of computer graphics is to represent the real objects on the computer display, various efforts have been made to efficiently generate the realistic appearance and motion of cloth [21, 28, 23].

In order to represent cloth, various kinds of methods such as finite elements [6], continuum mechanics [25] and particle system [4, 23] can be used. In this paper, the particle system is used for the cloth model because it compromises the conflicts between the accuracy and the efficiency. When the particle system is applied, a virtual cloth is represented with particles and links. The movement of each particle is determined by the internal forces exterted on the particle, and the force can be computed by considering the distances between the particles connected by the links.

2 Previous Work

Many researchers have proposed various methods for reproducing the accurate behavior of cloth model, and it is possible to reproduce the realistic appearance and the movement of the cloth on the

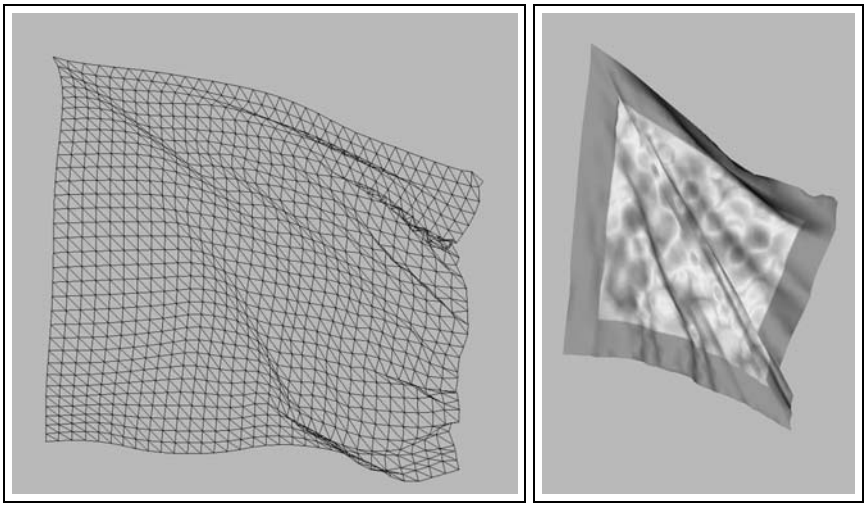


Figure 1. Complex virtual cloth models to be animated in real-time

computer displays with the noble contributions by various researchers [4, 5, 10, 25, 24, 26, 28, 29].

2.1 Geometric Approaches

In the early stage of cloth modeling in computer graphics, geometric approaches have been widely employed because physically-based correct simulation required too heavy computational cost for the hardware environments at that time. The approaches tried to model the geometric properties of the cloth such as folds, creases, and overall appearnce with geometrical equations [20].

The work by Weil [30] is regarded as the first attempt to geometrically represent the cloth objects[20]. The method generated the draped appearance of the cloth object by using multiple catenary curves. Therefore, the method can be applicable only to the hanged and draped cloth. Several research groups have proposed their own geometric approaches to the representation of the cloth model. Agui $et\ al.$ proposed a geometric approach to represent sleeves [1]. Hinds $et\ al.$ also proposed a geometric approach that generates the cloth model with geometric surfaces[13]. Ng $et\ al.$ proposed a geometric method which represents the cloth model with a series of cross sections composed of two layers, and they generated folds of the cloth model with sinusoidal curves [19]. However, the geometric methods do not

guarantee the plausibility of the result since they do not consider any physical properties of actual cloth. Moreover, each geometric method focuses on a specific condition of the cloth model so that flexible and dynamic motion cannot be achieved.

2.2 Physically-based Modeling

Feynman's model for generating the appearance of cloth is one of the earlist works that represents the cloth model on the physical basis [11]. The method was devised to find the final equilibrium state of the cloth model by minimizing the energy of the model. Since the method was devised to find the static equilibrium state of the cloth, it could be applied only to draped cloth models.

Terzopoulos $et\ al.$[25] characterized the cloth simulation as a deformable surface problem, and their approach employed physically-based modeling in order to generate the physically plausible animation [25]. Their method represents a cloth models as an elastic object, and its motion is described by a differential equation that determines the internal forces. The work by Terzopoulos $et\ al.$ provided a fundamental basis for the physically-based modeling for the animation of deformable objects, and their idea has been followed by various research groups. Although their method presented a new paradigm for the physically plausible animation of deformable objects, the method does not focus on the computational efficiency of the simulation process. Therefore, the method suffers from the instability problem that causes undesirable declination of the overall performance.

Breen $et\ al.$ proposed particle based method for generating the realistic appearance of the draped cloth. The method represents a cloth object as an interacting particle system [4]. The method was devised to achieve a restricted goal: generating the final equilibrium state of specific materials. The method employed energy minimization process to find the equilibrium position. The serious disadvantage of this method is that it cannot produce dynamically changing intermediate scenes between the initial state and the final equilibrium.

After the outstanding contributions, various methods has been proposed to represent the cloth model

more correctly [5, 28, 10]. Thalmann group has proposed various techniques for generating cloth on human character models [5, 28]. However, those methods used explicit integration schemes so that real-time or interactive animation of cloth model has been almost impossible until recent years.

2.3 Efficient Simulation with Implicit Integration

Many researchers have endeavored to devise efficient methods for cloth simulation, and one of the most important advances in recent years is the use of implicit integration with large steps [2]. The method could use arbitrarily large time steps during the simulation. Therefore, the method was much more efficient than any other previous physical simulation methods when the stiffness of the model is high enough to represent realistic cloth. However, the implicit integration method requires the solution of a linear system, which involves a large matrix with a size proportional to the square of the size of the input [2, 9, 18]. Therefore, it is impossible to generate real-time animation of cloth model even with the implicit method when the cloth model is finely discretized for the realistic appearance.

Some researchers have tried to devise efficient methods that can interactively manipulate the deformable objects in virtual environments, and a few methods have been proposed to surmount the limitation of the implicit method [9, 17, 15]. Desbrun $et\ al.$ proposed an efficient technique that approximates the matrix involved in the linear system as a constant matrix and precomputes the inverse matrix [9]. They applied the precomputed inverse matrix as a force filter at every time-step in order to update the states of the cloth model at interactive rates. Although this method is more efficient than the original implicit method, it also involves O(n^2)-sized matrix, and the precomputed inverse matrix is usually a dense matrix. Therefore, their method can be used only for moderately complex mesh structures.

A stable method that approximates the solution with a direct update formula has been also introduced. No matrix operations are involved in the method so that the method can update the state of the deformable object models in O(n) time with arbitrarily large time steps [15]. The damping effect of this method increases as the stiffness of the cloth model increases. Therefore, this method can be also applied only

to a moderately complex mesh.

The previous techniques for real-time animation sacrificed the accuracy in order to achieve real-time or interactive performance. Therefore, the physical plausibility of animation result cannot be guaranteed with those methods.

2.4 Hybrid Methods for Real-time Animation

The heavy computational cost of cloth simulation made researchers to devise various hybrid methods. Oshita and Makinouchi have proposed a geometric approach that generates wrinkly details on cloth models composed of sparse particles [22]. The method produces smooth cloth-like surfaces based on PN triangulation proposed in [27]. Texture mapping techniques have been also used for producing the realistic appearance of the cloth model[12]. However, the geometric or texture-based approaches cannot produce the realistic appearance of cloth because the geometric interpolation of the initial coarse mesh does not take the physical correctness into account at all, and the wrinkle patterns generated by textures are limited by the initial textures given by users.

Recently, Cordier and Magnenat-Thalmann proposed real-time animation techniques for fully dressed virtual human [8]. The fundamental idea of their method is the employment of different methods for different parts of the dress. However, the method uses conventional simulation method for the floating cloth like long skirt, and the animation of the floating cloth is the bottleneck of the performance.

Kang and Cho proposed an efficient method for rapid animation of mass-spring model 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 [14]. The method is efficient enough to be used for interactive applications such as computer games or VR environments. However, the bilayered structure is not intuitive and easy for implementation, and the adjustment of the location and the velocity in the method cannot be justified with any reasonable physical foundation.

3 Problem Formulation

One of the major difficulties in cloth animation lies in the stability of the system. As described in the review on the previous methods, the stability problem can be solved with implicit methods. Fortunately, the matrix involved in the implicit integration scheme for cloth animation is in general so sparse that the linear system can be efficiently solved with iterative methods. However, the exact solution still requires heavy computations when the cloth model is complex enough to represent the realistic details.

The stability of the animation process is an essential property of the real-time environments. Since the purpose of this paper is to propose a real-time cloth animation method, the problem should be formulated with an implicit integration scheme. As a consequence, the problem is transformed to a linear system problem, and the solution of the linear system produces stable animation result.

3.1 Physical Model for Cloth Behavior

In this paper, a mass-spring based cloth model is employed because it is the most intuitive and simple model for efficient animation of cloth. The mass-spring model assumes that an object is composed of mass-points and spring edges. Each spring connects two mass-points, and the structure of the spring connection can be arbitrarily constructed.

The forces caused by the springs attached to each mass-point determine the motion of the mass-point. The spring force can be easily computed with Hooke's law, and the movement of a particle can be described with Newton's second law. After the model is constructed and internal forces are computed, the cloth animation can be regarded as a numerical integration problem that integrates the force for computing the new velocity and finally integrates the new velocity for finding the new position of each mass-point. This mass-spring model is very intuitive approach to the representation of soft objects such as cloth.

Let $\mathbf{f}_i^j$ denote the force on the mass-point i caused by the spring between the mass-points i and j. If

the spring force is simply modeled with Hooke's law and κ_{ij} denotes the spring constant of the spring. $\mathbf{f}_i^j$ can be described as follows:

$$\mathbf{f}_i^j = \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{1}$$

where $\mathbf{x}_i$ denotes the location of i-th mass-point, and l_{ij}^0 the rest length of spring between i-th and j-th mass-point.

The total spring force on the i-th mass-point is then the sum of all the spring forces caused by all the springs linked to the mass-point, and the internal force $\mathbf{f}_i$ on the i-th mass-point can be easily calculated as follows:

$$\mathbf{f}_i = \sum_{(i,j)\in E} \mathbf{f}_i^j = \sum_{(i,j)\in E} \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{2}$$

where E is a set of edges which correspond springs between mass-points.

Real-time animation of simple cloth model is no more a hard problem because of the rapid improvement of hardwares and software techniques. However, the sufficiently complex models for wrinkly details cannot be interactively animated even with the current state-of-art techniques. Therefore, the geometric complexity of a cloth model in interactive animation systems is often reduced as low as possible, and the realistic details of the actual cloth can hardly be represented in the real-time systems. The proposed techniques in this paper is supposed to generate realistic animation in real-time.

3.2 Integration Scheme for Stable Animation

A straight-forward approach to the numerical integration of the spring force $\mathbf{f}_i$ is to use explicit Euler integration scheme as follows:

$$\begin{pmatrix} \mathbf{v}_i^{t+h} \\ \mathbf{x}_i^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i^t + \frac{h}{m}\mathbf{f}_i^t \\ \mathbf{x}_i^t + h\mathbf{v}_i^{t+h} \end{pmatrix} \tag{3}$$

where $\mathbf{v}_i^t$ denotes the velocity of i-th mass-point at time t, and $\mathbf{f}_i^t$ the force on the mass-point at time t. Similarly, $\mathbf{x}_i^t$ denotes the location of i-th mass-point at time t, and h denotes time interval between simulation steps.

This simple method enables system to locate any mass-point of a simulated cloth at the next time step $t + h$ with the given information of $\mathbf{x}_i^t$, $\mathbf{v}_i^t$ and the computed internal force $\mathbf{f}_i^t$. The internal force on the mass-point i can be easily computed with Eq. 2. However, the explicit integration cannot be applied to the real-time cloth animation system because stability of the system requires the step size to be decreased down to an sufficiently small value. The sufficiently small time step is often intolerable for the real-time environments. In other words, the instability problem disables us to employ a simple explicit integration method because it takes too much time to produce an animation result of mass-spring model [18, 16, 9]. Therefore, a stable implicit integration scheme should be employed if the efficiency of the computational performance is required.

The backward Euler method is an implicit integration scheme that guarantees the stability of the numerical integration process. Therefore, the mass-spring based objects can be stably animated by integrating the spring forces with the following backward Euler method [9, 16]:

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix} \tag{4}$$

where h denotes the time interval, and $\mathbf{v}$ denotes the vector of velocity values of the mass-points. Similarly, $\mathbf{f}$ and $\mathbf{x}$ are the vectors of forces and locations of the mass-points respectively. The matrix $\mathbf{M}$ is the mass matrix. The superscripts t and $t + h$ denote time, and semantically mean the current state and the next state respectively. The vectors ($\mathbf{v}$, $\mathbf{f}$, and $\mathbf{x}$) and the matrix $\mathbf{M}$ can be described as follows:

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n]^{\mathbf{T}}, \quad \mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^{\mathbf{T}}, \quad \mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]^{\mathbf{T}}$$

$$\mathbf{M}_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{M}_n \end{bmatrix} \tag{5}$$

The objective of computation is to find $\mathbf{x}^{t+h}$ (i.e., the new positions of the mass-points at the next step), and the vector $\mathbf{x}^{t+h}$ can be easily determined when the next velocities of the mass-points, $\mathbf{v}^{t+h}$, is known. The next velocity vector can be computed if we find the velocity change vector $\Delta\mathbf{v}^{t+h}$ that denotes $\mathbf{v}^{t+h} - \mathbf{v}^t$. Therefore, the cloth animation with Eq. 4 is eventually reduced to finding $\Delta\mathbf{v}^{t+h}$ as follows:

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t+h} \tag{6}$$

Although the mass-spring based animation can be described as a simple equation like Eq. 6, the real-time animation is not an easy problem. The difficulty arises from the unknown force vector at time $t+h$. Since Hooke's law can compute only $\mathbf{f}^t$, the force at the next time step should be approximated. Because the force at the current state can be easily computed, the force at the next time step can be approximated by applying Taylor expansion as follows:

$$\mathbf{f}^{t+h} = \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x}^{t+h} = \mathbf{f}^t + \mathbf{J}\Delta\mathbf{x}^{t+h} \tag{7}$$

where $\mathbf{J}$ denotes the Jacobian matrix of the force vector with respect to the position vector, and $\Delta\mathbf{x}^{t+h}$ is the positional change from time t to $t+h$. The size of the matrix $\mathbf{J}$ is $n \times n$, and each component

of $\mathbf{J}$ is a 3×3 sub-matrix. As shown in Eq. 1, the force does not depend on the velocity so that partial derivative with respect to the velocity does not need to be considered.

Because the positional change $\Delta \mathbf{x}^{t+h}$ can be rewritten as $(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})h$, the force at the next state can be also expressed as follows:

$$\mathbf{f}^{t+h} = \mathbf{f}^t + \mathbf{J}(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})h \tag{8}$$

With the force approximation shown in Eq. 8, the animation of the mass-spring model is finally reduced to a linear system solving as follows:

$$\Delta \mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^t + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^t + h^2\mathbf{M}^{-1}\mathbf{J}\Delta \mathbf{v}^{t+h} \tag{9}$$

The only unknown vector in this linear system is $\Delta \mathbf{v}^{t+h}$, and other matrices and vectors can be computed without solving the linear system. Rearrangement of Eq. 9 yields the linear system in the standard form of $\mathbf{A}\mathrm{x} = \mathrm{b}$ as follows:

$$(\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{J})\Delta \mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^t + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^t \tag{10}$$

By multiplying the mass-matrix $\mathbf{M}$ to the both side, the problem can be finally expressed as follows:

$$(\mathbf{M} - h^2\mathbf{J})\Delta \mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t \tag{11}$$

The stable animation of the mass-spring model can be achieved by solving the linear system shown in Eq. 11. However, the stability is not a complete solution to the real-time animation. In order to guarantee realistic virtual environments, sufficiently large number of mass-points must be used and the size of the linear system becomes too large to be solved in real-time.

4 Real-time Cloth Animation

In general, the matrix involved in the implicit integration techniques for cloth animation is so sparse that iterative methods can be successfully employed. Although the sparseness of the matrix does not guarantee the real-time performance, it is the most important property. The method proposed in this paper also exploits the sparseness of the matrix and other properties in order to compute the next state as fast as possible. This paper proposes an efficient approximation of the solution of the linear system incurred by the implicit integration scheme. Since the integration scheme itself is based on the implicit method, the result is stable enough for the real-time animation which requires large time-steps. Moreover, the computational complexity of the proposed method is linearly proportional to the number of mass-points, which is clearly the optimal computational complexity.

In addition, another attention should be paid to the derivative of the spring force. The implicit integration scheme involves the force at the next step which cannot be computed with the current state of the mass-spring model. In order to approximate the force at the next step, the first order derivative of the force at the current step is used. However, the force derivative can introduce another numerical instability when the spring is contracted. In this chapter, an approximation of the force derivative is introduced in order to avoid the instability.

4.1 Properties of the Problem

The size of the matrix $\mathbf{M} - h^2\mathbf{J}$ in Eq 11 rapidly increases as the number of mass-points increases. However, the linear system can be efficiently solved because of the properties of the matrix. Since $\mathbf{M}$ is a diagonal matrix and the Jacobian matrix $\mathbf{J}$ is sparse, it is obvious that $\mathbf{M} - h^2\mathbf{J}$ is also sparse. For the simplicity, let us define a matrix $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{M} - h^2\mathbf{J} \tag{12}$$

The components of the matrix $\mathbf{W}$ are 3×3 sub-matrices, which can be easily computed. The com-

ponent in the i-th row and j-th column, $\mathbf{W}_{ij}$, is $-h^2\mathbf{J}_{ij}$ if i and j are different from each other. The diagonal components, $\mathbf{W}_{ii}$, can be computed as $\mathbf{M}_i - h^2\mathbf{J}_{ii}$. In other words, the i-th diagonal component is the sum of the off-diagonal components in the row and mass sub-matrix $\mathbf{M}_{ii}$. Therefore, the diagonal components can be expressed as $\mathbf{M}_i + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}$ where E is the set of springs. The matrix $\mathbf{W}$ in the linear system has many important properties that can be exploited for better performance.

The vectors $\mathbf{f}$, $\mathbf{v}$, and $\mathbf{x}$ are all 3-dimensional vectors as described in Eq. 5. Therefore, the components of the Jacobian matrix $\mathbf{J}$ must be 3×3 matrices. The matrix in the linear system can be regarded as a block matrix composed of 3×3 sub-matrices. The previous approximate methods [9, 15] approximated these sub-matrices as scalar values, and the result is stable enough to use arbitrarily large time steps. However, such excessive approximation introduces unnecessary damping in every direction so that the animation results by these methods are not plausible. The method proposed in this paper does not employ the approximation of these sub-matrices into scalar values in order to maintain the plausibility of the cloth animation.

It is obvious that the matrix is symmetric because $\mathbf{J}_{ij}$ and $\mathbf{J}_{ji}$ are identical. One additional property is that each sub-matrix is also a symmetric matrix. The symmetry of the whole matrix and sub-matrices enables us to efficiently store the matrix, and the inverses of the sub-matrices can be computed in a more efficient way because they are 3×3 symmetric matrices.

The matrix $\mathbf{W}$ is sparse because $\mathbf{M}$ is diagonal matrix and a component of the Jacobian matrix $\mathbf{J}_{ij}$ has a non-zero matrix only when the mass-point i and j are linked with a spring. Therefore, the number of non-zero components (3×3 sub-matrices) in the i-th row is simply $n_i + 1$ where n_i denotes the number of springs linked to the mass-point i. In general, only a small number of springs are linked to a specific mass-point, and it is not affected by the complexity of the model, i.e., the number of total mass-points in the cloth model. Therefore, each row in the matrix has a limited number of effective components, and the sparseness of the matrix increases as the number of total mass-points increases.

The vector ($h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t$) in the right side of the Eq. 11 can be described as $h(\mathbf{f}^t + h\mathbf{J}\mathbf{v}^t)$ and the

additional forces $h\mathbf{J}\mathbf{v}^t$ can be considered as viscosity forces. Because of the sparseness of the Jacobian matrix, the vector $h\mathbf{J}\mathbf{v}^t$ can be efficiently computed. Let $\tilde{\mathbf{f}}$ be the internal force which is the sum of spring forces and viscosity forces ($\mathbf{f}^t + h\mathbf{J}\mathbf{v}^t$). Each component (3-dimensional vector) of the internal force vector can then be calculated with the consideration of linked mass-points as follows [9]:

$$\begin{aligned} \tilde{\mathbf{f}_i^t} &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}\mathbf{v}_j^t + h\mathbf{J}_{ii}\mathbf{v}_i^t \\ &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}(\mathbf{v}_j^t - \mathbf{v}_i^t) \end{aligned} \tag{13}$$

Then, the linear system in Eq. 11 can be rewritten as follows:

$$\mathbf{W}\Delta\mathbf{v}^{t+h} = h\tilde{\mathbf{f}^t} \tag{14}$$

The linear system shown in Eq. 14 is the actual problem that should be solved for the stable cloth animation. In the formal problem, the matrix $\mathbf{W}$ and the vector $\tilde{\mathbf{f}^t}$ can be efficiently computed with the current state of the cloth model without any difficulties. The problem is how to find a reasonable approximation of $\Delta\mathbf{v}^{t+h}$ that satisfies the linear system shown in Eq. 14.

4.2 Solution Approximation in Linear Time

Due to the properties of the matrix $\mathbf{W}$, the linear system of Eq. 14 can be expressed as following n equations:

$$\begin{aligned} \mathbf{W}_{11}\Delta\mathbf{v}_1^{t+h} &= h\tilde{\mathbf{f}_1^t} + h^2 \sum_{(1,j)\in E} \mathbf{J}_{1j}\Delta\mathbf{v}_j^{t+h} \\ \mathbf{W}_{22}\Delta\mathbf{v}_2^{t+h} &= h\tilde{\mathbf{f}_2^t} + h^2 \sum_{(2,j)\in E} \mathbf{J}_{2j}\Delta\mathbf{v}_j^{t+h} \\ &\vdots \\ \mathbf{W}_{nn}\Delta\mathbf{v}_n^{t+h} &= h\tilde{\mathbf{f}_n^t} + h^2 \sum_{(n,j)\in E} \mathbf{J}_{nj}\Delta\mathbf{v}_j^{t+h} \end{aligned} \tag{15}$$

All the equations in Eq. 15 have a similar form, and the i-th equation can be rewritten to highlight the velocity change of the i-th mass-point as follows:

$$\begin{aligned} \Delta \mathbf{v}_i^{t+h} &= \mathbf{W}_{ii}^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta \mathbf{v}_j^{t+h}) \\ &= (\mathbf{M}_{ii} - h^2\mathbf{J}_{ii})^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta \mathbf{v}_j^{t+h}) \end{aligned} \tag{16}$$

The matrices ($\mathbf{M}_i$, $\mathbf{J}_{ii}$, and $\mathbf{J}_{ij}$) involved in Eq. 16 are 3×3 symmetric matrices, and the computation of the vector $\sum \mathbf{J}_{ij}\Delta \mathbf{v}_j^{t+h}$ requires a small amount of computations if the velocity change of linked mass-points at the next time step $\Delta \mathbf{v}_j^{t+h}$ is known. Therefore, the motion of the cloth model can be computed in O($n + e$) time if the velocity change of each mass-point can be computed independently with Eq. 16. However, each equation, unfortunately, cannot be computed independently because it is related with the result of some other equations ($\Delta \mathbf{v}_j^{t+h}$). The method of this paper approximates the solution of the linear system by iteratively computing the velocity change of each mass-point with Eq. 16. In fact, Eq. 16 is the Jacobi iteration scheme if another superscript k is introduced as the number of iterations. Let $\Delta \mathbf{v}_j^{t+h^{(k)}}$ be the result after the k iterations, and suppose that the initial guess of velocity change of each mass-point is simply $\Delta \mathbf{v}_i^{t+h} = \mathbf{W}_{ii}^{-1}h\tilde{\mathbf{f}}_i^t$ by ignoring the velocity changes of linked mass-points. The iterative scheme can then be described as follows:

$$\begin{aligned} \Delta \mathbf{v}_i^{t+h^{(0)}} &= \mathbf{W}_{ii}^{-1}h\tilde{\mathbf{f}}_i^t \\ \Delta \mathbf{v}_i^{t+h^{(k+1)}} &= \mathbf{W}_{ii}^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta \mathbf{v}_j^{t+h^{(k)}}) \end{aligned} \tag{17}$$

During the iterative update of $\Delta \mathbf{v}_i^{t+h}$, the matrix $\mathbf{M}_i - h^2\mathbf{J}_{ii}$ (i.e., $\mathbf{W}_{ii}$) and the vector $h\tilde{\mathbf{f}}_i$ remain constant. Therefore, these matrix and vector need to be computed only once during the iterations for one time-step. Let us denote the constant matrix and vector as $\mathbf{W}_i$ and $\mathbf{p}_i$ as follows:

$$\begin{aligned} \mathbf{W}_i &= \mathbf{M}_i - h^2\mathbf{J}_{ii} \quad \in \mathbf{R}^{3\times 3} \\ \mathbf{p}_i &= h\mathbf{W}_i^{-1}\tilde{\mathbf{f}}_i \quad \in \mathbf{R}^3 \end{aligned} \tag{18}$$

The iterative scheme in Eq. 17, then, can be more compactly expressed as follows:

$$\begin{aligned} \Delta\mathbf{v}_i^{t+h^{(0)}} &= \mathbf{p}_i \\ \Delta\mathbf{v}_i^{t+h^{(k+1)}} &= \mathbf{p}_i + h^2\mathbf{W}_i^{-1}\sum_{(i,j)\in E}\mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h^{(k)}} \end{aligned} \tag{19}$$

The diagonal components of the matrix in Eq. 14 are usually more dominant than the off-diagonal components so that the initial guess of the iterative scheme is already a good approximation of the solution. Therefore, the satisfactory approximation of the solution can be obtained with a small number of iterations Since the iterative scheme is based on the implicit method, the animation result is sufficiently stable for real-time systems. In most cases, the approximate solution after only one iterative update ($\Delta\mathbf{v}_j^{t+h^{(1)}}$) was stable and plausible enough to be used in real-time system. Therefore, the velocity change of each mass-point can be approximated as follows:

$$\Delta\mathbf{v}_i^{t+h} \simeq \Delta\mathbf{v}_i^{t+h^{(1)}} = \mathbf{p}_i + h^2\mathbf{W}_i^{-1}\sum_{(i,j)\in E}\mathbf{J}_{ij}\mathbf{p}_j \tag{20}$$

The fast approximation of the velocity change of each mass-point with Eq. 20 enables real-time cloth animation.

The proposed method is efficient because it computes the velocity change of each mass-point by considering only the linked mass-points. The method requires the system to compute the inverse matrix ($\mathbf{W}_i^{-1}$) of diagonal sub-matrices $\mathbf{W}_i$ of which size is 3×3. Compared to the whole linear system solving, the burden of calculating the 3×3 inverse matrix is trivial. Thus, the proposed method provides

an efficient scheme to update the state of the whole cloth model in O(n) time with guaranteed stability. In other words, the computational cost of the method for computing the next step is as efficient as explicit method, of which time complexity is obviously optimal. Besides the efficiency, the computational result by the method is as stable as implicit method which makes it possible for a large time step to be used. The stability and the efficiency of the proposed method provides the optimal solution to the cloth animation problem in the real-time environments.

In the approximate solution shown in Eq. 20, the iteration parameter k was set to 1. Although the parameter can be arbitrarily increased, the increased number of the iteration impairs the real-time performance of the system. Therefore, the iteration parameter k should be small enough to guarantee the real-time performance. Apart from the real-time performance of the system, another important thing should be considered: the iteration parameter k should be an odd number. In other words, if the iteration parameter k is a small and even number, the system tends to become unstable.

This property becomes meaningless as the k increases up to a large value. However, k should be an odd number when it is restricted to be a sufficiently small number for the real-time animation. The reason can be explained with simple consideration of the difference between the initial guess $\mathbf{p}_i$ and desired solution δ. The vector $\mathbf{p}_i$ is an unstable approximation of the solution. Therefore, the magnitude of $\mathbf{p}_i$ tends to be much larger than a stable solution δ ($\mathbf{p}_i >> \delta$). By considering the linked mass-points (i.e., considering the term $h^2\mathbf{W}_i^{-1}\sum_{(i,j)\in E}\mathbf{J}_{ij}\mathbf{p}_j$), the approximate solution becomes stable and have a value similar to the stable solution δ. However, one more iteration (i.e., $k = 2$) adds unstable approximation ($\mathbf{p}_i$) to the previous stable solutions, and brings instability to the system. Therefore, the stability of the approximate solution alternates according to whether k is even or odd. Such instability vanishes as the number of the iteration increases. However, for the small k values, it is safe to use an odd k values. Even with extremely small odd numbers such as $k = 1$ or $k = 3$, the proposed method produced stable results.

4.3 Stabilized Force Derivative

In order to keep the simulation process stable, an implicit integration scheme should be employed. As shown in Eq. 4, the implicit integration scheme does not use the simple internal force computed by Hooke's law. The implicit method involves the force at the next simulation step. However, the force at the next step cannot be computed with the current state. Therefore, the force at the next time step was approximated with the first derivative as shown in Eq. 7. Since $\mathbf{f}$ is n dimensional vector of which elements are 3 dimensional vectors, The derivative of the force vector with respect to the positional vector $\mathbf{x}$ (i.e., $\mathbf{J}$) is $n \times n$ Jacobian matrix of which elements are 3×3 sub-matrices. The element at the i-the row and the j-th column in $\mathbf{J}$ is derivative of force on the mass-point i with respect to the location of the mass-point j (i.e., $\partial \mathbf{f}_i / \partial \mathbf{x}_j$), and can be expressed as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \kappa_{ij} \frac{|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|} \mathbf{I}_3 + \kappa_{ij} \frac{l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|} \left(\frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathbf{T}}}{|\mathbf{x}_j - \mathbf{x}_i|^2} \right) \tag{21}$$

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

Although Eq. 20 significantly increases the stability of animation system, Jacobian matrix in Eq. 21 incurs another type of instability as described in [7]. The force derivative shown in Eq. 21 does not incur any problem when the spring is stretched. However, the force derivative in Eq. 21 has serious problem when the distance between the mass-points i and j is smaller than l_{ij}^0. When the distance between two mass-points approaches to 0, the derivative of the force contains extremely large values as components, and it makes the system fail. In order to avoid the undesirable situations, the force derivative is approximated by assuming l_{ij}^0 has the same value as $|\mathbf{x}_j - \mathbf{x}_i|$ when the distance between two linked mass-points i and j is smaller than the rest length l_{ij}^0. Then the components of the force derivative are guaranteed to have smaller values than the spring constant of the spring. This assumption enables the system to avoid the failure, and the result animation is still plausible enough. In other words, the accurate force derivative is used when the spring is stretched and the stabilized approximate force

derivative is used when the spring is contracted. Therefore, the force derivative used for the proposed method can be expressed as follows:

$$\text{if } \; l_{ij}^0 < |\mathbf{x}_j - \mathbf{x}_i| \qquad (22)$$

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

$$\text{if } \; l_{ij}^0 \geq |\mathbf{x}_j - \mathbf{x}_i|$$

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

4.4 Efficient and Stable Damping

Although the implicit method is unconditionally stable during the simulation and the proposed approximate method shows a good stability property, Eq. 20 produces motion without any consideration of damping. However, the actual movement of any object in real world reaches the static equilibrium state by damping effect that dissipates the energy. Therefore, damping should be considered in order to produce realistic motion, and the simplest damping model is to employ damping force vector $\mathbf{f}_d$ as follows:

$$\mathbf{f}_d = -C_d \mathbf{v} \qquad (23)$$

where C_d is damping coefficient, which is a non-negative scalar value.

If the damping force is considered, the internal force on a mass-point depends not only on the locations of the mass-points but also on the velocities of the mass-points. Therefore, the derivative of the force used in the previous section should be adjusted in order to include the partial derivative with respect to

the velocity. Therefore, the force at the next time step can be expressed as follows:

$$\begin{aligned} \mathbf{f}^{t+h} &= \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x}^{t+h} + \frac{\partial \mathbf{f}_d}{\partial \mathbf{v}} \Delta \mathbf{v}^{t+h} \\ &= \mathbf{f}^t + \mathbf{J} \Delta \mathbf{x}^{t+h} - C_d \mathbf{I} \Delta \mathbf{v}^{t+h} \\ &= \mathbf{f}^t + \mathbf{J}(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})h - C_d \mathbf{I} \Delta \mathbf{v}^{t+h} \end{aligned} \tag{24}$$

Note that the damping affects only the diagonal components of the matrix. Therefore, the velocity change in Eq. 11 can be rewritten as follows:

$$(\mathbf{W} + hC_d\mathbf{I})\Delta \mathbf{v}^{t+h} = h\mathbf{f}^t + h^2 \mathbf{J}\mathbf{v}^t \tag{25}$$

Eq. 25 implies that a simple addition of hC_d to the diagonal components of the 3×3 sub-matrix $\mathbf{W}_i$ generates damping effects. The simple damping model dissipates the energy and improves the stability of the system. While the damping effect can improve the stability, the plausibility of the motion can be impaired because the simple model slows the motion in any cases. In order not to damp the motion of a mass-point when its movement has the same direction and the same velocity as those of the linked mass-points, the damping model can be modified as follows:

$$\mathbf{f}_d = -C_d \sum_{(i,j)\in E} (\mathbf{v}_i - \mathbf{v}_j) \tag{26}$$

The sparseness of the matrix $\partial \mathbf{f}_d / \partial \mathbf{v}$ is exactly the same as the sparseness of $\mathbf{J}$ because the 3×3 sub-matrix $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ij}$ is a non-zero matrix only when the mass-points i and j are linked by a spring edge. If two mass-points are linked, $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ij}$ becomes $C_d \mathbf{I}_3$.

The diagonal sub-matrix $(\partial \mathbf{f}_d / \partial \mathbf{v})_{ii}$ can also be simply computed as $-n_i C_d \mathbf{I}_3$ where n_i is the number of mass-points linked to the mass-point i. Note that this damping model only affects the diagonal components of the non-zero sub-matrices. Therefore, the velocity change in Eq. 16 can be modified

to include the damping effects without introducing any significant additional computations. For the simplicity, let us denote $\mathbf{W}_i + n_i h C_d \mathbf{I}_3$ and $\mathbf{J}_{ij} + h C_d \mathbf{I}_3$ as $\mathbf{W}'_i$ and $\mathbf{J}'_{ij}$ respectively. Then, the velocity change can be computed as follows:

$$\Delta \mathbf{v}_i^{t+h} = \mathbf{W'}_i^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum \mathbf{J}'_{ij} \Delta \mathbf{v}_j^{t+h}) \tag{27}$$

The iterative update scheme can be constructed with a simple modification of the definition of $\mathbf{W}_i^{-1}$ and $\mathbf{p}_i$ in Eq. 18, and the approximate solution can be obtained as follows:

$$\Delta \mathbf{v}_i^{t+h} \simeq \mathbf{W'}_i^{-1} h\tilde{\mathbf{f}}_i^t + h^2 \mathbf{W'}_i^{-1} \sum \mathbf{J}'_{ij} (\mathbf{W'}_j^{-1} h\tilde{\mathbf{f}}_j^t) \tag{28}$$

Eq. 28 enables the system to successfully deal with the damping force without any instability problem. The damping force which depends on the velocity is also numerically integrated with the implicit integration scheme in order to guarantee the stability of the system, and the derivative of the damping force is efficiently computed and included in the system by exploiting the the properties of the the damping force and its derivative. Fig. 2 shows the real-time animation result produced with Eq. 28. As shown in the figure, the proposed method generated plausible animation sequence in real-time environments.

5 External Forces

The motion generated with Eq. 20 generates an motion without the consideration of any external forces. However, an actual cloth in the real world experiences various external forces caused by gravity or interaction with air. Therefore, plausible animation cannot be achieved without the consideration of the external forces.

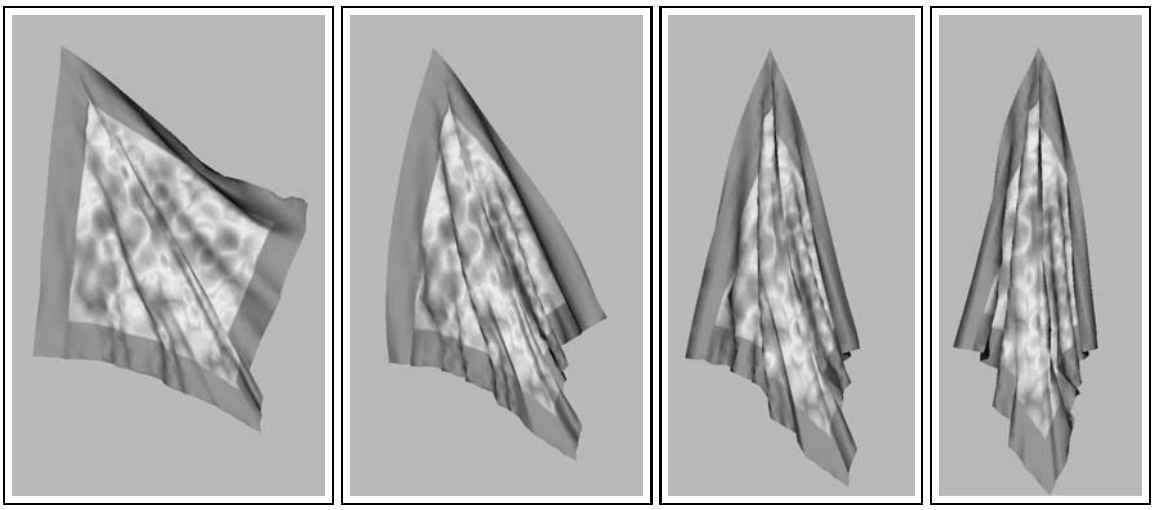


Figure 2. Real-time animation produced with the proposed techniques

5.1 External Forces 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 [3]:

$$|\mathbf{f}^{drag}| = \frac{1}{2} C_D \rho |\mathbf{v}^{wind}|^2 S \sin\theta \tag{29}$$

where $|\mathbf{f}^{drag}|$ denotes the magnitude of the drag force, C_D is the drag force coefficient, ρ is the density of a fluid, $\mathbf{v}^{wind}$ is the velocity of an object relative to the fluid, S is the area of object surface, and θ is the angle between $\mathbf{v}^{wind}$ 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:

$$|\mathbf{f}^{lift}| = \frac{1}{2} C_L \rho |\mathbf{v}^{wind}|^2 S \cos\theta \tag{30}$$

where $|\mathbf{f}^{lift}|$ 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.

In order to implement the drag and the lift forces, let us define $\hat{\mathbf{n}}_i$ as the unit normal vector of the cloth surface at the i-th mass-point, and $\hat{\mathbf{v}}_i$ as $\mathbf{v}_i/|\mathbf{v}_i|$. The angle between $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{v}}_i$ is then $\pi/2 - \theta$. Thus, $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i$ is $\sin\theta$ $(= \cos(\pi/2 - \theta))$. Therefore, the magnitude of the drag force is proportional to $|\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i|$. Since the direction of the drag force is the opposite direction of the velocity as shown in Eq. 29, the magnitude of the drag force can be implemented as follows:

$$\mathbf{f}_i^{drag} = -K_D|\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i||\mathbf{v}_i|^2\hat{\mathbf{v}}_i \tag{31}$$

where K_D is the control parameter for the drag force.

For the implementation of the lift force, the direction of the lift force should be determined. Let us denote $\mathbf{U}_i$ as the direction of the lift force on the i-th mass-point. Since the direction of the lift force is perpendicular to the direction of the relative velocity of the mass-point to the fluid, $\mathbf{U}_i$ can be defined as follows:

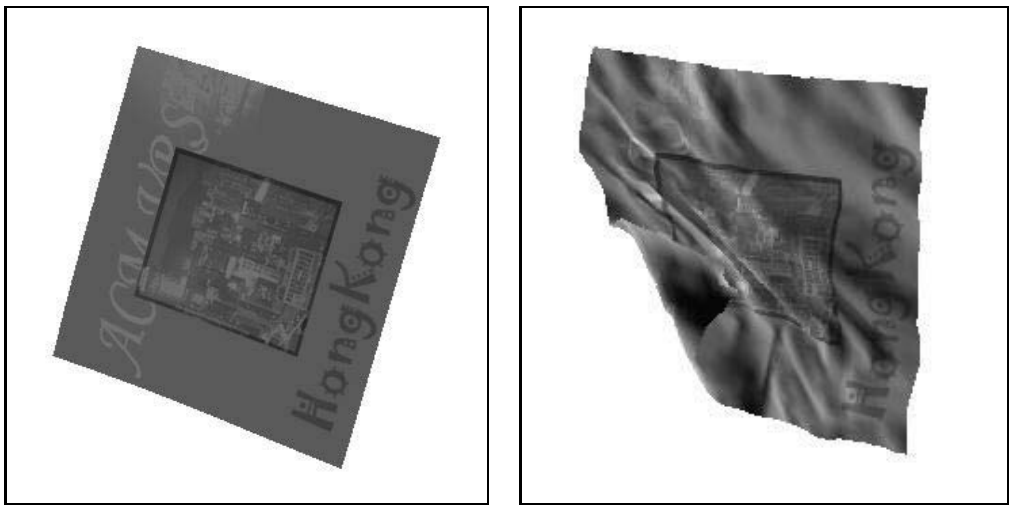
$$\begin{aligned} \mathbf{U}_i \;\; = \;\; & (\hat{\mathbf{n}}_i \times \hat{\mathbf{v}}_i) \times \hat{\mathbf{v}}_i \;\; , \;\; if \;\; \hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i > 0 \\ & (-\hat{\mathbf{n}}_i \times \hat{\mathbf{v}}_i) \times \hat{\mathbf{v}}_i \;\; , \;\; otherwise \end{aligned} \tag{32}$$

Once the direction of the lift force was determined, the lift force can be also determined only by computing the magnitude of the lift force. Therefore, the lift force $\mathbf{F}_i^{lift}$ on the i-th mass-point as follows:

$$\mathbf{f}_i^{lift} = (K_L|\cos\theta||\mathbf{v}_i|^2)\mathbf{U}_i \tag{33}$$

where K_L is the control parameter for the lift force.

However, the cosine function is too expensive function when the efficiency of the computation is more important than the physical correctness. As shown in Eq. 29 and Eq. 30, the magnitude of the drag force



(a) (b)

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

is proportional to $\sin\theta$ while that of the lift force is proportional to $\cos\theta$. In other words, the magnitude of the lift force is maximal when that of the drag force is minimal, and vice versa. In order to efficiently determine the lift force, the magnitude of the lift force was assumed to be proportional to $1 - |\sin\theta|$. This simple assumption increase the magnitude of the lift force as that of the drag force decreases. In the opposite situation, of course, the magnitude of the lift force is increased. As mentioned before, $|\sin\theta|$ is identical with $|\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i|$. Therefore, the lift force can be implemented as follows:

$$\mathbf{f}_i^{lift} = (K_L(1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i|)\mathbf{v}_i|^2)\mathbf{U}_i \tag{34}$$

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. The interaction between the cloth model and air significantly increases the realism of the result. Fig. 3 shows the animation results when the cloth model was released in the gravity field. Fig. 3 (a) shows the result when no interaction between the air and the cloth was assumed while (b) shows the result when the cloth was enforced to interact with the air. The air

interaction was implemented with the realism enhancement techniques in this section. The result animation demonstrates that the consideration of the interaction between the air and the cloth model generates more realistic and plausible results.

6 Implementation Details

Iterative update or approximation with Eq. 19 and 20 enables us to efficiently manipulate deformable objects of high complexity that have not been available in current virtual reality systems. However, in the actual implementations, more things should be considered. In this section, some important issues for the implementation will be explained.

6.1 Storage for Jacobian Matrix

Although the size of the Jacobian matrix $\mathbf{J}$ is huge, it is very sparse. The matrix can be efficiently stored by taking advantage of the sparseness with general techniques for sparse matrices. However, sparseness of the matrix is not the only property that can be exploited for efficient storage.

As mentioned early, the matrix is symmetric so that $\mathbf{J}_{ij}$ and $\mathbf{J}_{ji}$ are identical and have effective values only when the mass-point i and j are linked. Therefore, each effective sub-matrix of $\mathbf{J}$ except for the diagonal components ($\mathbf{J}_{ii}$) can be considered as an attribute of each spring. The number of diagonal sub-matrices $\mathbf{J}_{ii}$ is exactly the same as the number of mass-points. Therefore, each diagonal sub-matrix can then be considered as an attribute of each mass-points. Since the effective sub-matrices of the Jacobian matrix $\mathbf{J}$ are considered as attributes of either mass-points or spring edges, they can be easily stored and retrieved as member data of the mass-point and spring objects. The lower triangular components in sub-matrices are not stored because all the sub-matrices in our problem are symmetric.

Begin ComputeJacobian

for (i : from 1 to n) $\mathbf{J}_{ii} \leftarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

for (e : from 1 to $n(E)$) [{$n(E)$ is the number of total spring edges}

$i \leftarrow E(e).v_1$ {v_1 is the first mass-point of the spring edge $E(e)$}

$j \leftarrow E(e).v_2$ {v_2 is the second mass-point of the spring edge $E(e)$}

$\mathbf{J}_{ij} \leftarrow \partial \mathbf{f}_i / \partial \mathbf{x}_j$

$\mathbf{J}_{ii} \leftarrow \mathbf{J}_{ii} - \mathbf{J}_{ij}; \qquad \mathbf{J}_{jj} \leftarrow \mathbf{J}_{jj} - \mathbf{J}_{ij}$

]

End ComputeJacobian

Table 1. Jacobian matrix computation

6.2 Jacobian Matrix Computation

The animation technique proposed in this paper is finally reduced to finding the velocity change of each mass-points. The velocity change computation is composed of four modules that sequentially computes the Jacobian matrix $\mathbf{J}$, $\mathbf{W}_i^{-1}$, $\mathbf{p}_i$, and $\Delta \mathbf{v}_i$.

The first module computes the Jacobian matrix $\mathbf{J}$. Although the number of total 3×3 sub-matrices in the Jacobian matrix is n^2, the effective sub-matrices can be computed by considering the spring edges only. Therefore, the computation of Jacobian can be done as shown in Table 1. In order to simplify, the damping was not considered in the pseudo-code.

7 Experimental Results

The proposed method has been successfully implemented for an interactive cloth animation system. The cloth animation system has been developed on PC environments. We have tested the method in medium level PC environments (Pentium II 600 MHz), and the proposed method showed real-time performance for the animation of complex models with thousands of polygons. It is obvious that the time complexity of the proposed method is linearly proportional to the number of mass-points and spring edges because the update process for each mass-point considers only the neighboring mass-points linked by springs. Fig. 4 visually shows the time required to compute the next state according to the number of mass-points. As shown in the figure, the time required for updating the state of the cloth model is linearly proportional to the total number of mass-points in the cloth model.

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

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

$$\mathcal{E}_i(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{|\Delta \mathbf{x}_i^{ref} - \Delta \mathbf{x}_i^{exp}|^2}{|\Delta \mathbf{x}_i^{ref}|^2} \tag{35}$$

where $\mathbf{\Delta} x_i^{ref}$ denotes the motion of the mass-point i in the reference motion, and $\mathbf{\Delta} x_i^{exp}$ is, similarly, the motion of the mass-point i in the experimental motion.

The total error function was devised to be the arithmetic average of the specific error values of all the

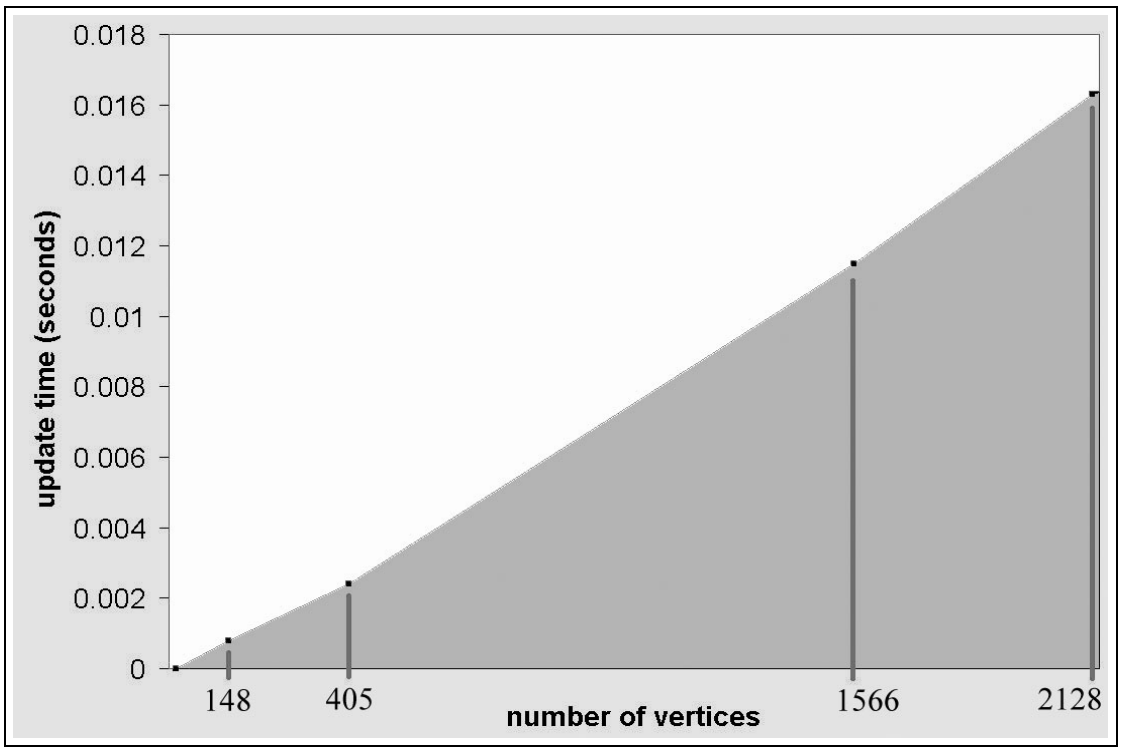


Figure 4. State update time according to the number of mass-points

mass-points as follows:

$$\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{\sum_{i=1}^{n} \mathcal{E}_i(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})}{n} \tag{36}$$

Therefore, the smaller the $\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$ values are, the more plausible the experimental motion is. However, the accurate reference motion cannot be easily obtained. In this paper, the reference motion is assumed to be a motion generated with an extremely small time step size such as 1/30000 sec. Although the reference motion and the experimental motion are generated with different simulation time steps, the check points should be synchronized.

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

$$\mathcal{S}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{\sum_{i=1}^{n} (\Delta \hat{\mathbf{x}}_i^{ref} \cdot \Delta \hat{\mathbf{x}}_i^{exp})^2}{n} \tag{37}$$

where $\Delta \hat{\mathbf{x}}_i^{ref}$ denotes the normalized unit vector along the direction of the reference motion $\Delta \mathbf{x}_i^{ref}$, and

Table 2. $\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$: reference motion with 1/30000 sec time steps and experimental motion with 1/150 sec, and time interval between the check points is 1/30 sec.

check points	1	2	3	4	5	6
$\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$	0.039510	0.003994	0.001276	0.001021	0.001297	0.002019
check points	7	8	9	10	11	12
$\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$	0.004688	0.001803	0.002885	0.009175	0.012875	0.011074
check points	13	14	15	16	17	18
$\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$	0.005238	0.004325	0.003962	0.004577	0.009778	0.010590

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

Table 2 shows the $\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$ values measured at check points. In order to measure the values, $1m \times 1m$ square cloth model was used. The mass of the cloth model is 0.05 kg, and the stiffness constant of each spring e_{ij} is $0.03/l_{ij}^0$ where l_{ij}^0 is the rest length of the spring. The reference motion was generated with time step size of 1/30000 sec and the experimental motion was generated with 1/150 sec. The time interval between the check points was fixed to be 1/30 sec. As shown in the table, $\mathcal{E}$ values of the experimental motion were small enough to produce a plausible animation.

Table 3 shows the $\mathcal{S}$ values under the same condition as Table 2. As shown in the table, the $\mathcal{S}$ values are almost 1 at every check point. Therefore, the direction of the experimental motion can be regarded to be almost the same as that of the reference motion, and the proposed method produces plausible animation.

Fig. 5 shows the $\mathcal{E}$ and $\mathcal{S}$ values measured at 43 check points under the same simulation condition as Table 2 and Table 3. As shown in the figure, the motion produced by the proposed method has small $\mathcal{E}$,

Table 3. $\mathcal{S}(\Delta \mathrm{x}^{ref}, \Delta \mathrm{x}^{exp})$**: reference motion with 1/30000** sec **time steps and experimental motion with 1/150** sec**, and time interval between the check points is 1/30** sec**.**

check points	1	2	3	4	5	6
$\mathcal{S}(\Delta \mathrm{x}^{ref}, \Delta \mathrm{x}^{exp})$	0.998158	0.997947	0.997957	0.997775	0.997444	0.996644
check points	7	8	9	10	11	12
$\mathcal{S}(\Delta \mathrm{x}^{ref}, \Delta \mathrm{x}^{exp})$	0.995828	0.996858	0.996032	0.993291	0.991465	0.995356
check points	13	14	15	16	17	18
$\mathcal{S}(\Delta \mathrm{x}^{ref}, \Delta \mathrm{x}^{exp})$	0.995162	0.995657	0.995733	0.995660	0.992599	0.992900

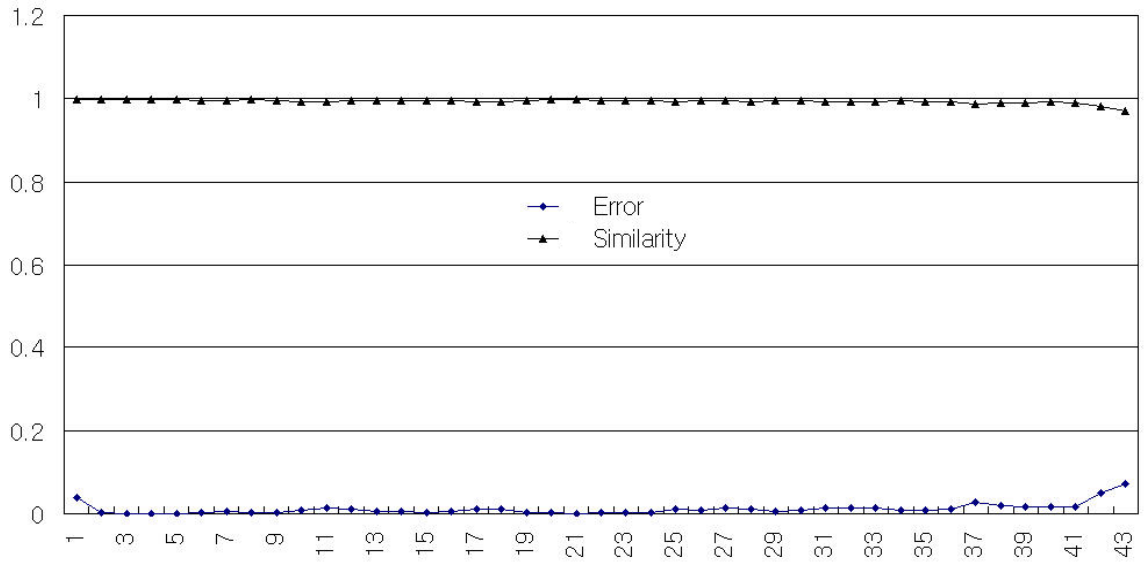


Figure 5. $\mathcal{E}$ **and** $\mathcal{S}$ **values for 43 check points: reference motion with 1/30000** sec **time steps and experimental motion with 1/150** sec**, and time interval between the check points is 1/30** sec**.**

Table 4. $\mathcal{E}$ and $\mathcal{S}$: reference motion with 1/30000 sec time steps and experimental motion with 1/30 sec, and time interval between the check points is 1/30 sec.

check point	1	2	3	4	5	6
$\mathcal{E}$	1.032871	0.093998	0.023254	0.032405	0.024684	0.021908
$\mathcal{S}$	0.998134	0.99776	0.997748	0.996827	0.994657	0.992595
check point	7	8	9	10	11	12
$\mathcal{E}$	0.013602	0.0161	0.024417	0.121476	0.05943	0.073172
$\mathcal{S}$	0.993204	0.994244	0.992554	0.984574	0.980214	0.983945

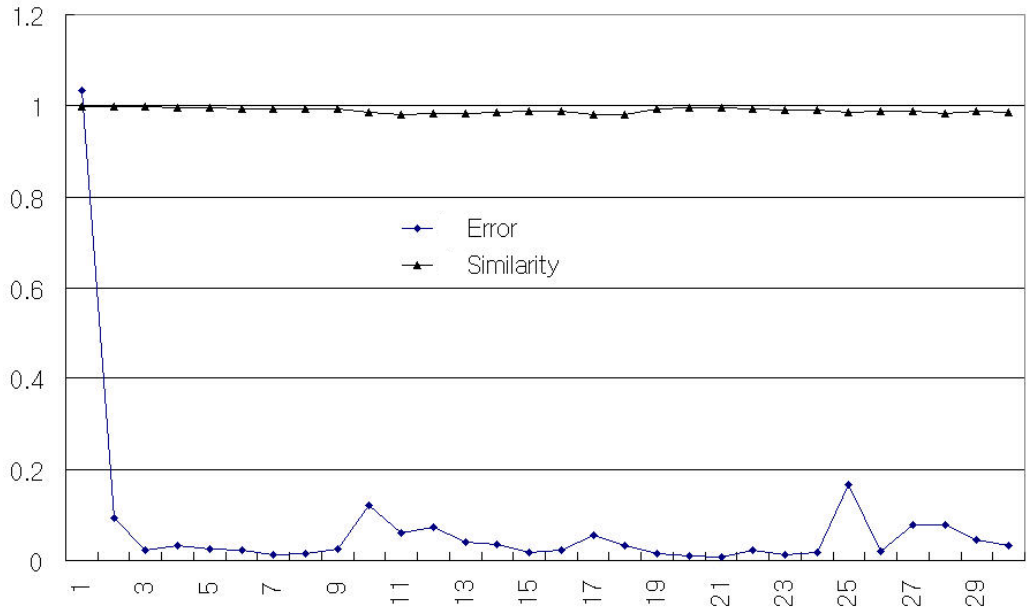


Figure 6. $\mathcal{E}$ and $\mathcal{S}$ values for 30 check points: reference motion with 1/30000 sec time steps and experimental motion with 1/30 sec, and time interval between the check points is 1/30 sec.

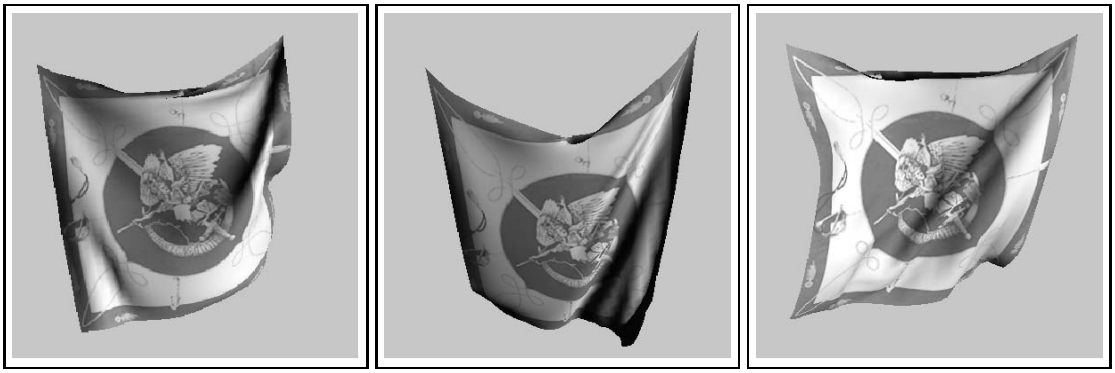


Figure 7. Plausible animation produced in real-time with the proposed method

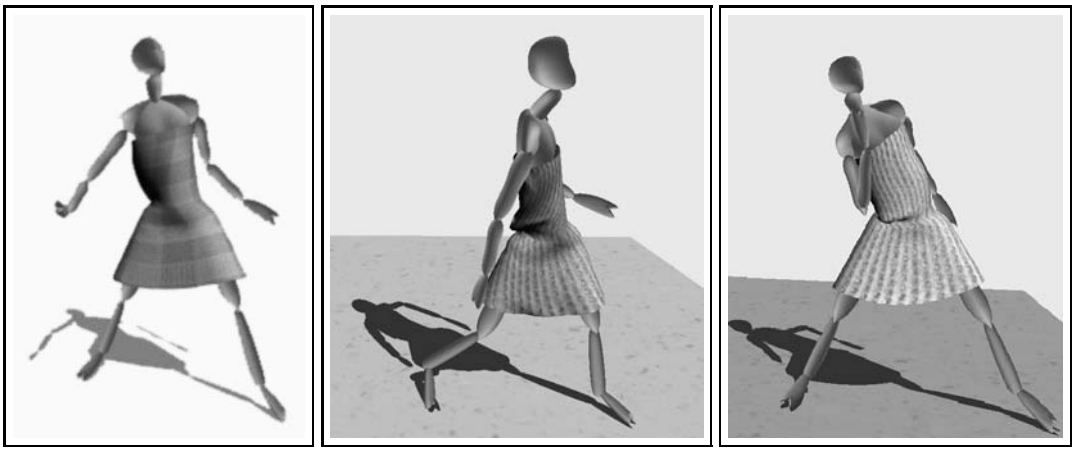


Figure 8. Interactive animation of dressed character

and large $\mathcal{S}$ values even with the large time steps of which size is 1/150 sec. Therefore, the proposed method can be successfully used for generating the plausible motion in real-time environments.

Table 4 and Fig. 6 shows the measured $\mathcal{E}$ and $\mathcal{S}$ values when the experimental motion was generated with the time step size of 1/30 sec. As shown in the table, the proposed method produced plausible result even under this severe condition. This experimental result shows that the proposed method is not only stable but also plausible when the system is supposed to generate animation sequence with large time steps. In order to measure the error and similarity shown in the experiment, the motion was generated with only one iteration. In other words, the iteration parameter k in Eq. 17 was set to be 1.

Fig. 7 shows the result of the real-time animation system implemented with the proposed method. As shown in the figure, the method produced physically plausible motion. The proposed method was

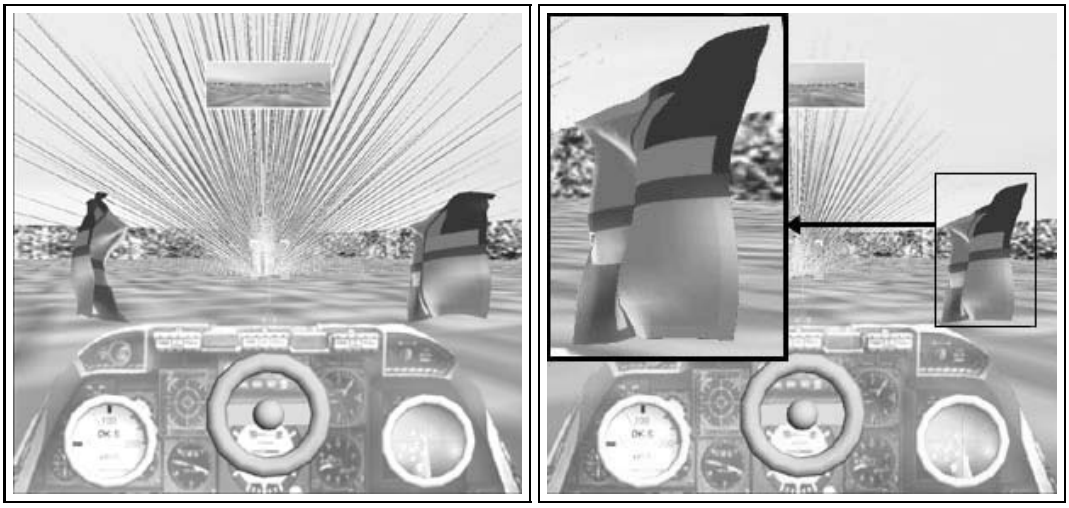


Figure 9. Real-time cloth animation in a game running on PC

applied to a flag model composed of 2048 triangles. As shown in the Fig. 7, the proposed method generated plausible appearance of the cloth model. The size of the time step was 1/300 sec when the result shown in the figure was generated. The geometric complexity of the model was complex enough to represent realistic virtual cloth in virtual environments. The proposed method successfully generated the plausible animation of the complex cloth model in real-time.

The computational efficiency and the physical plausibility of the proposed techniques enable the animation system to produce interactive virtual character with realistic dress. The virtual cloth models with complex geometry have been hardly animated in interactive systems because of the computational cost of the cloth simulation. However, the proposed method can successfully animate realistic cloth model at interactive rates. Fig. 8 shows the result of the interactive animation system implemented with the proposed method. The proposed method efficiently generated plausible and interactive animation of dressed human. Fig. 9 demonstrates the real-time performance of the techniques proposed in this paper. The cloth objects were integrated in an actual game application for PC environments, and the objects were animated without violating the interactivity of the game application.

8 Conclusion

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

One of the major difficulties in cloth animation lies in the stability of the system. The instability of a numerical integration method affects the overall performance of the cloth animation system because the unstable method requires extremely small time-steps. Therefore, real-time animation of a complex cloth model is not feasible with the unstable methods. For that reason, the stability is the first essential property of animation techniques in real-time environments such as VR systems.

Therefore, many researchers have been endeavored to devise stable animation techniques without violating the physical plausibility of the physically-based modeling, and one of the most important advances in recent years is the use of implicit integration. Although the implicit integration scheme is finally reduced to a huge linear system solving problem, the matrix in the system is in general sparse. Therefore, the linear system can be efficiently solved with iterative methods. However, the exact solution still requires too heavy computations to be used in interactive systems.

Recently, various techniques for real-time cloth animation have been proposed. Most of them generated the global motion of the cloth model with rough mesh with sparse particles and used special techniques for generating the detailed wrinkles. Those methods are, as a matter of course, efficient enough to produce cloth animation in real-time environments. However, their methods for the detailed appearance of the cloth do not usually take into account the physical correctness so that the results of the methods are not guaranteed to be physically plausible.

The method proposed in this paper provides sufficient stability because the approximate solution of the method is based on the implicit integration. The stability of the method has been obtained by

employing the implicit integration method for deriving the iterative state-update scheme that computes the next state of each mass-points. Therefore, the method can produce cloth animation with large time-steps. The proposed method efficiently approximates the solution of the problem by taking advantage of various properties of the problem. The proposed method independently computes the next state of each mass-point by considering linked mass-points only. Therefore, the efficiency and the stability of the method enable virtual environments to deal with realistic cloth model.

Apart from the real-time performance, the proposed method has additional important advantages. The proposed method is intuitive and easy to implement. The method can be easily implemented with simple modules for inverse computation of 3×3 matrices and multiplication of 3×3 matrices and vectors so that the method does not require developers to struggle with optimization of the storage and the performance of the linear system solver. The easiness of the implementation guarantees that the implementation of the proposed method in various real-time or interactive systems will be successfully achieved with minimum efforts.

References

[1] T. Agui, Y. Nagao, and M. Nakajima. An expression method of cylindrical cloth objects - an expression of folds of a sleeve using computer graphics. *Trans. Soc. of Electronics, Information and Communications*, J73-D-II:1095–1097, 1990.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[3] Peter J. Brancazio. Physics and sports: The aerodynamics of projectiles. In David Halliday and Robert Resnick, editors, *Fundamentals of Physics*, pages E6:1–E6:8. John Wiley & Sons, 1988.

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

[5] Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):99–104, July 1992.

[6] George Celniker and Dave Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):257–266, July 1991.

[7] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Transactions on Graphics: Proceedings of SIGGRAPH 2002*, pages 604–611, July 2002.

[8] Frédéric Cordier and Nadia Magnenat-Thalmann. Real-time animation of dressed virtual humans. *Proceedings of Eurographics 2002*, 2002.

[9] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pages 1–8, June 1999.

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

[11] C. Feynman. Modeling the appearance of cloth. *Master's thesis, Dept. of EECS, Massachusetts Institute of Technology*, 1986.

[12] Sunil Hadap, Endre Bangarter, Pascal Volino, and Nadia Magnenat-Thalmann. Animating wrinkles on clothes. *IEEE Visualization '99*, pages 175–182, October 1999.

[13] B. K. Hinds and J. McCartney. Interactive garment design. *The Visual Computer Journal*, 6:53–61, 1990.

[14] Young-Min Kang and Hwan-Gue Cho. Bilayered approach for efficient animation of cloth with realistic wrinkles. *Proceedings of Computer Animation 2002*, pages 203–211, 2002.

[15] Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho, and Chan-Jong Park. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, 17(3):147–157, 2001.

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

[17] Mark Meyer, Gilles Debunne, Mathieu Desbrun, and Alan H. Bar. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visulalization and Computer Animation*, 12:1–12, 2001.

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

[19] H. Ng and R. L. Grimsdale. Geoff- a geometrical editor for fold formation. *Lecture Notes in Computer Science*, 1024:124–131, 1995.

[20] Hing N. Ng and Richard L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics & Applications*, 16(5):28–41, September 1996.

[21] Hidehiko Okabe, Haruki Imaoka, Takako Tomiha, and Haruo Niwaya. Three dimensional apparel cad system. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):105–110, July 1992.

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

[23] Jan Plath. Realistic modelling of textiles using interacting particle systems. *Computers & Graphics*, 24(6):897–905, December 2000.

[24] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):269–278, August 1988.

[25] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205–214, July 1987.

[26] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics & Applications*, 8(6):41–51, November 1988.

[27] A. Vlachos, J. Peters, C. Boyd, and J. Mitchell. Curved PN triangles. *Symposium on Interactive 3D Graphics 2001*, pages 159–166, 2001.

[28] Pascal Volino, Martin Courshesnes, and Nadia Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Proceedings of SIGGRAPH 95*, pages 137–144, August 1995.

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

[30] J. Weil. The synthesis of cloth objects. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 20:49–54, 1986.

List of Figures