이 학 박 사 학 위 논 문

Real-time Cloth Animation with Physical Plausibility and Numerical Stability

2003년 2월

부 산 대 학 교 대 학 원

전 자 계 산 학 과

강 영 민

이 학 박 사 학 위 논 문

Real-time Cloth Animation with Physical Plausibility and Numerical Stability

지도교수 조 환 규

2003년 2월

부 산 대 학 교 대 학 원

전 자 계 산 학 과

강 영 민

강 영 민의 이학 박사 학위 논문을 인준함

2002년 12월 12일

주 심 권 혁 철 (인)

부 심 김 영 호 (인)

위 원 김 영 봉 (인)

위 원 이 도 훈 (인)

위 원 조 환 규 (인)

Contents

List of Figures

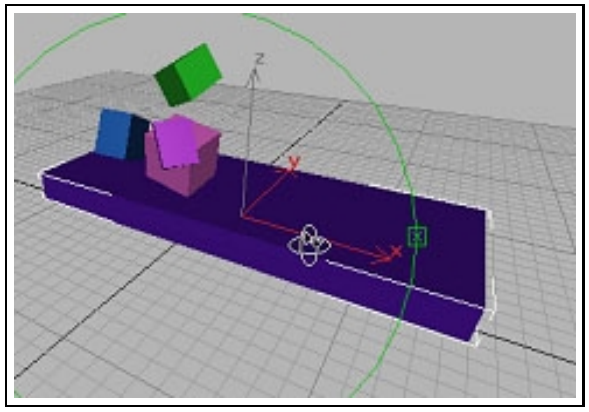
List of Tables

Chapter 1

Introduction

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-based modeling and simulation should be employed when the generated animation is supposed to reproduce the real world in virtual environemnts. Many researchers have proposed various physically-based animation techniques that enhance the realism of the animation and improve the computational performance [Bar90, BW92, WB94, TPBF87, KM90].

As metioned in [TPBF87], the methods to formulate and represent instantaneous shapes of objects are central to computer graphics. The objects in computer graphics literature can be classified into two major categories: rigid objects and non-rigid objects as shown in Fig 1.1. 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 spe-



(a) Rigid body animation

(b) Non-rigid body animation

Figure 1.1: Rigid body animation and non-rigid body animation

cial numerical problem known as instability problem. The instability problem requires the system to decrease the size of the simulation time step. As a consequence, the real-time animation of the deformation objects has been hardly possible in most computing environments, and the efficient modeling and animation of deformable objects have been one of the major research topics in computer graphics for more than a decade.

1.1 Rigid Objects and Deformable Objects

The shape of a rigid object does not change throughout the whole animation sequence. Therefore, the movement of such an object can be efficiently represented by computing the global translation and rotation according to the forces exterting on the object. In other words, the computational cost of updating the state of the rigid object is linearly proportional to the geometric complexity of the object. However, the next state of a deformable object is determined not only by the external forces excerting on the object

	Rigid Object	Non-rigid object
Shape definition	fixed local coordinate for each component	no fixed local coordinate
Animation	global orientation and translation	movement of each component
Crucial force	external force	internal force
Major issues	physical correctness	physical correctness + stability and efficiency

Table 1.1: Comparison of rigid object animation and non-rigid object animation

but also by the internal forces caused by the deformation of the object itself. The internal forces that incur the movement of the non-rigid deformable object can be represented with various kinds of models. However, the basic ideas are all similar. A given initial state of the deformable object is defined as zero-energy state, and the deformation of the object causes the internal forces that independently accelerate each component of the object toward the initial zero-energy state.

For the animation of deformable object, the animation system must compute the individual movement of each element in the model. Therefore, the animation of a deformable object requires heavier computation than the animation of a rigid object when the geometric complexities of both objects are similar. Unfortunately, the heavy cost for the deformable object animation is caused not only by computing the internal forces but also by numerical instability. If the simulation time step is too large, the numerical integration of the internal force easily becomes unstable. Therefore, the numerical integration pro-

cess should use sufficiently small time steps which guarantee the stability of the system. However, the size of the time step should be decreased as the stiffness of the deformable object increases. Even worse, the stiffness of the object can be increased up to an arbitrarily large value. In other words, extremely small time steps are often required in order to ensure the stability, and the small time steps make it impossible to efficiently generate the animation of deformable objects.

Table 1.1 compares the rigid object animation and the non-rigid object animation. The shape of a rigid object are usually defined by the local coordinate of each component. However, the geometric components of a non-rigid object do not have fixed local coordinates because the object undergoes deformation. Therefore, the animation of a rigid object can be performed by computing the global orientation and the global translation of the object. In the other hand, the non-rigid object animation system must compute the indivisual movement of each component. In the generation of the motion, the rigid object animation focuses on the external force on the object, while the non-rigid object animation takes into account the internal force as the crucial force. Therefore, the major issue in the rigid object animation is naturally the physical correctness itself while stability guarantee or efficiency of the computation are the most important issues for the non-rigid object animation.

1.2 Cloth Animation

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 graphics display, various efforts have been made to efficiently generate the realistic appearance and motion of cloth objects [OITN92, VCMT95, Pla00]. However, the cloth is a very stiff model which makes it more difficult for physical simulation methods to efficiently generate the motion. There are two major factors that determine the performance of the cloth animation systems. One of them is the physical model for representing the cloth behavior, and the other is the numerical integration method for generating the motion of the cloth model.

In order to represent cloth, various kinds of methods such as finite elements [CG91], continuum mechanics [TPBF87] and particle system [BHW94, Pla00] can be used. In this thesis, 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 force exterted on the particle, and the force can be computed by considering the distances between the particles connected by the links. There are various ways of constructing the particle system. The simplest method is to assume that the particles are the mass-points and the links behave like springs.

While the cloth representation method such as mass-spring model determines the characteristics of the motion of the cloth model, the numerical integration method for generating the motion of the model affects the computational performance of the ani-

mation system. Although there are various integration methods, those methods can be categorized into two major classes: explicit and implicit integration methods. The selection of the numerical integration methods for cloth animation cannot be made without considering the situation of an actual problem because each numerical integration method has its own advantages and disadvantages [VMT01].

Immersive virtual reality environments require realistic scenes that can interact with user behaviors in real-time. Therefore, the cloth model in virtual reality systems should be animated in an efficient way, while keeping the structure of the model complex enough to guarantee the plausible appearance. In this thesis, efficient animation techniques are proposed for the real-time animation of complex deformable objects. The proposed method exploits the stability of the implicit integration, and efficiently approximates the solution of the linear system. 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.

Chapter 2

Previous Work

Many problems in computer graphics are finally reduced to trade-offs between accuracy and efficiency, and the cloth animation is also the arbitration of the accuracy and the efficiency. 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 computer displays with the noble contributions by various researchers [BHW94, CYTT92, EWS96, TPBF87, TF88, TW88, VCMT95, VMT94]. However, the realistic simulation of cloth requires large amount of computations because the numerical integration easily tends to become unstable [BW98, WB94]. Therefore, it is very difficult to produce the animation of deformable objects in real-time environments such as VR or interactive game systems. Although there have been various models and techniques for the animation of deformable objects, the real-time animation of realistic cloth is still extremely difficult because the accurate method cannot achieve real-time performance if the structure of virtual model is complex enough for the realistic appearance

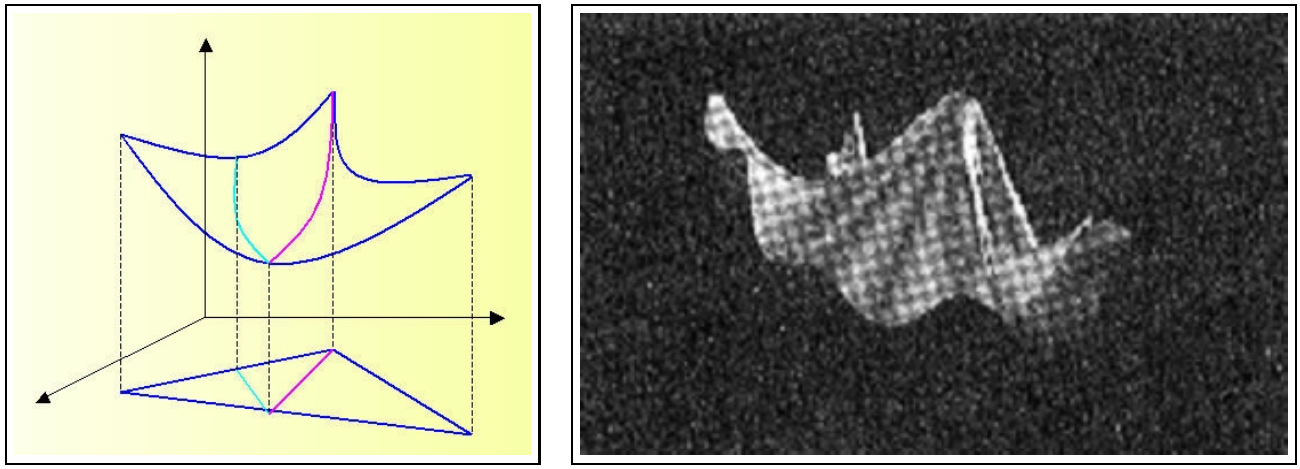
of cloth.

In the following sections, the previous techniques are categorized into several groups and briefly explained in order to provide overall insight to the cloth animation problem. The previous techniques are classified into five categories: geometric approach, physically-based modeling with explicit integration, physically-based modeling with implicit integration, approximate implicit method, and hybrid method.

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 geometric approaches do not represent the behavior of cloth based on dynamics. The approaches tried to model the geometric properties of the cloth such as folds, creases, and overall appearnce with geometrical equations [NG96].

The work by Weil [Wei86] is regarded as the first attempt to geometrically represent the cloth objects[NG96]. The method generated the draped appearance of the cloth object by using multiple catenary curves which are defined as $y = a\cosh(x/a)$ where a is the scaling factor of the curve. The method fits catenary curves between constraints points. The constraints points represent the hanged points of the cloth model, and the catenary curves are fitted between the points. In order to produce more realistic surfaces, the method subdivides a triangle into two parts and generated a new catenary curve along the



(a) Subdivision with catenary curves (b) Generated cloth model

Figure 2.1: Geometric approach with caternary curves proposed in [Wei86]

new edge as shown in Fig 2.1 (a). Therefore, the model could represent a realistic hanged cloth model as shown in Fig. 2.1 (b). However, the method cannot produce dynamic cloth behavior because the geometric catenary curves can represent only a draped deformable cable. Therefore, the method can be applicable only to the hanged and draped cloth as shown in Fig. 2.1 (b)

Several research groups have proposed their own geometric approaches to the representation of the cloth model [HM90, NG95]. Agui $et\ al.$ proposed a geometric approach to represent sleeve [ANN90]. They represented the sleeve with a series of circular rings, and generated the folds on the sleeve by enforcing the distance between adjacent particles to keep the given initial distance. Hinds $et\ al.$ also proposed a geometric approach that generates the cloth model with geometric surfaces. Their method was aimed to automate the garment manufacture, and the cloth model was defined by several patches named panels. Each panel was uniquely defined by its edges, and the detailed surface was generated

with geometric interpolation [HM90]. 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 [NG95].

The geometrical approaches can generate the cloth model in an efficient manner because they do not involve any time-consuming physical simulation and do not suffer from any instability problems. However, the method does 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 [Fey86]. The method was devised to find the final equilibrium state of the cloth model by minimizing the energy of the model. The energy was defined as follows:

$$E(P_{i,j}) = k_s E^{i,j}_{elast} + k_b E^{i,j}_{bend} + k_g E^{i,j}_{grav} \tag{2.1}$$

where k_s, k_b, and k_g are elasticity, bending, and density constants respectively, and $E^{i,j}_{elast}$,$E^{i,j}_{bend}$, and $E^{i,j}_{grav}$ are elasticity, bending, and gravitational energy respectively. Since the method was devised to find the static equilibrium state of the cloth, the method could generate only the draped cloth model.

Terzopoulos $et\ al.$[TPBF87] characterized the cloth simulation as a deformable sur-

face problem, and their approach employed physically-based modeling in order to generate the physically plausible animation. They formulated the motion of the cloth model as follows:

$$\frac{\partial}{\partial t}(\mu\frac{\partial \mathbf{r}}{\partial t}) + \gamma\frac{\partial \mathbf{r}}{\partial t} + \frac{\delta E(\mathbf{r})}{\delta \mathbf{r}} = \mathbf{f}(\mathbf{r}, t) \tag{2.2}$$

where $\mathbf{r}(\mathbf{a}, t)$ is the position of the particle $\mathbf{a}$ at time t, $\mu(\mathbf{a})$ is the mass density of the body at $\mathbf{a}$, $\gamma(\mathbf{a})$ is the damping density, and $\mathbf{f}(\mathbf{a}, t)$ represents the net externally applied force. $E(\mathbf{r})$ is a $functional$ which measures the net instantaneous potential energy of the elastic deformation of the body [TPBF87].

The simulation process of their work uses finite element or finite difference methods. 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 $et\ al.$ was the 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 [BHW94]. 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 as shown in Fig. 2.2. The serious disadvantage

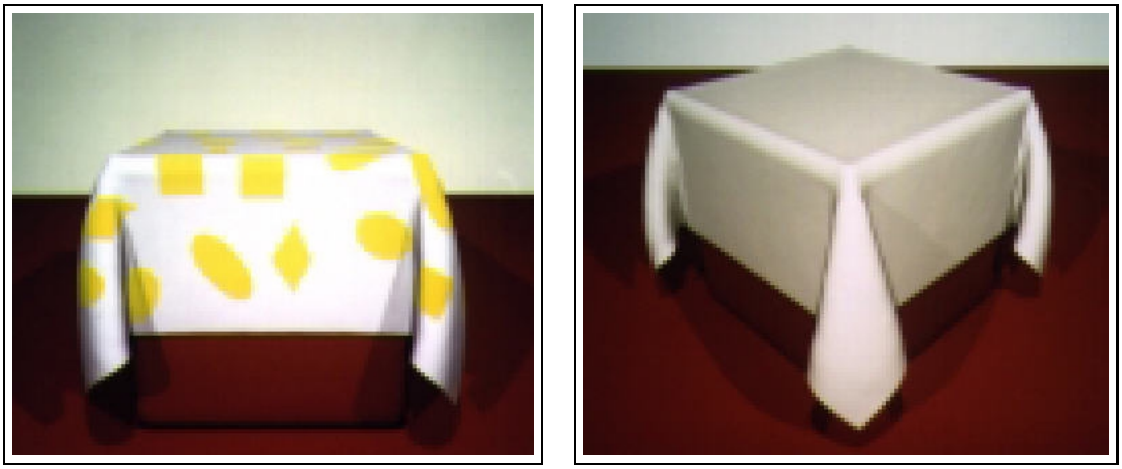
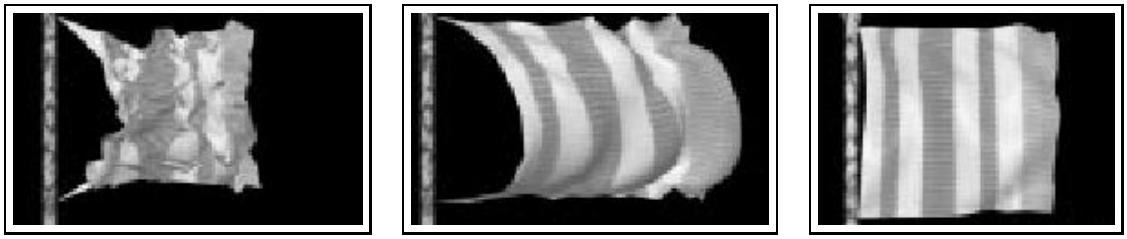


Figure 2.2: Draped cloth generated by energy minimization proposed in [BHW94]

of this method is that it cannot produce dynamically changing intermediate scenes between the initial state and the final equilibrium. Therefore, the method cannot be used for animation.

After the outstanding contributions, various methods has been proposed to represent the cloth model more correctly [CYTT92, VCMT95, EWS96]. Most of those methods are based on the deformable object model proposed by Terzopoulos $et\ al.$ [TPBF87], and the various methods have been devised to improve the visual effect by employing more correct computation and physical modeling of the internal forces. Thalmann group has proposed various techniques for generating cloth on human character models [CYTT92, VCMT95]. However, all those methods used explicit integration schemes so that real-time or interactive animation of cloth model has been almost impossible until recent years.

Since the correct simulation of the cloth requires huge amount of time in general, several methods have been proposed to efficiently generate the motion. Provot proposed



(a) high stiffness (b) low stiffness (c) constrained flag

Figure 2.3: Constrained deformation for mass-spring model proposed in [Pro95]

an simple cloth model that behaves like collection of mass-points linked with springs [Pro95]. However, the method suffers from the instability problem as shown in Fig. 2.3 (a), and decreased stiffness of the system produced the unrealistic animation as shown in Fig. 2.3 (b).

The stiffness cannot be increased to avoid the unrealistic stretch of the springs because the method used explicit integration method which severely suffers from instability of the stiff differential equations. Instead of increasing the stiffness, the method iteratively reduced the length of over-stretched spring edges in order to produce $semi - rigid$ cloth model as shown in Fig. 2.3 (c). However, the iterative adjustment of the spring edge was not based on physical justification. As shown in the work, the instability is the major obstruction to the efficient and plausible cloth animation. Without overcoming the problem, real-time cloth animation based on physical model is not possible.

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 [BW98]. The method formulated the cloth animation with the following implicit integration scheme:

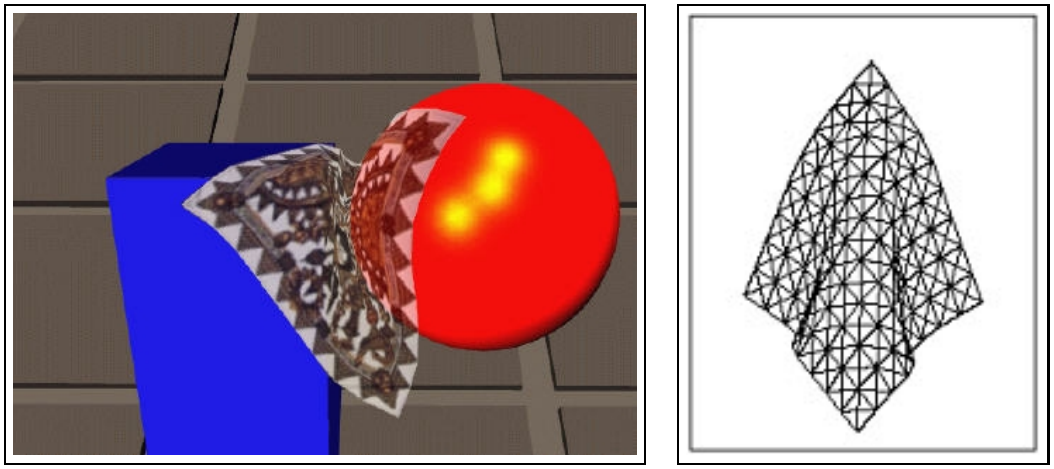
$$\Delta \mathbf{v} = h\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{v}_0 + \Delta\mathbf{v}) \tag{2.3}$$

where $\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{v}_0 + \Delta\mathbf{v})$ is the force that will be generated at the next state.

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. The time complexity of the method for computing the next state of the cloth model was about O($n^{1.5}$), and 2.23 sec was required for updating a cloth model with 2,602 vertices [BW98].

The implicit integration is unconditionally stable during the simulation, and the stability makes it possible to efficiently generate the cloth animation by using large time-steps. 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 [BW98, DSB99, Nak91]. 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 pro-



(a) Result of precomputed filter method (b) Mass-spring model

Figure 2.4: Real-time cloth animation with precomputed filter proposed in [DSB99]

posed to surmount the limitation of the implicit method [DSB99, MDDB01, KCCP01]. 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 [DSB99]. They applied the precomputed inverse matrix as a force filter at every time-step in order to calculate the states of the cloth model at interactive rates. Assume that the linear system involved in the cloth animation is

$$\mathbf{W}\Delta\mathbf{v} = \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ matrix, and n is the number of the mass-points in the cloth model. $\Delta\mathbf{v}$ is the vector of velocity changes, and $\mathbf{f}$ is the vector of forces. They then approximated the matrix $\mathbf{W}$ as a constant matrix $\mathbf{W}'$, and computed the inverse matrix $\mathbf{W}'^{-1}$. Finally, they obtained the vector of velocity changes by applying the inverse matrix to the force

vector at every frame as follows:

$$\Delta \mathbf{v} = \mathbf{W}'^{-1}\mathbf{f} \tag{2.4}$$

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, 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. Therefore, their method can be used only for moderately complex mesh structures as shown in Fig. 2.4 (b).

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 [KCCP01]. The direct update formula is based on the implicit method so that the method does not suffer from the instability problem. However, the method excessively damps the motion 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-step is used. The method uses the same approximation for the force derivative as the precomputed filter method. 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. Basically, the hybrid methods use simplified geometric model for the physical simulation and generate complex details by using geometric curves and surfaces. Oshita and Makinouchi have proposed a geometric approach that generates wrinkly details on cloth models composed of sparse particles [OM01]. The method produces smooth cloth-like surfaces based on PN triangulation proposed in [VPBM01]. Texture mapping techniques have been also used for producing the realistic appearance of the cloth model[HBVMT99]. 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. Therefore, those methods cannot produce natural wrinkles on the cloth model that undergoes various situations, and similar methods that employ multiple mesh structures have similar problems [ZY01, KC02].

Recently, Cordier and Magnenat-Thalmann proposed real-time animation techniques for fully dressed virtual human [CMT02]. The fundamental idea of their method is the employment of different methods for different parts of the dress. Their hybrid method uses geometric deformation for the cloth that is tightly attached to the skin, and catenary curve for efficient sleeve animation. 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.

	Animation Scheme	Advantage	Disadvantage
Geometric	Geometric curves	Efficient	Artificial
Physically-based (explicit)	Dynamics	Plausible	Unstable Inefficient
Physically-based (implicit)	Dynamics	Stable	Linear system
Approximate implicit	Approximate Dynamics	Efficient Stable	Artificial
Hybrid	Dynamics + α α : geometry, texture...	Efficient	Artificial

Table 2.1: Comparison of animation methods for virtual cloth model

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 [KC02]. 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.

Table 2.1 shows the comparison between the animation methods for virtual cloth model. Real-time VR environments require a physically plausible and numerically stable method. According to the Table 2.1, stability of the system can be achieved by using implicit integration scheme. However, the implicit integration scheme does not guarantee

the real-time performance. Real-time system for cloth animation requires a new efficient method which maintains the stability of the implicit integration. This thesis presents an efficient method for creating the animation of realistic virtual cloth.

Chapter 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 thesis 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. An efficient method for finding the reasonable approximation of the solution will be presented in the next chapter.

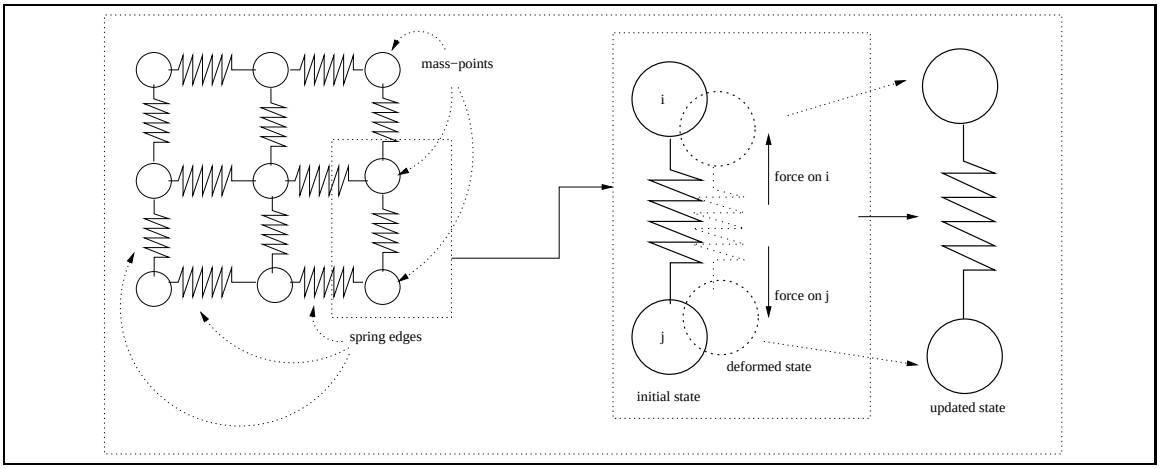


Figure 3.1: Conceptual structure of mass-spring model

3.1 Cloth Model

In this thesis, 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. The conceptual structure of the mass-spring model is shown in Fig. 3.1

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:

$$\mathbf{f}_i^j = \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{3.1}$$

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:

$$\mathbf{f}_i = \sum_{(i,j)\in E} \mathbf{f}_i^j = \sum_{(i,j)\in E} \kappa_{ij}(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{3.2}$$

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

Fig. 3.2 shows complex mass-spring models for realistic cloth animation. 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 complex models such as those shown in Fig. 3.2 cannot be animated interactively 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 thesis is supposed to generate realistic animation in real-time. Therefore, the cloth

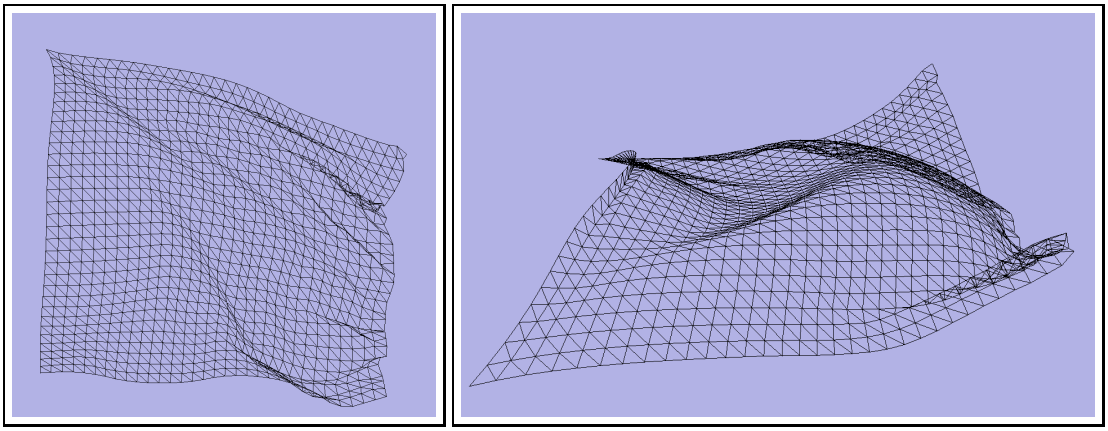


Figure 3.2: Complex mass-spring model

models are assumed to be sufficiently complex as shown in Fig. 3.2

3.2 Straight-forward Integration and Instability

Physically-based cloth simulation can be formulated as numerical integration of forces. The numerical integration schemes can be divided into two categories: explicit methods and implicit methods. The simplest numerical integration method is to use explicit Euler method. However, explicit methods cannot avoid instability problem because the motion of the cloth model is described as a stiff differential equation.

The most important factor in the selection of the numerical integration method is the size of the problem (i.e., the number of particles in the cloth model). The time complexity of an explicit integration method is linearly proportional to the number of particles. However, the time complexity of an implicit method rapidly increases as the number of the particles increases because the implicit method casts the original problem into a linear

system solving problem involving O(n^2) matrix where n is the number of the particles.

Another important factor is the requirements concerning the accuracy. If the system is supposed to generate the motion of a cloth model as accurate as possible, a high order explicit method with small time steps must be employed. However, many kinds of systems for cloth animation do not require physically accurate result. Physically plausible motion is sufficient for those usual animation systems, and efficiency becomes the most important criterion for the selection of the integration method. In order to efficiently generate the cloth motion, implicit integration scheme is regarded as the best solution because it provides unconditional stability and enables the system to employ arbitrary time-step sizes.

The explicit integration methods show better result than the implicit integration methods. However, the accuracy of an explicit method is guaranteed only when the stability of the simulation is maintained. Unfortunately, the stability of the explicit integration can be obtained with sufficiently small time-steps. Therefore, the explicit integration methods show worse performance than the implicit methods in most cases of the cloth animation problem.

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

$$\begin{pmatrix} \mathbf{v}_i^{t+h} \\ \mathbf{x}_i^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i^t + \frac{h}{m}\mathbf{f}_i^t \\ \mathbf{x}_i^t + h\mathbf{v}_i^{t+h} \end{pmatrix} \tag{3.3}$$

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. 3.2.

However, this simple integration scheme (i.e., explicit method) cannot be applied to cloth simulation unless the time step h is very small. The explicit Euler method described in Eq.3.3 is very easy and intuitive, but it has a fatal defect that it can be unstable[WB94, Kas95]. The stiffness of a differential equation causes the instability. The instability is the common problem in any kinds of explicit integration schemes. Therefore, the explicit integration can hardly be selected as a simulation technique for deformable objects, especially for stiff objects such as cloth.

3.3 Stable Animation with Implicit Integration

The behavior of the mass-spring model can be easily described with Hooke's law, and the motion of the model can be easily modeled with simple differential equations. However, as described in the previous section, 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 [Nak91, Kas95, DSB99].

The instability of the simple explicit Euler method is inevitable because it computes the next state by using a simple planar model assuming that force does not change during the time interval between the current step and the next step. When the stiffness values of the spring links and the sizes of the time steps increase, system turns unstable. Therefore, a stable implicit integration scheme should be employed if the efficiency of the computational performance is required.

In order to numerically solve a differential equation $dY/dt = f(Y)$, general implicit integration assumes that the derivative is not simply $f(Y^t)$ throughout the time interval but a weighted average of the derivative $f(Y^t)$ and $f(Y^{t+h})$ [Kas95]. If the weights for $f(Y^t)$ and $f(Y^{t+h})$ are 0 and 1 respectively, the integration scheme becomes backward Euler integration as follows:

$$Y^{t+h} = Y^t + hf(Y^{t+h}) \tag{3.4}$$

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 [DSB99, Kas95]:

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix} \tag{3.5}$$

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:

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

$$\mathbf{M}_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{M}_n \end{bmatrix} \tag{3.6}$$

where $\mathbf{f}_i$, $\mathbf{x}_i$, $\mathbf{v}_i$, and m_i denote the force, location, velocity, and mass of the mass-point i respectively. 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. 3.5 is eventually reduced to finding $\Delta\mathbf{v}^{t+h}$ as follows:

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^{t+h} \tag{3.7}$$

Although the mass-spring based animation can be described as a simple equation like Eq. 3.7, 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:

$$\mathbf{f}^{t+h} = \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x}^{t+h} = \mathbf{f}^t + \mathbf{J} \Delta \mathbf{x}^{t+h} \tag{3.8}$$

where **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 **J** is $n \times n$, and each component of **J** is a 3×3 sub-matrix. As shown in Eq. 3.1, 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:

$$\mathbf{f}^{t+h} = \mathbf{f}^t + \mathbf{J}(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})h \tag{3.9}$$

With the force approximation shown in Eq. 3.9, the animation of the mass-spring model is finally reduced to a linear system solving as follows:

$$\Delta \mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^t + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^t + h^2\mathbf{M}^{-1}\mathbf{J}\Delta \mathbf{v}^{t+h} \tag{3.10}$$

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. 3.10 yields the linear system in the standard form of $\mathbf{A}\mathrm{x} = \mathrm{b}$ as follows:

Model	Mass-spring model
Force	Spring force (dependent only on the locational information)
Force derivative	partial derivative respect to the locations
Integration scheme	Implicit integration scheme (backward Euler method)
Problem characteristics	Linear system solving with huge but sparse matrix

Table 3.1: Summary of the problem

$$(\mathbf{I} - h^2\mathbf{M}^{-1}\mathbf{J})\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}\mathbf{f}^t + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^t \tag{3.11}$$

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

$$(\mathbf{M} - h^2\mathbf{J})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t \tag{3.12}$$

The stable animation of the mass-spring model can be achieved by solving the linear system shown in Eq. 3.12. 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.

Table 3.1 shows the summary of the problem. The problem is to animate the mass-spring based cloth model. The force can be easily computed with Hooke's law, and

it depends only on the locational information of the geometric components (i.e., mass-points). Therefore, the derivative of the force is a partial derivative respect to the locations. In order to guarantee the stability, implicit integration scheme was employed so that the problem is transformed into linear system solving with a huge sparse matrix.

Chapter 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 thesis also exploits the sparseness of the matrix and other properties in order to compute the next state as fast as possible. This thesis 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 3.12 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:

$$\mathbf{W} = \mathbf{M} - h^2\mathbf{J} \tag{4.1}$$

The components of the matrix $\mathbf{W}$ are 3×3 sub-matrices, which can be easily computed. The component 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. 3.6. 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 [DSB99, KCCP01] 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 thesis 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. 3.12 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 [DSB99]:

$$\begin{aligned} \tilde{\mathbf{f}_i^t} &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}\mathbf{v}_j^t + h\mathbf{J}_{ii}\mathbf{v}_i^t \\ &= \mathbf{f}_i^t + h \sum_{(i,j)\in E} \mathbf{J}_{ij}(\mathbf{v}_j^t - \mathbf{v}_i^t) \end{aligned} \tag{4.2}$$

Then, the linear system in Eq. 3.12 can be rewritten as follows:

$$\mathbf{W}\Delta\mathbf{v}^{t+h} = h\tilde{\mathbf{f}^t} \tag{4.3}$$

The linear system shown in Eq. 4.3 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. 4.3.

4.2 Solution Approximation in Linear Time

Due to the properties of the matrix $\mathbf{W}$, the linear system of Eq. 4.3 can be expressed as following n equations:

$$
\begin{aligned}
\mathbf{W}_{11}\Delta\mathbf{v}_1^{t+h} &= h\tilde{\mathbf{f}}_1^t + h^2 \sum_{(1,j)\in E} \mathbf{J}_{1j}\Delta\mathbf{v}_j^{t+h} \\
\mathbf{W}_{22}\Delta\mathbf{v}_2^{t+h} &= h\tilde{\mathbf{f}}_2^t + h^2 \sum_{(2,j)\in E} \mathbf{J}_{2j}\Delta\mathbf{v}_j^{t+h} \\
&\vdots \\
\mathbf{W}_{nn}\Delta\mathbf{v}_n^{t+h} &= h\tilde{\mathbf{f}}_n^t + h^2 \sum_{(n,j)\in E} \mathbf{J}_{nj}\Delta\mathbf{v}_j^{t+h}
\end{aligned}
\tag{4.4}
$$

All the equations in Eq. 4.4 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:

$$
\begin{aligned}
\Delta\mathbf{v}_i^{t+h} &= \mathbf{W}_{ii}^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h}) \\
&= (\mathbf{M}_{ii} - h^2\mathbf{J}_{ii})^{-1}(h\tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j)\in E} \mathbf{J}_{ij}\Delta\mathbf{v}_j^{t+h})
\end{aligned}
\tag{4.5}
$$

The matrices ($\mathbf{M}_i$, $\mathbf{J}_{ii}$, and $\mathbf{J}_{ij}$) involved in Eq. 4.5 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. 4.5. 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 thesis approximates the solution of the linear system by iteratively computing the velocity change of each mass-point with Eq. 4.5. In fact, Eq. 4.5 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:

$$\begin{aligned} \Delta \mathbf{v}_i^{t+h^{(0)}} &= \mathbf{W}_{ii}^{-1} h \tilde{\mathbf{f}}_i^t \\ \Delta \mathbf{v}_i^{t+h^{(k+1)}} &= \mathbf{W}_{ii}^{-1} (h \tilde{\mathbf{f}}_i^t + h^2 \sum_{(i,j) \in E} \mathbf{J}_{ij} \Delta \mathbf{v}_j^{t+h^{(k)}}) \end{aligned} \tag{4.6}$$

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:

$$\begin{aligned} \mathbf{W}_i &= \mathbf{M}_i - h^2 \mathbf{J}_{ii} \quad \in \mathbf{R}^{3 \times 3} \\ \mathbf{p}_i &= h \mathbf{W}_i^{-1} \tilde{\mathbf{f}}_i \quad \in \mathbf{R}^3 \end{aligned} \tag{4.7}$$

The iterative scheme in Eq. 4.6, then, can be more compactly expressed as follows:

$$\begin{aligned} \Delta \mathbf{v}_i^{t+h^{(0)}} &= \mathbf{p}_i \\ \Delta \mathbf{v}_i^{t+h^{(k+1)}} &= \mathbf{p}_i + h^2 \mathbf{W}_i^{-1} \sum_{(i,j) \in E} \mathbf{J}_{ij} \Delta \mathbf{v}_j^{t+h^{(k)}} \end{aligned} \tag{4.8}$$

The diagonal components of the matrix in Eq. 4.3 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:

$$\Delta \mathbf{v}_i^{t+h} \simeq \Delta \mathbf{v}_i^{t+h^{(1)}} = \mathbf{p}_i + h^2 \mathbf{W}_i^{-1} \sum_{(i,j) \in E} \mathbf{J}_{ij} \mathbf{p}_j \tag{4.9}$$

The fast approximation of the velocity change of each mass-point with Eq. 4.9 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. 4.9, 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 Approximation of Force Derivative for Stability

In order to keep the simulation process stable, an implicit integration scheme should be employed. As shown in Eq. 3.5, 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. 3.8. 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 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:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \kappa_{ij} \frac{|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|} \mathbf{I}_3 + \kappa_{ij} \frac{l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|} \left(\frac{(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathbf{T}}}{|\mathbf{x}_j - \mathbf{x}_i|^2} \right) \tag{4.10}$$

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

The force derivative shown in Eq. 4.10 does not incur any problem when the spring is stretched. However, the force derivative in Eq. 4.10 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:

$$\begin{aligned}
&\text{if } \; l_{ij}^0 < |\mathbf{x}_j - \mathbf{x}_i| \qquad (4.11)\\
&\qquad \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \kappa_{ij}\frac{|\mathbf{x}_j-\mathbf{x}_i|-l_{ij}^0}{|\mathbf{x}_j-\mathbf{x}_i|}\mathbf{I}_3 + \kappa_{ij}\cdot\frac{l_{ij}^0}{|\mathbf{x}_j-\mathbf{x}_i|}\left(\frac{(\mathbf{x}_j-\mathbf{x}_i)(\mathbf{x}_j-\mathbf{x}_i)^{\mathrm{T}}}{|\mathbf{x}_j-\mathbf{x}_i|^2}\right)\\
&\text{if } \; l_{ij}^0 \geq |\mathbf{x}_j - \mathbf{x}_i|\\
&\qquad \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \kappa_{ij}\left(\frac{(\mathbf{x}_j-\mathbf{x}_i)(\mathbf{x}_j-\mathbf{x}_i)^{\mathrm{T}}}{|\mathbf{x}_j-\mathbf{x}_i|^2}\right)
\end{aligned}$$

Henceforth, $\mathbf{J}_{ij}$ denotes the approximate force derivative shown in Eq. 4.11, and $\mathbf{J}_{ii}$ denotes the negated sum of the approximate force derivatives (i.e., $\mathbf{J}_{ii} = -\sum_{(i,j)\in E}\mathbf{J}_{ij}$).

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. 4.9 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:

$$\mathbf{f}_d = -C_d\mathbf{v} \tag{4.12}$$

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:

$$\begin{aligned} \mathbf{f}^{t+h} &= \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x}^{t+h} + \frac{\partial \mathbf{f}_d}{\partial \mathbf{v}}\Delta\mathbf{v}^{t+h} \\ &= \mathbf{f}^t + \mathbf{J}\Delta\mathbf{x}^{t+h} - C_d\mathbf{I}\Delta\mathbf{v}^{t+h} \\ &= \mathbf{f}^t + \mathbf{J}(\mathbf{v}^t + \Delta\mathbf{v}^{t+h})h - C_d\mathbf{I}\Delta\mathbf{v}^{t+h} \end{aligned} \tag{4.13}$$

Note that the damping affects only the diagonal components of the matrix. Therefore, the velocity change in Eq. 3.12 can be rewritten as follows:

$$(\mathbf{W} + hC_d\mathbf{I})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t \tag{4.14}$$

Eq. 4.14 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:

$$\mathbf{f}_d = -C_d \sum_{(i,j)\in E} (\mathbf{v}_i - \mathbf{v}_j) \tag{4.15}$$

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. 4.5 can be modified to include the damping effects without introducing any significant additional computations. For the simplicity, let us denote $(\mathbf{W}_i + n_i h C_d \mathbf{I}_3)^{-1}$ 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:

$$\Delta \mathbf{v}_i^{t+h} = \mathbf{W'}_i^{-1} (h \tilde{\mathbf{f}}_i^t + h^2 \sum \mathbf{J}'_{ij} \Delta \mathbf{v}_j^{t+h}) \tag{4.16}$$

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. 4.7, and the approximate solution can be obtained as

follows:

$$\Delta \mathbf{v}_i^{t+h} \simeq \mathbf{W'}_i^{-1} h \tilde{\mathbf{f}}_i^t + h^2 \mathbf{W'}_i^{-1} \sum \mathbf{J}'_{ij} (\mathbf{W'}_j^{-1} h \tilde{\mathbf{f}}_j^t) \tag{4.17}$$

Eq. 4.17 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.

4.5 Optimization of Storage and Computation

Iterative update or approximation with Eq. 4.8 and 4.9 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.

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

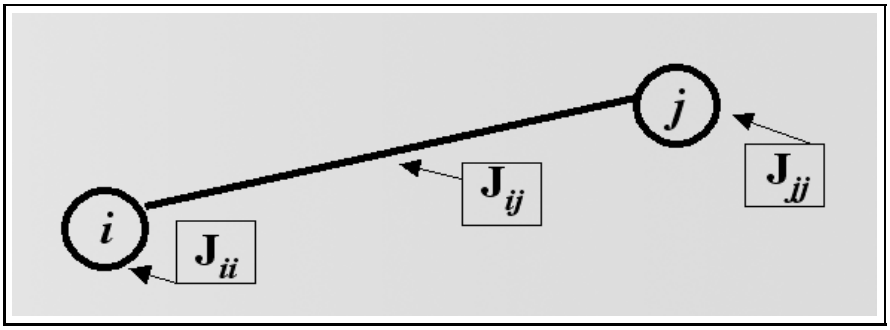


Figure 4.1: Storage of Jacobian matrix

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-points and spring objects as shown in Fig. 4.1. The lower triangular components in sub-matrices are not stored because all the sub-matrices in our problem are symmetric. All the other matrices have the same properties as the matrix $\mathbf{J}$ so that the identical strategy can be used for the storage of the matrices.

The animation technique proposed in this thesis is finally reduced to finding the velocity change of each mass-points. The velocity change computation is composed of four modules as shown in Table 4.1.

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

Begin ComputeVelocityChange
1. Compute $\mathbf{J}$ 2. Compute $\mathbf{W}_i^{-1}$ for each vertex 3. Compute $\mathbf{p}_i$ for each vertex 4. Compute $\Delta\mathbf{v}_i$ for each vertex
End ComputeVelocityChange

Table 4.1: The computation of velocity change

done as shown in Table 4.2. For the simplicity, the damping was not considered in the pseudo-code.

In the pseudo-code, $n(E)$ denotes the number of spring edges, and $E(e)$ denotes the e-th spring edge in the spring set E. Two mass-points linked with the spring edge $E(e)$ are denoted as $E(e).v_1$ and $E(e).v_2$. The Jacobian computation module initializes the diagonal components of the Jacobian matrix $\mathbf{J}_{ii}$ by setting all nine elements as 0. The module, then, computes the off-diagonal components of the Jacobian matrix with Eq. 4.10. As mention before, $\mathbf{J}_{ij}$ is identical with $\mathbf{J}_{ji}$, and $\mathbf{J}_{ij}$ can be regarded as an attribute of the spring edge that links the mass-point i and j. Therefore, the time complexity of the computation of the off-diagonal components is linearly proportional to the number of spring edges. As a diagonal component is the negated sum of the off-diagonal components in the same row, all the diagonal components can be also computed with the same time complexity.

The second module for computing the inverse of the matrix $\mathbf{W}_i$ is straightforward

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 $n(E)$) [

$i \leftarrow E(e).v_1 \qquad j \leftarrow E(e).v_2$

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

$\mathbf{J}_{ii} \leftarrow \mathbf{J}_{ii} - \mathbf{J}_{ij}$

$\mathbf{J}_{ii} \leftarrow \mathbf{J}_{ii} - \mathbf{J}_{ij}$

]

End ComputeJacobian

Table 4.2: Jacobian matrix computation

and can be performed in O(n) time because $\mathbf{W}_i$ can be regarded as an attribute of the mass-point i and the module simply computes the inverse matrices of n 3×3 matrices.

The third module computes the $\mathbf{p}_i$ that is defined in Eq. 4.7. The computation of $\mathbf{p}_i$ for each mass-point requires the consideration of the velocities of the linked mass-points. Therefore, the module can be implemented as shown in Table 4.3. Damping is ignored again in this pseudo-code. The module first initializes the $\mathbf{p}_i$ vector for each mass-point i. The vector $\mathbf{p}_i$ can be computed by summing the multiplication of the 3×3 matrix $h^2\mathbf{J}_{ij}$

Begin Compute_$\mathbf{p}_i$

for (i : from 1 to n) $\quad \mathbf{p}_i \leftarrow [0,0,0]^{\mathbf{T}}$

for (e : from 1 to $n(E)$) [

$i \leftarrow E(e).v_1$

$j \leftarrow E(e).v_2$

$\mathbf{p}_i \leftarrow \mathbf{p}_i + h^2\mathbf{J}_{ij}(\mathbf{v}_i - \mathbf{v}_j)$

$\mathbf{p}_j \leftarrow \mathbf{p}_j + h^2\mathbf{J}_{ij}(\mathbf{v}_j - \mathbf{v}_i)$

]

for (i : from 1 to n) $\mathbf{p}_i \leftarrow \mathbf{W}_i^{-1}(\mathbf{p}_i + h\mathbf{f}_i)$

End Compute_$\mathbf{p}_i$

Table 4.3: Computation of $\mathbf{p}_i$ vectors

and the 3 dimension vector which represents the velocity differnce between two linked mass-points i and j. Therefore, the computation time of the vector $\mathbf{p}_i$ for all the mass-point i is also linearly proportional to the number of spring edges. The fourth module that computes the velocity change of each mass-point can be simply implemented with the already obtained $\mathbf{W}_i$ and $\mathbf{p}_i$.

Table 4.4 shows the summary of the proposed animation method. The actual problem can be defined as a linear system solving with a huge sparse and symmetric block matrix. The stability was guaranteed by the implicit integration scheme and the stabi-

Actual problem	Linear system with a sparse, symmetric, and block matrix
Stability	Guaranteed by - implicit integration scheme - stabilized force derivative
Solution approximation	Iterative update with the consideration of linked mass-points
Physical Plausibility	Not impaired
Update complexity	Efficient: O(n)
Storage optimization	Matrix stored as attributes of mass-points and springs
Damping	No significant overhead cost

Table 4.4: Realtime animation method

lized force derivative. The physical plausibility was not significantly impaired because no severe approximation was used. The method is not only stable and plausible but also efficient because of the iterative update scheme that considers only the small number of linked mass-points. Storage was optimized and damping was also integrated without any significant overhead computation.

Chapter 5

Realism Enhancement and Collision

The motion generated with Eq. 4.9 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.

Integrating the gravitational force into the cloth animation system is straight-forward and it does not require any careful treatment if the stability of the system is guaranteed. Since the stability of the proposed animation technique is inherent from the implicit integration method, the consideration of gravitational force is a trivial problem. Apart from the gravity, the most significant external force on cloth object is the force caused by the interaction between the cloth and air. Even if there is no wind, the movement of the cloth object causes the interaction with air. The interaction, as a consequence, exerts drag and lift forces on the cloth model. Without the consideration of the external forces, the plausibility of the animation system will be severely impaired.

Another important issue in the realistic cloth animation is collision handling. The cloth animation cannot be realistic without handling the collision. Especially when the cloth animation is supposed to be used for character animation, collision handling is as crucial as numerical integration with regard to the computational cost. Usually, the collision handling is an extremely expensive task. Moreover, deformable objects produces a special collision case where some part of the model collides with another part of the same model. Efficient handling of the self-collision is also a difficult problem. Although many researchers have tried to tackle this self-collision handling problem, it still plays the role as an major obstacle to the real-time animation. In this chapter, the collision is dealt with in order to increase the realism of the animation, and an efficient collision handling method is proposed in order to be used in real-time animation systems.

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 [Bra88]:

$$|\mathbf{f}^{drag}| = \frac{1}{2} C_D \rho |\mathbf{v}^{wind}|^2 S \sin\theta \tag{5.1}$$

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:

$$|\mathbf{f}^{lift}| = \frac{1}{2} C_L \rho |\mathbf{v}^{wind}|^2 S \cos\theta \tag{5.2}$$

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 of 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. 5.1, the magnitude of the drag force can be implemented as follows:

$$\mathbf{f}_i^{drag} = -K_D |\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i| |\mathbf{v}_i|^2 \hat{\mathbf{v}}_i \tag{5.3}$$

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:

$$\begin{aligned} \mathbf{U}_i &= (\hat{\mathbf{n}}_i \times \hat{\mathbf{v}}_i) \times \hat{\mathbf{v}}_i \ , \ if \ \ \hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i > 0 \\ & (-\hat{\mathbf{n}}_i \times \hat{\mathbf{v}}_i) \times \hat{\mathbf{v}}_i \ , \ otherwise \end{aligned} \tag{5.4}$$

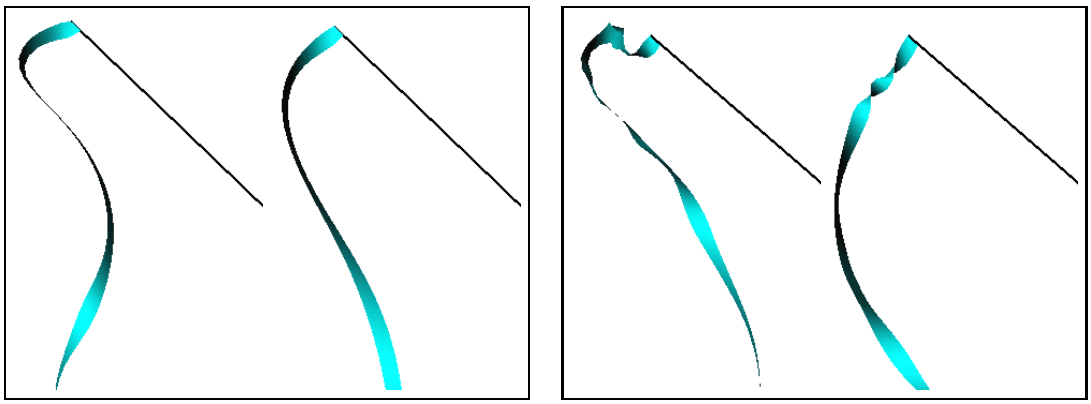
Once the direction of the lift force was determined, the lift force can be also determined only by computing the magnitude of the lift force. Therefore, the lift force $\mathbf{F}_i^{lift}$ on the i-th mass-point as follows:

$$\mathbf{f}_i^{lift} = (K_L |\cos\theta| |\mathbf{v}_i|^2)\mathbf{U}_i \tag{5.5}$$

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. 5.1 and Eq. 5.2, the magnitude of the drag force 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:

$$\mathbf{f}_i^{lift} = (K_L(1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i|)\mathbf{v}_i|^2)\mathbf{U}_i \tag{5.6}$$



(a) vacuum motion (b) motion in air

Figure 5.1: Fluid influence: (a) motion in vacuum (b) motion in the air

Fig. 5.1 shows the effect of the drag force and lift force. Fig. 5.1 (a) shows the sequence of ribbon animation when the drag and the lift forces are not taken into account, while (b) shows the movement of the ribbon affected by the drag and the lift forces. As shown in the figures, the drag and lift forces generate more natural animation.

5.2 Efficient Collision Handling

Collision detection and response are also essential for realistic animation, and collision resolution for cloth animation has been one of the major research subjects in computer graphics literature because it can be a computational bottleneck in interactive animation of cloth [VMT94, Pro97, HMB01]. Although the real-time animation techniques proposed in chapter 4 enables the system to update the state of cloth objects in real-time, the simulation method did not consider collision handling. Collision handling is an extremely expensive task if the objects that collide with each other are geometrically com-

plex. Therefore, real-time cloth animation system should detect and resolve the collision in an efficient way without violating the interactiveness of the system and the realism of the animation result.

In order to detect collisions between objects, the geometric relations of all the geometric components should be investigated. Therefore, an accurate collision detection method has the worst case time complexity of O(n^2) where n is the number of the geometric components. Therefore, real-time collision detection is almost impossible. Various hierarchical methods for efficient collision detection have been proposed [GLM96, KGL+98, LKC98]. Even with the hierarchical structures for alleviating the computational burden of the collision detection, real-time collision detection is still difficult since cloth models are usually composed of too large number of collision entities.

Therefore, in real-time environments, it is almost impossible to detect and resolve the collision between a finely discretized cloth model and complex geometric objects. Although hierarchical structures can be exploited to increase the collision detection process, the exact collision detection cannot avoid O(n^2) time complexity in the worst case. Some efficient collision handling techniques have been proposed by a few research groups [Pro97, VMT94, HMB01]. However, real-time collision handling is still a hard problem in real-time cloth animation environments.

With the currently available common hardware environments, the real-time cloth animation cannot be achieved without simplifying the character model or cloth model. In this thesis, character models are simplified as combination of simple geometric objects such as spheres and cylinders, and collision between a cloth object and a character model

is detected and resolved in real-time.

Let us denote the positional vectors of two end points of the cylinder axis as $\mathbf{x}_s$ and $\mathbf{x}_e$, and the positional vector of a mass-point i as $\mathbf{x}_i$. $\mathbf{x}_p$ denotes the projected point of $\mathbf{x}_i$ on the axis of the cylinder. The mass-point i is to be inspected whether it collide with the cylinder or not. The scalar values l, r and d denote the length of the cylinder, the radius of the cylinder, and the distance between the mass-point and the axis of the cylinder respectively. The collision normal vector $\mathbf{n}_{col}$ is simply approximated as the unit vector along the vector $\mathbf{x}_i - \mathbf{x}_p$, and the unit vector $\mathbf{u}$ was defined as $(\mathbf{x}_e - \mathbf{x}_s)/|\mathbf{x}_e - \mathbf{x}_s|$.

Apparently, the collision occurs when d has the smaller value than r and $0 < (\mathbf{x}_i - \mathbf{x}_s) \cdot \mathbf{u} < l$. The collision resolution is to keep d to be always larger than r. In other words, collision detection is simply the comparison of d and r, and collision resolution is to locate the mass-point out of the cylinder.

The responses after the collision detection should enforce the mass-point to be located out of the cylinder. If the mass-point enters the internal volume of the cylinder, collision resolution module moves the mass-point in the direction of the collision normal vector $\mathbf{n}_{col}$. The simplest positional adjustment of the penetrating mass-point can be achieved by locating the mass-point at $r\mathbf{n}_{col} + \mathbf{x}_p$. However, the positional adjustment does not guarantee the realistic movement of colliding movement because an actual colliding part of cloth model also changes its velocity.

In order to adjust the velocity of the colliding mass-point, the velocity of the colliding point of the character model should be taken into account. If the relative velocity between the colliding mass-point and the colliding point of the character model indicates that the

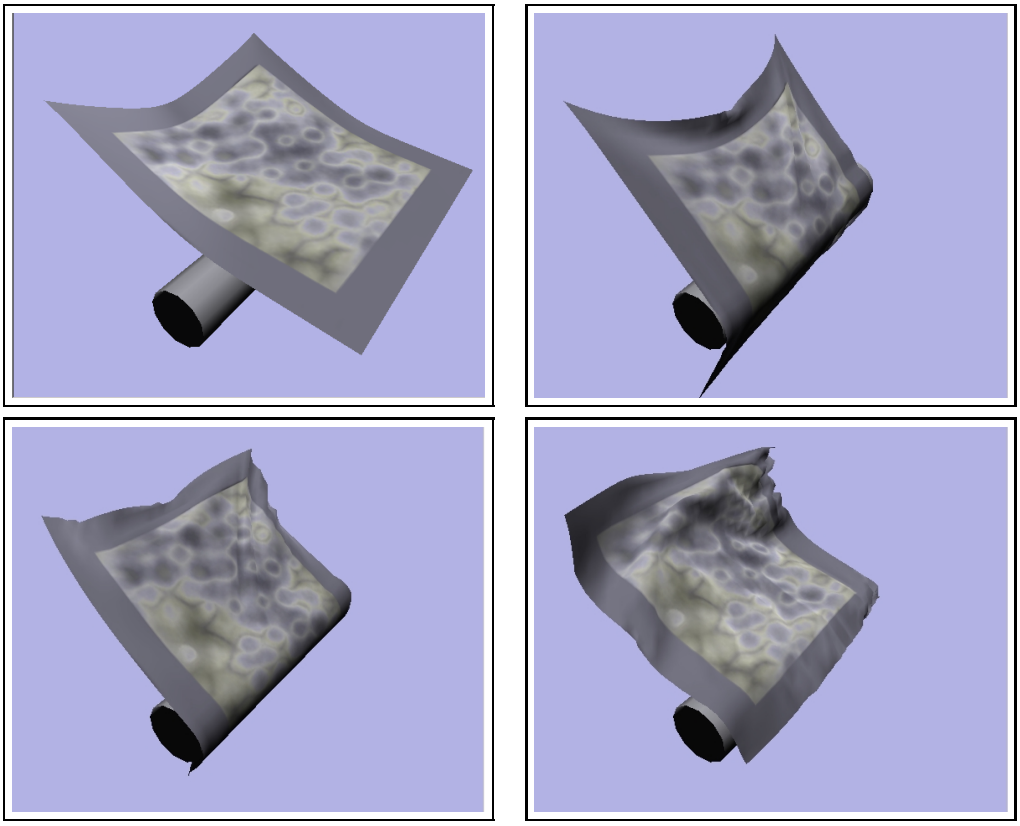


Figure 5.2: Collision handling result

mass-point keeps on moving into the cylinder, the velocity of the mass-point should be adjusted in order to slide on the cylinder. If, in the other hand, the relative velocity indicates that the colliding mass-point would go outside the cylinder, no velocity adjustment is necessary. Therefore, the colliding point of the cylinder and its velocity should be computed.

Let us denote the velocities of $\mathbf{x}_s$, $\mathbf{x}_e$, and the projected point $\mathbf{x}_p$ as $\mathbf{v}_s$, $\mathbf{v}_e$, and $\mathbf{v}_p$ respectively. The velocity at $\mathbf{x}_p$ can be simply approximated with linear interpolation between $\mathbf{v}_s$ and $\mathbf{v}_e$, and the relative velocity $\mathbf{v}_{rel}$ was defined as $\mathbf{v}_i - \mathbf{v}_p$.

Whether the mass-point will move into the cylinder or not can be easily investigated

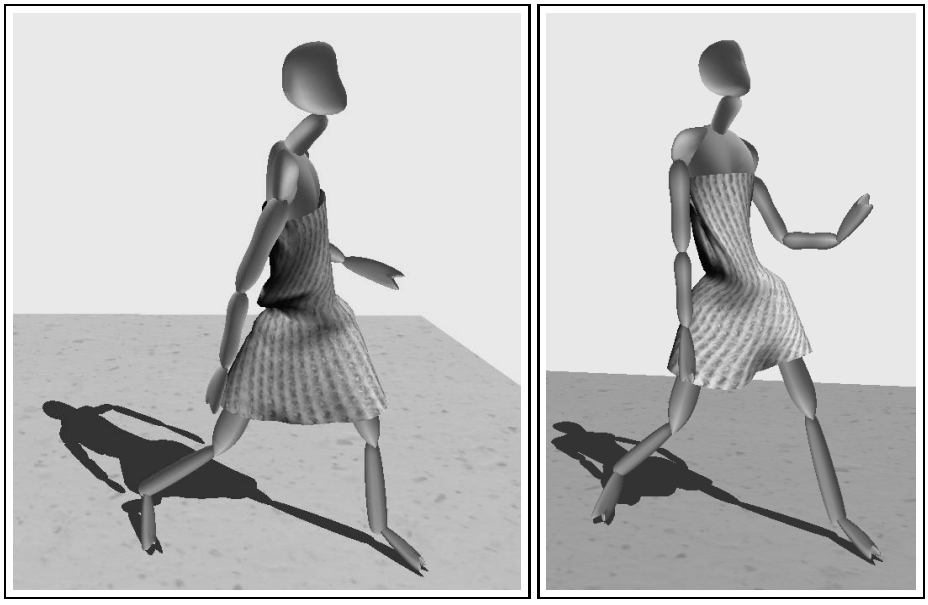


Figure 5.3: A dressed virtual model

by considering the relation between the relative velocity and the collision normal vector $\mathbf{n}_{col}$. If the inner product of $\mathbf{n}_{col}$ and the relative velocity is negative, the mass-point will penetrate the surface of the cylinder. Therefore, the velocity of the mass-point should be adjusted to be blocked by the surface. Therefore, the adjusted velocity of the mass-point can be computed as follows:

$$\mathbf{v}_i^{adjusted} = \mathbf{v}_i - (\mathbf{n}_{col} \cdot \mathbf{v}_{rel})\mathbf{n}_{col} \tag{5.7}$$

Fig. 5.2 shows the result when the simple collision handling technique was applied. As shown in the figure, the result of the collision handling method is plausible enough to be used in real-time applications such as games or interactive designing systems.

The method was also tested for the character animation. Fig. 5.3 shows the result when the collision handling techniques have been applied to a dressed virtual character.

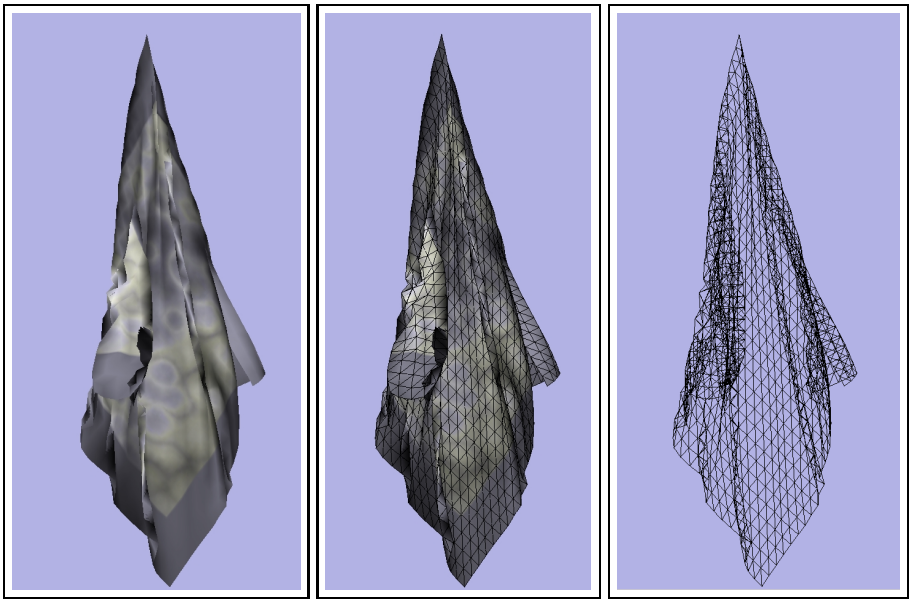


Figure 5.4: Result without the consideration of the self-collision handling

The shape of the model was approximated with cylinders and spheres. As shown in the figure, the technique produces plausible interaction between the cloth model and the character model.

5.3 Self-Collision Handling with O(n) Complexity

Dissimilarly to the rigid objects, deformable objects such as cloth can undergo special collision case where a part of the model collides with another part of the same model. This special collision case is known as self-collision situation, and collision handling techniques for cloth animation should deal with the self-collision.

Fig. 5.4 shows the possible result when the self-collision was not taken into account. As shown in the figure, the self-collision severely defects the realism of the cloth anima-

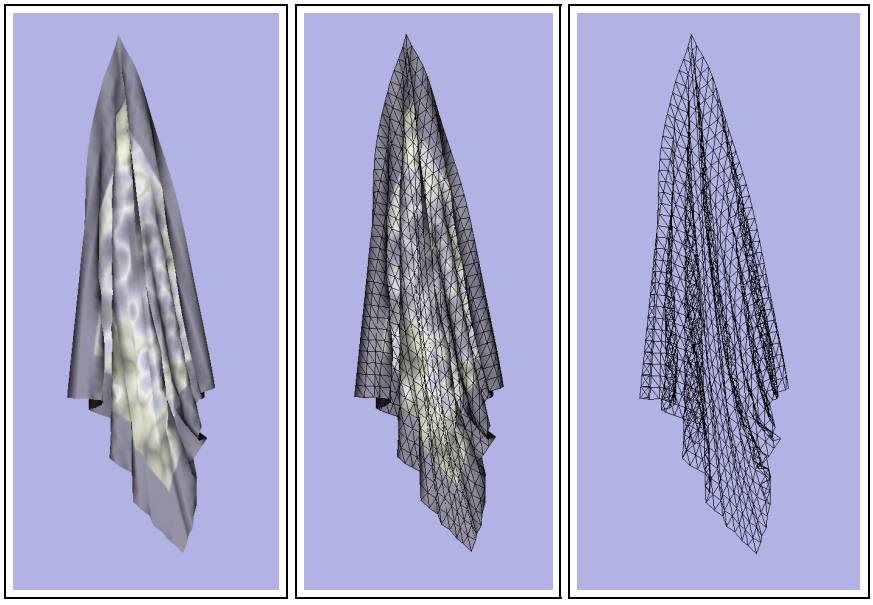


Figure 5.5: Result with the consideration of the self-collision handling

tion. Moreover, the self-collision handling is not a simple problem that can be solved in linear time. Therefore, handling the self-collision is one of the most important issues in cloth animation. Fig. 5.5 shows the result when the self-collision was detected and resolved. As shown in the figure, handling the self-collision drastically increase the realism of the cloth animation.

Brute-force techniques cannot avoid O(n^2) time complexity where n is the number of entities that can collide with each other. However, in the most cases, every possible pair of the entities does not need to be investigated.

The most prominent works on the self-collision handling have been proposed by Volino $et\ al.$ [VMT94] and Provot [Pro97]. Volino $et\ al.$ showed that self-collision detection can be efficiently performed by investigating the normal vector of the each face and

measuring the geometrical regularity, and Provot improved the idea [VMT94, Pro97]. Their techniques construct the hierarchical structure of the normal vectors, and avoid the unnecessary comparison.

In this thesis, an efficient self-collision handling method is proposed. This method uses a hash function to store n mass-points of a cloth model in 3-dimensional space into n buckets. Each bucket contains the collision candidates, and collision detection process is independently performed within each bucket. Therefore, the method can efficiently detect the self-collision. The computation of the exact collision response can be another bottle-neck in cloth animation. In order to guarantee the real-time performance, a collision avoidance method was employed instead of the exact collision resolution. The collision avoidance method represents each mass-point as a sphere. Therefore, the collision between two mass-points can be detected by measuring the distance between the mass-points, and the collision handling can be also efficiently performed by simulating the collision between the spheres.

The collision avoidance method constructs the bounding box of the cloth model, and the bounding box is divided into cubic cells along the coordinate axis. A mass-point will then be located in one of the cells. It is obvious that each cells can be described as three integers (i_x,i_y,i_z). The size of the cubic cell should be minimized in order to reduce the number of collision candidates. However, the number of the cells increases as the size of each cells decreases. If the number of the cells are large, the collision detection process should be performed more often and the storage requirement will be increased. In order to alleviate the computational and storage overhead, the 3 dimensional cell index (i_x,i_y,i_z)

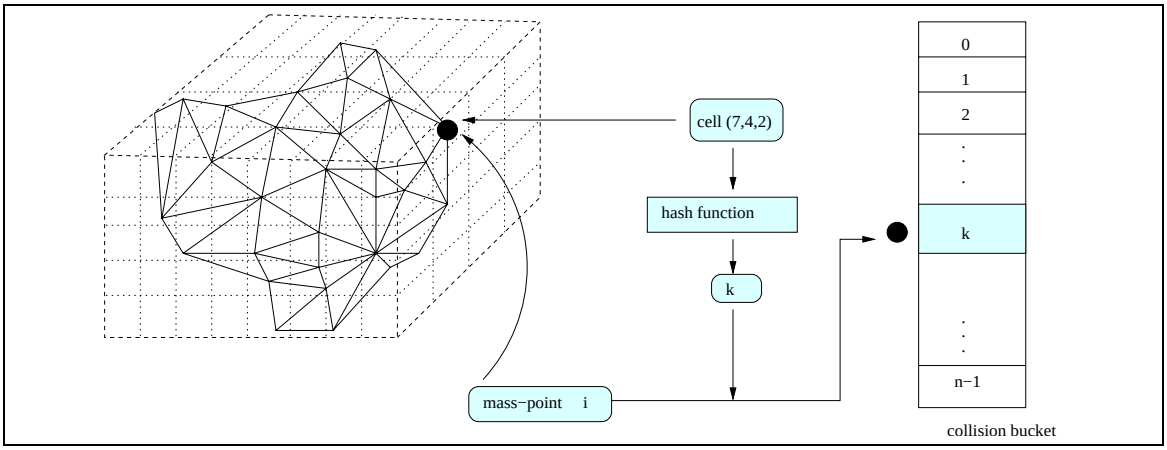


Figure 5.6: Storing a vertex index in a bucket

is transformed into 1 dimensional space. In other words, a hash function $k = h(i_x, i_y, i_z)$ has been devised where i_x, i_y, i_z, and k are all non-negative integers, and k is bound under n. In order to efficiently detect the collision, n buckets are prepared and each bucket contains collision candidates. The value k can then be used as bucket index, and each mass-point located in the cell (i_x,i_y,i_z) is stored in the bucket k as shown in Fig. 5.6.

Since n mass-points are stored in n buckets, each bucket contains O(1) mass-points in average. Therefore, the collision detection process can be performed in O(n) time. If all the mass-points are located in a single cell, the computational complexity can increase up to O(n^2). However, the collision handling method in this thesis enforce each mass-point not to approach to other mass-points within a certain distance so that the worst case cannot occur.

The serious problem of this method is that two mass-points can approach close to each other without being located in an identical cell. In order to avoid such an undesirable situation, the method employs two different cell division methods with different

Air interaction	Drag and lift force as external forces
Collision with object	Colliding object was approximated as cylinder or sphere
Self-collision Detection	Bucketing for O(n) collision detection
Self-collision Handling	Collision avoidance

Table 5.1: Realism enhancement for real-time environments

cell sizes. Therefore, each mass-point has two different cell indices (i_{x_1}, i_{y_1}, i_{z_1}) and (i_{x_2}, i_{y_2}, i_{z_2}), and the hash function then generates two different bucket indices k_1 and k_2. With the different bucket indices, the mass-point is stored in two different buckets. Although this method does not guarantee that two close mass-points will be stored in an identical bucket, the undesirable situation becomes rare during the animation. Moreover, the undesirable situations are usually resolved at the next animation step and it is almost impossible for a mass-point to penetrate the other cloth part without being detected.

Table 5.1 shows the summary of the realism enhancement techniques of this thesis. Air interaction was implemented by considering the drag and lift force as external forces. In order to efficiently handle the collision between the cloth model and other objects, colliding objects were simplified with cylinders and spheres. Self-collision handling was performed in real-time by allocating each mass-point into buckets according to its locational information. Accurate resolution of the self-collision was not employed because of the heavy computational cost. Instead, simple and effective collision avoidance method was used.

Chapter 6

Experimental Results

The proposed method has been successfully implemented for an interactive cloth animation system.

Table 6.1 shows the computational performance of the proposed method. The proposed method was tested in the medium level PC environments (Pentium II 600 MHz), and the method showed real-time performance for the animation of complex models with thousands of polygons even with the medium level PC environments. As shown in Table 6.1, it was possible to update the state of a complex cloth model with 3025 polygons (4-

number of vertices	148	405	1566	2128
number of edges	404	1161	4590	6231
number of triangles	257	757	3025	4104
update time (sec)	0.0008	0.0023	0.0108	0.0158

Table 6.1: Computational performance of the method

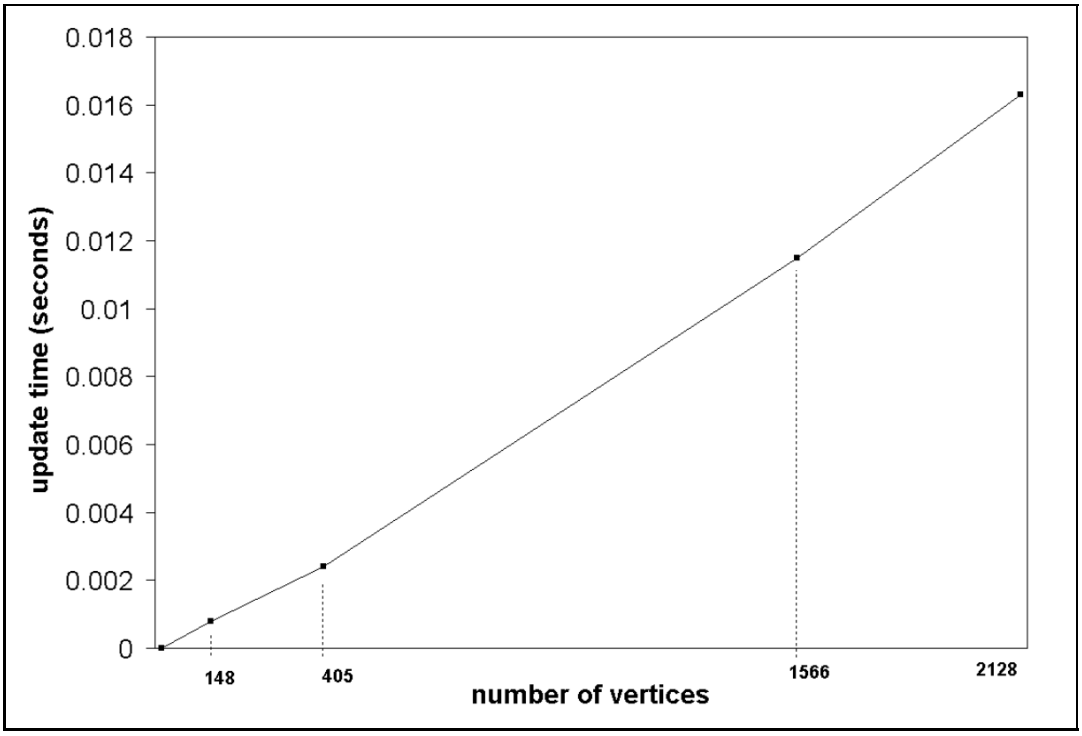


Figure 6.1: State update time according to the number of mass-points

th column) in 0.0115 seconds. Since the proposed method is stable enough to use large steps, it was also possible to generate the real-time animation of the complex model with 3025 polygons by using time-steps with the size of 1/120 second.

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. 6.1 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.

Fig. 6.2 also shows the time required for computing the next state of the cloth model.

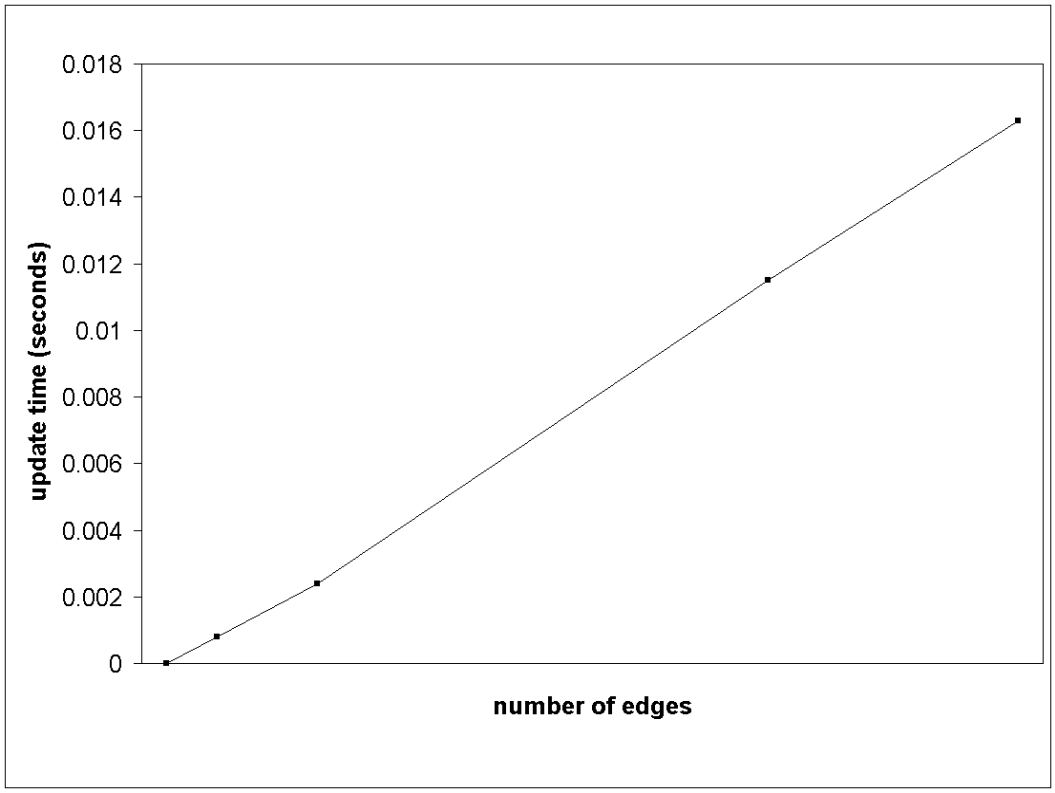


Figure 6.2: State update time according to the number of edges

In Fig. 6.2, the required time is compared with the total number of edges in the cloth model. As shown in the figure, the update time is also linearly proportional to the number of edges. Since the proposed method is based on the implicit integration scheme, the method was stable enough to be applied to stiff cloth model. Therefore, the method does not need to use small time steps which disables the system to efficiently generate the animation results. Therefore, the experimental data shown in Fig. 6.1 and Fig. 6.2 clearly show that the proposed method is optimal solution for the real-time cloth animation.

Although the proposed method is stable and efficient, another important criterion

Figure 6.3: Real-time animation of complex model

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. Fig. 6.3 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 applied to a flag model composed of 2048 triangles. As shown in the Fig. 6.3, 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.

In order to investigate the plausibility of the proposed method, two kinds of error evaluation functions were devised: Positional error $\mathcal{E}$ and angular similarity $\mathcal{S}$. Let us denote $\Delta \mathrm{x}^{ref}$ 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:

$$\mathcal{E}_i(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{|\Delta \mathbf{x}_i^{ref} - \Delta \mathbf{x}_i^{exp}|^2}{|\Delta \mathbf{x}_i^{ref}|^2} \tag{6.1}$$

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 mass-points as follows:

$$\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{\sum_{i=1}^{n} \mathcal{E}_i(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})}{n} \tag{6.2}$$

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 thesis, 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 of 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:

$$\mathcal{S}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp}) = \frac{\sum_{i=1}^{n} (\Delta \hat{\mathbf{x}}_i^{ref} \cdot \Delta \hat{\mathbf{x}}_i^{exp})^2}{n} \tag{6.3}$$

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

Table 6.2: $\mathcal{E}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$: 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.

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 $\Delta \hat{\mathbf{x}}_i^{exp}$ 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 6.2 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 6.3 shows the $\mathcal{S}$ values under the same condition as Table 6.2. As shown in the table, the $\mathcal{S}$ values are almost 1 at every check point. Therefore, the direction of

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

Table 6.3: $\mathcal{S}(\Delta \mathbf{x}^{ref}, \Delta \mathbf{x}^{exp})$: 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.

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. 6.4 shows the $\mathcal{E}$ and $\mathcal{S}$ values measured at 43 check points under the same simulation condition as Table 6.2 and Table 6.3. As shown in the figure, the motion produced by the proposed method has small $\mathcal{E}$, 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 6.4 and Fig. 6.5 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

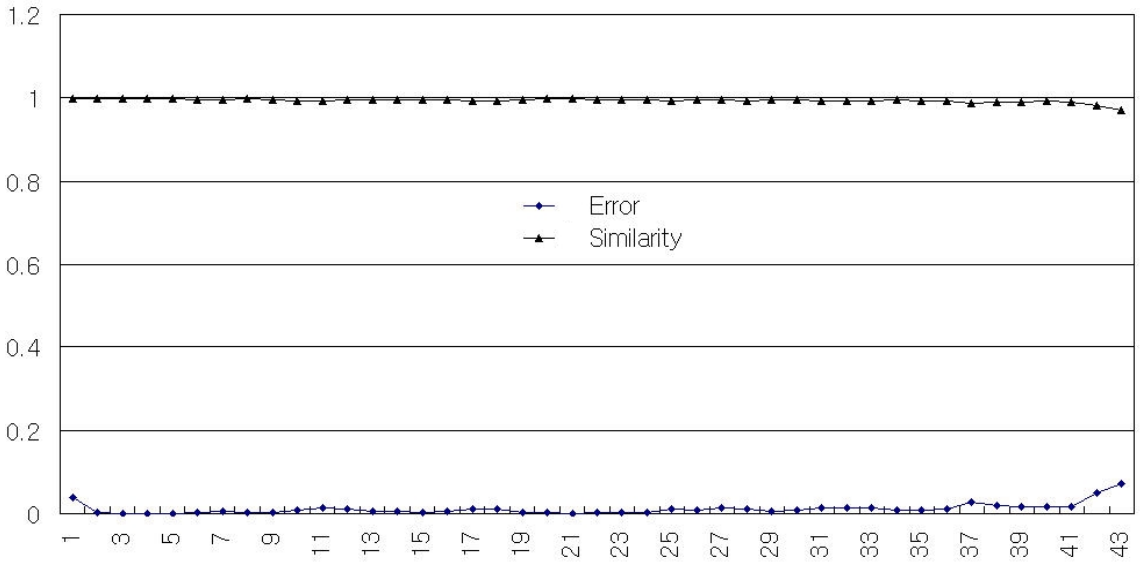


Figure 6.4: $\mathcal{E}$ and $\mathcal{S}$ values for 43 check points: reference motion with 1/30000 sec time steps and experimental motion with 1/150 sec, and time interval between the check points is 1/30 sec.

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

Table 6.4: $\mathcal{E}$ 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.

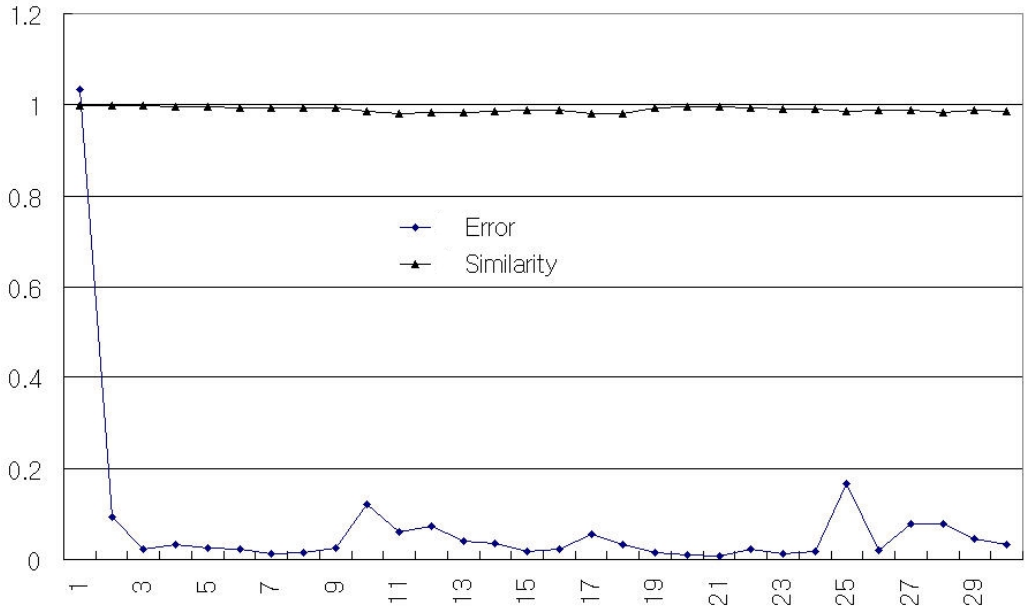


Figure 6.5: $\mathcal{E}$ and $\mathcal{S}$ values for 30 check points: reference motion with 1/30000 sec time steps and experimental motion with 1/30 sec, and time interval between the check points is 1/30 sec.

only one iteration. In other words, the iteration parameter k in Eq. 4.6 was set to be 1.

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 and applying the realism enhancement techniques described in Chapter 5. Fig. 6.6 is the animation results when the drag force and lift force are considered. As seen in the figure, the drag and lift force model generated plausible scene.

The interaction between the cloth model and air significantly increases the realism of the result. Fig. 6.7 shows the animation results when the cloth model was released in the gravity field. Fig. 6.7 (a) shows the result when no interaction between the air

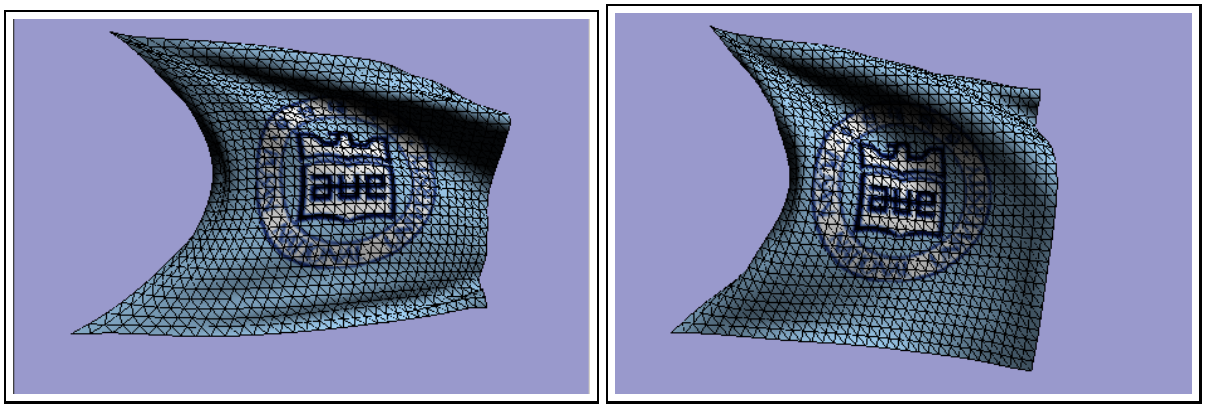
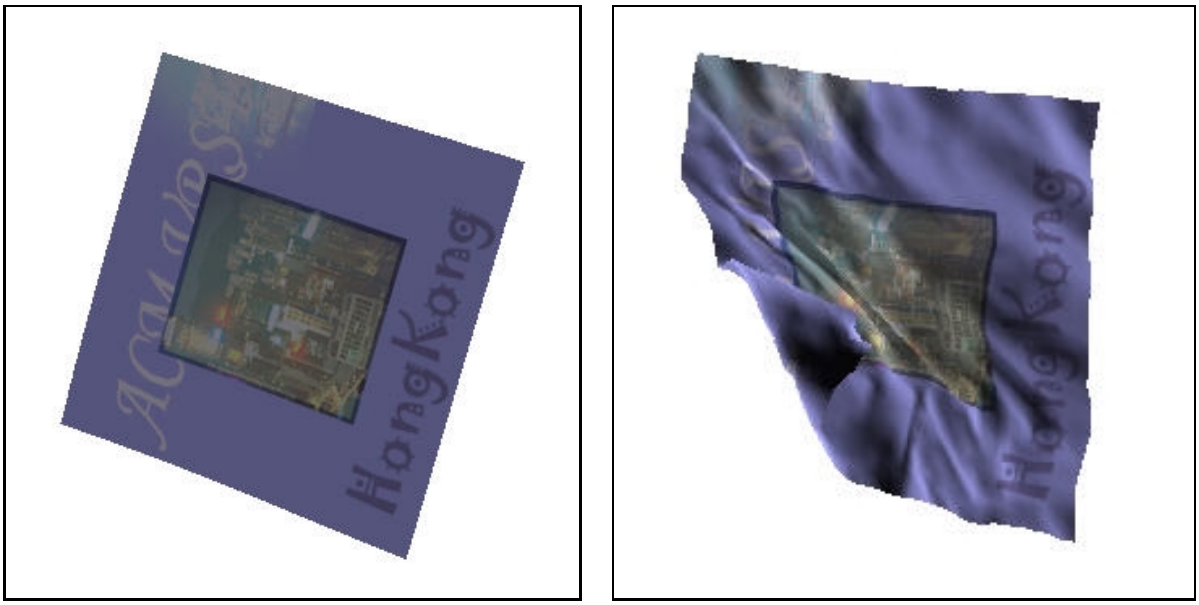


Figure 6.6: Flag in wind

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 techniques in Chapter 5. The result animation demonstrates that the consideration of the interaction between the air and the cloth model generates more realistic and plausible results. As show in Fig. 6.6 and Fig. 6.7, The proposed method was successfully integrated with the realism enhancement techniques.

The efficient collision handling techniques were also experimented. Fig. 6.8 shows the animation sequence with the efficient cloth-cylinder collision handling techniques. The complex cloth was placed on three cylinders and a sphere. Because of the simplicity of the collision handling technique, it is natural that the collision resolution was performed in real-time. Apart from the efficiency, the method generated very plausible interaction between the cloth model and the simple geometric objects.

Fig. 6.9 shows the animation sequence with the proposed self-collision handling method. By employing the bucketing method explained previously, the self-collision was also detected and resolved in O(n) time. Therefore, self-collision handling did not



(a) (b)

Figure 6.7: 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

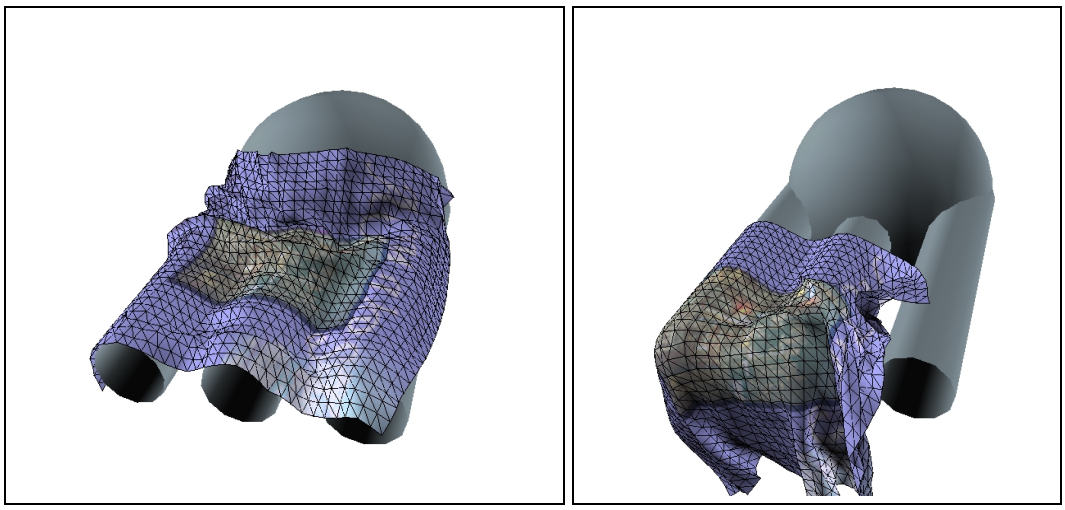


Figure 6.8: Animation sequence with collision with simple objects

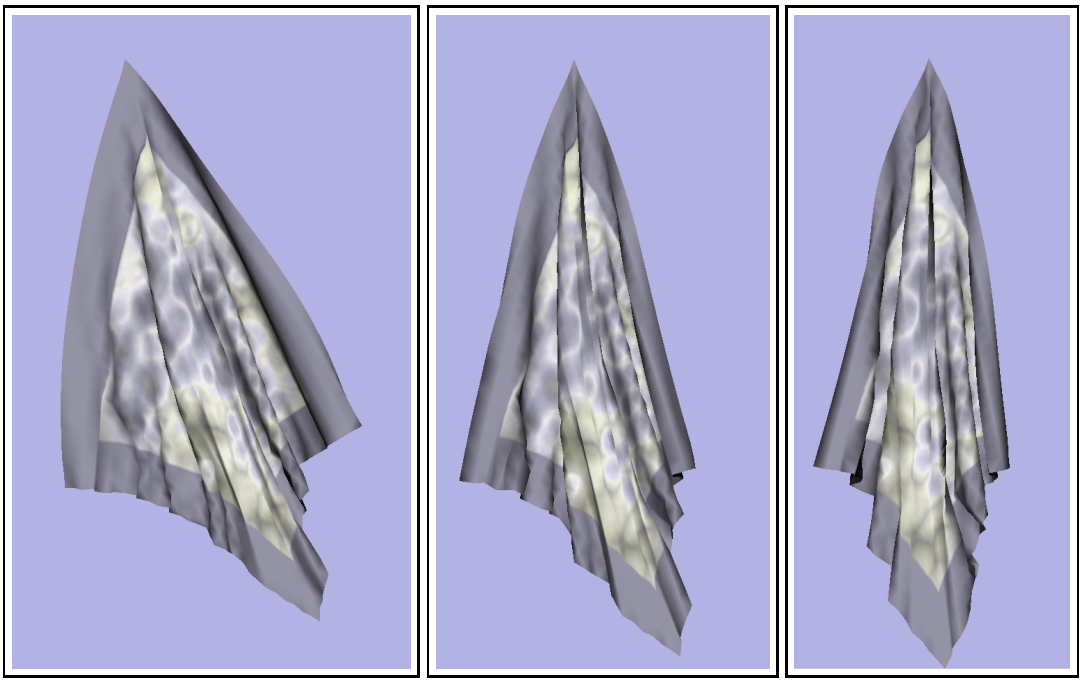


Figure 6.9: Animation sequence with self-collision handling

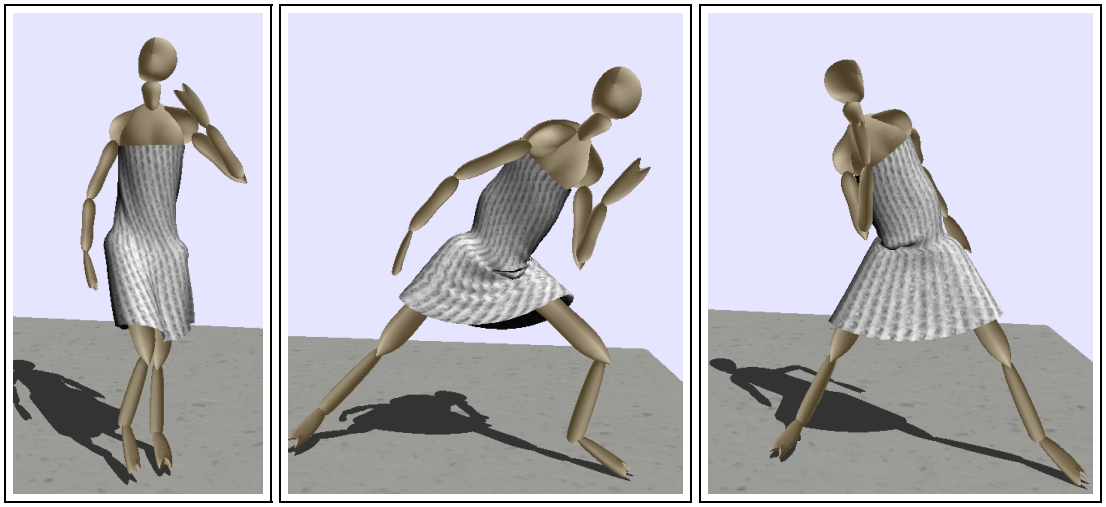


Figure 6.10: Interactive animation of dressed character

play a role as a computational obstacle in the generation of the real-time animation for the cloth model shown in Fig. 6.9.

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. 6.10 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.

Chapter 7

Conclusion

In this thesis, 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.

In the early stage of cloth animation research, geometric approaches have been widely employed because of the limitation of the computing environments. However, the geometric approaches could not reproduce the realistic motion of actual cloth because the methods are not based on the physics. As the rapid improvement of the computing environments has been made in the recent years, physically-based modeling has been employed in most cloth animation systems. Although the physically-based modeling provides realistic animation results, computational cost of the correct simulation is still too expensive for the real-time environments.

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 an efficient 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. However, the implicit integration scheme is finally reduced to a linear system solving problem that involves O(n^2)-sized matrix where n is the number of elements of the cloth model. Fortunately, the matrix is in general so sparse that 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.

In order to reduce the computational cost, some hybrid methods and approximation methods have been proposed. However, those methods usually does not guarantee the physical plausibility. 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 thesis provides a 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. Although the stability is the essential property, it is not a sufficient property for the real-time animation. The stability of the implicit method requires heavy computational burdens to obtain the next state of each mass-point because it casts the original problem into a linear system problem. However, the linear system for mass-spring animation derived by the implicit method has many properties, which can be exploited for improving the performance of the animation system. The proposed method efficiently approximates the solution of the linear system by taking advantage of these properties. The proposed method independently computes the next state of each mass-point by considering only linked mass-points so that the computation of the next state of the whole cloth model can be achieved in O($n + e$) time where n is the number of the mass-points and e is the number of the spring edges in the cloth model. 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

	stability	update complexity	physical justification
Explicit	no	O(n)	high
Implicit	yes	O($n^{1.5}$) ~	high
Approximate IM	yes	O(n) ~ O(n^2)	poor
Hybrid	unknown	O(n) ~	poor
Proposed Method	yes	O(n)	high

Table 7.1: Comparison of physically-based cloth animation methods

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.

Table 7.1 shows the comparison between the proposed method and other methods. In order to investigate the physical plausibility and the computational efficiency, three important properties of cloth animation methods were compared: the stability of the simulation, time complexity of the update process for one time-step, and the physical justification of the animation scheme. As shown in the table, the explicit methods can efficiently perform the state update process, and physical plausibility is also guaranteed if the system remains stable. However, the methods are so unstable that the overall performance is very poor. In the other hand, the implicit methods show unconditionally stable and plausible results even with very stiff models. However, the implicit methods cast the original problem into linear system problem so that real-time performance can be hardly achieved when the geometric complexity of the model is high. The approximate methods

based on implicit integration scheme can be applied to real-time applications because they are stable and efficient. However, the physical plausibility is significantly impaired. Although actual techniques for hybrid methods differ from each other, the most of them are often stable enough and the computational efficiency is also guaranteed. However, the methods also suffer from unrealistic behaviors of cloth models. The proposed method is based on implicit method, and computes the approximate solution without any severe approximation. Therefore, the proposed method is stable and efficient while the plausibility of the result motion is guaranteed.

Bibliography

[ANN90] 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.

[Bar90] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):19–28, August 1990.

[BHW94] 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.

[Bra88] 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.

[BW92] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):303–308, July 1992.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[CG91] 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.

[CMT02] Frédéric Cordier and Nadia Magnenat-Thalmann. Real-time animation of dressed virtual humans. *Proceedings of Eurographics 2002*, 2002.

[CYTT92] 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.

[DSB99] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pages 1–8, June 1999.

[EWS96] 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.

[Fey86] C. Feynman. Modeling the appearance of cloth. *Master's thesis, Dept. of EECS, Massachusetts Institute of Technology*, 1986.

[GLM96] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. Obb-tree: A hierarchical structure for rapid interference detection. *Proceedings of SIGGRAPH 96*, pages 171–180, August 1996.

[HBVMT99] Sunil Hadap, Endre Bangarter, Pascal Volino, and Nadia Magnenat-Thalmann. Animating wrinkles on clothes. *IEEE Visualization '99*, pages 175–182, October 1999.

[HM90] B. K. Hinds and J. McCartney. Interactive garment design. *The Visual Computer Journal*, 6:53–61, 1990.

[HMB01] Suejung Huh, Dimitris N. Metaxas, and Norman I. Badler. Collision resolutions in cloth simulation. *Proceedings of Computer Animation 2001*, pages 122–127, 2001.

[Kas95] Michael Kass. An introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Note: Physically-based Modelling*. ACM SIGGRAPH, 1995.

[KC02] 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.

[KCCP01] 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.

[KGL+98] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using shelltrees. *Computer Graphics Forum*, 17(3):315–326, 1998.

[KM90] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):49–57, August 1990.

[LKC98] Jen-Duo Liu, Ming-Tat Ko, and Ruei-Chuan Chang. A simple self-collision avoidance for cloth animation. *Computers & Graphics*, 22(1):117–128, 1998.

[MDDB01] 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.

[Nak91] S. Nakamura. Initial value problems of ordinary differential equations. In *Applied Numerical Methods with Software*, pages 289–350. Prentice-Hall, 1991.

[NG95] H. Ng and R. L. Grimsdale. Geoff- a geometrical editor for fold formation. *Lecture Notes in Computer Science*, 1024:124–131, 1995.

[NG96] Hing N. Ng and Richard L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics & Applications*, 16(5):28–41, September 1996.

[OITN92] 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.

[OM01] 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.

[Pla00] Jan Plath. Realistic modelling of textiles using interacting particle systems. *Computers & Graphics*, 24(6):897–905, December 2000.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Graphics Interface '95*, pages 147–154, May 1995.

[Pro97] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design. *Computer Animation and Simulation '97*, pages 177–190, September 1997.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):269–278, August 1988.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205–214, July 1987.

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics & Applications*, 8(6):41–51, November 1988.

[VCMT95] 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.

[VMT94] 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.

[VMT01] Pascal Volino and Nadia Magnenat-Thalmann. Comparing efficiency of integration methods for cloth simulation. *Proc. of Computer Graphics International 2001*, pages 265–272, June 2001.

[VPBM01] A. Vlachos, J. Peters, C. Boyd, and J. Mitchell. Curved PN triangles. *Symposium on Interactive 3D Graphics 2001*, pages 159–166, 2001.

[WB94] Andrew Witkin and David Baraff. Differential equation basics. In *SIGGRAPH Course Note: Physically-based Modelling*. ACM SIGGRAPH, 1994.

[Wei86] J. Weil. The synthesis of cloth objects. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 20:49–54, 1986.

[ZY01] Dongliang Zhang and Matthew M.F. Yuen. Cloth simulation using multi-level meshes. *Computers & Graphics*, 25:383–389, 2001.

물리적 사실성과 수치적 안정성을 가진 실시간 옷감 애니메이션

강 영 민

부산대학교, 전자계산학과

요 약

본 논문은 질량 스프링 모델에 기반한 옷감 모델의 동작을 개인용 컴퓨터 환경에서 실시간에 생성하기 위한 효과적인 기법을 제안한다. 많은 연구자들이 다양한 기법들을 제안했지만, 옷감 객체를 가상 현실 환경에서 실시간으로 움직이게 하는 일은 여전히 어려운 문제이다. 이는 옷감의 움직임을 표현하는 운동 방정식을 수치적분하는 것이 안정적으로 수행되지 않기 때문이다. 수치 해석 분야에서 알려져 있는 바와 같이 암시적 적분법를 통해 이러한 불안정성 문제를 해결할 수 있지만, 암시적 적분법은 수치적분 문제를 선형 시스템 풀이 문제로 바꾸기 때문에 복잡한 옷감 객체의 움직임을 실시간에 생성하기가 쉽지 않다. 비록 일반적인 옷감 애니메이션에 나타나는 선형 시스템이 희소행렬로 표현되기 때문에 그 해를 효율적으로 구할 수 있기는 하지만, 옷감 모델이 많은 수의 질점으로 이루어진 경우에는 문제의 크기가 급격히 커지게 된다. 따라서, 실제 옷감과 유사한 모습을 보일 수 있을 정도로 복잡한 옷감 모델을 실시간에 애니메이션 하는 것은 현재의 시뮬레이션 기법으로는 거의 불가능하다. 이러한 문제점들을 해결하고 실시간 혹은 상호작용적 애니메이션을 얻기 위해 몇 가지 근사 기법들이 제안되었으나, 지나친 근사에 기초하고 있기 때문에 사실적인 동작을 생성하지 못하였다. 옷감과 같이 높은 경직도를 가진 변형가능 물체의 실시간 혹은 상호작용적 애

니메이션을 생성하기 위해서는 안정적인 수치적분 기법이 필수적으로 요구된다. 실시간 옷감 애니메이션 기법을 제안하는 것이 본 논문의 목표이다. 따라서, 본 논문은 암시적 적분법을 통해 옷감 애니메이션 문제를 정의하고, 이를 통해 얻어지는 선형 시스템의 해를 빠르게 근사하는 기법을 제안한다. 옷감 애니메이션을 위해 선형시스템의 해를 빠르게 근사하는 방법은 이전에도 몇 가지 방법이 제안되었으나, 이전 기법들은 안정성을 위해 지나친 근사를 수행하여 생성된 동작의 사실성이 많이 떨어진다는 단점을 가지고 있었다. 본 논문의 기법은 애니메이션을 생성하는 과정에서 안정성을 유지하면서도 입력의 크기에 선형적으로 비례하는 계산 시간으로 상태 갱신이 가능하다. 이러한 안정성과 효율성을 얻기 위해 이전의 연구들은 과도한 댐핑(damping)을 이용하거나 힘의 미분치를 지나치게 근사하는 방식을 사용하였다. 따라서, 이러한 이전 기법들을 통해 사실적인 옷감 동작을 생성하기 위해서는 옷감 모델의 기하적 복잡도를 제한할 수 밖에 없었다. 본 논문의 기법은 생성 애니메이션의 사실성을 떨어뜨리는 이러한 요소들을 도입하지 않으면서도 수치적분의 안정성을 보장한다. 따라서, 이 기법은 매우 사실적인 옷감 주름을 표현할 수 있을 정도의 기하적 복잡도를 가진 모델을 실시간 애니메이션 환경에서 사용할 수 있도록 한다. 본 논문의 기법은 가상 현실 환경에서 시스템 전체의 상호작용성을 해치지 않으면서도 사실적인 옷감 모델 및 동작을 포함할 수 있는 방법으로 3차원 그래픽스를 이용한 영상산업, 게임, 의류 디자인 분야 등의 다양한 분야에 응용될 수 있다.

감사의 글

짧은 사회 생활을 접고 학교로 돌아온지 여섯 해가 지나갑니다. 이제 새로운 선택을 했던 그때를 다시 되돌아 봅니다. 후회의 마음이 아니라, 그때 가졌던 다짐에 부끄럽지 않은 여섯 해를 보내었나를 확인해야 하기 때문입니다. 어느 길이 옳았는지 아직 알지 못합니다. 그때의 다짐에 부끄러운 시간들도 많았습니다. 그동안 이룬 일은 처음에 마음 먹은 일의 반에도 미치지 못합니다. 하지만, 그동안의 시간이 저를 얼마나 키웠는가를 되돌아 보면 많은 분들께 감사를 드리지 않을 수 없습니다.

사람은 자신의 뜻과 재능만으로 살아갈 수 없습니다. 누구를 만나 같이 살아가느냐가 개인의 능력과 의지보다 오히려 더 중요할지도 모릅니다. 석사 과정과 박사 과정이라는 짧지 않은 기간 동안 한결 같이 애정과 질책으로 부족한 제자를 다듬어 주신 스승 조환규 교수님께 감사를 드립니다. 저의 지난 학업 과정은 교수님을 스승으로 모신 것만으로도 이미 성공이었다고 생각합니다.

항상 빠듯한 일정으로 바쁘신 가운데에도 흔쾌히 논문 심사를 맡아 주시고, 고마운 조언과 지도를 아끼지 않으셨던 권혁철 교수님, 김영호 교수님께 진심으로 감사를 드립니다. 가깝지 않은 거리에서 불편을 마다하시지 않고 심사를 맡아 주셨던 김영봉 교수님께도 특별한 감사를 드립니다. 역시 먼 거리에서 심사를 위해 불편을 감수하시고 심사의 안과 밖에서 세심히 조언해 주셨던 이도훈 교수님께도 감사를 드립니다.

연구실에서 훌륭한 동료들과 같이 지낼 수 있었던 것에도 감사합니다. 처음 연구실에 발을 들여 놓았을 때, 따뜻하게 이끌어 주셨던 선배 조미경 박사님, 박동규 박사님, 유영중 박사님께 감사합니다. 연구실 선배로서 많은 도움을 주었던 상현, 재권, 민아, 용빈에게 아주 늦은 감사를 드립니다. 먼저 학교를 떠난 동기들, 훈희, 성수, 세영,

그리고 오랫동안 같이 공부했던 정호열 박사와의 우정도 소중히 간직할 것입니다. 때로는 신경질적이고 너그럽지도 못한 선배에게 항상 성실한 모습을 보였던 많은 후배들, 양수, 환철, 선진, 경수, 판규, 혜선, 희정, 미녕, 재필, 은미, 미경, 영복, 지형, 그들 모두의 넉넉한 마음에도 일일이 감사를 드립니다. 연구실의 맞형으로 여러 가지 일에 몸을 돌보지 않고 신경 쓰며 노력했던 정현 형에게 크고 작은 일들을 떠맡기기만 하지 않았나하는 미안함과 함께 고마운 마음을 전합니다.

힘들 때 찾으면 언제나 제게 편안한 마음과 기쁨을 선사했던 벗들, 또한 앞으로의 어떤 상황에서든지 저를 지지할 소중한 그 벗들에게도 사랑을 담아 감사합니다. 제가 사랑했던 이들, 그리고 저를 사랑했던 모든 이들에게 감사합니다.

누구보다도 저를 신뢰하고 지원해주신 부모님께 가장 애틋한 고마움을 드리고 싶습니다. 존경하는 부모님과 사랑하는 동생들에게 글로 다할 수 없는 감사와 사랑을 바치면서 보잘 것 없는 이 논문 하나를 같이 바칩니다.