

國立交通大學

多媒體工程研究所

碩士論文

採用小波轉換加速的即時布料動態模擬

Using Wavelet Transform to Speedup Real-Time Cloth
Simulation

研究生：紀佳吟

指導教授：施仁忠 教授

中華民國九十六年六月

採用小波轉換加速的即時布料動態模擬

Using Wavelet Transform to Speedup Real-Time Cloth Simulation

研究生：紀佳吟

Student：Chia-Ying Chi

指導教授：施仁忠

Advisor：Zen-Chung Shih

國立交通大學

多媒體工程研究所

碩士論文



Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Multimedia Engineering

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

採用小波轉換加速的即時布料動態模擬

研究生：紀佳吟

指導教授：施仁忠 教授

國立交通大學多媒體工程研究所



物理方面的模擬，例如動態布料模擬，對於增加應用軟體的視覺效果上有相當大的助益。之前有關於這方面的研究大多致力於更穩定的數值積分方法，以及更快速的系統架構。本篇論文提出一個節省更多時間的系統架構：基於根據重要性採樣的概念，先將質點的法向量資料做小波轉換，藉由查詢轉換過的階層資訊，找出法向量差別較小的區域，然後以內插法代替數值積分法來計算區域內的質點位置。此外，為了配合以上的前處理方式，我們修改近似隱積分法，使得不需要內插的質點能經由穩定而快速的計算得到正確的結果。同時因為內插帶來的改變，許多在質量-彈簧系統中形成困擾但不易解決的問題也獲得改善。

Using Wavelet Transform to Speedup Real-Time Cloth Simulation

Student: Chia-Ying Chi

Advisor: Dr. Zen-Chung Shih

Institute of Computer Science and Engineering

National Chiao-Tung University



Physically-based simulation such as cloth simulation is helpful to fertilize visual effect in many applications. Techniques have been proposed are mostly aiming at more stable numerical method, or less time-consuming framework. In this thesis which based on spring-mass system, we will introduce a fast and stable numerical integrator from merging verlet method and approximate implicit integrator. Both of them have the advantage of short computational time and high stability. Additionally, in order to decrease the whole simulation time for a cloth object, we first find flat segments before each simulation, then, interpolate positions and velocities of nodes within each segment, saving time spent by using integration method. Thus we can have efficient and stable real-time simulation system without sacrificing visual quality.

Acknowledgements

First of all, I would like to express my gratitude to my advisor, Prof. Zen-Chung Shih for his guidance and patience, and. Also, I appreciate all the members in Computer Graphics and Virtual Reality Laboratory for their comments and help in these days.

I dedicate the achievement of this work to my family and friends, and thanks for their support and encouragement. Special thanks go to my boy friend. Without his support and company, I couldn't fully focus on my study.



Contents

ABSTRACT (in Chinese)	I
ABSTRACT (in English)	II
Acknowledgements	III
Contents	IV
List of Figures	V
CHAPTER 1 Introduction	1
<i>1.1 Motivation</i>	<i>1</i>
<i>1.2 Overview</i>	<i>1</i>
<i>1.3 Thesis Organization</i>	<i>2</i>
CHAPTER 2 Background	4
<i>2.1 Mass-Spring System</i>	<i>4</i>
<i>2.2 Wavelet Transform</i>	<i>6</i>
CHAPTER 3 Related Works	7
<i>3.1 Integration method</i>	<i>7</i>
<i>3.2 Mesh Adaptation</i>	<i>9</i>
CHAPTER 4 Fast and Stable Integrator	11
<i>4.1 Approximated Implicit Euler Integrator</i>	<i>11</i>
<i>4.2 Verlet Integrator</i>	<i>15</i>
<i>4.3 Combine Two Integration Method</i>	<i>17</i>
CHAPTER 5 Speed Up by Segmentation and Interpolation	19
<i>5.1 Segmentation use Wavelet transform</i>	<i>20</i>
<i>5.2 2D interpolation</i>	<i>23</i>
<i>5.3 Following Benefit</i>	<i>24</i>
CHAPTER 6 Implementation and Results	27
CHAPTER 7 Conclusion and Future Works	36
<i>7.1 Conclusion</i>	<i>36</i>
<i>7.2 Future Works</i>	<i>37</i>
Reference	39

List of Figures

Figure 1.1 The System flowchart.....	3
Figure 2.1 Three representations of internal forces	5
Figure 2.2 Structure of mass-spring syste.....	5
Figure 3.1 Triangular subdivision in [14]:	
(a) Initial face of j-1 level (b) the new mesh created by splitting step	
(c) the mesh of j level created by the averaging step.....	10
Figure 4.1 Staggered grids for the Verlet method	16
Figure 4.2 (a) Use approximate implicit method only	
(b) After our modification from verlet method.....	18
Figure 5.1 2D wavelet transform in hierarchy	22
Figure 5.2 Normal differences	22
Figure 5.3 Preprocessing of interpolation.....	23
Figure 5.4 Interpolate internal nodes	24
Figure 5.5 Force propagation in mass-spring system	25
Figure 5.6 Spring length correction after interpolation	25
Figure 5.7 Bending force in wrong direction while small angle.....	25
Figure 5.8 Better effect result after our interpolation	26
Figure 6.1 Cloth simulation result of our system.....	27.28.29
Figure 6.2 Time comparison between original simulation and after our improvement	
(a) For 32×32 cloth (b) For 64×64 cloth.....	29.30
Figure 6.3 Proportion of regions which is suitable for interpolation	
(a) For 32×32 cloth (b) For 64×64 cloth.....	30.31
Figure 6.4 (a) Spring stretch without interpolation	
(b) Spring stretch after our improvement.....	32
Figure 6.5 (a) Simulation result with threshold=0.022	
(b) Simulation result with threshold=0.03	33
Figure 6.6 Other simulation result in VR system	33.34.35

CHAPTER 1

Introduction

1.1 Motivation

Physically-based simulation is on highly demand for many interactive applications, such as games, e-commerce, virtual reality, and so on. One kind of simulation object, Cloth, is often required for human character. What can not be ignored is how to simulate them efficiently and correctly. Since Cloth is rigid when stretch, large strength on cloth may occur when springs between two nodes elongate too much. Over-large strength will make simulation fail from divergence of integration. To avoid this, small time step is an easy way, but that also means more simulation times, and low performance. On the other hand, to get balance between visual quality and efficiency, conception of multi-level resolution has been implemented for years, and it's still worth to borrow.

In this thesis, we propose an algorithm which combined verlet method and approximate implicit method to assure higher stability. Additionally, we use multi-level segmentation to save simulation time with interpolation.

1.2 Overview

Our system flowchart is shown in Fig 1.1. In our system, first, we create a cloth based on mass-spring system. In each simulation, we segment out nodes in surface

which is flat enough and do interpolation. Finally, compute positions and velocities of the other nodes using integration method.

In segmentation and interpolation, we compute curvature of each node, and use wavelet transform to build tree structures. Then we traverse the tree structure to find flat segments. Positions and velocities of nodes in the segment will be interpolated for the next time step, while the others were computed by integrator, which is more time consuming but correct.

In integration, we have a combination algorithm to compute. Approximate implicit integrator compute force more correctly and efficiently, while verlet method increase stability by taking previous position in consideration. We merge this two algorithm to compute positions and velocities of nodes for the next time step..

Finally, when simulation process completes, system will render the cloth, and go on simulation for the next time step repeatedly.

1.3 Thesis Organization



The rest of this thesis is organized as follows. In chapter 2, we introduce some background knowledge for cloth simulation and wavelet transform. In chapter 3 there are some related works about integration method and adaptive mesh proposed before. In chapter 4, we explain the combined algorithm for integration. Chapter 5 has our new approach as content, including process of segmentation and interpolation to save times. Then experimental results of our system are shown in chapter 6. Finally, chapter 7 is the conclusions and future works.

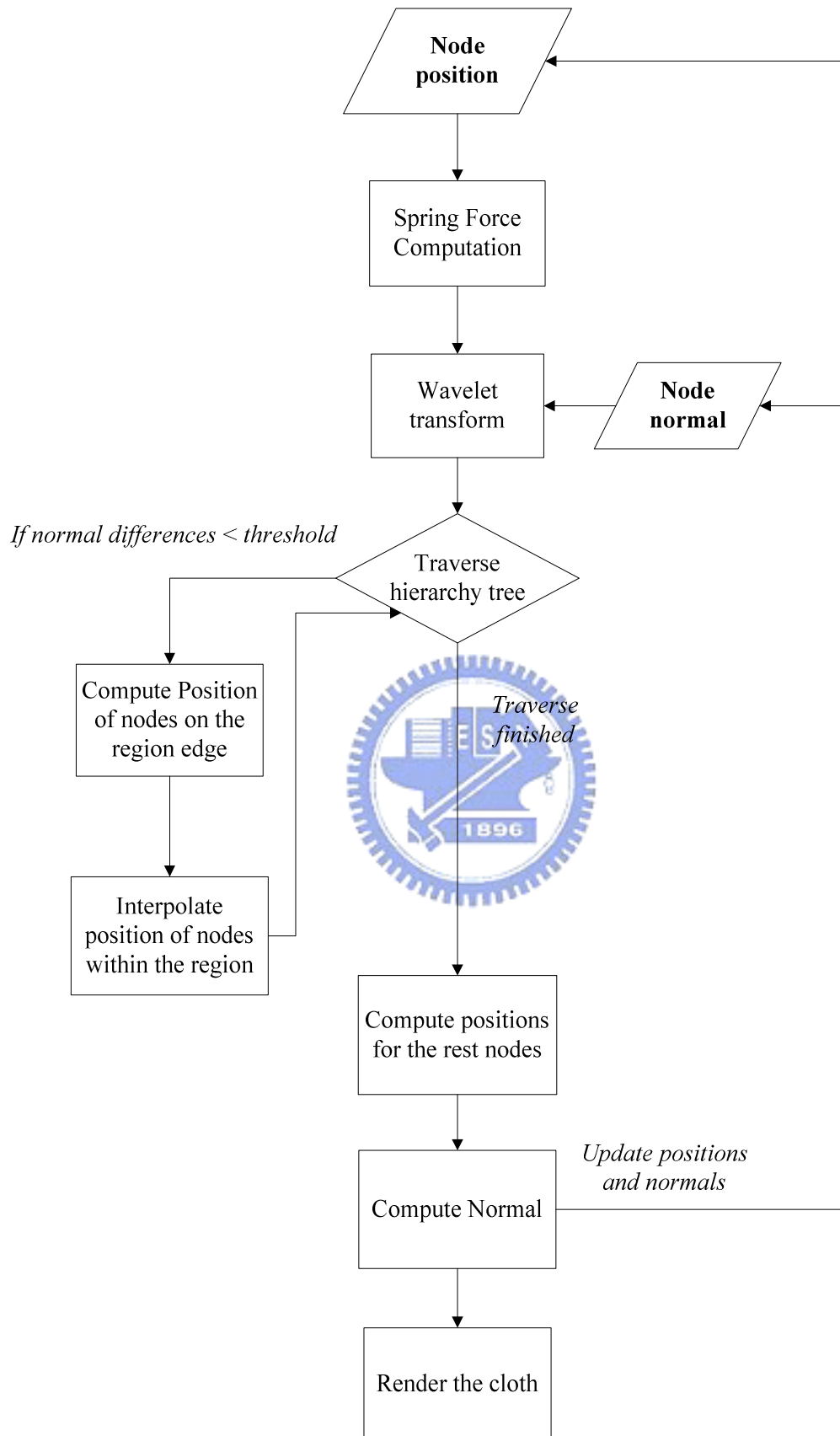
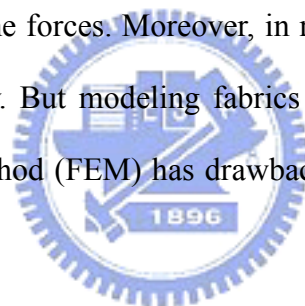


Figure 1.1 The System flowchart

CHAPTER 2

Background

In this chapter, we introduce the background knowledge of real-time cloth simulation and wavelet transform. We simulate cloth as a deformable object because it folds and wrinkles easily for out-planes forces. But on the other hand, cloth has large strain and stress under in-plane forces. Moreover, in real-time simulation, we have to compute the result efficiently. But modeling fabrics as a traditional continuum and employing finite element method (FEM) has drawbacks. This approach needs a very fine mesh or high resolution.



2.1 Mass-Spring System

There are many different approaches to simulate cloth. Mass-Spring system is one of them which are efficient and easy to implement. It was widely-adopted for real-time cloth simulation recently. We assume a piece of cloth as a group of arranged nodes. Then, we apply springs between nodes to produce and propagate internal forces. Since forces between nodes can be classified into three types: stretch, shear, and bend, as shown in Fig 2.1.

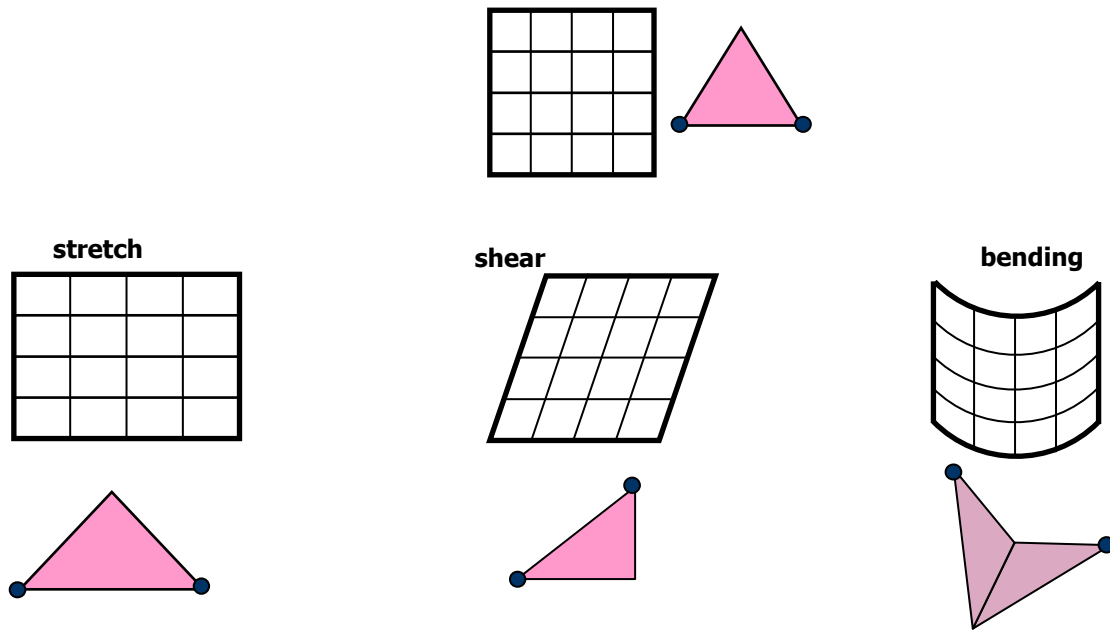


Figure 2.1 Three representations of internal forces

We also add springs in three ways of connection to simulate these forces, as shown in Fig 2.2.

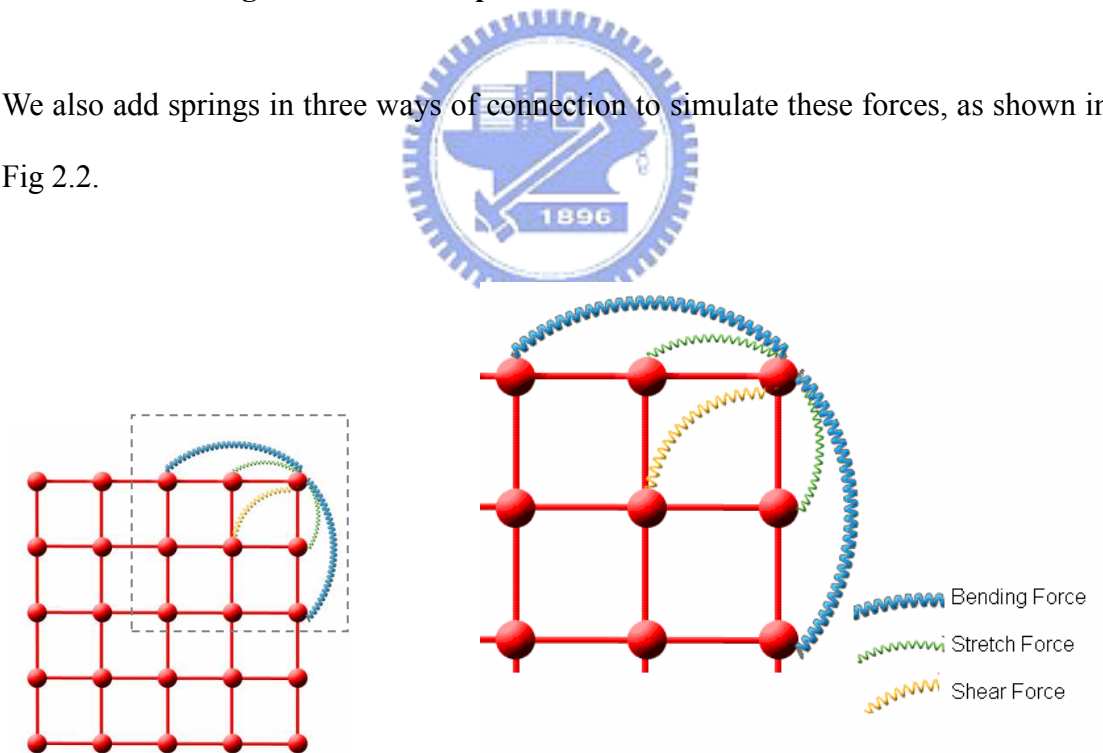


Figure 2.2 Structure of mass-spring system

When simulation, we use Hooke's law to calculate spring forces between nodes:

$$\vec{F} = -k\vec{x}$$

For common simulation, we first integrate spring forces as internal forces

according to spring length. Then, we use these forces to compute velocities and positions of each node for the next render.

2.2 Wavelet Transform

Wavelet transform is widely used in digital signal processing for many years. Wavelet transforms are classified into discrete wavelet transforms (DWTs) and continuous wavelet transforms (CWTs). CWTs operate over every possible scale and translation whereas DWTs use a specific subset of scale and translation values. In our approach, since the dimension of cloth is an integer, we use the discrete wavelet transforms.

The Haar wavelet is the first known wavelet and was proposed in 1909 by Alfred Haar. It is also the simplest possible wavelet. The disadvantage of the Haar wavelet is that it is not continuous and therefore not differentiable. The Haar wavelet finds averages and differences between samples repeatedly. For a two-dimensional array of values, we can perform a 2D Haar transform by first performing a 1D Haar transform on each row, and then on each column.

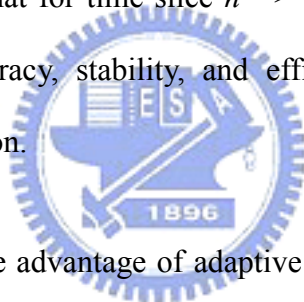
Since the Haar wavelet transform is fast, simple, and does not occupy extra memory space, we chose it for building the hierarchy structure in our approach.

CHAPTER 3

Related Works

3.1 Integration Method

In this chapter, we introduce related works about numerical integration method used in cloth or deformable object simulation. All useful methods should be convergence, which means, that for time slice $h \rightarrow 0$, the numerical solutions meet the analytical. Besides, accuracy, stability, and efficiency are also the necessary characteristic of a good solution.



Additionally, we can take advantage of adaptive mesh to make simulation more flexible. By refining or simplifying part of the mesh in different situation, we can both have coarse mesh to get efficiency, and delicate one to get detail accuracy.

3.1.1 Euler Method

The oldest and most simple method of integration is the so called forward or explicit Euler method, used by Carignan *et al.* [4]. This method computes the state of the next time step out of a direct “extrapolation” of the previous states using derivative evaluations. Thus it has high efficiency, but not accurate comparing with Implicit Euler method.

Implicit simulation methods are currently widely used in applications involving garment simulation, from real-time animation systems in Virtual Reality applications [5] [11] [13] to accurate garment simulation for design and prototyping applications [17]. They were introduced by Baraff et al [1] in the field of cloth simulation. Stability is the most advantage of this method. Simulation errors, which usually break the stability of explicit integration methods, only appear there as a form of “numerical damping” that does not prevent convergence to equilibrium. Therefore, we can use larger simulation timesteps and decrease computation times as a result.

Many different implicit method are now available for cloth simulation, they are differ in complexity, accuracy, and stability. The most widely-used method is approximated implicit Euler method [1, 5, 11, 13]. Since in implicit method, to simulate a cloth object with N mass-points, we have to compute an $N \times N$ matrix H for each time step, but H_{ij} is 0 when the i -th and the j -th mass-points are not linked with a spring. Thus, implicit method can be approximated by computing only non-zero elements of the matrix.

Desbrun in [13] splits the forces acting on a node into two parts, the linear one and nonlinear one. The linear part is easy to integrate, while the non-linear force just rotate without varying in magnitude. Thus, in order to preserve momentum, it also needs to correct linear and angular momentum additionally. However, in [10], Cho and Choi simplified the formula of velocity change directly, and makes the model working in $O(n)$ times with high stability.

3.1.2 Verlet Method

Usually the problem of stability is rapid changes in particle positions caused

e.g. by collision response or use input lead to instabilities. To cope with this, Desbrun [13] proposed to correct the velocities by position change after each time step. The effect of this correction is that the velocities are given only by the particle positions and forces at that time. So changes in position directly affect the velocity of the next time step, therefore increases the stability. This integration scheme is equal to the Verlet integration [15] which updates the position without computing any velocities.

This method has been very popular in molecular dynamics for decades and has recently been proposed in the context of physically-based simulation of cloth [7, 12]. Although Verlet method may not have credible accuracy, it has advantages of stability and efficiency for its brief formula.



3.2 Mesh Adaptation

Physical accuracy of mass-spring system is lower than that of deformable models using Finite Element Methods (FEM) for its coarse sampling. However, FEM requires solving a large sparse system which results in incompatible with real-time applications. Some previous work uses LOD (Level Of Detail) to locally refine a deformable model in regions of interest. Most of them concentrate on how to build reversible hierarchy structure to manage varying mesh topologies, and algorithms about polygonal refinement.

Mass-spring structure will produce inaccurate results if too coarse a mesh is employed, especially when collision occurs. Hutchinson et al. [8] represent a piece of draped cloth by adaptive mass-spring model which refines the regions of high

curvature. Their approach first finds the point which has inaccurate position. Between each node linked to the inaccurate one, new points are activated and new masses are introduced. Choi and Hong [2] refine meshes locally using surface wavelet to reduce the computational effort while ensuring a same global behavior for the deformable object as shown in Fig 3.1. Volkov and Li describe a general method which can be used with a variety of regular refinement rules [16]. This approach costs less time for the whole simulation which includes hierarchy construction and destruction as mesh adaptation. However, it still can not reach the speed as interactive or real-time simulation.

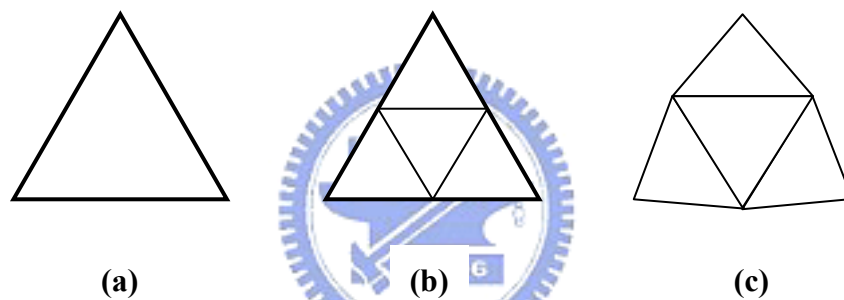


Figure 3.1 Triangular subdivision in [14]:

- (a)Initial face of $j-1$ level (b) the new mesh created by splitting step
(c) the mesh of j level created by the averaging step.**

Totally, in contrast to these approaches which refine the regions of high curvature, our method tends to rebuild the hierarchy every time and down-sampling regions of the mesh to save computation. Because our method does not change mesh topologies, taxing operations to update and reverse the hierarchy representation, which needs frequent disk access, is unnecessary.

CHAPTER 4

Fast and Stable Integrator

In this chapter, we introduce our new integrator from combining approximated implicit Euler method and verlet method, in order to get higher stability for real-time simulation.

4.1 Approximated Implicit Euler Integrator



4.1.1 The explicit method

A simple straight-forward approach for cloth simulation is using the explicit Euler integration:

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

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

simple method, we can easily calculate \mathbf{x}_i^{t+h} , the location of the i -th mass-point at the next time step $t+h$ with current state values \mathbf{x}_i^t , \mathbf{v}_i^t and \mathbf{F}_i^t .

However, this simple integration scheme can not be applied to cloth simulation unless the time step h is very small, because the explicit method becomes unstable easily when the stiffness (as spring constant) is increased. This unstable problem is revealed from the assumption of the fixed forces during the time interval between the current simulation and the next simulation. Therefore, the explicit method is not an appropriate integration model for the real-time or interactive cloth animation systems which require very large time steps to achieve fast animation.

4.1.2 The implicit method

In consideration of stability and accuracy, it is generally regarded that the implicit method is a much better choice for the fast simulation of cloth.

With the implicit Euler method, the updated formula can be rewritten as follows:

$$\begin{aligned}\mathbf{v}_i^{t+h} &= \mathbf{v}_i^t + \mathbf{F}_i^{t+h} \frac{h}{m_i} \\ \mathbf{x}_i^{t+h} &= \mathbf{x}_i^t + \mathbf{v}_i^{t+h} h\end{aligned}\tag{2}$$

The only change is that \mathbf{F}_i^t is replaced by \mathbf{F}_i^{t+h} but it has been proven that this simple change enables unconditionally stable integration [16][17].

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

$$\mathbf{F}_i^{t+h} = \mathbf{F}_i^t + \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \Delta \mathbf{x}_i^{t+h}\tag{3}$$

where F_i^t denotes the internal forces consisting of all the internal forces F_i^t on the i -th mass-point (i.e. $F^t = [F_1^t, F_2^t, \dots, F_n^t]^T$), and similarly, $\Delta x^t = [\Delta x_1^t, \Delta x_2^t, \dots, \Delta x_n^t]^T$.

4.1.3 The time-consuming matrix computation

$\partial F / \partial x$ is the negated Hessian matrix of the system[5], denoted as H . Now substituting Eq.3 into Eq.2, and by writing: $\Delta x^{t+h} = x^{t+h} - x^t = (v^t + \Delta v^{t+h})h$, we find:

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

If Δv^{t+h} can be calculated, velocity and location of each mass-point at the next step can easily be updated with Eq.3. Therefore, the simulation can be reduced to finding the value of Δv^{t+h} . After ignoring $h H v^t$ as viscous forces, Eq.3 can be written

with the inverse of $\left(I - \frac{h^2}{m} H \right)$ as follows:

$$\Delta v^{t+h} = \left(I - \frac{h^2}{m} H \right)^{-1} \frac{F^t h}{m} \quad (5)$$

However, the critical problem is that Eq.4 involves $\left(I - \frac{h^2}{m} H \right)$ which is an $O(n \times n)$ matrix.

4.1.4 Approximation method for efficiency and stability

Desbrun [13] approximated the Hessian matrix H by spitting the spring force into two parts and considering only the linear part. The approximated Hessian matrix is as follows:

$$\begin{cases} H_{ij} = k_{ij} & \text{If } i \neq j \\ H_{ii} = -\sum_{j \neq i} k_{ij} \end{cases} \quad (6)$$

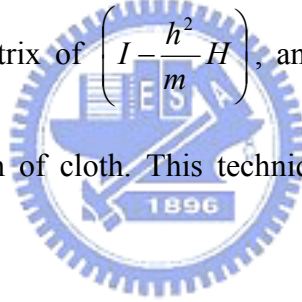
where H_{ij} denotes the entry of H at the i -th row and the j -th column. Then the

matrix $\left(I - \frac{h^2}{m} H\right)^{-1}$ remains constant during simulation if spring constant of each

spring, the time step, and the mass of each mass-point are constant. Desbrun

pre-computed the inverse matrix of $\left(I - \frac{h^2}{m} H\right)$, and used the inverse matrix as a

force filter for the simulation of cloth. This technique produces stable result with simple calculations.



However, the inverse matrix of $\left(I - \frac{h^2}{m} H\right)$ is not necessarily a sparse matrix,

even though $\left(I - \frac{h^2}{m} H\right)$ is sparse. Thus, the calculation would require $O(n^2)$ times

as long as the settings are changed.

The velocity change of the i -th mass-point can be updated by considering only the linked mass-points, because H_{ij} is 0 when the i -th and the j -th mass-points are

not linked to a spring. Therefore, Choi [10] proposed another approximation scheme.

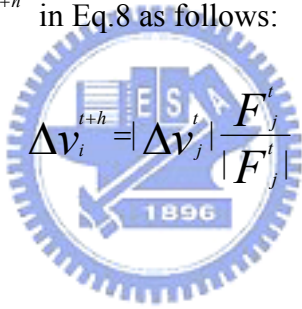
First, the implicit update equation was written as follows:

$$\left(I - \frac{h^2 \mathbf{H}_{ii}}{m_i} \right) \Delta \mathbf{v}_i - \frac{h^2}{m_i} \sum_{\forall j|(i,j) \in E} (\mathbf{H}_{ij} \Delta \mathbf{v}_j) = \frac{\mathbf{F}_i^t h}{m_i} \quad (7)$$

If the uniform spring constant k for all the spring-link is assumed, and n_i denotes the number of mass-points that are linked to the i -th mass-point, the Hessian matrix can be written as $\mathbf{H}_{ij} = k$ and $\mathbf{H}_{ii} = -k n_i$ from Desbrun in [4]. The update formula was written as follows:

$$\Delta \mathbf{v}_i^{t+h} = \frac{\mathbf{F}_i^t h + kh^2 \sum_{(i,j) \in E} \Delta \mathbf{v}_j^{t+h}}{m_i + kh^2 n_i} \quad (8)$$

Here, Choi in [8] predict $\Delta \mathbf{v}_i^{t+h}$ in Eq.8 as follows:

$$\Delta \mathbf{v}_i^{t+h} = \left| \Delta \mathbf{v}_j^t \right| \frac{\mathbf{F}_j^t}{\mathbf{F}_j^t} \quad (9)$$


4.2 Verlet Integrator

In this section, we introduce the Verlet method, and show how we apply this method to enhance stability.

4.2.1 The Verlet Method

The second integration scheme we will consider is commonly referred as leapfrog or Stoermer-Verlet method, which uses centered differences at a staggered grid to derive results.

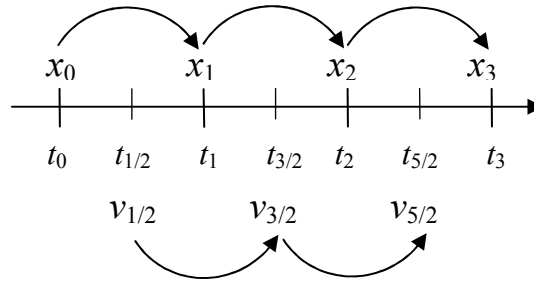


Figure 4.1 Staggered grids for the Verlet method

Assume that we now approximate v at $t+(2i+1)h/2$ and x at $t+ih$ by centered differences as shown in Fig 4.1:

$$\frac{v_i^{t+h/2} - v_i^{t-h/2}}{h} = \frac{F_i^t}{m_i} \quad (10)$$

$$\frac{x_i^{t+h} - x_i^t}{h} = v_i^{t+h/2} \quad (11)$$

Thus

$$v_i^{t+h/2} = v_i^{t-h/2} + h \frac{F_i^t}{m_i} \quad (12)$$

$$x_i^{t+h} = x_i^t + hv_i^{t+h/2} \quad (13)$$

Then substitute Eq.12 into Eq.13 resulting in the second order centered difference

$$x_i^{t+h} - 2x_i^t + x_i^{t-h} = h^2 \frac{F_i^t}{m_i} \quad (14)$$

From this equation, an alternative formulation of the Verlet scheme as a multistep method can be derived:

$$x_i^{t+h} = -x_i^{t-h} + 2x_i^t + h^2 \frac{F_i^t}{m_i} \quad (15)$$

The effect of this equation is that the positions for the next time step can be calculate

from previous position and forces but velocities. If the velocities are in need for collision detection, it can be compute easily from position differences in the time interval. As described in [7], the Verlet integration scheme is one of the best choices for problems with low or no damping because of efficiency, stability, and accuracy.

4.3 Combine the Integration Methods

Since both approximate implicit method and Verlet method are stable and efficient, we try to merge them and result in a suitable integrate scheme for our multi-level system.

4.3.1 Position Change Calculation

As the common integrate scheme, when $\Delta \mathbf{v}_i^{t+h}$ has been determined, \mathbf{v}_i^{t+h} and \mathbf{x}_i^{t+h} are also available from Eq.1. However, because our system changes node positions directly that may cause spring shivering especially with high stiffness. We need more stable method to compute positions and ease the effect of interpolation.

Since Verlet method has high stability for cloth simulation, we want to take advantage of this method and provide advance in numerical integration. However, what the point of Verlet method is that velocity change is not necessary to be calculated. Additionally, if we use only the Verlet method as integration, our system will not simulate correctly, because the vertex position in consideration may be unreliable after interpolation. In order to merge $\Delta \mathbf{v}_i^{t+h}$ from approximate implicit method, we substitute Eq.1 into Eq.15 and then yield:

$$\mathbf{x}_i^{t+h} = \mathbf{x}_i^t + h(\mathbf{v}_i^t + h \frac{F_i^n}{m_i}) \quad (16)$$

Although Eq.16 loses the characteristic of Verlet method and has no difference with explicit method, the correct experiment result of this formulation inspires us with a new idea. After calculating $\Delta \mathbf{v}_i^{t+h}$ approximately, we did not use it to calculate new positions directly. However, we leave it for the next time-step simulation, and update position with current velocity using explicit Euler method, because errors of position from interpolation should not affect other nodes in the next time step. On the other side, position change after interpolation may not be reliable, so we can not calculate velocities from current position. Therefore, the velocity change $\Delta \mathbf{v}_i^{t+h}$ is used to update next time-step velocity, which will also be used to calculate position in the next simulation. With this modification, our system has higher stability when normal difference threshold increased. The simulation results show the enhancement in Fig 4.2.

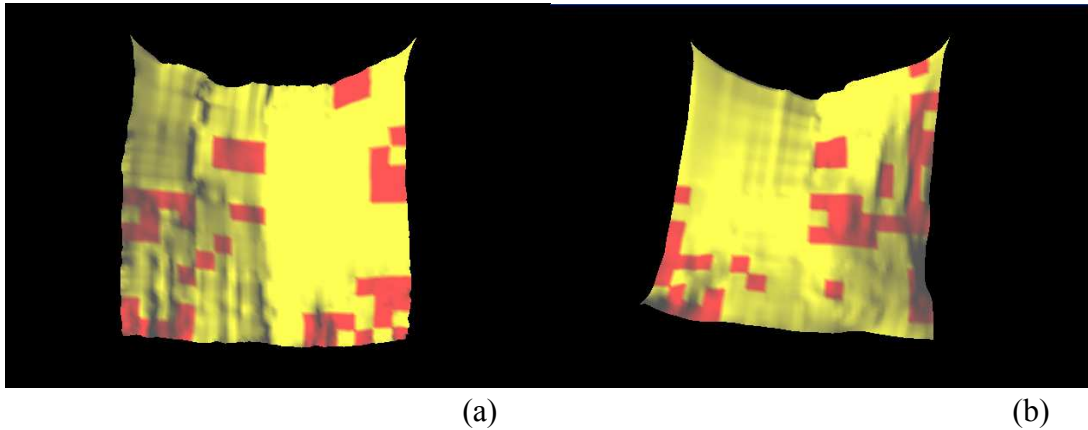


Figure 4.2 (a) Use approximate implicit method only
(b) After our modification from verlet method

This approach reveals the question that why do not use actual velocity to calculate but approximate one. As mention in Section 4.2.1, Meyer [13] proposed the

method to correct the velocity in each simulation by

$$\mathbf{v}_i^t = \frac{\mathbf{x}_i^t - \mathbf{x}_i^{t-h}}{h} \quad (17)$$

However, in our system, positions are possibly to be interpolated directly, and errors caused by interpolation should not effect the following simulation. Since the position change may not follow physical principles, it is better to use velocity approximated previously than calculate from positions as in Eq.17.

Therefore, our integration method was complete for multi-level interpolating system. Even under large time step and stiffness, our simulation remains stable and efficient.



CHAPTER 5

Speed Up by Segmentation and Interpolation

This chapter introduces another contribution about multi-level interpolation. First, we use wavelet transform to built hierarchical structure of the cloth object, and traverse it to find regions which are suitable for interpolation. Then the rest node without being interpolated will be calculated accurately by numerical integration. By this approach, we do not need to do any integration for each particle, and save time with interpolation instead.

5.1 Segmentation Use Wavelet Transform

In original, since cloth has bending forces and limited buckling as a result, regional surface with low or zero curvature should be possibly and frequently appear during cloth simulation. Therefore, we want to find segments like this and thus we can use interpolation to get approximate position for these nodes. The segmentation must meet these requires:

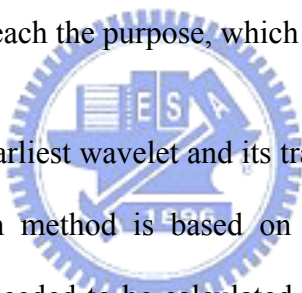
1. Fast: Usually segmentation method costs considerable times for large amount of

samples, while cloth object in simulation is a fine mesh for good visual quality. In order to compatible with our real-time system, the segmentation method must not cost too much times.

2. Nodes in the same segment must be connected: If segment element as nodes are separate one from another, interpolation can not performs efficiently and have more complexity if some elements are connected.
3. Not all of the nodes should be classified into a segment: There are still nodes may have high curvature with neighboring ones. For these element, we should calculate position for them accurately.

From above, many traditional segmentation methods do not meet these characteristics.

We use wavelet transform to reach the purpose, which is simple and fast.



The Harr wavelet is the earliest wavelet and its transform is simple to implement. Basic 2D image compression method is based on Haar wavelet transform. Only averages and differences are needed to be calculated to construct higher level. Small differences will be ignored as 0, and only notable part remains. The structure after transform is closed to a hierarchy tree. Mesh vertices represent as leave node, and each parent node has its child leave nodes connected. Therefore, from traversing the tree, we can easily find connected segment for interpolation. If traverse goes to the leave nodes, their position should be calculated accurately.

For the cloth object, we choose 2D wavelet transform to build the hierarchical structure. Because what we want to find is a continuous region, and 1D transform will lost the particle topology information about rows and columns. Fig 5.1 shows leveling relation of 2D wavelet transform.

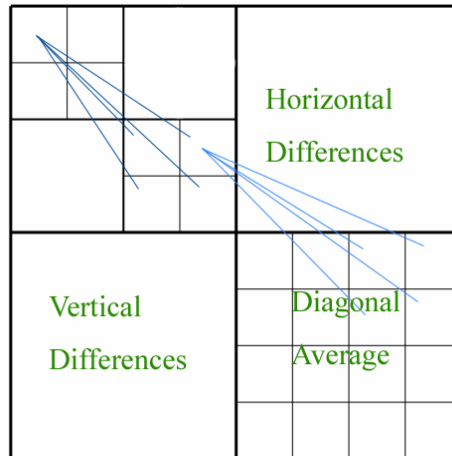


Figure 5.1 2D wavelet transform in hierarchy

When local surface is closed to flatness, normals between neighboring nodes will have small differences in direction. To analysis degree of surface flatness, we take normal of vertices to do wavelet transform, thus each higher level has average normal and normal differences between average normal and vertex normal of lower level. Therefore, the normal differences information can be use to determine if the region is suitable for interpolation as shown in Fig 5.2.

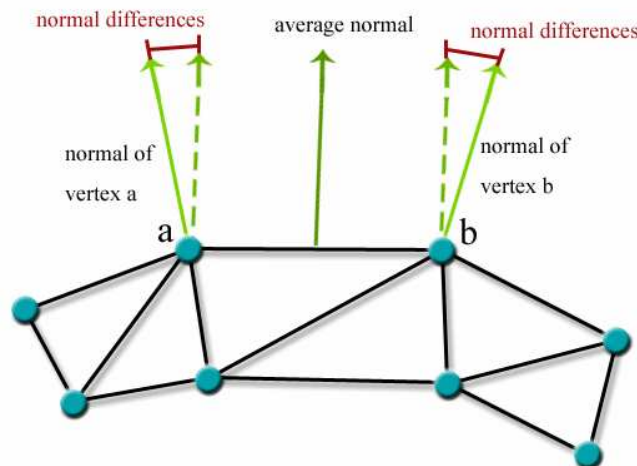


Figure 5.2 Normal differences

While transform completes, the system will traverse the hierarchical structure in

depth-first order and determine if the region can be interpolated for approximation. Once the value of a tree node is smaller than the threshold, traverse no any child node.

5.2 Interpolation of Particle Positions

When the normal differences of a region are small, it is suitable for interpolation. Our interpolate scheme is simple in order to save more time. However, if we use interpolation for all nodes within the region, it will result in hackly configuration full of flatness. When two interpolated regions share the same edge, definitely it is not proper to use interpolation directly. What should be done first is finding correct position of the edge, and then interpolating nodes within. Steps are shown below.

1. Calculate the accurate position of the node on the edge of the region, as show in Fig 5.3.

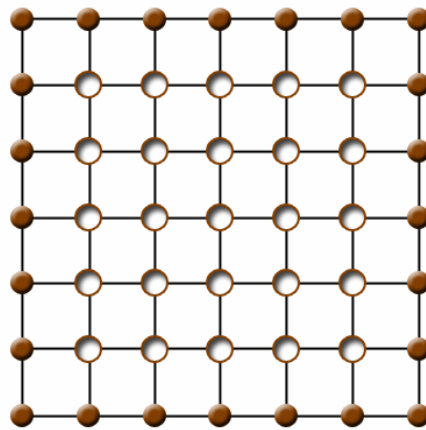


Figure 5.3 Preprocessing of interpolation

2. Use bilinear interpolation to fill positions of internal nodes, as show in Fig 5.4

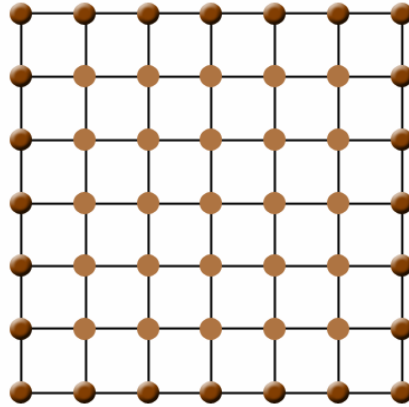


Figure 5.4 Interpolate internal nodes

As we use 2D wavelet transform, normal differences between rows and columns are available individually. This makes the system be able to do bilinear interpolation with different weights in rows and columns. For example, if normal differences between nodes in the same row are smaller than those in the same column, row interpolation will have higher weight when averaging with column interpolation.



5.3 Following Benefit

Except time saving, interpolation also brings another enhancement in stability and accuracy. In mass-spring system, as shown in Fig 5.5, forces are propagated by springs iteratively for a couple of simulations. Over-length springs due to rapid forces usually make integration fail, or produce inaccurate visual results with too much flexibility. In order to correct the elongation, position-based and velocity-based corrections are proposed. However, both of them cost too much for iteratively operation, and may effect or be effected by collision response. For the same purpose, bilinear interpolation here also provides correction in spring length as shown in Fig 5.6, hence our system have higher tolerance with external forces and stiffness.

For another aspect about bending forces, our interpolation method also provides help. There existing a problem in mass-spring system that when connected polygons bend in a small angle, as shown in Fig 5.7, bending forces will cause nodes moving in deflective direction. Many proposed approaches aim to this disadvantage and tried to solve it. However, that usually costs too much time for accurate result. With our method, the polygons will be possibly chosen for interpolation for its small cross angle, as shown in Fig 5.8. Therefore, our approach will prevent part of nodes from stretched by bending forces.

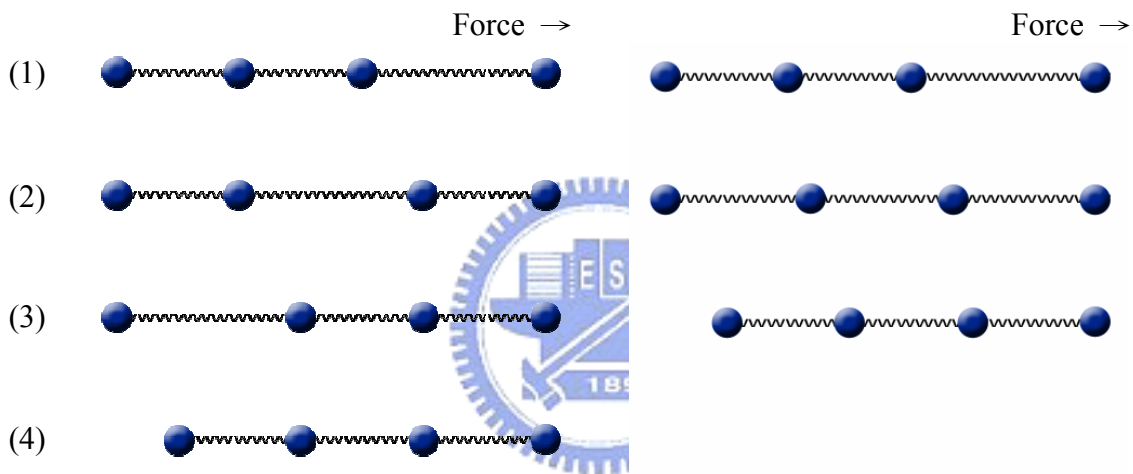


Figure 5.5 Force propagation in mass-spring system

Figure 5.6 Spring length correction after interpolation

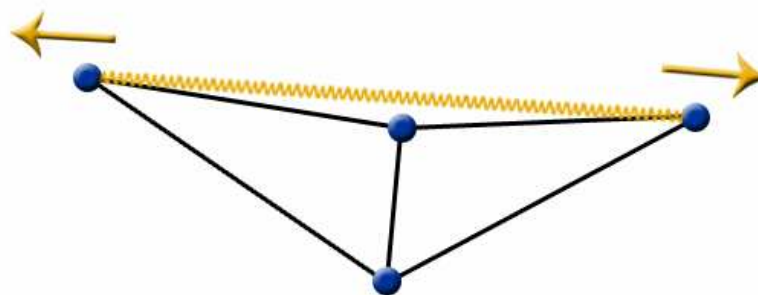


Figure 5.7 Bending force in wrong direction while small angle

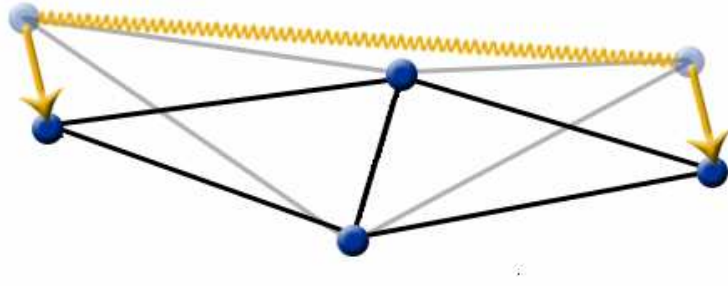


Figure 5.8 Better effect result after our interpolation

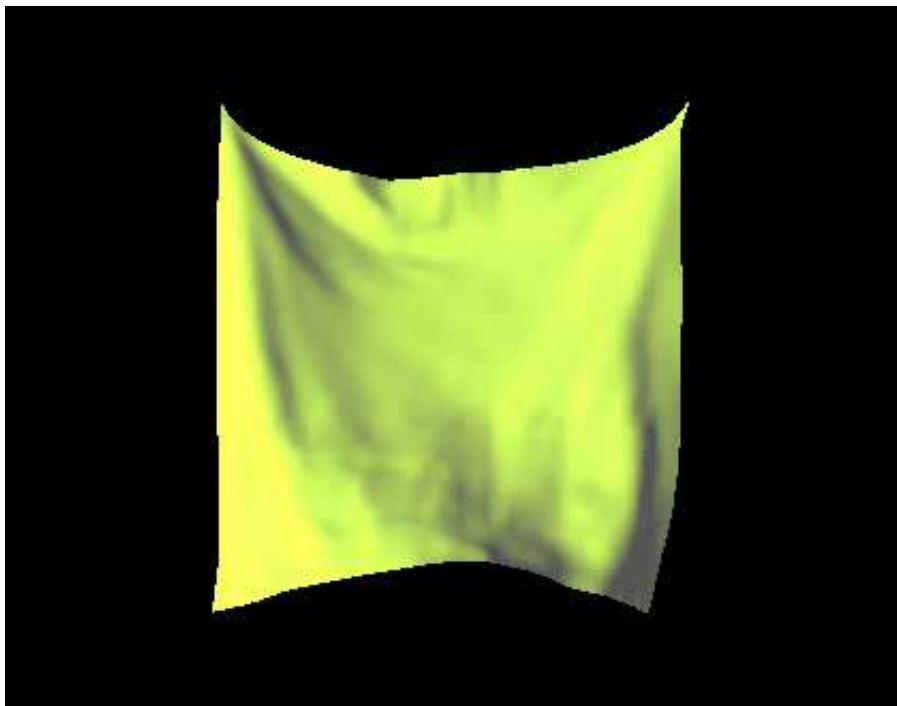


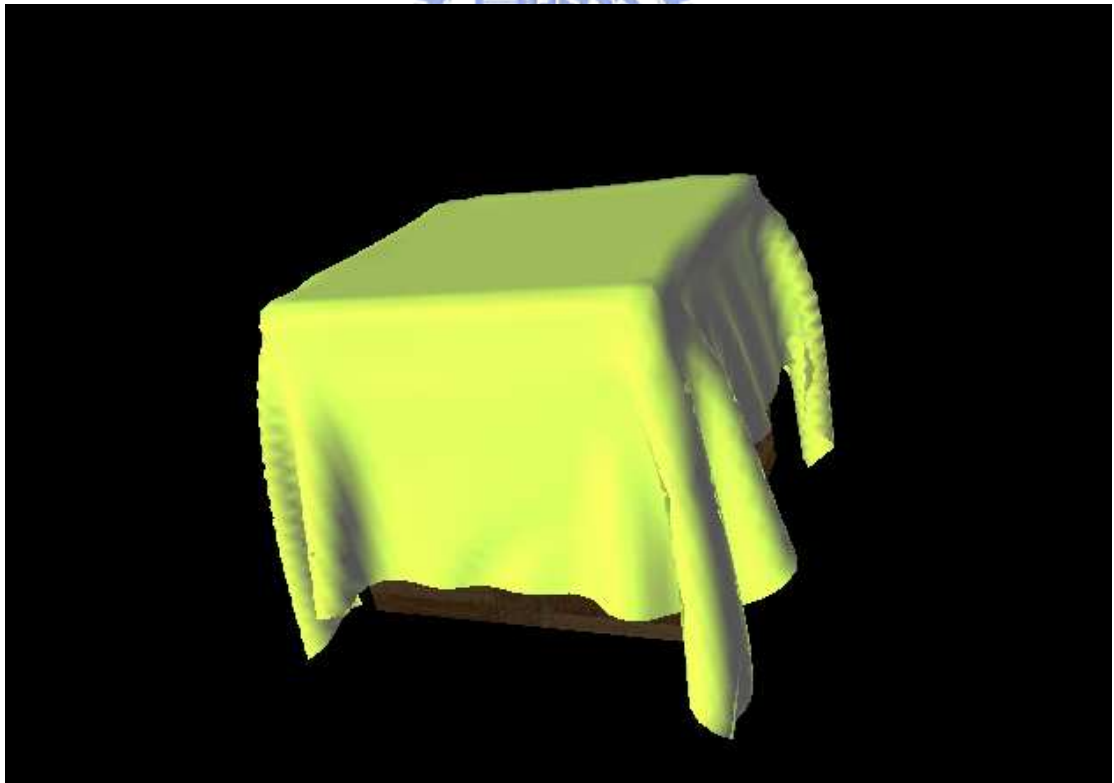
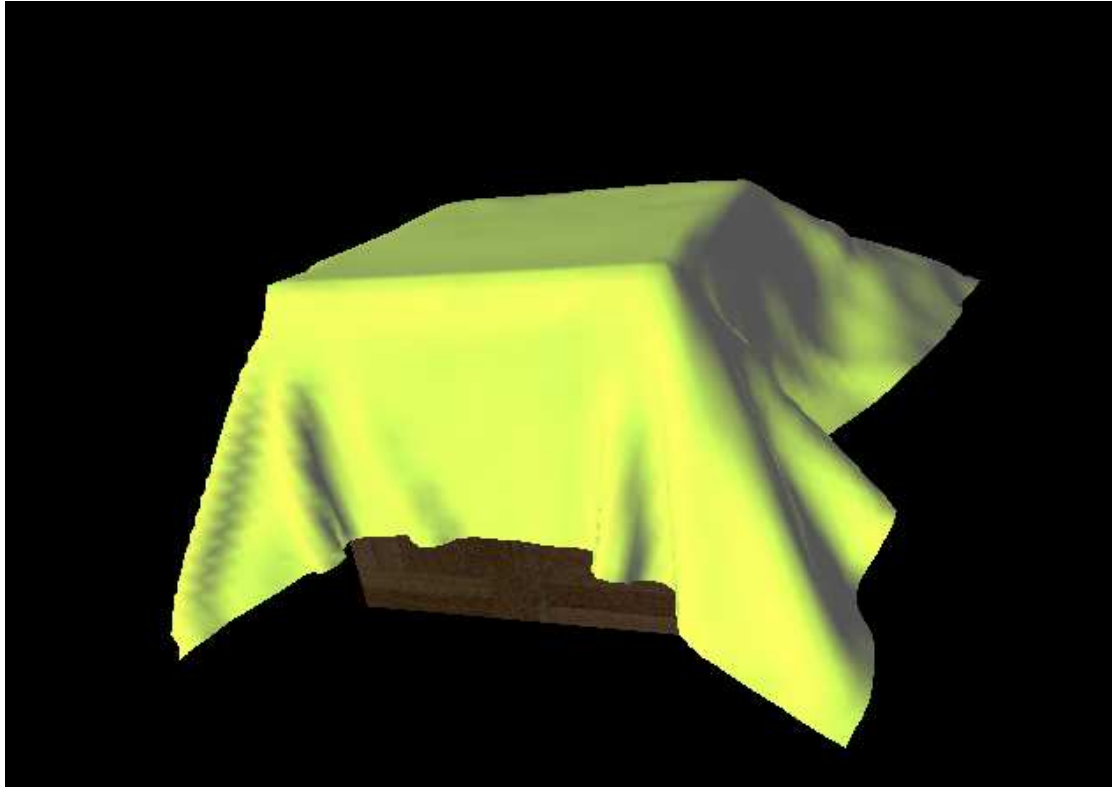
CHAPTER 6

Implementation and Results

In order to evaluate the effectiveness of animation, we implemented cloth simulation based on mass-spring system with and without our improvement. The simulation runs on a Pentium 4 PC with 3.4GHz CPU and 2Gbyte RAM, alone with NVIDIA GeForce 6600 GT graphic card.

The test case is hanging a piece of even cloth on two top corners as initial, then it will fall and swing when simulation starts. Fig 6.1 shows our simulation result. Cloth resolution is 64×64 , with spring constant in 2000, and vertex normal threshold 0.025.





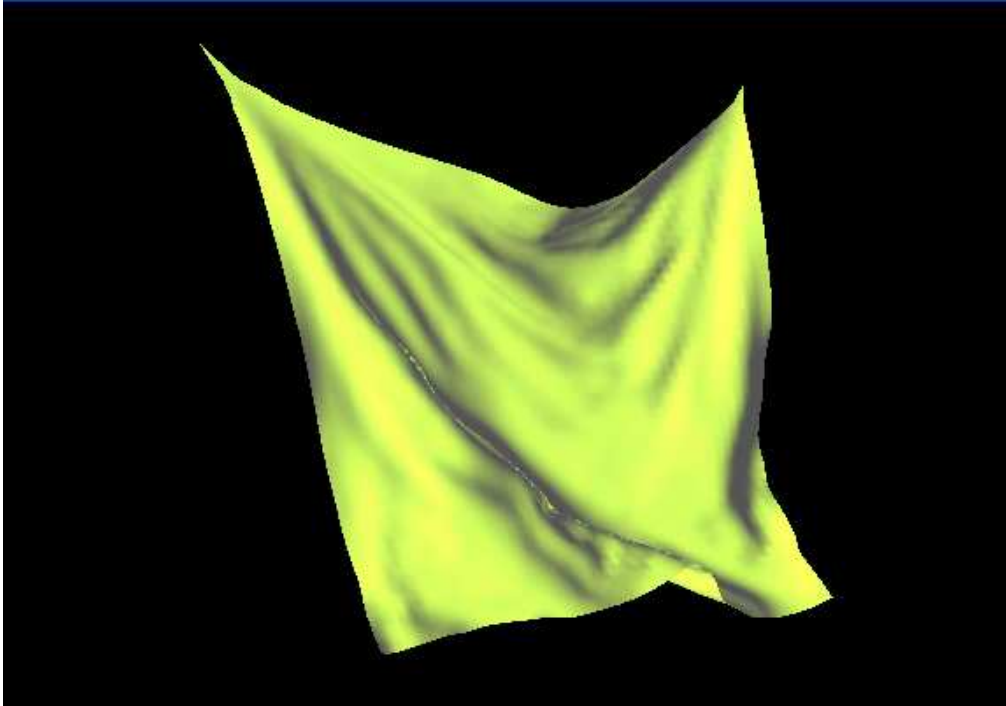
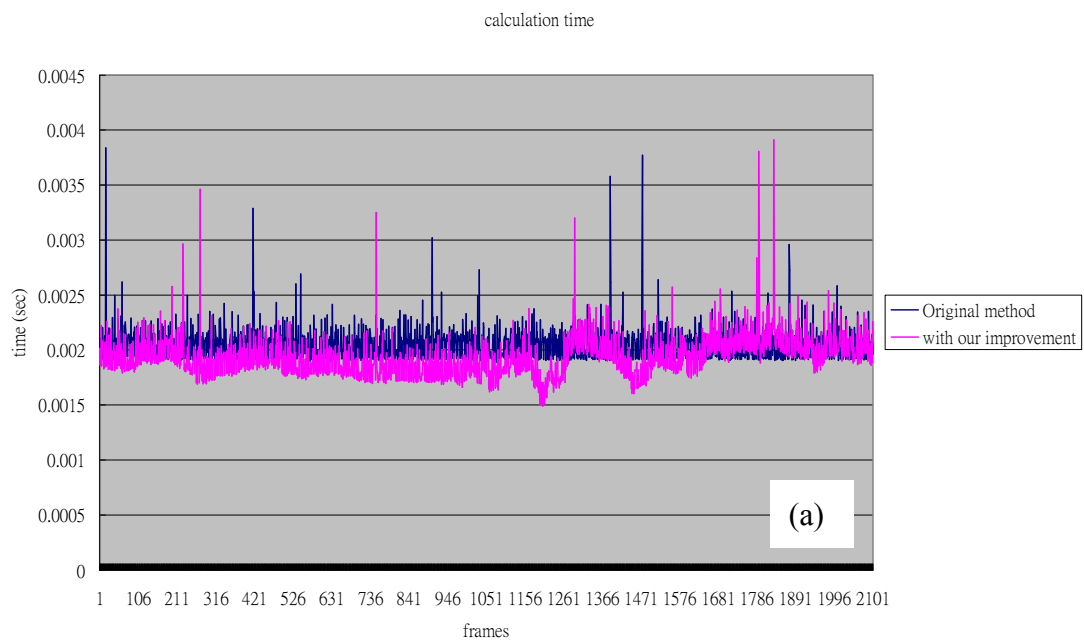


Figure 6.1 cloth simulation result of our system

Fig 6.2 shows the time comparison between with and without our improvement. It can be seen that mostly our system spends less time than traditional method which performs integration for each node.



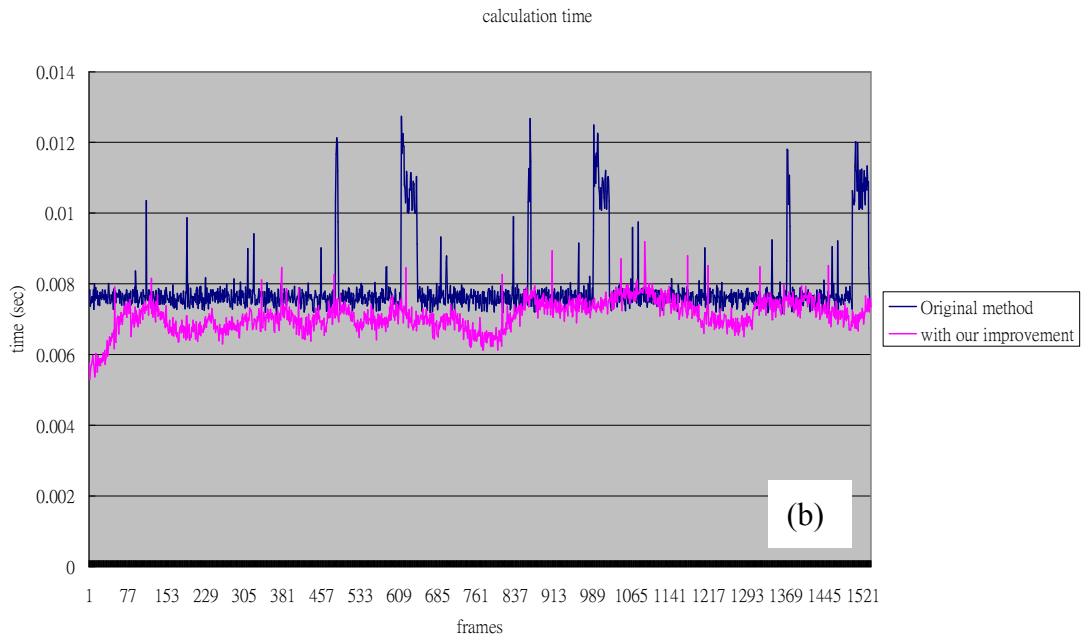
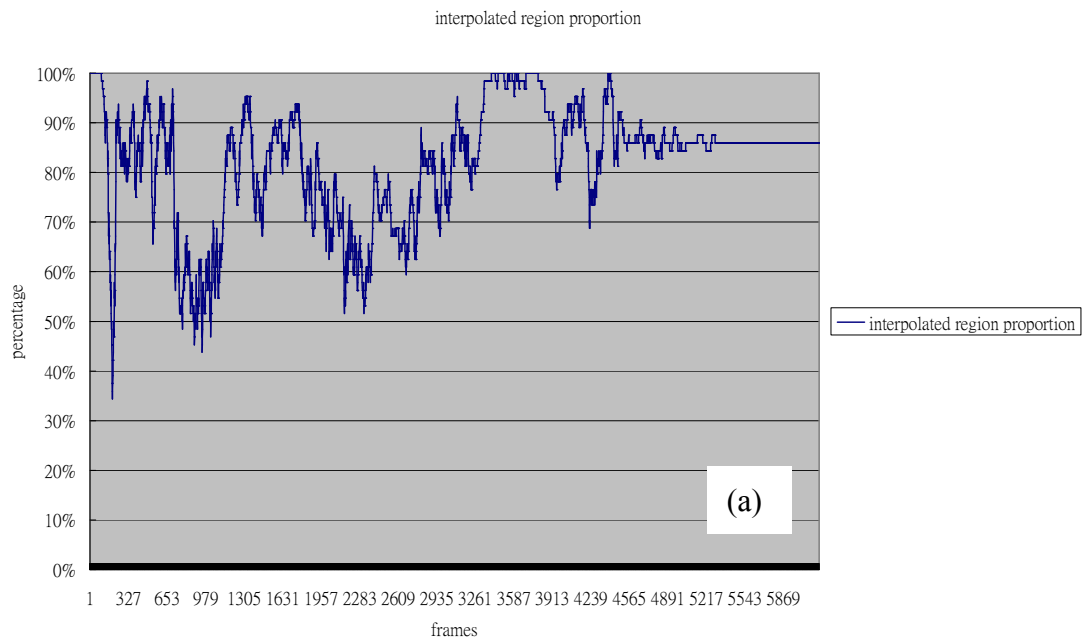


Figure 6.2 time comparison between original simulation and after our improvement applied. (a) for 32x32 cloth (b) for 64x64 cloth

Fig 6.3 shows the proportion of region which is suitable for interpolation approach during the simulation. For difference cloth configuration, the percentage changes as well.



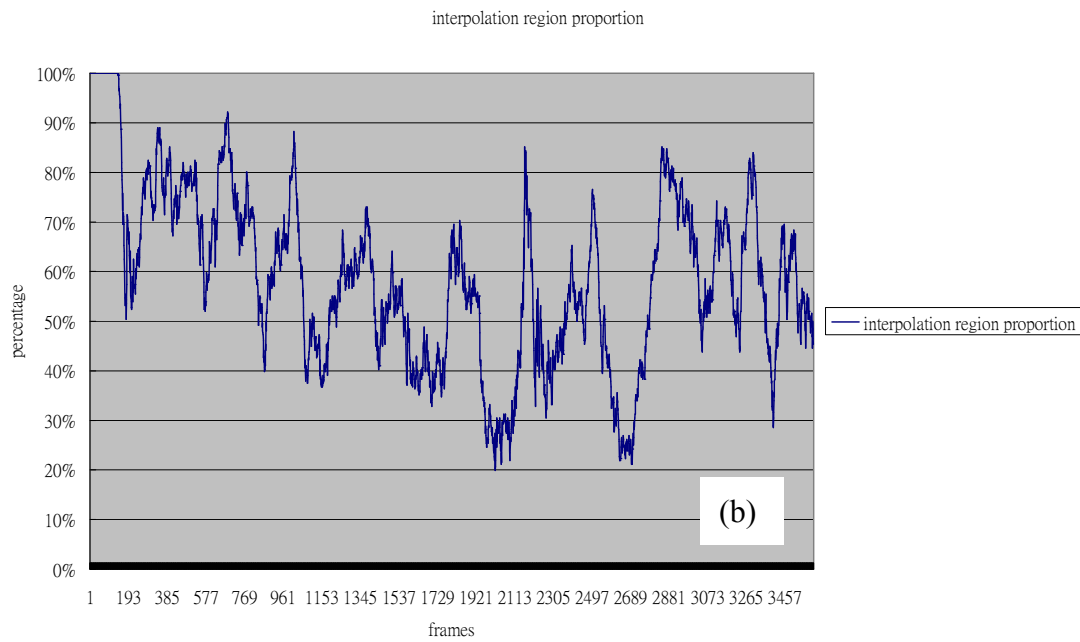


Figure 6.3 proportion of regions which is suitable for interpolation

(a) for 32x32 cloth (b) for 64x64 cloth

Fig 6.4 shows visual quality enhanced by our approach. The problem of spring over-length will be correct by bilinear interpolation without any extra operation. Besides, with traditional method as in Fig 6.4(a), spring forces will make the whole cloth object moving up and down. This can be eliminated with large spring constant, but that will also cause integration fail easily. However, in our method, the problem is solved and crumples were produced near the middle of top edge more naturally.

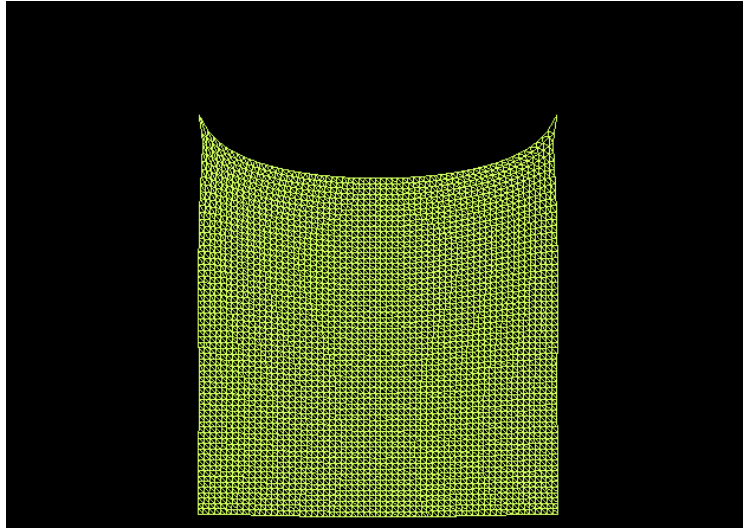


Figure 6.4(a) spring stretch without interpolation

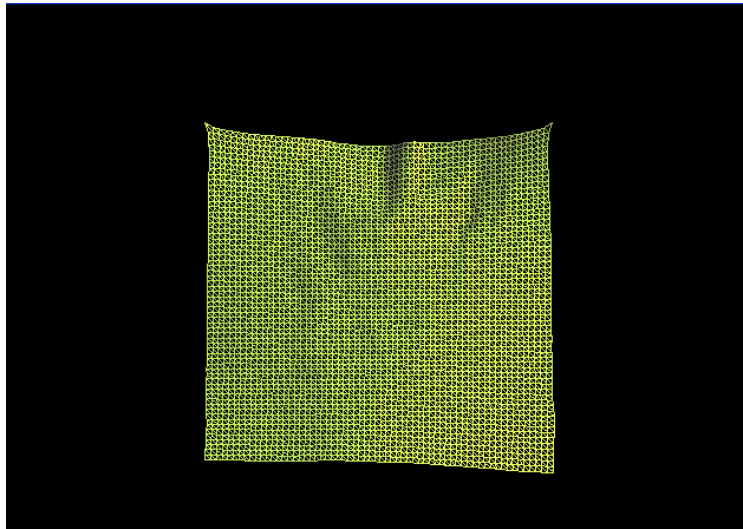


Figure 6.4(b) spring stretch after our improvement

Results above shows that our system has high efficiency, with cloth details preserved. Additionally, the always existing problem of spring stretch is solved at one time. Other simulation result is shown in below: Fig 6.5(a) shows a 64×64 cloth in wireframe with threshold = 0.022, while (b) and (c) has different value as 0.025 and 0.03. Vertices in yellow color is using interpolation, and others in red color is calculated by integration as original. It is notable that cloth details are still reserved.

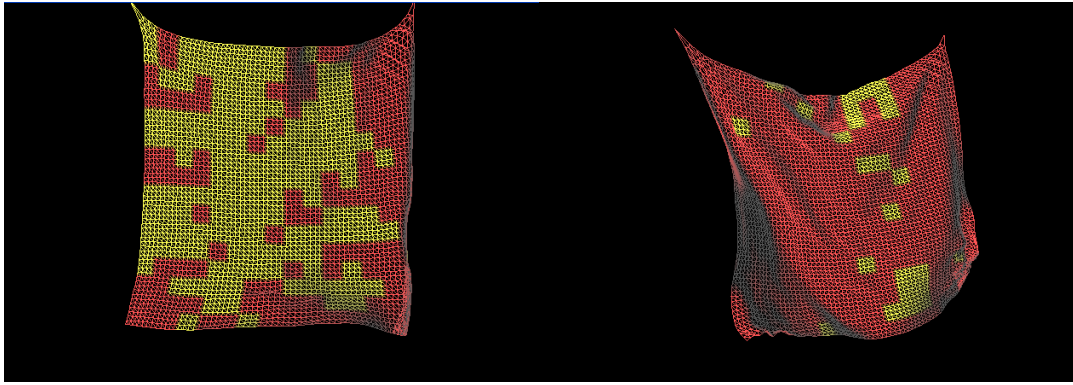


Figure 6.5(a) simulation result with threshold=0.022

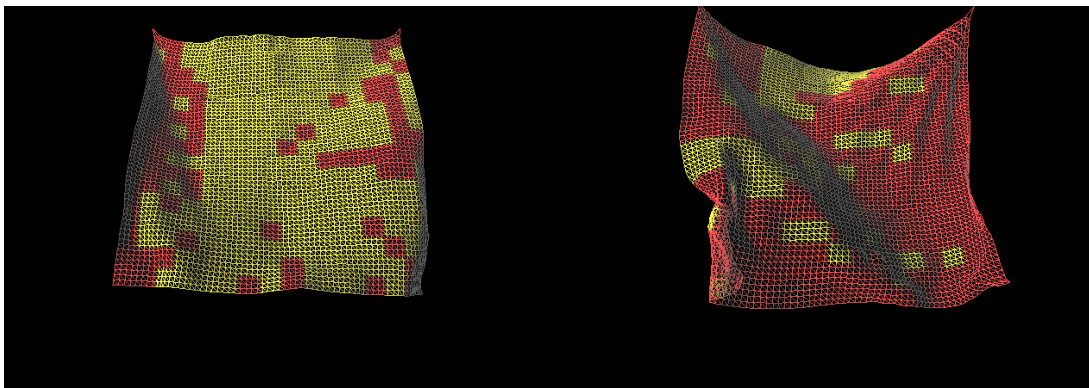


Figure 6.5(b) simulation result with threshold=0.03

In the following, Fig 6.6 shows our approach applied in a human action system.







Figure 6.6 Other simulation results in VR system



CHAPTER 7

Conclusion and Future Works

7.1 Conclusion

We have proposed a time-saving approach and a suitable algorithm for simulating cloth in real-time. As we applied approximated implicit method, which performs integration with a constant computation time, our system ensures the ability to simulate delicate cloth in real-time for interactive environment. From the concept in the Verlet method, we modify the algorithm and make it suitable for our time-saving preprocessing.

Since cloth is always tempted to have flat region, it is useful to interpolate internal nodes instead of calculating the integration repeatedly. We first use wavelet transform to efficiently build hierarchical tree about vertex normal. Then approximate the vertex positions by using interpolation, which takes less time than integration. As a result, there is no need to do integration for all internal nodes.

Not only saving times, our approach also provides corrections in spring length and bending force. After interpolation, spring length is averaged, which means particular elongation will be distributed and prevent the integrator from failure. For lightly bending edges, our method enables more correct operation as bending forces,

while original bending forces as spring force has the defects of direction at small bending angle. This improvement is our main contribution which is unrevealed previously.

7.2 Future Works

To get higher quality and efficiency, there are still some techniques can be applied.

1.Temporal coherence: Our method need to rebuild hierarchical tree before each simulation. Although wavelet transform is fast, always rebuild may not be necessary. Therefore, we can assume that if all vertex normal did not change too much, the last hierarchical tree will be adopted, and rebuild can be ignored.

2.Feature vector: Information of local curvature can be derived from vertex normal. However, other factors, like external force, original object speed, and collision response, may also affects local configuration as well. To take all other factors into consideration, vertex normal can be substituted by feature vector, which includes force and speed of the vertex. Thus our method should be more suitable and tolerable.

3.Better interpolation method: Linear interpolation will cause more errors when vertex normal difference is larger. Small threshold for tree traverse will restrict the problem. However, it will also decrease the efficiency because fewer nodes can be interpolated. Power method is simple and able to interpolate smoothly in curves, but for near-flat region, distance between nodes after interpolation will not be as

uniform as initial, and springs will be changed into wrong length. A suitable interpolation method should give proper positions to the nodes in short time, and produce visual result properly as well.



Reference

- [1] D. Baraff, A. Witkin, “Large Steps in Cloth Simulation”, *Computer Graphics (SIGGRAPH’98 proceedings)*, Addison-Wesley, 32, pp 106-117, 1998.
- [2] Yoo-Joo Choi, Min Hong, Min-Hyung Choi, Myoung-Hee Kim, “Adaptive Mass-Spring Simulation Using Surface Wavelet”, *Proceedings of International Conference on Virtual Systems and MultiMedia*, Sept. 2002
- [3] K.J. Choi, H.S. Ko, “Stable but Responsive Cloth”, *Computer Graphics (SIGGRAPH’02 proceedings)*, Addison Wesley, 2002.
- [4] M. Carignan, Y. Yang, N. Magenenat-Thalmann, and D. Thalmann. “Dressing animated synthetic actors with complex deformable clothes”, *Computer Graphics (Proc. SIGGRAPH)*, pages 99–104, 1992.
- [5] M. Desbrun, P.Schröder, A. Barr, “Interactive Animation of Structured Deformable Objects”, *Proceedings of Graphics Interface*, A K Peters, pp 1-8, 1999.
- [6] A. Fuhrmann, C. Gross, V. Luckas, “Interactive Animation of Cloth Including Self Collision Detection,” *Proc. Of WSCG’03*, University of West Bohemia, Czech Republic, pp. 141-148, 2003.
- [7] M. Hauth, O. Etmuss, B. Eberhardt, R. Klein, R. Sarlette, M. Sattler, K. Daubert, J. Kautz, “Cloth Animation and Rendering,” *Eurographics Tutorials*, 2002.
- [8] D. Hutchinson, M. Preston, T. Hewitt, “Adaptive Refinement for Mass-Spring Simulations”, *Proc. Of the Eurographics Workshop on Computer Animation and Simulation*, pages 31-45, Sep. 1996
- [9] M. Kass. “An introduction to continuum dynamics for computer graphics”. In *SIGGRAPH Course Note*. ACM SIGGRAPH, 1994

- [10] Young-Min Kang , Jeong-Hyeon Choi , Hwan-Gue Cho , Chan-Jong Park, “Fast and Stable Animation of Cloth with an Approximated Implicit Method”, *Proceedings of the International Conference on Computer Graphics*, p.247, June 19-24, 2000
- [11] Y.M. Kang, J.H. Choi, H.G. Cho, D.H. Lee, C.J. Park, “Real-Time Animation Technique for Flexible and Thin Objects”, *WSCG proceedings*, pp 322-329, 2000.
- [12] Z. Kacic-Alesic, M. Nordenstam, D. Bullock, “A Practical Dynamics System,” *Proc. of Symposium on Computer Animation*, San Diego, California, pp. 7-16, 2003.
- [13] M. Meyer, G. Debunne, M. Desbrun, A. H. Barr, “Interactive Animation of Cloth-like Objects in Virtual Reality”, *Journal of Visualization and Computer Animation*, John Wiley & Sons, 2000.
- [14] S. Nakamura. “Initial value problems of ordinary differential equations”. In *Applied Numerical Methods with Software*, pages 289–350. Prentice-Hall, 1991.
- [15] Loup Verlet. “Computer experiments on classical fluids:. i. thermodynamical properties of lennard-jones molecules. *Physical Review*, 159(1):98–103, 1997.
- [16] Volkov, V. and Li, L. (2003), “Real-time refinement and simplification of adaptive triangular meshes”. *Proc. IEEE Visualization 2003*, pp. 155–162
- [17] P. Volino, N. Magnenat-Thalmann, “Accurate Garment Prototyping and Simulation”, *Computer-Aided Design and Applications*, CAD Solutions, 2(5), pp 645-654, 2005.
- [18] 劉振鐸, 施仁忠, “可形變之布的動態模擬” , 1993
- [19] 劉振鐸, 施仁忠, “以電腦繪圖的技術模擬布的動態之研究” , 1998