Adaptive Mass-Spring Simulation using Surface Wavelet

Yoo-Joo Choi[1], Min Hong[2], Min-Hyung Choi[3], Myoung-Hee Kim[1]

[1]*Dept. of Computer Science and Engineering, Ewha Womans University, Seoul, Korea*
[2] *Bioinformatics, University of Colorado Health Sciences Center, Denver, USA*
[3] *Dept. of Computer Science and Engineering, University of Colorado, Denver, USA*

[1]*{choirina, mhkim}@ewha.ac.kr*
[2]*Min.Hong@UCHSC.edu*, [3]*minchoi@acm.org,*

Abstract. This paper presents an efficient approach to represent the animation of the deformable objects using adaptive spatial resolution. We introduce the multi-resolutional mass-spring model that is locally refined using surface wavelet to reduce the computational effort while ensuring a same global behaviour for the deformable object. In the animation of deformable objects, the part of the object that gets the external force is refined in higher level of detail and the mass values of mass nodes in the local part are changed in order to preserve the total mass of the object. Triangular mesh surface can effectively be subdivided or simplified using local synthesis filters and analysis filters. The local synthesis and analysis filters are constructed in on-line processing when the large magnitude of object deformation is detected. We exploit "winged-edge" data structure in order to detect the neighbouring faces and edges in constant time. The proposed method which is independent of the regularity of the initial mesh reduces computational cost in effect while ensuring behaviour of the object in the refined model.

1. Introduction

The real-time interaction with deformable objects plays a crucial role in many interactive virtual reality applications. The applications related to surgical simulation particularly demand the real-time animation of deformable object according to a user's interaction. Among the several methodologies for physically-based modeling, mass-spring models are widely applied in the virtual reality application because they are effective in computational cost and simple to implement. However, even the mass-spring models including many mass nodes and springs for representing more complex deformation makes it difficult to simulate the deformation in real time. In the case of the large mass-spring models, the nodes largely affected by the external force are usually the subset of total nodes and the rest of nodes mostly remain still or oscillate invisibly. However, computation for deformation covers the entire nodes composed the object and it makes the real-time simulation difficult. Therefore, combining locally refinement techniques with physically-based modeling have been applied in order to reduce large amount of computation while ensuring the global behaviour of detailed model. In this paper, we propose the adaptive mass-spring model which is locally animated at different levels of details using surface wavelet which is effective to subdivision and simplification of surface mesh.

The remaining sections of this paper are as follows. In section 2, we briefly review the researches on the deformable modeling and adaptive method in physically-based animation. We discuss a conventional mass-spring models and surface subdivision method using surface wavelet in section 3 and 4. We present our adaptive refinement scheme combined with mass-spring model in chapter 5. In chapter 6, we discuss the result of implementation and we finally suggest the conclusion and future works in chapter 7.

2. Related Work

Physical accuracy and real-time performance are trade-off in animating deformable objects. In the case of mass-spring system, the equations of motion are integrated independently for each particle, which make calculations fast. However, physical accuracy of mass-spring system is lower than that of deformable models using Finite Element Methods (FEM). Conventional FEM uses exact differential operators which require solving a large sparse system. As a result, it is less compatible with real-time applications. Explicit finite element method uses similar animation way to mass-spring case, in which each node of the FEM mesh dynamically integrates its motion from the positions of adjacent nodes. This method gives more accurate results than mass-spring system. However, computing deformation at a fine discretization level requires a fine sampling that prevents real-time performances. Some previous work uses LOD (Level Of Detail) to locally refine a deformable model in regions of interest. Hutchinson et al. represent a piece of draped cloth by adaptive mass-spring model which refines the regions of high curvature [1]. O' Brien simulates the fracture of glass by refining a tetrahedral mesh along fracture lines [2]. Wu et al. propose a scheme for mesh adaptation based on an extension of the progressive mesh concept to produce a diagonal mass matrix that allows real time computation [3]. Debunne et al. propose a method of physically-based modeling that combines a linear finite volume-based mechanical model with a non-hierarchical refinement technique based on Voronoi concept [4]. The relationships between disparate resolution meshes are maintained using ghost nodes that transmit the mechanical behaviour of a mesh section at one resolution to adjacent mesh sections of different resolutions. Kawai builds "hierarchical mass-spring models" in order to manipulate the elastic object using coarse-to-fine representation of mass-spring models based on the degree of deformation [5].

The most of previous works assume that the initial fine mesh is of good quality. For example, Wu et al. assume that angles within each triangle should be at least 20° and the valence of each vertex should be less than 10. In this paper, we propose an adaptive mass-spring method based on a surface wavelet concept, which is not independent of the regularity of the initial mesh and can effectively subdivide or simplify the surface using analysis filters or synthesis filters. The triangular mesh in the area that has a large magnitude of object deformation is locally refined by multiplying by local synthesis filters.

3. A conventional Mass-Spring Model

The demands for real-time performance in modeling and simulation of deformable objects have lead many researchers to concentrate on simple elastic models. Mass-spring model is a simple discrete approximation of a continuous model assuming that the mass is concentrated on nodal points and the nodes are connected with neighbor nodes with mass-less springs. Mass-spring model is well studied and widely used for many simulations and animations. It is a simple physical model, easy to program, and it demonstrates reasonably fast simulation speed. Despite the drawbacks such as problems in finding proper spring coefficients and dealing with incompressibility, mass-spring model is called for when the importance of numerical performance and interactivity outweighs physical accuracy and general applicability. Mass-Spring model is well suited for the interactive surgery simulation and medical animation where real-time feedback and prompt responses are required. Simulating human organ using a mass-spring system consists of a number of mass points and springs that connect the nodal points to propagate the energy. The following figure illustrates the liver model with surface oriented mass-spring systems which consist of mass points (green circles) and springs (yellow line).

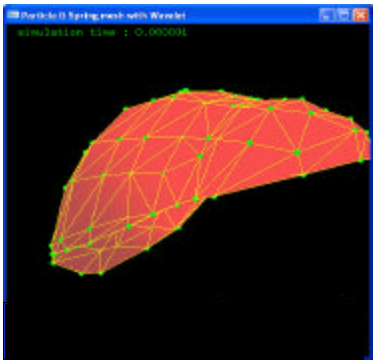


Figure 1.1. Liver model with mass-spring model.

The governing Lagrange's equation for mass-spring model is

$$m_i \ddot{x}_i + g_i \dot{x}_i + k_i x_i = f_i + g_i$$

where $x_i, \dot{x}_i, \ddot{x}_i$ are the position, the velocity and the acceleration respectively. m_i, g_i, k_i are the nodal mass, the damping coefficient and stiffness coefficient at node i, and g_i is total force on node i due to spring connecting node i to neighboring nodes. f_i is the external force at node i. We use explicit Euler method to numerically integrate the ODE (Ordinary Differential Equation). When the initial value of x is $x_0 = x(t_0)$, the estimate value of x at a later time $t_0 + h$ is $x(t_0 + h)$, where h is a parameter for step-size.

$$x(t_0 + h) = x_0 + h\dot{x}(t_0)$$

Using the Newton's law $f = ma$, acceleration and velocity of mass points can be written as a pair of coupled first-order ODE:

$$\dot{v} = \ddot{x} = f / m$$
$$\dot{x} = v$$

To calculate spring forces g_i, the spring length at resting stage r is pre-computed from the initial positions of nodes that define an object's original shape. The spring forces g_1 and g_2 between a pair of mass points 1 and 2 at position x_1 and x_2 with velocity v_1 and v_2 are

$$g_1 = -\left[k_s\left(|\Delta x| - r\right) + k_d\left(\frac{\Delta v \cdot \Delta x}{|\Delta x|}\right)\right]\frac{\Delta x}{|\Delta x|}$$
$$g_2 = -g_1$$

where k_s and k_d are spring constant and damping constant respectively. $\Delta x = x_1 - x_2$ is the difference between the two points' positions, and $\Delta v = v_1 - v_2$ is the difference between the two points' velocities. Given the external forces, the new position of every mass points is computed by numerically solving the ODE.

4. Multi-resolutional analysis using Surface Wavelets

The starting point for multi-resolutional analysis is a nested set of linear function spaces $V^0 \subset V^1 \cdots \subset V^n$ with the resolution of function in V^j increasing with j. The basis function for the space V^j are called scaling functions. The next step in multi-resolution analysis is to define the wavelet spaces, denoted by W^j. Each wavelet space W^j to be the complement of V^j in V^{j+1}. Thus, any function in V^{j+1} can be written as the sum of a unique function in V^j and a unique

function in W^j. The functions we choose as a basis for W^j are called wavelets. Matrix notation is used to perform a wavelet transform [6][7]. We consider a function in some approximation space V^j and assume that we have the coefficients of this function in terms of some scaling function basis. We write these coefficients as a column matrix of values $c^j = \left[c_0^j \ldots\ldots c_{v(j)-1}^j\right]^T$. The coefficients c_i^j be thought of as the x-, y-,z- coordinates of a node points in $\Re^3$. To create a low-resolution version c^{j-1} of c^j with a smaller number of coefficients v(j-1) values of c^{j-1} is to use some form of linear filtering and down-sampling on the v(j) entries of c^j. This process can be expressed as a matrix equation

$$c^{j-1} = A^j c^j \qquad \text{<eq. 4.1>}$$

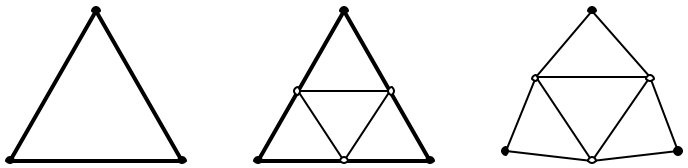
where A^j is a constant $v(j-1)\times v(j)$ matrix.
Since c^{j-1} contains fewer entries than c^j, It is clear that some amount of detail is lost in this filtering process. If A^j is appropriately chosen, it is possible to capture the lost detail as another column matrix d^{j-1}, computed by

$$d^{j-1} = B^j c^j \qquad \text{<eq. 4.2>}$$

where B^j is a constant $w(j-1)\times v(j)$ matrix related to A^j. The pair of matrices A^j and B^j are called analysis filters. The process of splitting the coefficients c^j into a low-resolution version c^{j-1} and detail d^{j-1} is called analysis of decomposition. If A^j and B^j are chosen appropriately, then the original coefficients c^j can be recovered from c^{j-1} and d^{j-1} by using the matrices P^j and Q^j:

$$c^j = P^j c^{j-1} + Q^j d^{j-1} \qquad \text{<eq. 4.3>}$$

For triangular subdivision, the splitting step breaks each face of $j-1$ level into four sub-faces by introducing edge midpoints to create a new mesh. The averaging step then perturbs the resulting collection of vertices to create a mesh for j level according to the wavelet coefficients. Figure 4.1 depicts the averaging and perturbation of triangular mesh.



(a) (b) (c)

Figure 4.1 Triangular subdivision :
(a)Initial face of j-1 level (b) the new mesh created by the splitting step
(c) the mesh of j level created by the averaging step.

5. Adaptive Refinement

The main idea of proposed method is based on the dynamic combination of different detail levels of mass-spring models according to the location of manipulator node in order to reduce the mass nodes and springs for computation and to preserve the overall behaviour of object. The neighbouring area of manipulator node with the external force is locally refined by adding extra masses and splitting the springs. And the masses of existed nodes in coarse level are modified in order to preserve the overall mass of object.

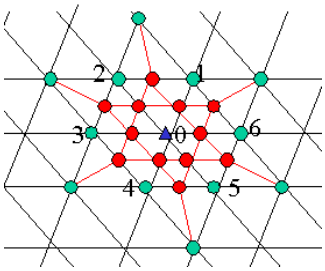


Figure 5.1 the new mesh after locally refinement
Blue triangle: the forced node, **Red circle** : the inserted nodes in refined level,
Cyan circle: the initial nodes in coarse level

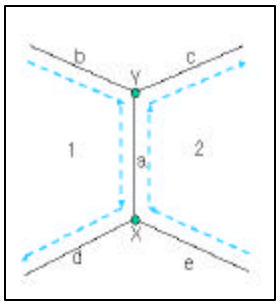
Figure 5.1 depicts how to bcally refine the neighbouring area of manipulator node. The blue triangle represents the manipulator node and the cyan circles represent the existed mass nodes in coarse detail level. And the red circles express the inserted extra nodes and the red edges are the inserted springs. The springs between two cyan circles-the existed mass nodes are divided into two short springs by the extra nodes. The positions and the number of the extra nodes are decided by the subdivision surface algorithm. We apply a surface lazy-wavelet as a subdivision method. As a result of subdivision using lazy wavelet, each face at the distance of one spring from the manipulator node is subdivided into four sub-faces and each face at the distance of two springs which includes the extra nodes is subdivided into two sub-faces like Figure 5.1. In the case of that the valence of the manipulator node is six, twelve extra nodes are added in the area at the distance of one spring from manipulator node. That is, seven mass nodes are broken into nineteen mass sub-nodes. Each mass of nineteen nodes is computed by

$$MASS_{new,j} = \frac{(\sum_{i=0}^{valence} MASS_{old,i} + 1)}{valence + NumNewNodes} \qquad \text{<eq. 5.1>}$$

where, valence means the valence of manipulator node. i is an index for existed nodes in the neighbouring area and j means an index for nineteen nodes. *NumNewNodes* is the number of the extra nodes. We construct "winged-edge" data structure for mass-spring models in order to effectively detect the neighbouring faces and edges. Winged-edge data structure is the boundary representation method using edge's ordered relation. The following information are saved for each edge in the edge table.

- Start and end vertices of this edge
- Its left and right faces
- The predecessor and successor of this edge when traversingits left face
- The predecessor and successor of this edge when traversing its right face

Clockwise ordering is used for traverse. Figure 5.2 shows the information for the entry of edge a. The four edge b, c, d and e are the wings of edge a and edge a is "winged".



Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred.	Succ.	Pred.	Succ.
a	X	Y	1	2	b	d	e	c

Figure 5.2 Winged-edge Data Structure and Edge table

The winged-edge data structure requires two more tables, the vertex table and the face table. These two are very simple. The vertex table has one entry for each vertex which contains an edge that is incident to this vertex. The face table has one entry for each face which contains an edge that is one of this face's boundary edges. Using winged-edge data structure, we can easily answer the question: which vertices, edges, faces are adjacent to each face, edge, or vertex. The winged-edge data structure can answer these queries in constant time [8].

We construct the synthesis and analysis matrices for local lazy wavelets in order to decide the extra mass nodes when the manipulator node is selected by user interaction. We can easily find the neighboring node and neighboring edges of the manipulator node using winged-edge data structure. The synthesis matrix for decideing the extra nodes which break the edges in area of onterest into two subedges is as follows.

$$P_{lazy} = \begin{bmatrix} I \\ \hline P_m \end{bmatrix} \qquad Q_{lazy} = \begin{bmatrix} 0 \\ \hline I \end{bmatrix}$$

where, P_m is matrix to decide the midpoints of edges, I means the indenty matrix.

In order to construct P_m matrix, we search the edges in the distance of one edge from the manipulator node by searching the edge table and write ½in the cell of matrix for start and end vertex of related edge. A sample for P_{lazy} and Q_{lazy} is as follows.

$$P_{lazy} = \begin{bmatrix} 1 & . & . & . \\ . & 1 & . & . \\ . & . & 1 & . \\ . & . & . & 1 \\ \frac{1}{2} & \frac{1}{2} & . & . \\ \frac{1}{2} & . & \frac{1}{2} & . \\ . & \frac{1}{2} & \frac{1}{2} & . \\ . & \frac{1}{2} & . & \frac{1}{2} \\ . & . & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & . & . & \frac{1}{2} \end{bmatrix} \qquad Q_{lazy} = \begin{bmatrix} . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & . & . & . & . & . \\ . & 1 & . & . & . & . \\ . & . & 1 & . & . & . \\ . & . & . & 1 & . & . \\ . & . & . & . & 1 & . \\ . & . & . & . & . & 1 \end{bmatrix}$$

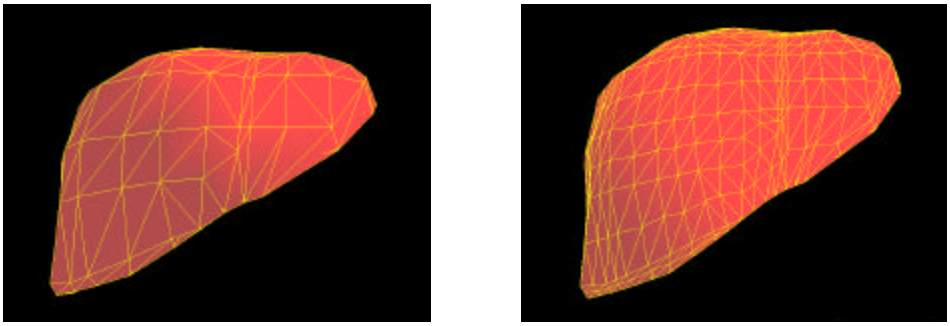
Using synthesis matrices and V_{old} column matrix, we can calculate the new set of mass nodes as follows.

$$V_{new}^{j} = P_{lazy}^{j} V_{old}^{j-1} + Q_{lazy}^{j} d^{j-1} \qquad \text{<eq. 4.3>}$$

where, V_{new}^{j} is new mass nodes and d^{j-1} is column matrix consisted of detail coefficients. Detail coefficients are defined in the pre-processing step for constructing coarse model and refined model.

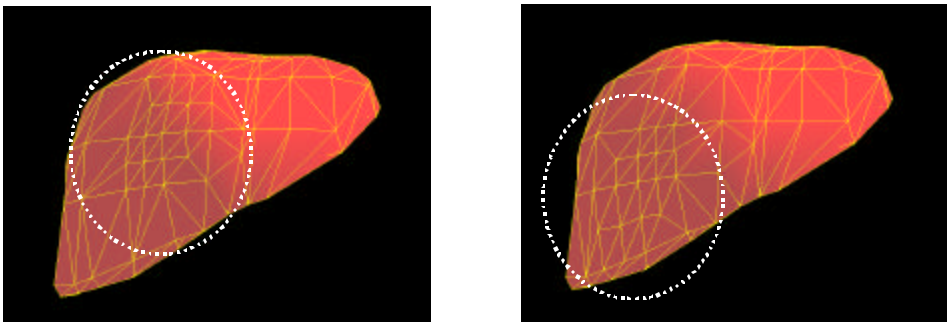
6. Results

In order to evalute the effectiveness of animation, we implemented the proposed adaptive and non-adaptive mass-spring method in Pentium III PC with 1GHz CPU and 1Gbyte RAM. We compared the execution time and the behaviour of coarse, refined and locally refined models based on the displacement variance of nodes which have the large magnitude of deformation. Figure 6.1 shows the original mass-spring model in coarse and refined detail levels. Figure 6.2 shows the locally refined mesh of original mass-spring model by proposed method. Table 6.1 summarizes the number of springs and the execution time until 1000 simulation time steps for each model.



(a) coarse model (b) refined model

Figure 6.1 Original mass-spring model



(a) manipulator node ID: 100 (b) manipulator node ID : 104

Figure 6.2 Locally-refined model

		No. of nodes	No.of springs	Execution time (sec.) for 1000 time steps
Conventional method	Refined model	550	1644	18.096
	Coarse model	139	411	6.249
Proposed method	Locally Refined Model	151	447	7.36

Table 6.1. Computation time comparison with conventional Mass-Spring method

In order to compare the overall behaviour of object, we computed the accumulated displacement of nodes in the refined area and displacement variance of the manipulator node according to simulation time. Figure 6.3 shows the accumulated displacement values and Figure 6.4 depicts the behaviour of the manipulator node.

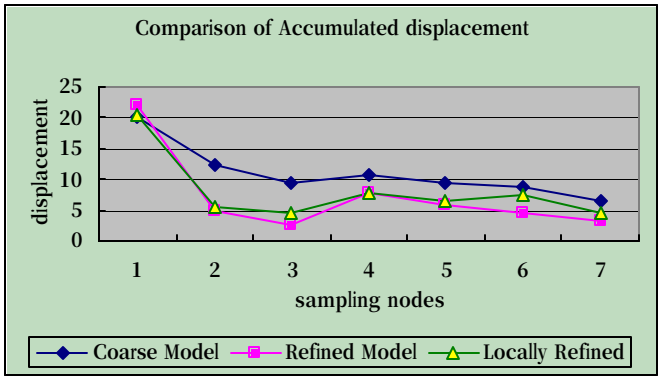


Figure 6.3 Comparison of accumulated displacement of sampling nodes

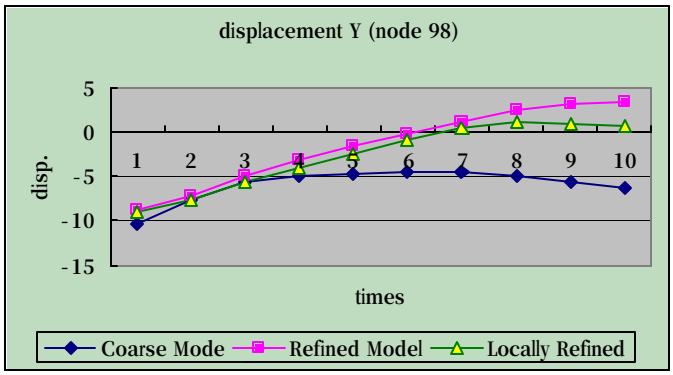


Figure 6.4 Comparison of variance of y-coordinates for manipulator node

The computation time is reduced about 60 percent of conventional refined mass-spring models and overall behaviour of refined model is preserved. Figure 6.5 depicts the geometrical deformation when we pull the manipulator node (green circle).

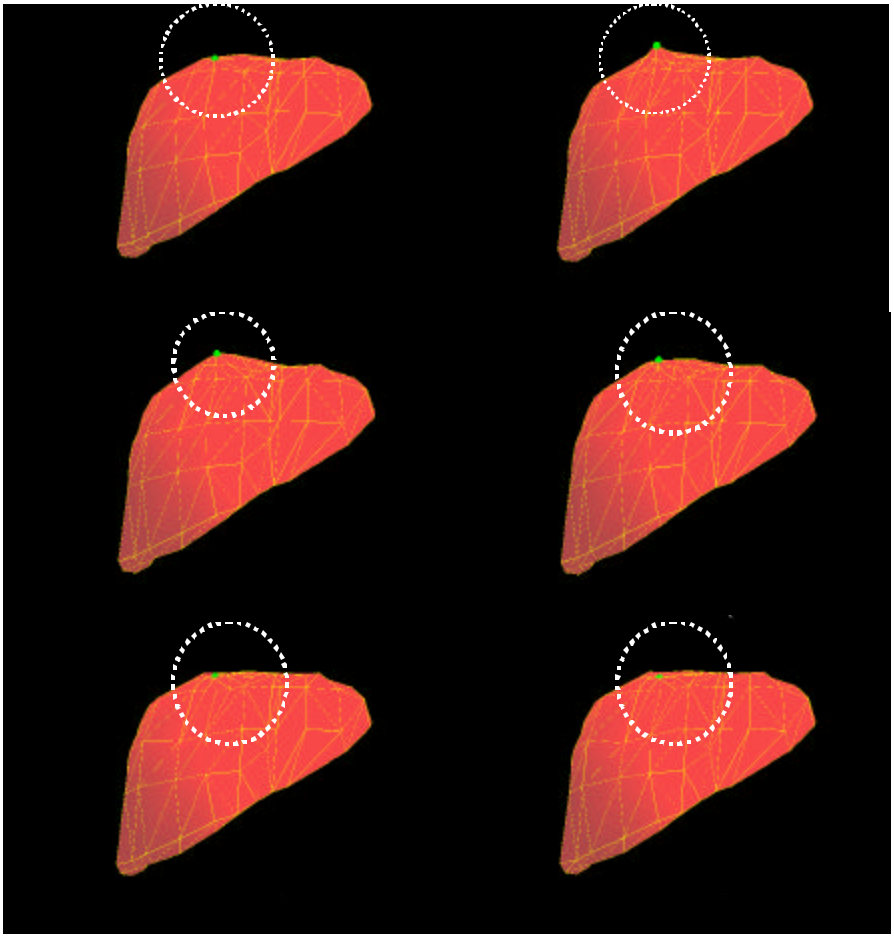


Figure 6.5 Geometrical deformation by locally refined method

7. Conclusion

We have developed an adaptive mass-spring model that can be animated at different levels of detail in order to reduce the computational effort while ensuring a same global behaviour for the

refined deformable model. We apply the surface wavelet method for the local subdivision surface. The local synthesis and analysis filters are dynamically constructed for wavelet transformation based on the location of the manipulator node. The winged-edge data structure enables us to detect the adjacent nodes, faces and springs in the constant time, which is necessary to construct synthesis and analysis filters. We can define the extra mass nodes by multiplying the existed mass nodes by synthesis filters. As the results of comparison with conventional and adaptive mass-spring model, the behaviour pattern of the locally refine model using the adaptive method is very similar to fully refined mass-spring model, while the computational time is reduced about 60 percent. The performance gain is more substantial as the number of nodes and springs increase. Dynamically determining the optimal area for local refinement based on the magnitude of deformation of each node remains as a future work. Moreover, we'll apply our adaptive approach to FEM and extend surface meshes to tetrahedral meshes in order to improve the realism of deformation.

Acknowledgement

This work was supported in part by the Korean Ministry of Science and Technology under the National Research Laboratory(NRL) Program and in part by the Korea Science and Engineering Foundation under Engineering Research Center (ERC) Program.

References

[1] D. Hutchinson, M. Preston, T. Hewitt, Adaptive Refinement for Mass/Spring Simulations, In Proc. Of the Eurographics Workshop on Computer Animation and Simulation, pages 31-45, Sep. 1996

[2] J. O'Brien and J. Hodgins. Graphical model and animation of brittle fracture, In Proceedings of SIGGRAPH'99, pages 137-146, 1999.

[3] Xunlei Wu, Micahel S. Downes, Tolga Goktekin, Frank Tendick, Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes, EUROGRAPHICS 2001, Volume 20 (2001), Number 3.

[4] G. Debunne, M. Desbrun, M. –P. Cani, and A. Barr. Adaptive simulation of soft bodies in real-time. In Proc. Computer Animation 2000. Phyladelphia, PA, May 2000, pp 15-20

[5] Hirofumi Kawai, Elastic Object Manipulation Using Coarse-To-Fine Representation of Mass-Spring Models, SIGGRAPH2001 Conference Abstracts and Applications, 2001.

[6] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle, Multiresolution Analysis of Arbitrary Meshes, In *Proceedings of SIGGRAPH '95*, pages 173-182. ACM, New York, 1995.

[7] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin**,** Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann, San Francisco, 1996.

[8] http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html

[9] Gilles Debunne, Mathieu Desbrun, Alan Barr, Marie-Paule Cani, Interactive Multiresolution Animation of Deformable Models, In 10th Eurographics Workshop on Computer Animation and Simulation, Milano, 1999.

[10] Gilles Debunne, Mathieu Desbrun, Marie_Paule Cani, Alan H. Barr, Dynamic Real-Time Deformations using Space & Time Adapative Sampling, ACM SIGGRAPH 2001, Los Angeles, 12-17 August, 2001.

[11] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, Roberto Scopigno, A Multiresolution Model for Soft Objects supporting interactive cuts and lacerations, Eurographics 2000, volume 19 (2000), Number 3

[12] James F. O'Brien, Jessica K. Hodgins, Graphical Modeling and Animation of Brittle Fracture, In SIGGRAPH 99 Conference Proceedings, pages 137-146, 1999.

[13] Xavier Provot, Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior, Graphics Interface, pp 147-154, 1995

[14] Gullaume Picinbono, Herve Delingette, Nicholas Ayache, Non-linear and anisotropic elastic soft tissue models for medical simulation, IEEE International Conference of Robotics and Automation, 2001.

[15] Mathieu Desbrun, Peter Schroeder, Alan Barr, Interactive Animation of Structured Deformable Objects, Technical Report 034, Center for simulation of dynamic response of materials, 1999.

[16] S. Gibson and B. Mirtich. A Survey of Deformable Modeling in Computer Graphics, Tech. Report No. TR-97-19, Mitsubishi Electric Research Lab., 1997

[17] D. Baraff. Linear-time dynamics using Lagrange multipliers. Computer Graphics Proceedings, Annual Conference Series: 137-146, 1996