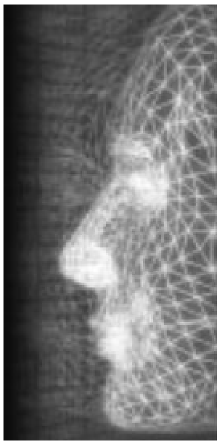
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2005; **16:** 69–83
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.57

Adaptive surface-deformable model with shape-preserving spring



*By Yoo-Joo Choi, Min Hong, Min-Hyung Choi and Myoung-Hee Kim**

This paper presents a multi-resolutional surface deformable model with physical property adjustment scheme and shape-preserving springs to represent surface-deformable objects efficiently and robustly. In order to reduce the computational complexity while ensuring the same global volumetric behaviour for the deformable object, we introduce a multi-resolutional mass-spring model that is locally refined using the modified-butterfly subdivision scheme. For robust deformation, a shape-preserving spring, which helps to restore the model to the original shape, is proposed to reduce the animation instability. Volume and shape preservation is indirectly achieved by restoring the model to the original shape without computing the actual volume and associated forces at every iteration. Most existing methods concentrate on visual realism of multi-resolutional deformation and often neglect to maintain the dynamic behavioural integrity between detail levels. In order to preserve overall physical behaviour, we present a new scheme for adjusting physical properties between different levels of details. During the animation of deformable objects, the part of the object under external forces beyond a threshold or with large surface curvature variations is refined with a higher level of detail. The physical properties of nodes and springs in the locally refined area are adjusted in order to preserve the total mass and global behaviour of the object. The adequacy of the proposed scheme was analysed with tests using practical mesh examples. Experimental results demonstrate improved efficiency in object deformation and preservation of overall behaviour between different levels. Copyright © 2005 John Wiley & Sons, Ltd.

Received: September 2003; Revised: June 2004

KEY WORDS: physically based modelling; mass-spring deformable model; spatial adaptation; adaptive refinement; multi-resolution; subdivision

Introduction

Real-time interaction with deformable objects plays a crucial role in many interactive virtual reality applications. Applications related to surgical simulation particularly demand real-time animation of deformable objects according to user interaction. Among the several methodologies for physically based modelling, mass-spring models are widely applied in virtual reality applications because they are effective in terms of computational cost and topological modification. However, since the accuracy of the model and simulation is dependent on the number of nodes and springs, a realistic simulation using a large data set still poses a significant challenge.

In the case of large mass-spring models, the nodes affected by external forces are usually a small subset of the total nodes, while the rest of the nodes usually remain still or oscillate invisibly. When all the nodes of an object are used to compute the deformation, the numerical system becomes unnecessarily large, hampering real-time performance. Therefore, combining level-of-detail (LOD) techniques with physically based modelling has been studied in order to reduce the computational load.[1–15] Most existing methods concentrate on the visual realism of multi-resolution deformation and often neglect to maintain dynamic behavioural integrity between detail levels.

*Correspondence to: Prof. Myoung-Hee Kim, Department of Computer Science and Engineering, Ewha Woman's University, Seodaemun-gu, Daehyun-dong 11-1, Seoul 120-750, South Korea. E-mail: mhkim@ewha.ac.kr

Contract/grant sponsors: Korean Ministry of Science and Technology; Korean Ministry of Information and Communication.

In this paper, a spatial adaptation scheme for mass-spring deformable models is presented that locally refines or simplifies the surface-deformable models at different LODs based on the modified-butterfly interpolation subdivision. This is a locally supported interpolation subdivision scheme and creates a smooth surface of arbitrary topology. Additionally, physical property adjustment scheme is proposed to preserve the overall behaviour of the object between different levels of detail. The preservation of dynamic behaviour between detail levels is particularly important for hollow, thin-shell type objects. Hollow objects, such as balloons or stomachs, can often be modelled with a surface mesh structure without internal nodes to reduce the total number of nodes. However, simple surface-deformable models combined with multi-resolution techniques without volumetric information may collapse more easily when mesh elements of different sizes and physical properties are involved. Since there is no volumetric structure to preserve the overall shape, maintaining the dynamic integrity between levels is a greater challenge. Therefore, we propose a shape-preserving spring that reduces instances of collapse and helps to maintain dynamic integrity between different levels. Volume and shape preservation is indirectly achieved by restoring the model to its original shape, and bypassing the expensive computation of element-level direct volume preservation at every iteration. The proposed shape-preserving spring combined with the new physical properties adjustment scheme enables us to use the layers of detail level effectively and stably.

The next section briefly reviews research on deformable modelling and adaptive approches in physically based modelling. The third section offers an overview of conventional mass-spring models. The adaptive deformable model combined with the physical property adjustment scheme is presented in the fourth section, and shape-preserving springs are introduced in the fifth section. Experimental results are shown in the sixth section and our conclusions are presented in the final section.

Related Work

Physical accuracy and real-time performance are conflicting goals in animating deformable objects. In the case of the mass-spring system, the equations of motion are integrated independently for each particle, which increases the speed of calculations. However, physical accuracy of the mass-spring system is not as good as that of the continuum-based finite element method (FEM). Conventional FEM uses exact differential operators that require solving a large sparse system. As a result, it is less compatible with real-time applications.

In the explicit FEM, each mesh node dynamically integrates its motion with the position of adjacent nodes. This method gives more accurate results than the mass-spring system. However, computing deformation at a fine discretization level requires fine sampling that prevents real-time performance. Some previous work[1–15] used LOD to locally refine a deformable model in regions of interest. Thingvold *et al.*[1] represented a table cloth as a B-spline surface, where particles corresponded to control points. Regions were refined where the simulation detected excessive stress. However, the B-spline surface can only represent a heavily restricted class of surface topologies. Hutchinson *et al.*[2] proposed an adaptive mesh refinement scheme for the rectangular mass-spring network. They represented a piece of draped cloth with the adaptive mass-spring model which refined the regions of high curvature. However, no method for simplification was provided. O'Brien[3] simulated glass fractures by refining a tetrahedral mesh along fracture lines. Meseure *et al.*[4] subdivided the mesh to make it more precise in the contact zone with rigid bodies such as surgery tools. Wu *et al.*[5] proposed a scheme for mesh adaptation based on an extension of the progressive mesh concept to produce a diagonal mass matrix that allowed for real-time computation. Wu *et al.* assumed that angles within each triangle should be at least 20 degrees and the valence of each vertex should be less than 10. Debunne *et al.*[6–8] proposed a method of physically based modelling that combined a linear finite volume-based mechanical model with a non-hierarchical refinement technique based on Voronoi's concept. The relationships between disparate resolution meshes were maintained using ghost nodes that transmit the mechanical behaviour of a mesh section at one resolution to adjacent mesh sections of different resolutions. Generally, the subdivision methods based on Voronoi's concept are less effective than other subdivision methods. Kawai[9] built hierarchical mass-spring models in order to manipulate elastic objects using a coarse-to-fine representation of mass-spring models based on the degree of deformation. Ganovelli *et al.*[10] presented a tetrahedral decomposition scheme of space to ensure required speed-up and to support dynamic changes of the topology due to cuts or lacerations in the represented tissue. Grinspun *et al.*[11] introduced a framework for the construction of adaptive solvers for PDEs partial differential equations (PDEs) based on basis function refinement in order to reduce the difficulty of implementing adaptive simulations. Most of the existing multi-resolutional mesh schemes in cloth animation assume that the initial fine mesh should be categorized in the heavily restricted class of surface topologies.[2,12,13] Further, most existing soft object deformation schemes combined with a LOD technique often neglect to maintain dynamic behavioural integrity between detail levels.

In this paper, an adaptive surface-deformable model suitable for hollow objects with arbitrary topology is proposed which shows the dynamic behavioural integrity between different detail levels. The proposed model includes the spatial adaptation scheme for arbitrary topology based on modified-butterfly interpolation subdivision and a physical property adjustment scheme to preserve the overall behaviour of the object between different detail levels. It has a shape-preserving spring to reduce collapsing induced by mesh elements under different conditions and to increase dynamic integrity maintenance between different levels by enhancing restoration of the original shape in each level.

Overview of Mass-Spring Deformable Model

Demands for real-time performance in modelling and simulation of deformable objects have led many researchers to concentrate on simple elastic models. The mass-spring model is a simple discrete approximation of a continuous model which assumes that the mass is concentrated on nodal points and the nodes are connected to neighbour nodes by massless springs. The mass-spring model has been widely studied and used in surgery simulations and cloth animation because it is effective when representing topological changes along with the possibility for reasonably fast simulation speed. The mass-spring model is well suited for interactive surgery simulation and medical animation where real-time feedback and prompt responses are required. The simulation of human organs using a mass-spring system consists of a number of mass points and springs. The mass point, hereafter referred to as nodes, is connected by springs to propagate the energy. The governing Lagrange's equation for the mass-spring model is

$$m_i\ddot{x}_i + \gamma_i\dot{x}_i + k_i x_i = f_i + g_i \tag{1}$$

where x_i, $\dot{x}_i$, and $\ddot{x}_i$ are the position, the velocity and the acceleration vectors of node i, respectively. m_i, γ_i, and k_i are scalar values to represent the nodal mass, the damping and stiffness coefficient at node i. g_i is the spring force vector to be applied on node i through the springs connecting node i, and f_i is the external force vector on node i.

In order to select a pertinent ordinary differential equation (ODE) solver, we comparatively analysed several ODE solvers from the perspective of numerical stability and efficiency. In case of a large time step and a stiff object, implicit methods are unconditionally more robust and more efficient than explicit methods. However, implicit methods may be inefficient when a small step size is applied because inversion of a large sparse matrix is required in each time step. In contrast, explicit methods are only conditionally stable, and therefore tend to converge more slowly than implicit methods. In our proposed adaptive method, the time step could become small during dynamic refinement and adjustment of physical properties. Therefore, explicit integration is often more appropriate for our methods. As a result of experimentation under various conditions, the explicit fourth-order Runge–Kutta method[16,17] proved to be effective and robust. We chose to use this method becuase it is a good general candidate for numerical solving of differential equations when combined with an intelligent adaptive step-size routine.

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

$$\begin{aligned} g_1 &= -\left[k_s(|\Delta x| - r) + k_d\left(\frac{\Delta v \cdot \Delta x}{|\Delta x|}\right)\right]\frac{\Delta x}{|\Delta x|} \\ g_2 &= -g_1 \end{aligned} \tag{2}$$

where g_1 and g_2 are the spring forces on *node 1* and *node 2*, respectively. k_s is a spring constant and k_d is a damping constant. $\Delta x = x_1 - x_2$ is the actual distance between the two nodes, and $\Delta v = v_1 - v_2$ is the difference between the two nodes' velocities. r is the rest length to be pre-computed based on the initial positions of nodes. k_s, k_d and r are scalar values, and spring forces Δx and Δv are vector components. The operation represents an inner product of two vectors. In equation (2), the spring force magnitude affected by the spring constant is proportional to the difference between the actual length and the rest length of the spring, while the damping force magnitude related to the damping constant is proportional to the nodes' speed of approach.

Building an Adaptive Deformable Model with Property Adjustment

Spatial Adaptation using Modified-Butterfly Interpolation Subdivision

Our spatial adaptation scheme is based on the dynamic combination of different detail levels in any given region that receives excessive external forces or has large surface curvature variation. This reduces the number of mass nodes and springs for computation. Various surface subdivision methods can be applied locally for surface refinement. Because approximation-based subdivision schemes shrink the subdivided surface from the original coarse surface mesh, these methods make it hard to preserve deformation integrity between detail levels. On the other hand, interpolation-based subdivision schemes prevent subdivided surfaces from shrinking because all nodes in the initial mesh are contained in the final subdivision surface.

The butterfly subdivision scheme[18] generates a smooth surface that interpolates the vertices of a triangular mesh by decomposing polygons on the mesh. Because this method is an interpolation-based scheme, the distances between the vertices of the initial mesh are maintained after subdivision. A triangle on the mesh is decomposed into four triangles, where new vertices are created on each edge of the original triangle. Although this scheme generates C^1 continuous surfaces near the regular vertices, some degeneracy can appear near the irregular vertices. To resolve this degeneracy, Zorin *et al.*[19] proposed a modified butterfly subdivision that maintains the eigenstructure of the regular cases near the irregular vertices. We applied Zorin's modified-butterfly method to guarantee smooth subdivision of the irregular mesh, which defines an arbitrary topology, without shrinking. This was implemented to overcome heavy restrictions in the existing spline-based adaptive animation methods to the geometric topology and mesh properties. A new scalar value for each edge midpoint of the triangulation is computed as a position of a new vertex. The positions of new vertices are determined from different schemes according to the regularity of the adjacent vertices on the edge. Edges to be subdivided can be divided into four classes.

- *The edge connects two vertices of valence six.* In this regular case, the position of the new vertex is determined from subdivision mask of equation (3) and Figure 1(a).

$$q^{k+1} = \frac{1}{2}(p_1^k + p_2^k) + \frac{1}{8}(p_3^k + p_4^k) - \frac{1}{16}(p_5^k + p_6^k + p_7^k + p_8^k) \tag{3}$$

- *The edge connects a K-vertex ($K \neq 6$) and a 6-vertex.* The 1-neighbours of the K-vertex are used in stencil as indicated in Figure 1(b). In this case, the following formula is used:

$$q^{k+1} = \sum_{i=0}^{N-1} s_i p_i^k \tag{4}$$

In this formula,

$$S_i = \frac{\frac{1}{4} + \cos\left(\frac{2\pi i}{N}\right) + \frac{1}{2}\cos\left(\frac{4\pi i}{N}\right)}{N}, \qquad 0 \leq i \leq N-1 \tag{5}$$

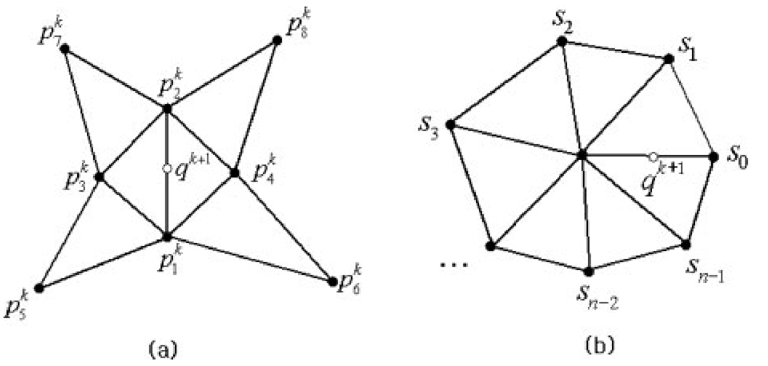


Figure 1. Modified butterfly subdivision mask: (a) regular cases; (b) irregular cases.

for $N \geq 5$. In the case of $N=3$, $s_0 = \frac{5}{12}$, $s_1 = s_2 = -\frac{1}{12}$ and, in the case of $N=4$, $s_0 = \frac{3}{8}$, $s_2 = -\frac{1}{8}$, $s_1 = s_3 = 0$.

- *The edge connects two extraordinary vertices.* If both of the vertices are irregular, the above formula is applied to both vertices and the results are averaged.
- *Boundary edges.* Boundary edges are subdivided using the one-dimensional four-point scheme $\left(s_{-1} = \frac{-1}{16},\ s_0 = \frac{9}{16},\ s_1 = -\frac{9}{16},\ s_2 = -\frac{1}{16}\right)$.

The proposed spatial adaptation scheme is divided into an off-line pre-processing phase and an on-line animation phase. In the pre-processing phase, the multi-level mesh structure is constructed from zero-level (un-subdivided) mesh based on the modified-butterfly scheme. Each node and edge in the initial mesh has zero as its LOD value. For the first-level mesh, new faces, mass nodes, edges and edge table information are created with one as the LOD value. Each mass node preserves the natural support set for the modified-butterfly subdivision. The natural support set refers to the minimal set of elements, which contain parametric support of the basis function for the modified-butterfly subdivision. Each face has pointers to the parent and child faces.

Dealing with the mass of new nodes and properties of springs is very important to preserve the overall dynamic behavior. Detailed methods are described in the next section. All nodes and springs in the same detail level have the same mass and physical properties. The velocity and force for each node is set to zero in the pre-processing step. In the pre-processing step, all elements—nodes, faces and edges—of all detail levels are pre-computed and stored in proprietary data structures to be selectively used later in the animation phase. The nodes, faces and edges at the zero level are activated and other elements belonging to upper detail levels are deactivated in the inital state. In the on-line animation phase, the criteria for refinement and simplification are checked for each node. If the criteria are satisfied, the neighbouring faces of the node are subdivied or simplified. The algorithms for subdivision and simplification are illustrated in Figure 2.

In the subdivision algorithm, the neighbouring faces at the same detail level as the node under large external forces or with large surface curvature variation are selected. Elements of the selected faces are deactivated. The subfaces of the selected faces and the edges of the subfaces are activated. Midpoint nodes of deactivated

```
Algorithm Subdivision ( int NodeID)
{
  LD = LOD-level for NodeID;
  Find incident faces of NodeID in LD level;
   For each incident face, FaceID {
     Increase the number of refinement for FaceID;
     If (faceID is deactivated already) Continue;
     Deactivate current face FaceID;.
    Deactivate current edges for FaceID;
    Activate sub-faces;
    Active sub-edges;
    Active mid-point node of edge;
    Adjust the position of mid-point node of edge;
    Adjust velocity and force of mid-point node of
         edge;
   }
   LOD-level for NodeID = LD +1;
}
```

```
Algorithm Simplification ( int NodeID)
{
  LD = LOD-level for NodeID;
  Find incident faces of NodeID in LD level;
   For each incident face, FaceID {
     Decrease the number of refinement for FaceID;
      If ( The number of refinement for FaceID is
             greater than 0 ) Continue;
    Find the parent face of FaceID;
    Find the sibling face of FaceID;
    Deactivate the sibling faces and FaceID;
    Deactivate edges of the sibling faces and
         FaceID;
    Deactivate mid point nodes of edges of parent
         face;
    Activate the parent face of FaceID;
    Activate the edges of parent face;
    }
   LOD-level for NodeID = LD-1;
}
```

Figure 2. Algorithm for subdivision and simplification in the animation phase.

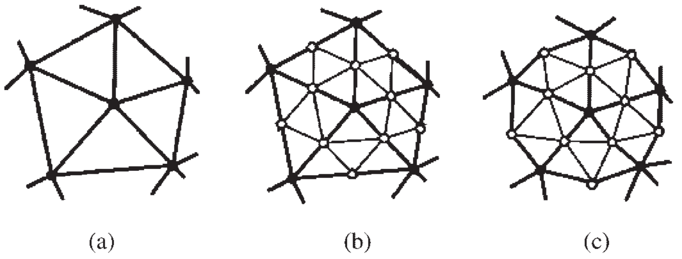


Figure 3. Local spatial subdivision: (a) original mesh; (b) splitting step; (c) averaging step.

edges are activated and positions of those nodes are recomputed using a natural support set for modified butterfly subdivision by equations (3), (4) and (5). The force and velocity for midpoint nodes are computed by averaging the adjacent nodes on the corresponding edge. Figure 3(a) depicts the original mesh and Figure 3(b) illustrates the splitting step which explicitly activates the midpoint node for each edge. Figure 3(c) explains the averaging step which computes the weighted average position for each midpoint node using the modified butterfly subdivision equation.

The mass of each node is recomputed in each simulation step. Details are explained in the next section.

The cracks on the subdivided surface can occur due to hanging nodes that are generated by the adaptive subdivision. Adaptive rendering is required in order to eliminate the artificial crack status generated by hanging nodes. In the adaptive rendering process, active faces that include hanging nodes are decomposed into proper triangles using compensating edges. Active faces in any refinement level are classified into four types according to the status of component edges:

- *Type 1.* Active faces in the first type have three active component edges because the refinement levels of the active face and three adjacent faces are all the same as shown in Figure 4(a). In this case, the face decomposition process is not required because any hanging nodes will not be found in the active face.
- *Type 2.* In the second type, a neighbouring face of an active face is in the higher refinement level. As a result, one hanging node is generated. A compensating edge between the hanging node and the normal node on the opposite side is added and the current active face is divided into two triangles as shown in Figure 4(b).
- *Type 3.* An active face of the thrid type has two hanging nodes due to two neighbouring faces in a higher refinement level. A compensating edge between two hanging nodes and another compensating edge that divides the active face into two equal triangles are generated. An active face in the third type is represented by three triangles as shown in Figure 4(c).
- *Type 4.* In the fourth type, an active face has three hanging nodes due to three neighbouring faces in the higher refinement level. Three compensating edges that connect the hanging nodes are added and the active face is decomposed into four component triangles as shown in Figure 4(d).

The compensating edge that is generated during the adaptive rendering proces is not used in the spring force computation. Figure 5 depicts a crack status due to a hanging node and the result of adaptive rendering using a compensating edge. The blue edge in Figure 4(b) represents the added compensating edge.

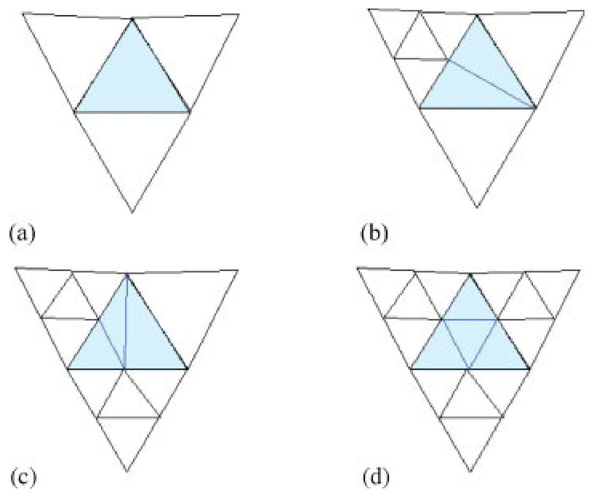


Figure 4. Adaptive rendering of an active face: (a) without a hanging node; (b) with a hanging node; (c) with two hanging nodes; (d) with three hanging nodes.

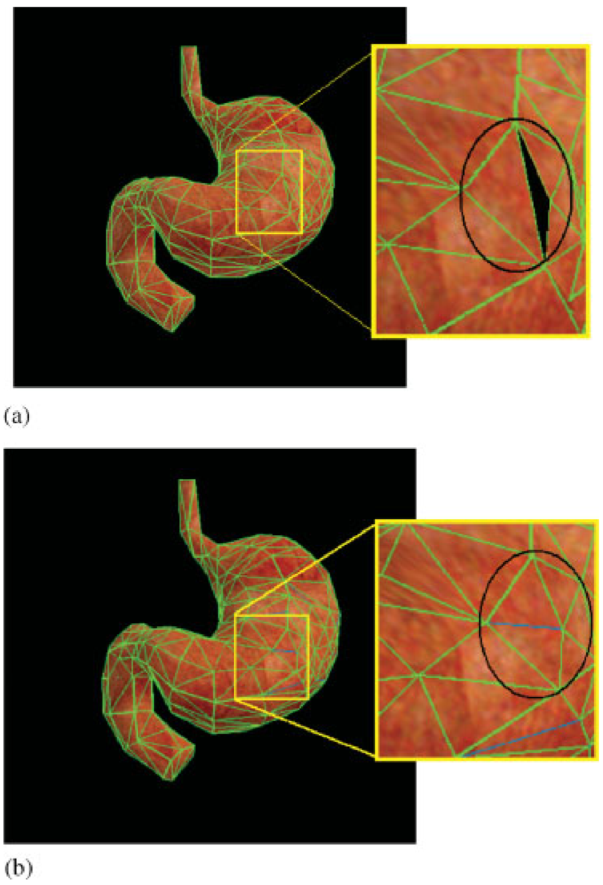


Figure 5. Comparison of active face rendering: (a) without adaptive rendering of active faces; (b) with adaptive rendering of active faces.

Adjustment of Physical Properties

An important problem in multi-resolutional deformable models is how to infer the physical properties, such as mass, velocity, acceleration and stiffness, of the nodes and springs in the simulation. In this section, we will explain how to adjust the physical properties in the different levels of detail.

Mass adjustment methods are classified into two categories. The first category uses extra nodes with the same mass, which increase overall mass of the object. The other category preserves the overall mass of the object after refinement by adding an extra node of a lesser mass. Hutchinson *et al.*[2] added extra nodes with same mass and springs in refinement. In this method, the total mass of the object is not maintained after refinement. In an interactive case, the mass of the region with external force is suddenly increased and could behave differently from that of other regions. On the other hand, Thingvold and Cohen[1] added nodes of lesser mass to preserve the overall mass of an object. In Thingvold's method, all the original control points and the refined control points respectively had a unit mass value. Therefore, the mass of the refined region is larger than that of the coarse region.

To preserve the mass of an entire object, the mass of each node can be determined proportionally to the area of neighbouring faces or tetrahedra.[13,21] However, in this case, skinny or sharp triangles that could be made during simulation caused the model to be numerically sensitive in a specific direction. Cotin *et al.*[22] pointed out that this method tends to create ill-conditioned iterative schemes when mesh elements are of different sizes. In order to maintain the total mass of object and to reduce the numerical instability, we calculate the mass per node for each detail level in the pre-processing step and redistribute the mass of nodes in a refined or simplified region only at the time of refinement or simplification, and maintain the value during simulation. The unit mass value of a node for each refinement level is determined by equation (6) in the pre-processing phase. In the animation phase, the mass of each node in the refined region is modified according to refinement level of neighbour edges. As a result, the mass of the refined region and overall mass of an object is preserved during dynamic refinement.

$$M_{\text{refine}} = \frac{\sum M_{\text{coarse}}}{N_{\text{refine}}} \tag{6}$$

In this equation, M_{refine} and M_{coarse} respectively are the masses for each node in the refined level and in the zero level. N_{refine} is the number of nodes in the refined level. In the on-line animation step, nodes in different levels of detail coexist in the same simulation time. In order to preserve the overall mass and uniformly distribute the mass over an object, the mass of each node is recomputed in each simulation step based on the detail levels of the active adjacent edges.

For example, the circle node in Figure 6(a) has five zero-level edges as adjacent edges, and therefore the mass of the circle node is the same as the mass of the zero-level node. The node marked with a red triangle in Figure 6(b) has two zero-level edges and three one-level edges as its adjacent edges. Therefore, the mass of the triangle node is computed as follows:

$$\begin{aligned} M = ((&\text{Mass of zero-level node}) \\ &\times 2 + (\text{Mass of one-level node}) \times 3)/5 \end{aligned} \tag{7}$$

The circle node in Figure 6(b) has the mass of a one-level node, because it has five adjacent first-level edges.

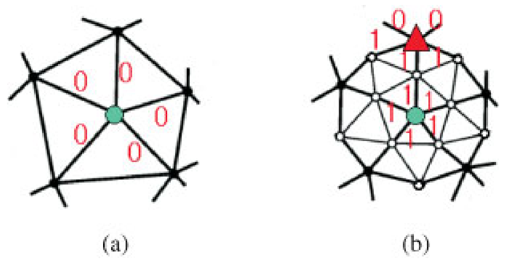


Figure 6. Adjustment of mass in the on-line simulation phase: (a) nodes in the same detail level; (b) nodes in multiple detail level.

To compute the stiffness coefficient of the spring, Gelder[20] proposed a derivation of stiffness that varies with the triangle area over the edge length squared in a non-multi-resolutional model. However, this approach tends to create ill-conditioned iterative schemes when mesh elements of different sizes are involved. Villard[13] used the LoD to calculate the internal forces. This had the same effect as using the LoD to adjust the stiffness and damping factors.

The acceleration of a node is determined by the force to be imposed on the node and by its mass. The force is affected by spring stiffness, damping parameter and spring length. In our sheme, the mass is adjusted to preserve the total mass of the object during animation. In order to maintain the acceleration of a node between different detail levels, stiffness and damping parameters are modified according to the adjusted mass and variation in springs length. Therefore, we recalculate the stiffness and damping factors based on the LoD of the current spring. We vary the mass per node of the current level based on the mass of the initial refinement level. In the pre-processing step, the stiffness and damping factors for a zero-level spring are determined by the two-dimensional Young's modulus and those for the refined-level springs are calculated using equation (8). The values for each spring are preserved in the online animation phase:

$$
\begin{aligned}
KS_{\text{refine}} &= 2^{l} \times KS_{\text{zero-level}} \\
&\quad \times \frac{\left(\sum_{v \in P_{\text{refine}}} \text{Mass}(v)\right)/N_{\text{refine}}}{\left(\sum_{v \in P_{\text{zero-level}}} \text{Mass}(v)\right)/N_{\text{zero-level}}} \\
KD_{\text{refine}} &= KD_{\text{zero-level}} \times \frac{KS_{\text{refine}}}{KS_{\text{zero-level}}}
\end{aligned}
\tag{8}
$$

where KS_{refine} and KD_{refine} respectively are the stiffness and damping values for a spring of a given refinement level. N is the number of mass nodes for each detail level and l is the current refinement level. $\sum_{v \in P_{\text{refine}}} \text{Mass}(v)$ is the total mass of nodes at the refinement level P_{refine} and $\sum_{v \in P_{\text{zero-level}}} \text{Mass}(v)$ is the total mass of nodes at the initial refinement level. Mass per node for each refinement level is pre-computed. The mass per node ratio with respect to that of the initial refinement level is used in the computation of the stiffness constants for each refinement level. The damping constant is computed according to the ratio of the stiffness constant in the high refinement level to the stiffness constant in the initial refinement level.

The time-step for each simulation step is set according to the most refined level. In the pre-processing phase, the time-step for the zero-level model is determined and in the on-line animation step the time-step for each simulation phase is recomputed using equation (9):

$$
TS = \frac{TS_{\text{zero-level}}}{2^{l}} \tag{9}
$$

where l is the most refined level in the current simulation step.

Typically, the velocity and force of the added nodes are computed by interpolating the unknown values from nodes present in the representation at the previous step.[6,10,14] In our method, the force and velocity for the midpoint node that is added during the simulation are computed by averaging forces and velocities of adjacent nodes on the corresponding spring.

Criterion for Refinement and Simplification

We have applied three kinds of criteria for refinement and simplification. Any node that satisfies one among three criteria allows the neighbour faces to be refined to a proper level. Otherwise, the neighbour faces are simplified.

Criterion Based on Force Concentration. In surgical simulation, large force concentrations accumulate where instruments press or pull tissue. If a node receives an external force beyond a threshold, the surfaces incident to that node are subdivided.

Criterion Based on Velocity Field. Occasionally, a node has a high velocity even after the external force is removed and the total force on the node is reduced due to its propagation into the neighbour nodes. In those cases, the faces adjacent to the node are subdivided.

Criterion Based on Curvature Variation. We apply curvature variation to find discontinuities on the surface. Nodes with a large change of surface curvature in comparison to the initial curvature in the rest state are identified. If variations in the curvature are larger than a threshold, the faces incident to the vertex are subdivided by one refinement level.

Shape-Preserving Spring

In soft tissue simulation or surgery simulation, volumetric objects with internal material such as the liver are constructed with internal regions and surface regions using tetrahedral meshes or cells. The cavity objects without internal material, such as the stomach and intestines, often can be modelled with a surface mesh structure without internal nodes to reduce the total number of nodes. In this case, only surface neighbouring nodes are joined together by damping springs. Such a model is very simple to construct and easy to combine with multi-resolution techniques. Real-time interaction with deformable models favours surface mesh-based models because surface mesh has fewer springs and nodes than tetrahedral mesh. However, simple surface mesh-based deformable models can collapse more easily than tetrahedral mesh models after strong external force is imposed for some time and removed or when mesh elements of different size and physical properties are involved because of lack of volume coherency. This makes it difficult to restore the model to its initial shape after deformation and to maintain dynamic integrity between detail levels.

To solve this problem, we propose a shape-preserving spring that reduces the collapsing and helps to maintain dynamic integrity between different levels by restoring the model to the original shape. The proposed shape-preserving spring is a zero-length rest spring that prevents each corresponding node from an excess of displacement. Each mesh node in the surface is joined to a virtual node with one extra spring. The position of the virtual node accords with the initial position of the corresponding mass node. Meseure[4] also used zero-length rest springs to handle deformable models submitted to rigid motions such as translation and rotation. They linked the rigid component to a damped spring surface mesh. In case of contact, the rigid and deformable components separate each from the other. When the interaction stops, the two components are joined again. In our proposed scheme, a zero-length rest spring for surface mesh nodes is always joined during deformation of an object, which prevents excessive displacement of the node and is helpful in restoring it to its initial state without collapsing. Figure 7 shows a surface mass-spring model with shape-preserving springs.

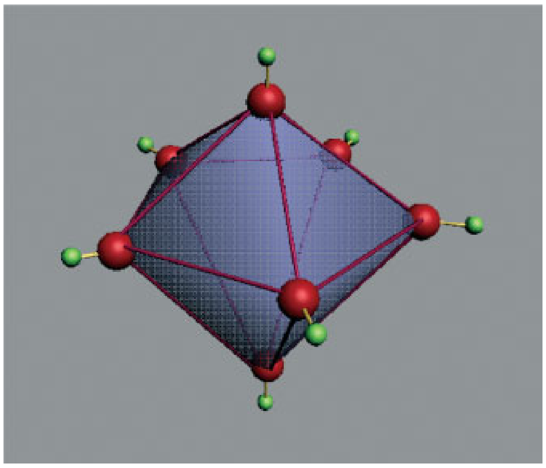


Figure 7. Surface mass-spring model with shape-preserving spring.

The red spheres represent the mass nodes on the surface and the red lines are the springs between mass nodes. The green spheres show the virtual nodes that preserve the initial shape and the yellow lines depict zero-length rest springs.

The spring force between mass node and virtual node is computed after the shape-preserving spring has lengthened. The shape-preserving spring force that affects the mass node is computed as equation (10):

$$g_1 = -\left[k_s'|\Delta x| + k_d'\left(\frac{v_1 \cdot \Delta x}{|\Delta x|}\right)\right]\frac{\Delta x}{|\Delta x|} \tag{10}$$

In this equation, k_s' is a spring constant and k_d' is a damping constant for the shape-preserving spring. Constant values are determined by the heuristic method according to the material properties of the object. $\Delta x = x_1 - x_2$ is the position difference between the mass node and the corresponding virtual node. v_1 is the current velocity of the mass node. In equation (10), all components except the spring and damping constants are vectors. The shape-preserving spring force magnitude is proportional to the displacement between the current node and virtual node. Therefore, when an excessive displacement of a node occurs, a large shape-preserving spring force is added to the node in the direction of the corresponding virtual node.

Implementation and Experimental Results

To evaluate the effectiveness of animation, we implemented the proposed adaptive and non-adaptive mass-spring method on a Pentium III PC with 1 GHz CPU with 1 GB RAM. We constructed a winged-edge[23] data structure for mass-spring models in order to effectively detect the neighbouring faces and edges. The winged-edge data structure is a boundary representation method using the edge's ordered relationship. Using the winged-edge data structure, we could easily find which

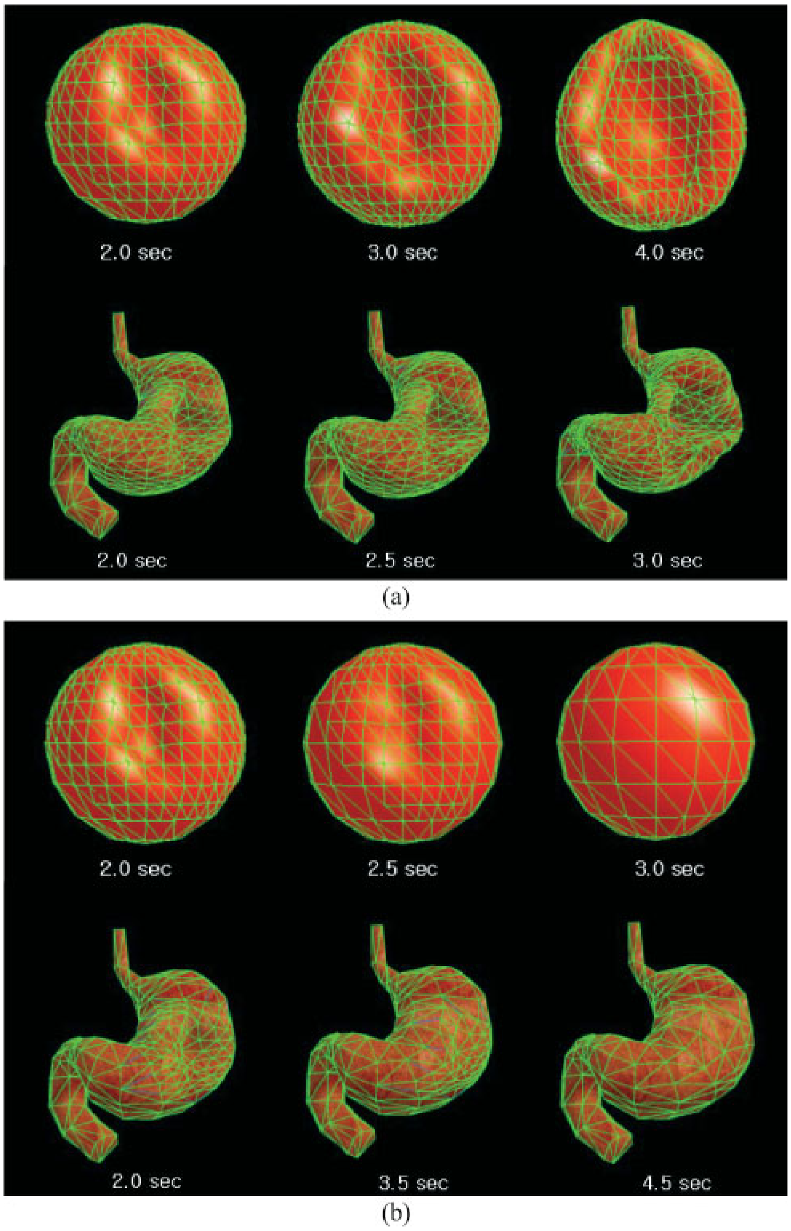


Figure 8. Comparison of mass-spring models with and without shape-preserving spring: (a) model without shape-preserving spring; (b) model with shape-preserving spring.

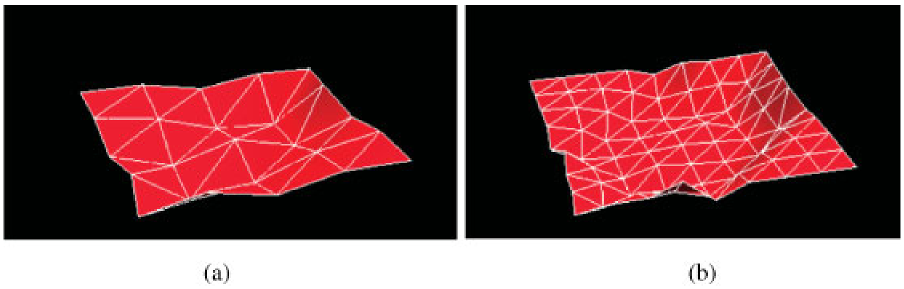


Figure 9. Visual comparison of deformation of coarse and refined models: (a) coarse model; (b) refined model in which the physical parameters are adjusted using the proposed adjustment scheme.

vertices, edges and faces were adjacent to each face, edge or vertex. The following information was saved for each edge in the edge table with the vertex, face and edge list:

- start and end vertices of this edge;
- its left and right faces;
- the predecessor and successor of this edge when traversing its left face;
- the predecessor and successor of this edge when traversing its right face.

In order to show the usefulness of the proposed shape-preserving spring, we studied sphere models with and without the shape-preserving spring. We pressed some region of the spheres for a simulation time of 2 s and then removed the pressure. Figure 8 depicts the snapshots of two deformation models.

Figure 8(a) shows that the model without the shape-preserving spring is not restored to its initial shape and remains distorted even after the external force is removed. The model with the shape-preserving spring in Figure 8(b) restored itself to its initial shape after we removed the pressure.

We also experimented with a simple surface mass-spring model—a simulated rectangle of stiff fabric—in order to validate the adjustment of physical properties between different levels of details. We wholly refined the coarse mass-spring model using the proposed adjustment scheme and applied pressure on the central part of the model for 1 s. We compared the overall behaviour of the coarse and refined models. Figure 9 depicts the results of deformations of the coarse and refined models using the same simulation time for visual comparison. Figure 10 shows the displacements of the nodes to which forces were applied in the coarse and the refined models with respect to time. The deformation shape of the two models and their node displacements nearly concide.

We compared the execution time and the behaviour of the coarse, the refined and the proposed adaptive

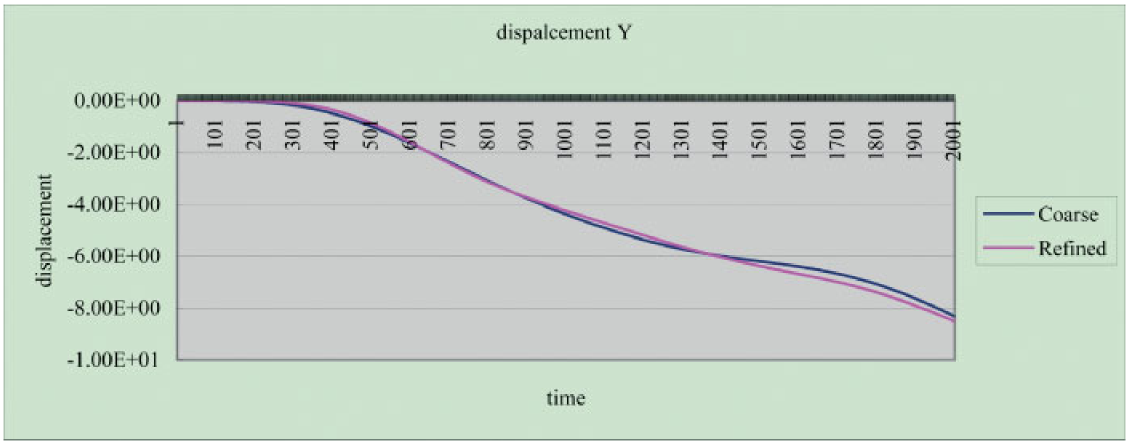


Figure 10. Comparison of displacement of forced nodes in the coarse and the refined model with the proposed adjustment scheme.

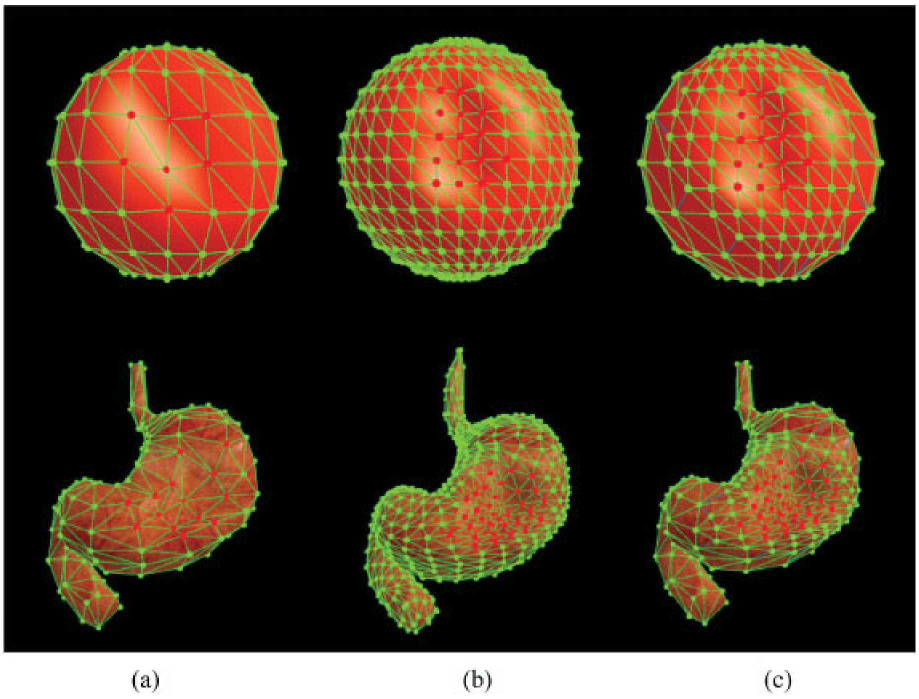


Figure 11. Visual comparison of the deformation of the three models: (a) coarse model; (b) refined model; (c) proposed adaptive model with adjustment scheme and shape-preserving spring.

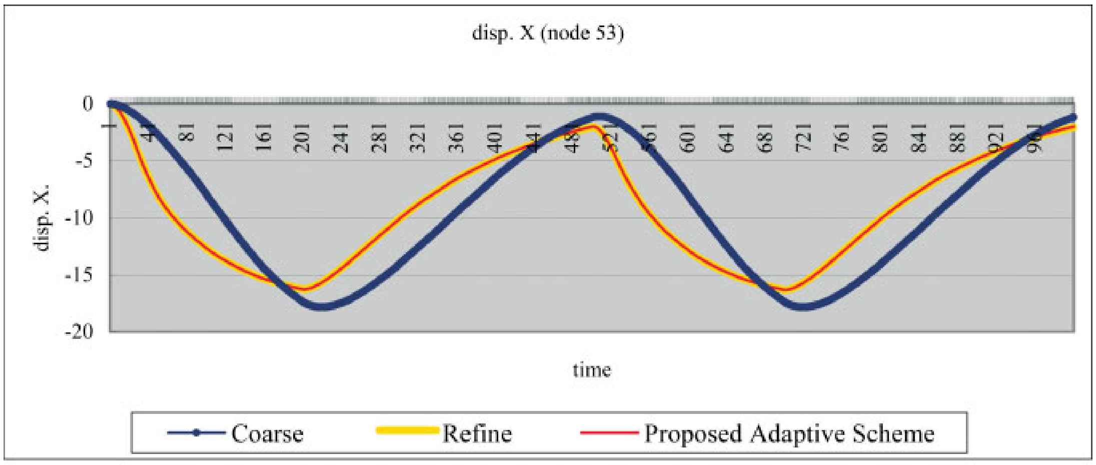
model. The overall behavior of each model was evaluated by visual comparison and the displacement variance of nodes that showed large magnitudes of deformation. Figure 11 shows the results of deformation of the three models after 2.3 s. The red nodes in Figure 11 represent nodes that have displacements beyond the threshold value. The similarity of the red node range indicates that the deformation behaviour is preserved between the coarse, the refined and the adaptive models after the proposed adjustment scheme of physical properties is applied.

The execution time for each model is shown in Table 1. Figure 12 compares the displacement of forced nodes in the coarse, the refined and the proposed adaptive sphere models. The displacement graph of the forced node for the refined and the proposed adaptive models coincide, and the forced node in the coarse model animates similarly to the forced node in other models.

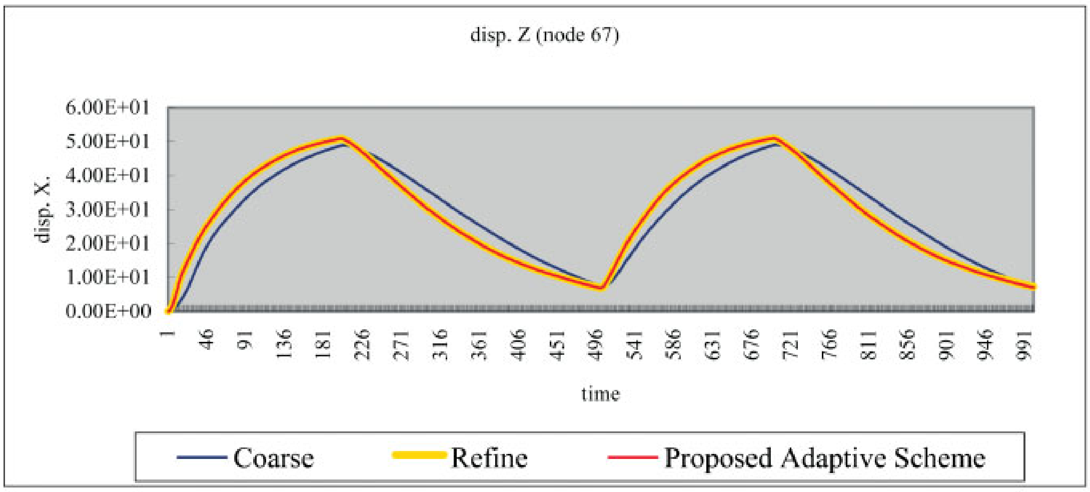
We applied texturing to the stomach model, which increased the realism of the deformation. Figure 13 shows a deformation of the stomach model with the texture.

Model		Conventional mass-spring			Proposed adaptive method		
		No. of nodes	No. of spring	ExeTime (s)	No. of nodes	No. of spring	Exe. Time (s)
Sphere model	Coarse	114	336	2.334	114∼133	336∼385	3.136
	Wholly Refined	450	1344	10.078			
Stomach model	Coarse	188	558	3.328	188∼253	558∼737	6.906
	Wholly Refined	746	2232	14.422			

Table I. Execution time comparison with conventional methods and the proposed method

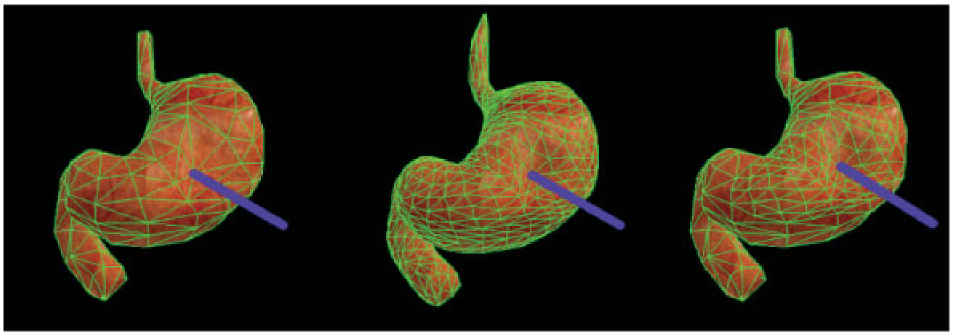


(a)



(b)

Figure 12. Comparison of displacement of forced nodes in the coarse, the refined and the proposed adaptive model: (a) for the sphere model; (b) for the stomach model.



(a) (b) (c)

Figure 13. Three stomach models with textures. (a) Coarse model (number of nodes and springs: 188, 558). (b) Refined model (number of nodes and springs: 746, 2232). (c) Proposed adaptive model (number of nodes and springs: 188～240, 558～700).

Conclusion

In this paper, we presented an efficient and robust approach representing surface-deformable objects using spatial adaptation and a shape-preserving spring. Moreover, a new adjustment scheme of physical properties between different levels of detail to preserve overall physical behaviour was described. To reduce the computational complexity while ensuring the same global volumetric behaviour for the deformable object, we introduced a multi-resolutional mass-spring model that is locally refined using a modified butterfly interpolation subdivision scheme and effective transition of physical properties between detail levels. For robust deformation, a shape-preserving spring was applied which reduces instability in animation. Volume and shape preservation is indirectly achieved by restoring the model to the original shape, without computing the actual volume and associated forces at every iteration. The proposed adaptive surface-deformable model is suitable for cavity objects with arbitrary topologies and guarantees the dynamic behavioural integrity between different detail levels.

The proposed adaptive method was validated in the current study using several tests on practical mesh examples. First, strong external forces were imposed on sphere models with and without the shape-preserving spring. The proposed spatial adaptation and physical property adjustment scheme was applied to two sphere models. The experimental result indicates that the proposed shape-preserving spring prevents abnormal distortion and is useful in restoration of the object to its initial shape and in preventing collapse. In the second experiment, the proposed adjustment scheme was validated by comparison of behaviour of the coarse model and the refined model adjusted by the proposed scheme. Finally, execution time and object behaviour for coarse, entirely refined and adaptive models were compared. Comparison of results showed that the proposed adaptive scheme reduced computational time by about 52–62% compared with the refined model while preserving the behaviour of the original model. Performance gains are more substantial as the number of nodes and springs increase. In future work, an adaptive approach to FEM will be applied and surface meshes will be extended to tetrahedral meshes in order to improve the realism of deformation.

ACKNOWLEDGEMENTS

This work was supported by the Korean Ministry of Science and Technology under the National Research Laboratory (NRL) Program, and by the Information Technology Research Center (ITRC) Program of the Korean Ministry of Information and Communication.

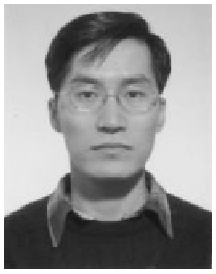
References

1. Thingvold JA, Cohen E. Physical modeling with B-spline surfaces for interactive design and animations. In *Symposium on Interactive Computer Graphyics* 1990; 129–137.
2. Hutchinson D, Preston M, Hewitt T. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* 1996; 31–45.
3. O'Brien J, Hodgins J. Graphical model and animation of brittle fracture. In *Proceedings of SIGGRAPH* 1999; 137–146.
4. Meseure P, Chaillou C. Deformable body simulation with adaptive subdivision and cuttings. In *Proceedings of International Conference in Centeral Europe on Computer Graphics, Visualization and Computer Vision* (WSCG 97), 1997; 361–370.
5. Wu X, Downes MS, Goktekin T, Tendick F. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Eurographics* Vol. 20(3), 2001.
6. Debunne G, Desbrun M, Barr A, Cani M-P. Interactive multiresolution animation of deformable models. In *10th Eurographics Workshop on Computer Animation and Simulation*, Milan, 1999.
7. Debunne G, Desbrun M, Cani M-P, Barr A. Adaptive simulation of soft bodies in real-time. In *Proceedings of Computer Animation 2000*, Philadelphia, PA, May 2000; 15–20.
8. Debunne G, Desbrun M, Cani M-P, Barr AH. Dynamic real-time deformations using space and time adapative sampling. *ACM SIGGRAPH 2001*, Los Angeles, 12–17 August 2001.
9. Kawai H. Elastic Object manipulation using coarse-to-fine representation of mass-spring models. In *SIGGRAPH 2001 Conference Abstracts and Applications*, 2001.
10. Ganovelli F, Cignoni P, Montani C, Scopigno R. A multiresolution model for soft objects supporting interactive cuts and lacerations. In *Eurographics*, Vol. 19(3), 2000.
11. Grinspun E, Krysl P, Schroeder P. CHARMS: a simple framework for adaptive simulation. In *ACM SIGGRAPH 2002*, Vol. 21(3).
12. Volkov V, Ling L. Adaptive local refinement and simplification of cloth meshes. In *Proceedings of 1st International Conference on Information Technology and Applications (ICITA 2002)*, Bathurst, Australia, 25–28 November 2002.
13. Villard J, Borouchaki H. Adaptive meshing for cloth animation. In *Proceedings of 11th International Meshing Roundtable*, Ithaca, NY, 15–18 September 2002.
14. Cignoni P, Ganovelli F, Scopigno R. Introducing multiresolution representation in deformable object modeling. In *SCCG '99 Conference Proceedings*, Buderice, Slovakia, 28 April–1 May 1999; 149–158.
15. Ganovelli F, Cignoni P, Montani C, Scopigno R. Enabling cuts on multiresolution representation. *The Visual Computer* 2001; **17**: 274–286.
16. Cartwright JHE, Piro O. The dynamics of Runge–Kutta methods. *International Journal of Bifurcations and Chaos* 1992; **2**: 427–449.

17. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerial Recipies in C++: The Art of Scientific Computing* (2nd edn). Cambridge University Press: Cambridge, UK, 2002; 715–727.
18. Dyn N, Levin D, Gregory J. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics* 1990; **9**(2): 160–169.
19. Zorin D, Schroeder P, Sweldens W. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of SIGGRAPH 1996*, 189–192.
20. Van Gelder A. Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools* 1998; **3**(2): 21–41.
21. Bro-Nielsen M. Finite element modeling in surgery simulation. *Proceedings of the IEEE: Special Issue on Surgery Simulation* 1998; **86**: 490–503.
22. Cotin S, Delingette H, Ayache N. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer* 2000; **16**: 437–452.
23. Arvo J. *Graphics GEMS II*. Academic Press: New York, 1991; 191–201.

Authors' biographies:

Yoo-Joo Choi has been a PhD student in the Department of Computer Science and Engineering at Ewha Women's University of Korea since 1999. She received her BS and MS degrees from Ewha Women's University in 1989 and 1991, respectively. She was an assistant researcher at the R&D department of KCI Co. and POSDATA Co. of Korea between 1991 and 1999. Her current research interests include medical image processing, physically based modeling, visual simulation and virtual reality applications.



Min Hong is a PhD student in Bioinformatics at the University of Colorado Health Sciences Center. He received his MS degree in Computer Science from the University of Colorado at Boulder in 2001. His current research interests are physically based modeling and simulation for medical and bioinformatics applications.

Min-Hyung Choi received his MS and PhD from the University of Iowa in 1996 and 1999 respectively. He joined the Computer Science and Engineering Department at the University of Colorado at Denver in 1999. His research interests are in Computer Graphics, Scientific Visualization and Human Computer Interaction with an emphasis on physically based modeling and simulation for medical and bioinformatics applications. Currently he is the Director of Computer Graphics and Virtual Environments Laboratory and an Associate of the Center for Computational Biology. He has been the acting chair of ACM SIGGRAPH Boulder/Denver Chapter since 2001.

Myoung-Hee Kim is a professor in the Department of Computer Science and Engineering at Ewha Women's University, Korea. She received her MS in Computer Science from Seoul National University of Korea in 1979 and her PhD from Universitaet Goettingen of Germany in 1986. Since 1987 she has been working at Ewha Women's University. She is now both director of the Center for Computer Graphics and Virtual Reality designated by the Korean Ministry of Information and Communication under the Information Technology Research Center Program and the Medical Image Visual Computing Laboratory designated by the Korean Ministry of Science and Technology under the National Research laboratory Program at Ewha University. Her current research interests include medical image processing, simulation, computer graphics and virtual reality.