Physically-based Animation of Volumetric Objects

Yan Chen, Qing-hong Zhu and Arie Kaufman

Center for Visual Computing and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

This paper presents a voxel-based animation technique which employs either a mass-spring model or a finite element model. The voxel-based mesh is generated in a pre-processing step. Two volumetric objects, a voxelized chair and scanned muscle are used as case studies with different models. The mass-spring model is used to show an animation sequence of a falling and bouncing chair. Wireframe display and volume rendering are used to display a real-time animation of the process. In addition, a real-time simulation is carried out by Finite Element Method (FEM) of a voxel-based multi-resolution muscle mesh. Two techniques, a direct integration and a simplified modal analysis method are discussed in the context of applying FEM for muscle deformation. Local deformation optimization with modal analysis for higher resolution muscle volumetric animation, which allows accurate prediction of muscle deformation changes, has been applied. The physiological muscle force has been considered and a biomechanically-based 3D FEM muscle model has been implemented. Realistic animations have been produced based on the FEM simulation with various graphics techniques.

Keywords: voxel-based, volumetric animation, mass-spring model, finite element method, biomedically-based muscle modeling, multiresolution, volume graphics.

1 Introduction

Physically-based animation has been one of the primary areas of research in computer graphics since the early 1980's [1,2,3,4,9,10,12,13,14,15]. These sited methods have the following limitations:

• Most existing models are surface-based [9,10,12,13,15], where physical principles are applied to surface primitives or spline control points. It is simple and fast, but it discards internal structures, cannot be used to represent objects with poorly defined edges or surfaces, and is sensitive to segmetation errors [5].

• Some use mass-spring models [3,14,15], which are simple and give good, interactive first impressions. Most of the previous work on mass-spring models are limited to 2D modeling [14,15] or 3D rigid object modeling [3]. For example, Christensen et al. [3] have applied mass-spring model for automatic motion synthesis, where no "flexion springs" have been considered for bending.

• Others use FEM [1,2,9,10]. It is easier to control and more accurate than the mass-spring model. There are two common ways to solve dynamic FEM problems. The most common is to use a direct integration algorithm such as center difference, Newmark algorithm and Wilson algorithm, but they are computationally expensive. They can only produce the surface animation [9,10] or a low-resolution volume animation [1,2]. Another common way of solving a dynamic FEM system is by using modal analysis.

In our approach, we adopted the voxel-based object modeling and rendering to solve the first limitation. Gibson el al. [4] have proposed the use of a voxel-based data representation not only for visualization, but also for modeling objects and structures derived from volumetric data. More recently, Gibson [5] has suggested the 3D ChainMail algorithm to propagate deformation through a volume rapidly. She also listed advantages of the volumetric object representation in surgical simulation as: same data organization as the acquired data, avoiding errors introduced by fitting surfaces or geometric primitives to the scanned images, and incorporating detailed information about the internal anatomical or physiological structure of organs and tissues. However, no physical models are considered in her algorithm.

To solve the second limitation, we have extended the mass-spring models to 3D elastic object modeling (see Section 3). We have experimented with a falling and bouncing voxelized chair (see Section 4).

FEM is a well established engineering method. Terzopoulos et al. [17] have used the finite-differences method and FEM in modeling physical deformations based on elasticity theory. Chen et al. [2] have simulated a 3D biomechanical FEM. In this model, skeleton kinematics and physiological effect have been considered, but only twenty node brick FEM mesh has been developed. In both of these systems, the volumetric case has not been implemented. In our volumetric system, hierachical modeling, modal analysis, and local deformation optimization are applied to speed up the process. By modal transformation, we can form a generalized eigenvalue problem. After we obtain the eigenvalues and eigenvectors of the dynamic system, the decoupled differential equations can be solved completely and separately running much faster than the direct integration method. Although the computation of eigenvalues and eigenvectors may be time consuming, this step can be done as a pre-computing process. It is advantageous for computer graphics and animation to compute only a linear function, and thus, making realistic animation feasible for high-resolution dataset (see Section 5). We have applied the FEM approach on a case study of scanned anconeus muscle deformation (see Section 6). We have implemented various volume graphics techniques for real-time realistic rendering of the data using 3D wireframe, polygon surface rendering and volume rendering.

2 Voxel-based Mesh Generation

Volume data are 3D entities that may have information inside them, might not consist of tangible surfaces and edges, or might be too voluminous to be represented geometrically. Volume data can be obtained by sampling, simulation, or modeling techniques [7]. In the case study of the anconeus muscle (Section 6), the sampled data from the Visible Human Dataset (VHD) [11] are a sequence of 2D slices of MRI. We preprocess the data, and then set up the voxel-based mesh for further simulation. We discuss it as an example below. In computational fluid dynamics and finite element analysis, the results of a simulation, typically running on a supercomputer, are often visualized as volume data for analysis and verification. The dataset for the case study of the falling chair is the result of a voxelization of its geometric model. The geometric objects are converted from their continuous geometric representation into a set of voxels that "best" approximate the continuous object [7, Chapter 5].

In this section we focus on the voxel mesh generation of the anconeus muscle as an example. First, we process the muscle image through manual or semi-automatic segmentation. The algorithm uses a multi-scale filtering (wavelet) and fuzzy-based image segmentation approach. In this method, the wavelet filtering technique is utilized to automatically find the optimal segmentation threshold, and the classification of the image is performed by means of multi-threshold and fuzzy clustering techniques. The muscle reconstructed voxel mesh is based on the segmented images.

Consequently, a hierarchical voxel mesh is reconstructed and serves as a multiresolution volumetric approximation of the original muscle data. For each level of detail, we first read the muscle raw file generated from the manual segmentation, and then after filtering, construct the voxel mesh based on the threshold value of the muscle image. The basic operation consists of constructing the voxel mesh inside of the 3D muscle data and discarding other information outside of it. In order to carry out the FEM simulation, we need to build the voxel mesh according to the data structure determined by the FEM simulator which contains 3D geometric coordinates and topological information between each voxel.

In order to obtain a high-resolution image, each voxel is subdivided into micro-voxels using mip-mapping, resulting in multiresolution voxel meshes. Based on the different resolution requirements of the application, we construct different levels of detail of the muscle meshes. Because it takes a relatively long time to simulate high-resolution muscle deformation using 3D FEM, it is advantageous to have muscle shape change in low or medium-resolution and apply FEM simulation on it. With the muscle shape deformation at low or medium-resolution, it is then relatively easy to predict a high-resolution deformation by comparison, mapping and interpolation. In our system, similarity theory, which is widely used in engineering design, is used to compare the deformation pictures in low, medium and high-resolution images. Pieper et al. [12] have used another approach by carrying out FEM simulation with a smaller number of nodes and elements, and then subdivided each element into micropolygons to create a high-resolution image.

The voxel mesh itself is also used in the rendering, so we can integrate real-time simulation and rendering based on a uniform mesh structure.

3 Volumetric Mass-Spring Model

4.1 Mass-Spring Lattice

Our elastic model is a mesh of $l \times m \times n$ virtual masses. Our approach is a 3D extension of the discrete mass-spring meshes of Provot [15]. Each mass is linked to its neighbors by massless springs of rest length greater than zero. The linkage inbetween neighbors is achieved in three different ways (Fig. 1):

• Springs linking masses $[i,\ j,\ k]$ and $[i{+}1,\ j,\ k]$, masses $[i,\ j,\ k]$ and $[i,\ j{+}1,\ k]$, and masses $[i,\ j,\ k]$ and $[i,\ j,\ k{+}1]$, are referred to as "structural springs",

• Springs linking masses $[i,\ j,\ k]$ and $[i{+}1,\ j{+}1,\ k{+}1]$, masses $[i{+}1,\ j,\ k]$ and $[i,\ j{+}1,\ k{+}1]$, masses $[i,\ j{+}1,\ k]$ and $[i{+}1,\ j,\ k{+}1]$, and masses $[i,\ j,\ k{+}1]$ and $[i{+}1,\ j{+}1,\ k]$ are referred to as "shear springs",

• Springs linking masses $[i,\ j,\ k]$ and $[i{+}2,\ j,\ k]$, masses $[i,\ j,\ k]$ and $[i,\ j{+}2,\ k]$, and masses $[i,\ j,\ k]$ and $[i,\ j,\ k{+}2]$ are referred to as "flexion springs".

Indeed, under pure shear stresses, only the "shear springs" are constrained; under pure compression or traction stresses (i.e., stretching), only the "structural springs" are constrained; whereas, under pure flexion stresses (i.e., bending), only the "flexion springs" are constrained.

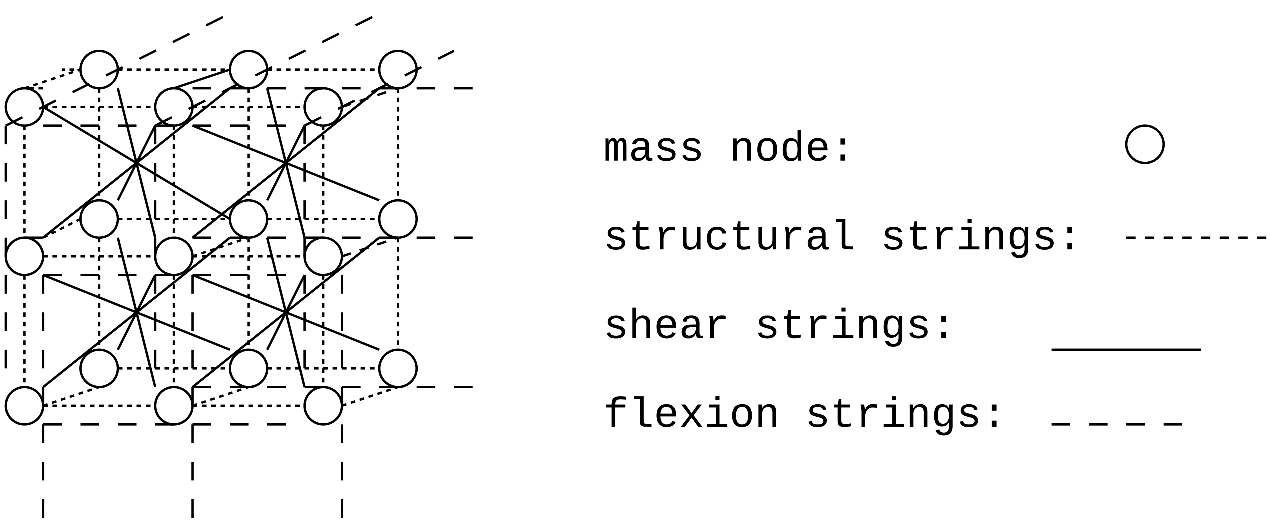

Figure 1: *A Volumetric Mass-Spring Lattice*

4.2 Voxel-based Forces

For the voxel-based deformable object, each node is connected to its nearest neighbors in three different directions $X,\ Y,\ Z$. As forces are applied to the object, those connections are maintained while the object deforms. The connections act as linear springs which draw the voxel mesh towards an equilibrium state. In the dynamic system, we consider volumetric parameters for each node: displacement $\vec{x}_{[i,j,k]}$, velocity $\vec{v}_{[i,j,k]}$, acceleration $\vec{a}_{[i,j,k]}$ are all 3D vectors:

- Every voxel $n = [i, j, k]$ is composed of eight nodes with neighbors of $\vec{p}_{[i,j,k]}$.

- For each voxel n, the gravity:

$$\vec{f}_g = \vec{g}m_{[i,j,k]} \tag{1}$$

is added, where $\vec{g}$ is a gravitational constant and $m_{[i,j,k]}$ is the mass of the voxel.

• The energy functions are conservative and hence result in perfectly elastic collisions. By introducing a damping force, we have,

$$\vec{f}_{d[i,j,k]} = -b\vec{v}_{[i,j,k]} \tag{2}$$

where b is a damping coefficiency for the deformable object.

• Within each voxel, the total internal forces for each node, which include the interaction force on node $[i,\ j,\ k]$ due to the system of voxels, is the sum of the individual tension of the node with each neighboring node $[i{+}1,\ j,\ k]$, $[i,\ j{+}1,\ k]$, $[i,\ j,\ k{+}1]$, etc. In our volumetric mass-spring lattice, four diagonal springs and six flexion strings have been added for each voxel for the stability of the whole system. Thus, there are sixteen springs connected to each node. The interaction force $f_{t[i,j,k]}$ for each node among these springs can be described as:

$$\vec{f}_{t[i,j,k]} = \sum_{j,k=1}^{16} \vec{f}_{\psi}(\vec{r}_{ijk}) \tag{3}$$

The total force imposed on each voxel $[i,\ j,\ k]$ is:

$$\vec{f}_{[i,j,k]} = \vec{f}_{g[i,j,k]} + \vec{f}_{d[i,j,k]} + \vec{f}_{t[i,j,k]} \tag{4}$$

4.3 Voxel-based Dynamics

The movement of each node $[i,\ j,\ k]$ in time is governed by Newton's law of motion:

$$\vec{a}_{[i,j,k]} = \frac{\vec{f}_{[i,j,k]}}{m_{[i,j,k]}} \tag{5}$$

$$\vec{a}_{[i,j,k]} = \frac{d\vec{v}_{[i,j,k]}}{dt} \tag{6}$$

$$\vec{v}_{[i,j,k]} = \frac{d\vec{x}_{[i,j,k]}}{dt} \tag{7}$$

where $\vec{f}_{[i,j,k]}$ is the sum of all the forces on node $[i,\ j,\ k]$, $m_{[i,j,k]}$ is the mass, $\vec{a}_{[i,j,k]}$ is the acceleration, $\vec{v}_{[i,j,k]}$ is the velocity, and $\vec{x}_{[i,j,k]}$ is the position of each node $[i,\ j,\ k]$. Given the initial position $\vec{x}_{[i,j,k]}(t_0)$ and the initial velocity $\vec{v}_{[i,j,k]}(t_0)$ of voxel n, for n = 1,2,...,N, voxel-based dynamics can be simulated with the given initial condition and volumetric constraints.

4.4. Algorithms

We use two differential schemes to compute the force on each node position $\vec{p}_{[i,j,k]}$. The first is the explicit Euler method and the second is a modified midpoint method. In the explicit Euler method, we compute the force $\vec{f}_{[i,j,k]}(t)$ on point $\vec{p}_{[i,j,k]}$ at any time by solving the dynamic equation of dynamics. The explicit differential scheme is employed here:

$$\vec{v}_{[i,j,k]}(t+\Delta t) = \vec{v}_{[i,j,k]}(t) + \Delta t \vec{a}_{[i,j,k]}(t+\Delta t) \tag{8}$$

$$\vec{x}_{[i,j,k]}(t+\Delta t) = \vec{x}_{[i,j,k]}(t) + \Delta t \vec{v}_{[i,j,k]}(t+\Delta t) \tag{9}$$

where Δt is a chosen time step.

The modified midpoint method advances the current velocity and position over the time step. The time interval δt is divided into n equal substeps of size $h = \delta t/n$. The explicit formulation for the system is given by:

$$z_{q+1} = z_{q-1} + 2h\vec{a}_{[i,j,k]}(t+qh) \tag{10}$$

for q=1,2,...n-1

$$\vec{v}_{[i,j,k]}(t+\Delta t) = \frac{1}{2}[z_n + z_{n-1} + h\vec{a}_{[i,j,k]}(t+\Delta t)] \tag{11}$$

$$y_{q+1} = y_{q-1} + 2h\vec{v}_{[i,j,k]}(t+qh) \tag{12}$$

for q=1,2,...n-1

$$\vec{x}_{[i,j,k]}(t+\Delta t) = \frac{1}{2}[y_n + y_{n-1} + h\vec{v}_{[i,j,k]}(t+\Delta t)] \tag{13}$$

for computing new velocity and displacement.

4 Case Study of Falling Chair

This case study simulates a voxelized chair falling down and bouncing on the ground. It runs on an an SGI Power Challenge GR, R10000, 195MHz, 3GB. We use both a wireframe model for showing the mass-spring lattice and volume rendering for realistic animation. Fig. 2 displays a chair of 888 nodes and 6326 strings. The whole process took 93 seconds for 936 frames. The chair dataset of Fig. 3 is a volume of 240 cells and 504 nodes. Here we used the accelerated ray-casting algorithm for curvilinear volumes developed at Stony Brook [6]. A solid was defined in curvilinear coordinates and the physical space based ray-casting method was applied to render this solid. The color indicates the stress density of each voxel. To do this, the maximum value of denisty in each frame was found and the density was set between *bias* and 255, based on its ratio to the maximum density. *Bias* is a constant that can be set to determine minimum density values, which in our experiment are the stress. The color set for *bias* is green and for 255 is red. For other density values, the color value is linearly interpolated between green and red. There are 1826 frames with a running time of 178 seconds.

5 Dynamic FEM Simulation for Muscle Animation

5.1 Direct Integration for FEM Dynamic System

Using Lagrangian dynamics and elastic theory, the deformable model equations of motion can be expressed in 3D vector form by the second-order ordinary differential equations:

$$M\frac{\partial^2 \vec{x}}{\partial t^2} + C\frac{\partial \vec{x}}{\partial t} + K(\vec{x})\vec{x} = \vec{f}(\vec{x}, t) \tag{14}$$

where $\vec{x}$ is a $3n$ vector of nodal displacement, M, C and K are $3n\times 3n$ matrices describing the mass, damping and stiffness between nodes within the volumetric object respectively, and f is a $3n$ vector of forces applied to each node.

The solution of the displacement, velocity and acceleration at time $t + \Delta t$ can be obtained by using center and Euler finite differences written by Lagrangian equation for the full system as:

$$K^* = K(x_t) + \frac{M}{\Delta t^2} + \frac{C}{2\Delta t} \tag{15}$$

$$u_t = (x_t - x_{t-1})/\Delta t. \tag{16}$$

$$f^*_{t+\Delta t} = (\frac{1}{2\Delta t}C + \frac{M}{\Delta t^2})x_t + (\frac{M}{\Delta t} - \frac{C}{2\Delta t})u_t + f_{t+\Delta t}. \tag{17}$$

Substituting Eqs. 15, 16 and 17 in Eq. 14, we can obtain,

$$K^*(x_{t+\Delta t}) = f^*_{t+\Delta t} \tag{18}$$

Consequently, the explicit procedure evolves the dynamic solution from the given initial conditions x_0 and u_0 by solving a time sequence of static equilibrium problems for the instantaneous configurations x_{t+1}. Thus, the original nonlinear partial differential equations have been reduced to a sequence of sparse linear algebraic equations.

In order to solve the sparse linear system equations quickly, we assume $K^*_t = K$ is time-variant. Thus the matrix decomposition solver needs only to perform a single initial decomposition of K which significantly reduces the total amount of computation required. Furthermore, the mass matrix and damping matrix are generated at each node by using diagonal damping and mass matrices,

$$M_{ii} = \frac{1}{3}\rho V, C_{ii} = \beta M_{ii} \tag{19}$$

where ρ is the mass-density, β is a scaling factor, and V is the volume.

The dynamic algorithm is as follows:

(1) Initial calculations:

- Form the 3D brick stiffness matrix K, mass matrix M, and damping matrix C.
- Calculate the initial displacement, velocity and acceleration: $x^0, \dot{x}^0, \ddot{x}^0$.

- Store the matrix value in memory.
- Calculate the static deformation based on the frontier solver.

(2). Precomputing:

- Select time step size Δt and calculate an integration constant.
- Derive the dynamic difference scheme based on the static deformation and its matrix.
- Store the matrix in a one dimension form using back substitution.

(3). Time loop:

- Calculate a new sparse matrix based on the stored matrix.
- Solve for displacements at time $t + \Delta t$ using the explicit differential scheme based on the stored matrix values (K, M, C).
- Calculate the new velocity and acceleration at time $t + \Delta t$.

5.2. A Simplified Modal Analysis Method for Volume Animation

The dynamic elastic FEM can be solved in two ways: one is by performing direct integration as we have discussed in the previous section. The second is using modal analysis. By mathematical transformation, we can change the base of the matrices so that the equations of motion can be uncoupled.

Time history modal superposition is probably the most commonly used method to calculate dynamic response of systems of finite extent that are subjected to loads with a known time variation for a large number of nodes. It is advantageous to use this method to show realistic animation for high-resolution datasets. The basic algorithm for modal analysis is described as follows:

(1). Formulation of the equations of motion.
(2). Provision of the modal matrix and natural circular frequencies.
(3). Uncouping of the equations of motion.
(4). Combination of the modal responses.

For a 3-degree-of-freedom system, this can be written as:

$$x_1(t) = \phi_{11} z_1(t) + \phi_{12} z_2(t) + \phi_{13} z_3(t) \tag{20}$$

$$x_2(t) = \phi_{21} z_1(t) + \phi_{22} z_2(t) \phi_{23} z_3(t) \tag{21}$$

$$x_3(t) = \phi_{31} z_1(t) + \phi_{32} z_2(t) + \phi_{33} z_3(t) \tag{22}$$

where

$$z_1(t) = \frac{c_0}{\omega_1^2}\left[1 - e^{-\beta\omega_1 t} f_1(sin(\beta, \omega t))\right] \tag{23}$$

$$z_2(t) = -\frac{c_0}{\omega_2^2}\left[1 - e^{-\beta\omega_2 t} f_2(sin(\beta, \omega t))\right] \tag{24}$$

$$z_3(t) = -\frac{c_0}{\omega_3^2}\left[1 - e^{-\beta\omega_3 t} f_3(sin(\beta, \omega t))\right] \tag{25}$$

β is the dumping coefficient, ω_1, ω_2 and ω_3 are eigenvalues, and $\phi_{11}, \phi_{12}, \phi_{13}, \phi_{21}, \phi_{22}, \phi_{23}, \phi_{31}, \phi_{32}, \phi_{33}$ are eigenvector components. These coefficients can be obtained by solving an eigenvalue problem. For a large number of nodes $3n$ of volumetric case, computational time for each node displacement x_i is very small, and therefore it is possible for us to obtain a realistic animation for high-resolution datasets. In our experiment, we used a simplified model and assumed that the eigenvalues have been obtained while the displacement is the function $f_2(e^{g(t)})$. We applied this method to local deformation for muscle animation while the direct integration method was used for muscle global deformation and animation.

6 Case Study of Anconeus Muscle

The anconeus muscle from VHD [11] is used to demonstrate our techniques [18]. The dataset consists of 71 slices, with the resolution of 140x220x71. The simulation and rendering are carried out on an SGI Power Onyx/RE2, R10000, 195MHz, 640MB. Our FEM simulation incoprates Stern's [16] biomechanical model for precise prediction of the muscle shape change. The algorithms, written with C/OpenGL, are integrated with shaded polygon graphics so that real-time display of shaded polygons is realized in our animation.

Stern [16] devised a mathematical model incorporating mechanical and physiological parameters of an idealized bone-muscle system, where the movement of a limb is under the action of a muscle. The model considers two aspects of muscle physiology: the effect of shortening (or lengthening) on tension development and of length on tension. Given some assumptions, the force produced by a muscle at maximum activation obeys the following expression:

$$P = \left\{ \frac{(P_0)_l/a + 1}{\frac{1}{K_i}\frac{1}{(B/C+2+C/B)}\frac{sin(\alpha_i \dot{\alpha}_i)}{j.Y.K_l} + 1} - 1 \right\} a \tag{26}$$

where
K_i = the straight line distance from one attachment point of the muscle on the bone to the other divided by the sum of the distances from the joint to each attachment point.
K_l = the value of K_i when the joint is positioned so that the length of the contractile tissue is L_0.
j = the value of b expressed in muscle lengths per sec.
α_i = the angle between the bones.
$\dot{\alpha}_i$ = the angular velocity of movement.
Y = the fraction of the distance A_i devoted to contractile tissue when the joint is positioned so that the contractile tissue is at L_0.

Fig. 4 shows five different muscle images according to different applied forces. The bottom of the muscle (blue points) is fixed, while forces are applied at the top (yellow points) in three directions. The FEM simulation is completely volumetric so that we can see changes of the muscle shape in the various cases. It is very important to both know and view the muscle geometric change imposed by the external force when we carry out, for example, surgery simulation and cutting. The images of Fig. 4 were generated using OpenGL shaded polygons with 1,840 voxels.

Fig. 5 presents several snapshots of our muscle animation rendered with OpenGL shaded polygons. The dynamic model shows the dumping process of the muscle when a sudden external force is applied at the top of the muscle (yellow points) while the bottom (blue points) is fixed at the lowest points. The muscle is oscillating with decreasing frequency.

The FEM simulation is carried out with 450 nodes and 219 voxels. The whole process took 0.76 seconds for 22 frames.

Local deformations of the muscle animation using the modal analysis method are shown in the three different resolutions (Fig 6) which contain 450, 2665 and 17723 voxels respectively (refer to Table 1). Compared with global deformation by a direct integration method, local deformation concerns only part of the muscle. Thus, we apply modal analysis method for only part of the muscle mesh and other parts remain unchanged. In a practical medical application, such as cutting, it is vital to know the local changes of the skin and the muscle. With application of modal analysis to local deformation, we have a better understanding of muscle animation dynamics.

Table 1: *Hierarchical voxel meshes for both 3D finite element simulation input and volume rendering*

Voxel Size	No of Voxels	No of Nodes	Fixed Points	Loaded Points
16 x 16 x 16	28	91	4	4
8 x 8 x 8	219	450	4	4
4 x 4 x 4	1840	2664	9	4
2 x 2 x 2	14578	17723	11	9
1 x 1 x 1	116630	128924	31	22

7 Conclusion and Future Work

We have presented a physically-based voxel-based animation for both mass-spring models and FEM. Voxel-mesh generation, as the pre-processing step, was discussed with the anconeus muscle as an example. Voxel-based elastic mass-spring models were set up and the falling voxelized chair was used to demonstrate the performance of our approach. We also have proposed the voxel-based FEM. We used a hierarchical voxel mesh which is suitable for both FEM simulation and volume graphics. Additionally, we applied local deformation optimization with modal analysis for higher-resolution muscle volumetric animation. The anconeus muscle dataset from VHD was used as a case study. We implemented a biomechanically-based 3D FEM muscle model which allows accurate prediction of muscle deformation changes. The physiological muscle force was considered and we simulated linear elastic muscle by FEM and generated a realistic animation with various graphics techniques.

We are currently working on direct curvilinear rendering of the deformed muscle and paralellizing the simulation and rendering. We also plan to simulate the cutting and tearing of tissues, taking advantage of volumetric modeling and rendering.

8 Acknowledgements

We would like thank Shigeru Muraki, Suya You, Lichan Hong, Robert Pokorny, Akio Doi and Ikuko Takanashi for their earlier work on this project. Also, thanks to Jeffrey Ge and Jack Stern for their useful suggestions for our work. Kathleen McConnell advised us greatly in writing this paper. The project was partially supported by Mitsubishi Electric Research Lab, Office of Naval Research under grant N000149710402, and National Science Foundation under grant MIP-9527694.

References

1. Bro-Nielsen, M., and Cotin, S., "Real Time Volumetric Deformable Model for Surgery Simulation using FE and Condensation," *Proc. Eurographics '96*, pp.C57–C66.

2. Chen, D., and Zeltzer, D. "Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method," *Proc. SIGGRAPH'92*, pp. 89–98.

3. Christensen, J., Marks, J. and Ngo, J. T., "Automatic motion synthesis for 3D mass-spring models", *The Visual Computer*, 1997, No. 13, pp. 20–28.

4. Gibson, S.F.F., Samosky, J., Mor, A., Fyock, C., Grimson, E., Kanade, T., Kikinis, R., Lauer, H., McKenzie, N., Nakajima, S., Ohkami, H., Osborne, R., Sawada A., "Simulating Arthroscopic Knee Surgery using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback", MERL Tech Report TR96-19.

5. Gibson, S.F.F., "Beyond Volume Rendering: Visualization, Haptic Exploration, and Physical Modeling of Voxel-based Objects," *Eurographics Workshop on Visualization in Scientific Computing*, 1995, pp. 171–201.

6. Hong, L. and Kaufman, A., "Accelerated Ray-Casting for Curvilinear Volumes", SUNY Stony Brook Tech Report TR CVC 970315, 1997.

7. Kaufman, A. (ed), "*Volume Visualization*", IEEE CS Press, 1991.

8. Koch, R., Gross, M., Carls, F., Buren, D., Fankhauser, G. and Parish, Y. "Simulating Facial Surgery Using Finite Element Models", *Proc. SIGGRAPH '96*, pp. 421–428.

9. Lee, Y., Terzopoulos, D. and Waters, K., "Realistic Modeling for Facial Animation," *Proc. SIGGRAPH '95*, pp. 55-62.

10. Metaxas, D. and Terzopolus, D., "Dynamic Deformation of Solid Primitives with Constraints", *Proc. SIGGRAPH '92*, pp. 309–312.

11. National Library of Medicine, "The Visible Human Project", with URL: http://www.nlm.nih.gov/research/visible/visible_human.html, 1997.

12. Pieper, S., Rosen, J. and Zeltzer, D., "Interactive Graphics for plastic Surgery: A Task-level Analysis and Implementation," *1992 Symposium on Interactive 3D Graphics*, pp. 127–134.

13. Platt, J. C. and Barr, A. H., "Constraint Methods for Flexible Models", *Proc. SIGGRAPH '88*, pp. 279–288.

14. Promayon, E., Baconnier, P. and Puech, C., "Physically-Based Deformations Constrained in Displacements and Volume", *Proc. Eurographics '96*, pp. C155–C164.

15. Provot, X. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior", *Proc. Graphics Interface '95*, pp. 147–154.

16. Stern, J., "Computer Modeling of Gross Muscle Dynamics," *Journal of Biomechanics*, 1974, Vol. 7, pp. 411-428.

17. Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K., "Elastically Deformable Models," *Proc. SIGGRAPH '87*, pp. 205–214.

18. Zhu, Q., Chen, Y. and Kaufman, A., "Real-time Biomechanically-based Muscle Volume Deformation using FEM", SUNY Stony Brook Tech Report TR CVC 980109, 1998.

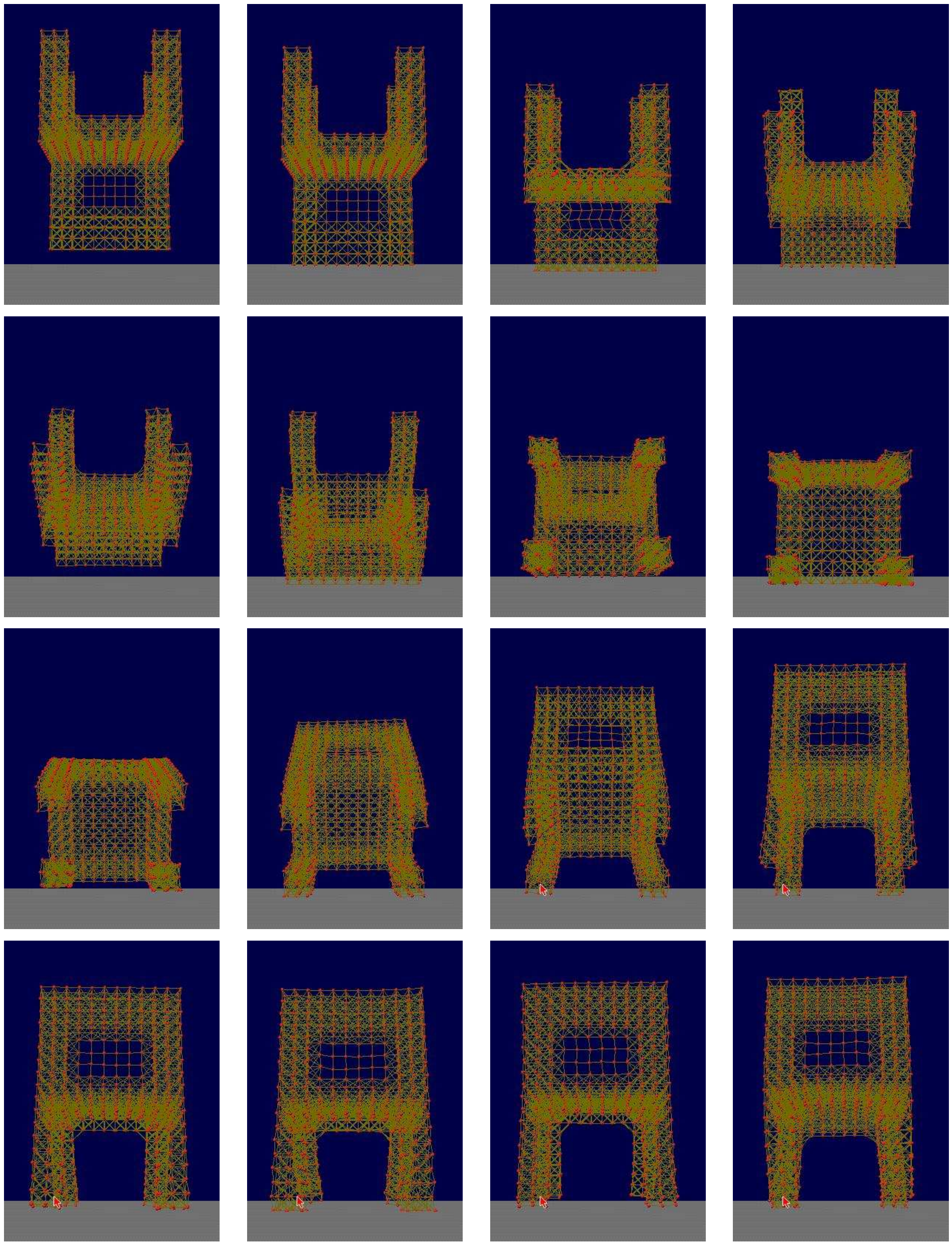

Figure 2: *Wireframe display for a falling voxelized chair.*

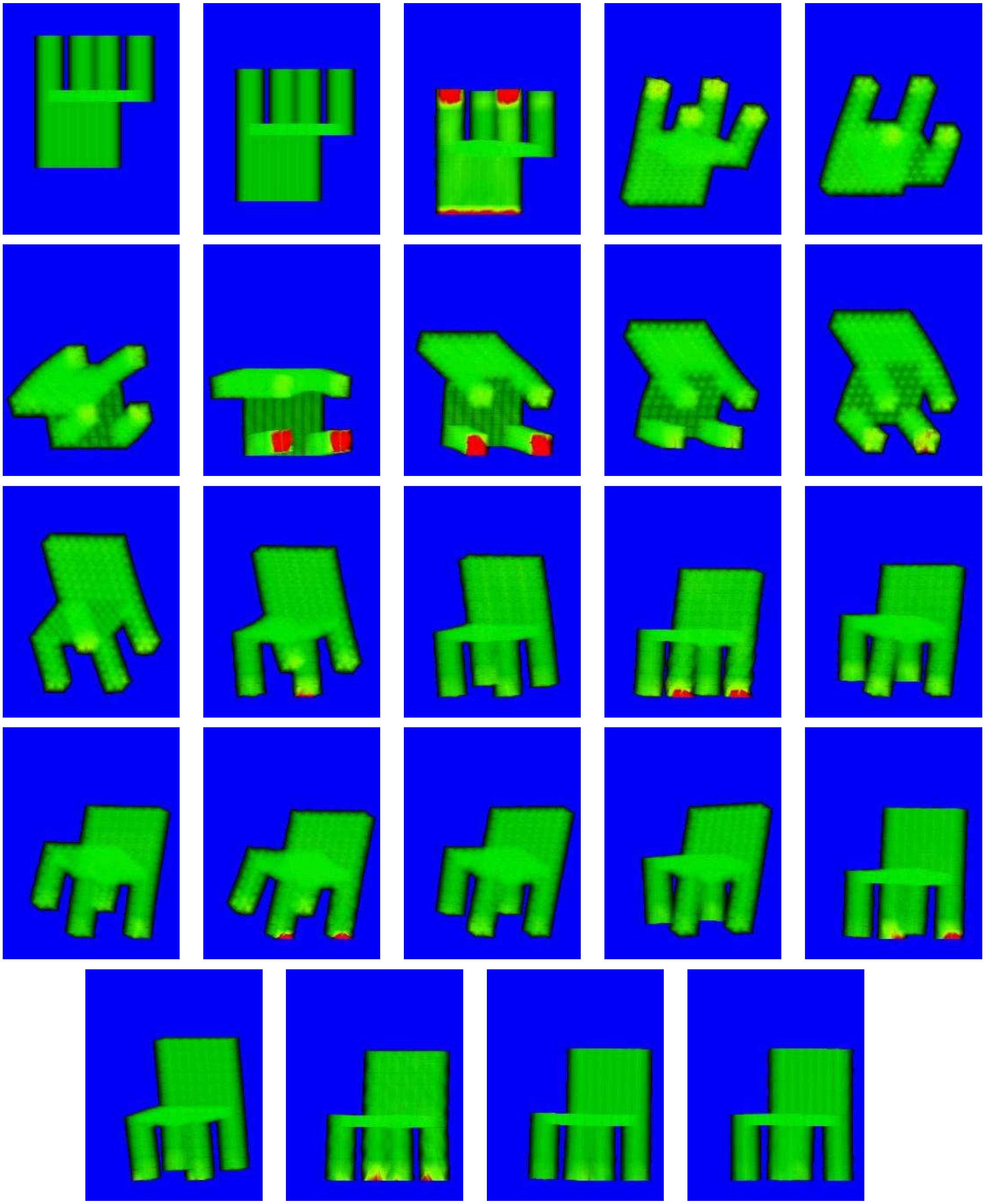

Figure 3: *Curvilinear ray-casting of a falling voxelized chair.*

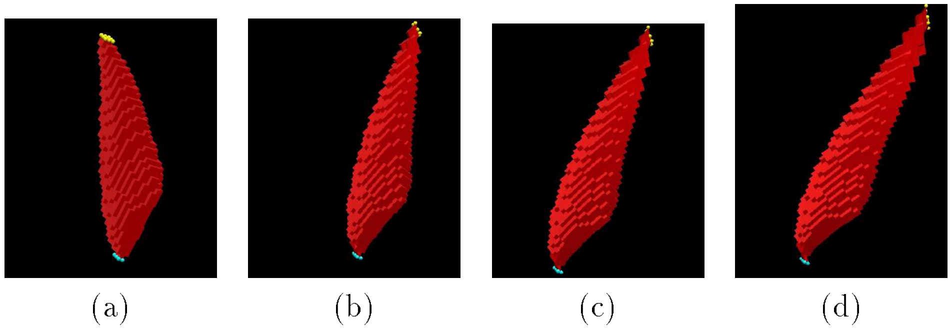

Figure 4: *Medium resolution of static muscle model with: (a) 0 Newtons; (b) 500 Newtons; (c) 750 Newtons; (d) 1000 Newtons of force applied.*

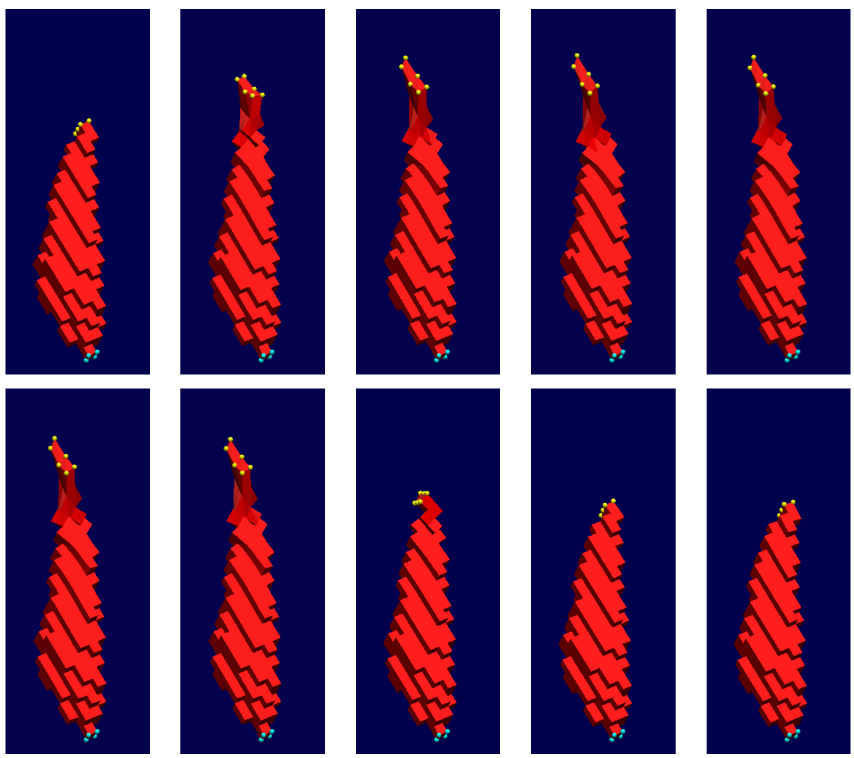

Figure 5: *Medium resolution of dynamic muscle model*

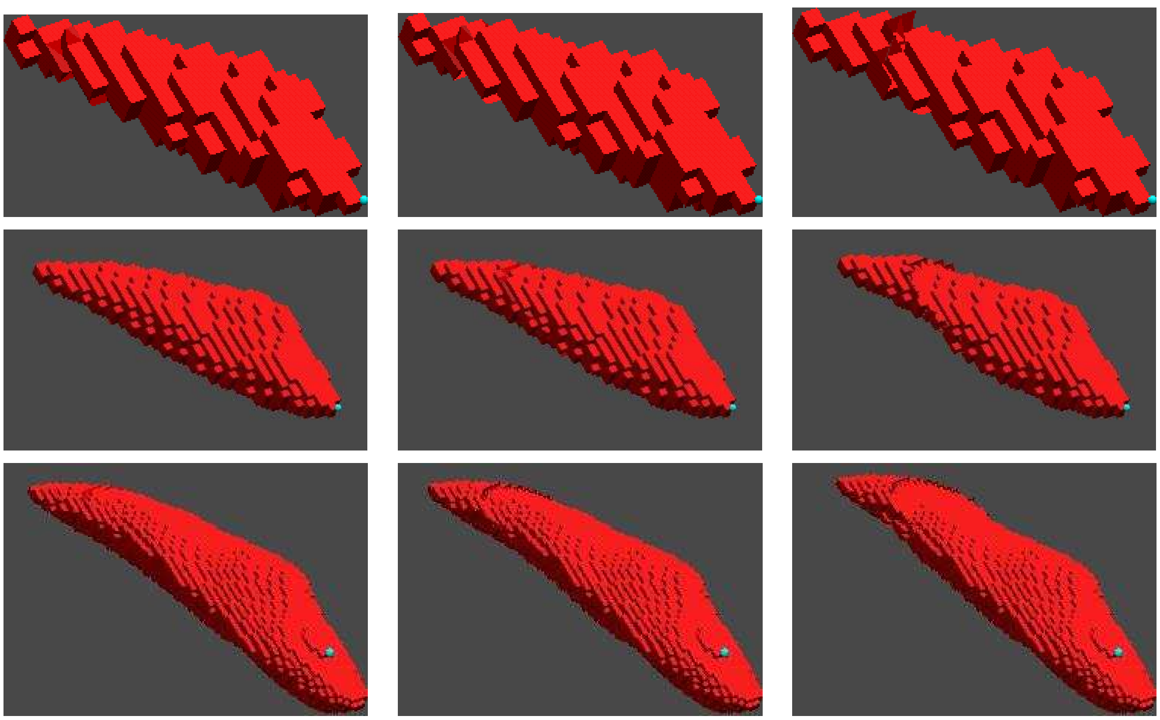

Figure 6: *Multiresolution animation of muscle local deformation*