3D ChainMail: a Fast Algorithm for Deforming Volumetric Objects

Sarah F. F. Gibson
MERL - A Mitsubishi Electric Research Lab
201 Broadway, Cambridge, MA, 02139
gibson@merl.com

ABSTRACT

An algorithm is presented that enables fast deformation of volumetric objects. Using this algorithm, rigid, deformable, elastic and plastic materials can be modeled by adjusting deformation limits for indifvidual elements. An interactive system that combines the deformation algorithm with collision detection and an energy minimizing elastic relaxation step is described. Using this system, objects containing up to 125,000 elements have been deformed interactively on an SGI Indy.

INTRODUCTION

Surgical simulation requires interactive modeling and visualization of complex, 3D anatomical structures. For example, surgery of the abdomen involves probing and cutting through organs and tissues that have complex shapes and material properties. Because modeling the deformation and cutting of tissue requires a representation of interior structure, volumetric object representations are well suited for surgical simulation. A volumetric representation can incorporate detailed information about internal anatomical or physiological structure. This detailed information can be used to model tissue deformation more accurately than models which represents the object surface and assumes a homogeneous interior. Because a volumetric representation uses the data produced by 3D medical scanners directly, errors that are introduced by fitting polygonal surfaces to the discrete image data can be avoided.

In a volumetric object representation, the object is stored as a discrete 3D array of sampled data elements. Each data element can consist of several bytes of information including visual properties, such as color or transparency, or material properties, such as tissue type or elasticity. The major disadvantage of volumetric representations is that objects can consist of millions of volume elements. This large data requirement poses challenges for memory storage and access, for real-time rendering, and for physically realistic modeling of object interactions. In this paper we present a fast algorithm for modeling the deformation of volumetric objects. The algorithm can model a range of materials including rigid, deformable, elastic, and plastic substances. In addition, the method can model anisotropic materials, such as muscle, which have different material properties along different axes.

BACKGROUND

The basic technologies that have influenced this work are: Volume Graphics; physics-based graphics; and soft-tissue modeling with Finite Element (FEM) and other methods.

Volume Graphics

Volume Graphics[13, 14] deals with the synthesis[11], modeling[28], manipulation, and rendering of volumetric objects. Prior work in Volume Graphics includes the development of techniques to replace the traditional graphics pipeline of polygon graphics with new methods for volumetric data. For example: shading[23, 24]; antialiasing[29]; and rendering algorithms[12]; are replaced by their volumetric counterparts. New algorithms and hardware implementations in volume rendering have begun to address the need for interactive rendering of regular volumes[15, 16, 21]. Recently, attention in Volume Graphics has been given to object manipulation, including haptic interaction with volumetric objects[1] and physically realistic modeling of object interactions[9].

Physics-based Graphics

There is a growing interest in physically realistic modeling of object interactions in the graphics community. This includes both detecting object collisions and modeling the energy and momentum transfer between colliding objects - problems that have been addressed for real-time interactions of rigid objects in systems with surface polygon object representations[2, 19] and to some extent with volumetric objects[9]. Much work is needed in the area of modeling interactions between highly deformable objects.

Soft-tissue Modeling

Finite Element Modeling (FEM) can be used to model complex materials. Careful selection of element nodes and accurate knowledge of the material properties at each node enables accurate simulation of complex mechanical behaviors. FEM has been applied to modeling the skin and muscle layers of the face[17, 26, 27], skeletal muscle[4], and the eye[10]. However, because of computational requirements, FEM cannot be used in interactive applications unless the number of node points is small. Techniques such as multigrid methods[25] and modal analysis[20, 7, 18] have been used to reduce the required computation for FEM applied to deformable object modeling. However, the computational complexity of FEM remains a bottleneck for interactive soft tissue modeling.

Other techniques that have been used to model soft tissue include: free-form deformation[8, 22]; active surfaces[5] or active cubes[3]; using a "zone of influence" to predefine the effect that displacement of a given node point will have on neighboring nodes[30]; and using implicit surfaces to model soft substances[6]. These techniques are useful because of their speed but they have limited accuracy for modeling complex tissues and object structures.

DEFORMATION SYSTEM

A system for interactive manipulation of deformable objects has been implemented. The system consists of five procedures: 1) an interactive control loop that monitors selection and control of the object; 2) the 3D ChainMail algorithm which stretches or contracts the object when a selected element is moved; 3) an elastic relaxation algorithm which adjusts relative element positions to minimize the system energy; 4) collision detection to check for collisions and prevent interpenetration of objects; and 5) rendering of the deformed object for visual feedback.

Data Structures

In this implementation, the object data structure consists of the object size, its type or classification, a pointer to the object elements, and the deformation and elasticity parameters (which are assumed to be constant throughout the object in this implementation). Volume element data structures consist of a element color (r,g,b), a position vector (x,y,z), and pointers to the six nearest neighbors: top, bottom, front, back, left, and right. An additional data structure keeps track of previous positions of moved elements. This structure is used for fast backtracking after a collision is detected or after reaching an object configuration that is not permitted.

Control Loop

The system consists of two phases: an initialization phase to read in and initialize data structures; and an X-Event control loop that continuously monitors and responds to user inputs. Button clicks and releases are used to select and deselect elements in the object. If an element is selected, mouse movements control displacements of the selected element. The largest possible step is taken towards the desired position if the desired position results in an object that is not permissible (due to collisions with other objects or the boundaries of the virtual space). The user interface monitors changes in the deformation parameters, changes in rendering engines, and termination of the session.

Object Deformation: 3D ChainMail

The large number of elements in a volumetric object poses a significant challenge for interactive applications that model physically realistic object deformation. One approach is to perform FEM calculations on a lower resolution grid[10]. However, this does not take advantage of the high resolution data produced by medical scanners. Here, we introduce an algorithm that uses the original data resolution but performs relatively simple deformation calculations for each element. When the volume is manipulated, the object stretches or contracts to satisfy maximum and minimum allowable distances between neighboring elements. The movement of each element depends only on the positions of its nearest neighbors, allowing fast propagation of the deformation through the volume. Because the motion constraints are similar to those of a set of linked elements in a chain, this algorithm has been dubbed 3D ChainMail.

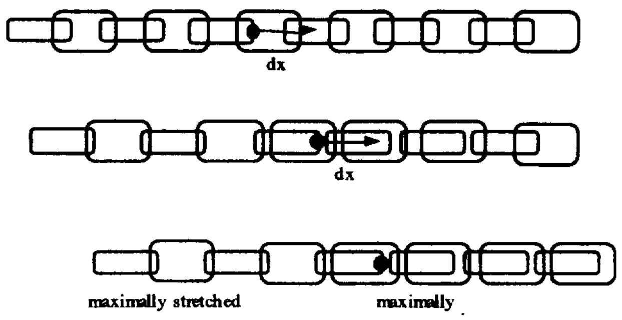

Figure 1. Deformation of a 1D chain when the selected link is moved to the right by dx.

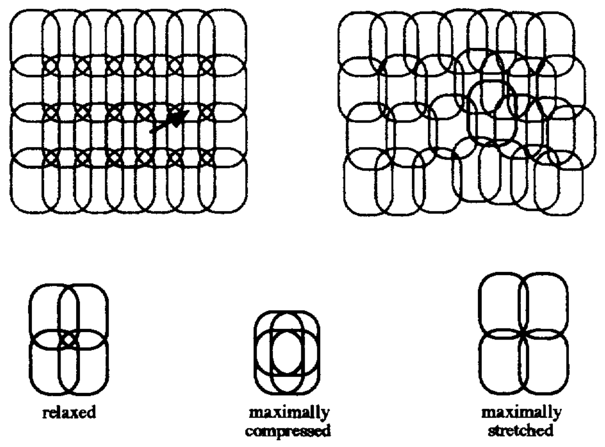

Figure 2. Deformation of 2D chain mail when the selected link is moved.

In the 3D ChainMail algorithm, volume elements are linked to their six nearest neighbors. When one node of the structure is pulled or pushed, neighboring links absorb the movement by taking up slack in the structure. If a link between two nodes is stretched or compressed to its limit, displacements are transferred to its neighboring links. In this way, small displacements of a selected point in a relatively slack system result in only local deformations of the system, while displacements in a system that is already stretched or compressed to its limit causes the whole system to move (see Figures 1 and 2). Much like the links in a chain, neighbors only respond to a given element's movement if constraints on the distances between elements are violated. Changing the constraints on link lengths allows us to model both rigid and deformable objects.

Although the discussion will focus on 2D objects, the extension to 3D is straightforward. Both 2D and 3D versions of this system have been implemented and examples from both implementations are reported in Section 4.

Two types of lists are maintained in this algorithm: a list consisting of the previous positions and pointers to elements that have been moved; and four lists of candidates for movement that are classified according to whether they are a top, left, bottom, or right neighbor of their sponsoring element. Each candidate element is processed in turn, starting from the selected element and then proceeding in order through the right, left, top, and bottom lists of candidate points. To process an element, the deformation constraints are checked against its sponsoring element. If the deformation limits are exceeded, the element is moved a minimum distance until the constraints are satisfied. When an element is moved -- either under direct control of the user or indirectly in response to a neighbor's movement -- the element becomes a sponsor to its unmoved neighbors and these neighbors are appended to their respective movement candidate lists.

The deformation limits are defined as follows: each element must lie within a horizontal range of *minDx* and *maxDx* from its left and right neighbors and within a vertical range of *minDy* and *maxDy* from its top and bottom neighbors. These limits control stretching and contraction of the material. In addition, each element must lie within +/- *maxHorizDy*, from its horizontal (left and right) neighbors and within +/- *maxVertDx*, from its vertical (top and bottom) neighbors. These limits control the maximum amount of shear that is possible in the material. The definition of these limits are illustrated in Fig. 3.

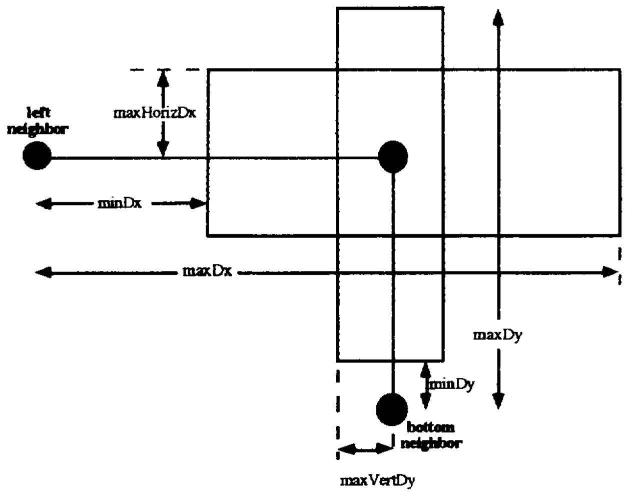

Figure 3. The regions in which the element can move relative to its left and bottom neighbors.

Algorithm Outline
The basic 3D ChainMail algorithm is as follows:
1) When the user-selected element is moved, the element and its old positions are added to the list of moved elements, its x, y positions are updated, and its four nearest neighbors (top, left, bottom, right) are added to the appropriate lists of candidates for movement.

2) The lists of candidate elements are processed in turn until all of the candidate lists are exhausted or the system is not permissible (in which case those elements that were moved are returned to their previous positions and a smaller step towards the desired position is attempted). The candidate lists are processed in the following order: right, left, top, bottom.

3) The right candidate list is processed in the following manner: beginning with the first element in the list, the stretch and shear constraints are checked between the list element and its sponsoring element (always its left neighbor). If the constraints are violated, the element is moved a minimum distance until the constraints are satisfied. The new position is calculated as follows:

if $(x - x_{left}) < minDx$, $x = x_{left} + minDx$;
else if $(x - x_{left}) > maxDx$, $x = x_{left} + maxDx$;

if $(y - y_{left}) < - maxHorixDy$, $y = y_{left} - maxHorixDy$;
else if $(y - y_{left}) > maxHorixDy$, $y = y_{left} + maxHorixDy$;

If the element is moved, its top, right and bottom neighbors are added to their respective candidate lists. (Since the element was sponsored by its left neighbor there is no need to add the left neighbor to the candidate list.) Each right candidate is processed in turn until no right candidates remain.

4) The left list is processed in a similar way except that left elements are sponsored by their right neighbors and movement of a left element causes its bottom, left, and top neighbors to be added to the candidate lists.

5) The top and bottom lists are also processed in a similar manner except that the top and bottom elements are sponsored by their bottom and top elements respectively and movement of a top (or bottom) element causes only a top (or bottom) element to be added to the correct candidate list.

This algorithm must be modified slightly for non-convex objects. In non-convex objects, if the right or left neighbor of a moved top (or bottom) element does not have a bottom (top) element, it should be added to the appropriate candidate list. Note that this may require that candidate lists be visited more than once to exhaust all the elements of all candidate lists.

The algorithm is especially fast for three reasons: 1) each element in the volume is considered at most once for each deformation, 2) each element is compared to only one neighbor (its sponsoring neighbor) to determine if and how it must be moved and 3) the deformation is propagated outwards from the selected point and the propagation is terminated as soon as possible. Of these, 1) and 3) result from the way elements are added to the candidate lists. The second point, 2) results from the following theorem.

Theorem
In the 3D ChainMail algorithm, each element can be compared to a single neighbor when the object has constant deformation limits throughout its volume.

Proof
The starting position of each element in the candidate lists already satisfies the constraints of neighbors that have not moved. Hence, the new position of a movement candidate depends only on neighbors that have been moved. (If a candidate is moved, then its unmoved neighbors are later considered for movement.)

For elements in the left (or right) candidate lists, only the sponsoring right (left) neighbor is moved prior to movement consideration. Hence, for left and right candidates only one neighbor must be considered.

For top (or bottom) neighbors, it is possible that both the sponsoring bottom (or top) neighbor and its left (or right) neighbor were moved prior to consideration. However, it is shown here that when an element from the top candidate list satisfies deformation constraints relative to its sponsoring bottom neighbor, then it automatically satisfies the constraints of its left neighbor as long as the left neighbor was previously placed to satisfy its own bottom neighbor.

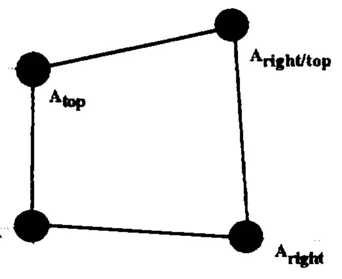

Figure 4. Grid configuration to show the relationship between points A, A_R, A_T, and $A_{R/T}$.

If the top and right neighbors of A satisfy the deformation constraints with respect to A (see Figure 4), then

$$A_R(x,y) \; \varepsilon \; ([x_0+minDx, x_0+maxDx], [y_0-maxHorizDy, y_0+maxHorizDy]) = ([x_{min_{AR}}, x_{max_{AR}}], [y_{min_{AR}}, y_{max_{AR}}])$$

and

$$A_T(x,y) \; \varepsilon \; ([x_0-maxVertDx, x0+maxVertDx], [y_0+minDy, y_0+maxDy]) = ([x_{min_{AT}}, x_{max_{AT}}], [y_{min_{AT}}, y_{max_{AT}}]).$$

The top neighbor of A_R, $A_{R/T}$, must satisfies the deformation constraints with respect to A_R. Hence,

$$\begin{aligned} A_{R/T}(x,y) \; \varepsilon \; & ([x_{min_{AR}}-maxVertDx, x_{max_{AR}}+maxVertDx], [y_{min_{AR}}+minDy, y_{max_{AR}}+maxDy]) \\ &= ([x_0+minDx-maxVertDx, x_0+maxDx+maxVertDx], [y_0-maxHorizDy+minDy, y_0+maxHorizDy+maxDy]) \\ &= ([x_{min_{AT}}+minDx, x_{max_{AT}}+maxDx], [y_{min_{AT}}-maxHorizDy, A_{top}(y)_{max}+maxHorizDy]). \end{aligned}$$

which also satisfies the deformation constraints with respect to the left neighbor, A_T, of $A_{R/T}$. Hence, when considering $A_{R/T}$, an element of a top candidate list, only one neighbor must be considered to satisfy both sets of constraints. A similar argument can be made for bottom candidate lists.

Elastic Relaxation
Even when a deformation is allowable, the resultant object shape may not be a minimum energy configuration. The system energy depends on the spacing between elements and the elastic properties of the object. A fully elastic object can be deformed, but it has a single configuration for which the object's energy is minimal. In contrast, a plastic object can reach a minimal energy state in a new shape. This system applies an energy relaxation algorithm between applications of the 3D ChainMail algorithm and whenever processing time is available. Constraints similar to those used for deformation are used to determine if an element satisfies elastic constraints relative to its neighbors. If not, the element's position is adjusted to reduce the energy of the object.

Collision Detection
Each time an element is moved, the system checks for a collision between the moved element and other objects in the system. If a collision occurs, the system is not permissible and the moved elements are returned to their original positions. The step size towards the desired position is reduced until an allowable system is found or a minimum step size is reached. In the examples presented here, objects only collide with the bounding walls of the viewing window and hence, in this implementation, a simple check of the element's x and y values is used to detect collisions. This system has also been implemented with similar results for more complex environments using a collision detection algorithm for volumetric objects[9].

Visualization

In the current system, a number of techniques based on OpenGL have been used to render the 2D and 3D objects for real-time visual feedback. In 2D, object elements were displayed as either points, connecting grid lines, or 4-sided shaded polygons defined by neighboring elements (see Figure 6). In 3D, either all of the object elements or just the surface elements were displayed as points.

High quality volume rendering is slow because each of the millions of volume elements can contribute to the final image. However, new algorithms and hardware implementations[15, 16, 21] have begun to address the need for interactive volume rendering of regular volumes. Volume rendering of irregular volumes at interactive rates remains a significant challenge and it is the focus of a related project at our laboratory.

RESULTS

This system was implemented in C and uses OpenGL, Tcl/Tk and togl in the user interface. The current implementation runs on SGI platforms. The examples and results reported here were run on an SGI Indy (MIPS R5000 processor with XZ graphics) and SGI Challenge (using one of 8 MIPS R10,000 processors). The size of the largest data volume reported here (125,000 elements) was chosen so that frame rates of at least three frames per second could be achieved on the SGI Indy.

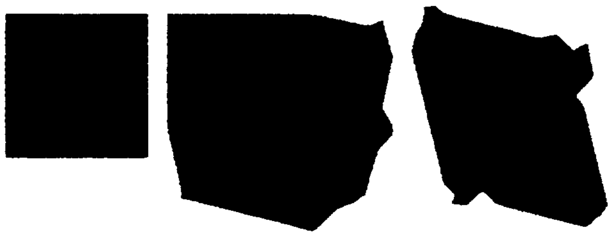

Figure 5. Sequential deformation of a 2D square object of size 180x180.

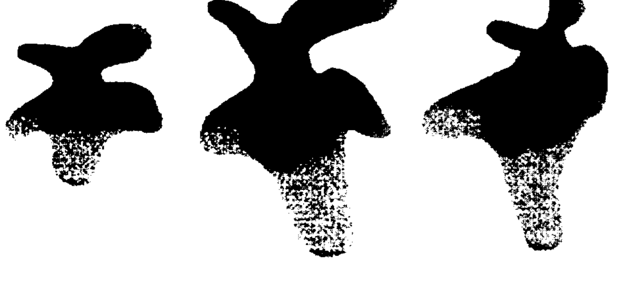

Figure 6. Deformation of a non-convex 2D object.

Figures 5 and 6 and Plates 1 to 4 show deformation of various 2D objects. We were able to achieve acceptable frame rates for an object of size 180x180 elements. In addition, we were able to deform non-convex objects and objects in which the position of some elements were fixed. Plate 4 shows the deformation of a 3D cube of size 50x50x50. When only surface points were rendered, frame rates of at least three per second were achieved on the SGI Indy.

SUMMARY AND DISCUSSION

We have presented an algorithm that enables fast deformation of objects containing hundreds of thousands of volumetric elements. This algorithm can model a range of substances including; rigid, deformable, elastic, and plastic materials. Unlike other work where deformation is modeled with complex calculations on a small number of elements, this algorithm performs simple calculations on a very large number of elements to achieve complex behavior.

ACKNOWLEDGMENTS

Special thanks to J. Samosky who suggested the name 3D ChainMail and to E. Gibson, H. Pfister, J. Marks, and B. Mirtich for helpful suggestions.

BIBLIOGRAPHY

1. R. Avila and L. Sobierajski, personal communication, 1996.
2. D.Baraff "Analytical methods for dynamic simulation of non-penetrating rigid bodies", (proc. SIGGRAPH), Computer Graphics, Vol. 24, pp. 19-28, 1989.
3. M. Bro-Nielsen, "Modeling elasticity in solids using active cubes - application to simulated operations", in Computer Vision, Virtual Reality and Robotics in Medicine, (proc. CVRMed '95), ed. Nicholas Ayache, pp. 535-541.
4. D. Chen, "Pump it up: computer animation of a biomechanically based model of muscle using the finite element method", PhD thesis, Media Arts and Sciences, MIT, 1991.
5. S. Cover, N. Ezquerra, Ja. O'Brian, R. Rowe, T. Gadacz, E. Palm, "Interactively deformable models for surgery simulation", IEEE Computer Graphics and Applications, Vol. 13, 6, pp. 68-75, 1993.
6. M. Desbrun, M.-P. Gascuel, "Animating soft substances with implicit surfaces", Computer Graphics (proc. SIGGRAPH), pp. 287-290, 1995.
7. I. Essa, S. Sclaroff, A. Pentland, "Physically-based modeling for graphics and vision", in Directions in Geometric Computing, ed. Ralph Martin, Information Geometers, U.K., 1993.
8. W. Hsu, J. Hughes, H. Kaufman, "Direct Manipulation of Free-form deformations", Computer Graphics (proc. SIGGRAPH), Vol 26, 2, pp. 177-184, 1992.
9. S. Gibson, "Beyond volume rendering: visualization, haptic exploration, and physical modeling of element-based objects", in Visualization in Scientific Computing (proc. Eurographics

workshop on ViSC), eds. R. Scateni, J. van Wijk, and P. Zanarini, Springer-Verlag, pp. 10-24, 1995.

10. I. Hunter, T. Doukoglou, S. Lafontaine, and P. Charette, "A teleoperated microsurgical robot and associated virtual environment for eye surgery", Presence, Vol. 2, pp. 265-280, 1993.
11. A. Kaufman, "Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes", Computer Graphics, Vol 21, 4, pp. 171-179, 1987.
12. A. Kaufman, Volume Visualization, IEEE Computer Society Press, Los Alamitos CA, 1991.
13. A. Kaufman, Daniel Cohen, Ronald Yagel, "Volume Graphics", IEEE Computer, Vol 23, 7, pp. 51-64, 1993.
14. A. Kaufman, "Volume Visualization", CRC Handbook of Computer Science and Engineering, 1996.
15. G. Knittel, "A scalable architecture for volume rendering", Comput. and Graphics, Vol. 19, No. 5, pp. 653-665, 1995.
16. P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transform", proc. SIGGRAPH, Computer Graphics, pp. 451-457, 1994.
17. Y. Lee, D. Terzopoulos, and K. Waters, "Realistic modeling for facial animation", Computer Graphics (proc. SIGGRAPH), pp. 55-62, 1995.
18. D. Metaxas, D. Terzopoulos, "Dynamic deformation of solid primitives with constraints", Computer Graphics (proc. SIGGRAPH), Vol 26, 2, pp. 309-312, 1992.
19. B. Mirtich, J. Canny, "Impulse-based simulation of rigid bodies", proc. 1995 Workshop on Interactive 3D Graphics, pp. 181-188, April, 1995.
20. A. Pentland, J. Williams, "Good vibrations: modal dynamics for graphics and animation", Computer Graphics, Vol 23, 3, pp. 215-222, July, 1989.
21. H. Pfister, " ", Ph.D. thesis, SUNY at Stony Brook, Aug. 1996.
22. T. Sedeberg and S. Parry, "Free-form Deformation of Solid Geometric Models", Computer Graphics (proc. SIGGRAPH) Vol 22, 4, Aug. 1986, pp. 151-160.
23. L. Sobierajski and A. Kaufman, "Volumetric ray tracing", proc. Volume Visualization Symposium, Washington, DC, pp. 11-18, 1994.
24. L. Sobierajski, A. Kaufman, "Volumetric radiosity", Technical Report 94.01.05, Computer Science, SUNY Stony Brook, 1994.
25. D. Terzopoulos, J. Platt, A. Barr, K. Fleischer "Elastically deformable models", Computer Graphics, Vol 21, 4, pp. 205-214, July, 1987.
26. D. Terzopoulos, Kurt F., "Modeling inelastic deformation: viscoelasticity, plasticity, fracture", Computer Graphics, Vol 22, 4, pp. 269-278, Aug., 1988.
27. D. Terzopoulos, K. Waters, "Physically-based facial modeling, analysis, and animation", J. Visualization and Computer Animation, Vol. 1, pp. 73-80, 1990.
28. S. Wang and A. Kaufman, "Volume sculpting", ACM Symposium on Interactive 3D Graphics, Monterey, CA, pp. 151-156, April 1995.
29. S. Wang and A. Kaufman, "Volume sampled elementization of geometric primitives", Proceedings Visualization '93, San Jose, CA, pp. 78-84, October 1993.
30. K. Waters, "A Muscle model for animating three-dimensional facial expression", Computer Graphics, Vol. 21, 4, July, 1987, pp. 17-24.