MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining

Sarah F. Frisken-Gibson

TR2000-24 December 2000

Abstract

In Volume Graphics, objects are represented by arrays or clusters of sampled 3D data. A volumetric object representation is necessary in computer modeling whenever interior structure affects an objectś behavior or appearance. However, existing volumetric representations are not sufficient for modeling the behaviors expected in applications such as surgical simulation, where interactions between both rigid and deformable objects and the cutting, tearing, and repairing of soft tissues must be modeled in real time. 3D voxel arrays lack the sense of connectivity needed for complex object deformation while finite element models and mass-spring systems require substantially reduced geometric resolution for interactivity and they can not be easily cut or carved interactively. This paper discusses a linked volume representation that enables physically realistic modeling of object interactions such as: collision detection, collision response, 3D object deformation, and interactive object modification by carving, cutting, tearing, and joining. The paper presents a set of algorithms that allow interactive manipulation of linked volumes that have more than an order of magnitude more elements and considerably more flexibility than existing methods. Implementation details, results from timing tests, and measurements of material behavior are presented.

IEEE Trans. on Visualization and Compuer Graphics, Vol. 5, No. 4, 1999, pp. 333-348

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining

Sarah F. Frisken-Gibson*

TR-2000-24 June 2000

Abstract

In Volume Graphics, objects are represented by arrays or clusters of sampled 3D data. A volumetric object representation is necessary in computer modeling whenever interior structure affects an object's behavior or appearance. However, existing volumetric representations are not sufficient for modeling the behaviors expected in applications such as surgical simulation, where interactions between both rigid and deformable objects and the cutting, tearing, and repairing of soft tissues must be modeled in real time. 3D voxel arrays lack the sense of connectivity needed for complex object deformation while finite element models and mass-spring systems require substantially reduced geometric resolution for interactivity and they can not be easily cut or carved interactively.

This paper discusses a linked volume representation that enables physically realistic modeling of object interactions such as: collision detection, collision response, 3D object deformation, and interactive object modification by carving, cutting, tearing, and joining. The paper presents a set of algorithms that allow interactive manipulation of linked volumes that have more than an order of magnitude more elements and considerably more flexibility than existing methods. Implementation details, results from timing tests, and measurements of material behavior are presented.

ålso published in IEEE Trans. Visualization and Computer Graphics, Vol. 5, No. 4, 1999, pp. 333-348

Publication History:–

1. First printing, TR-2000-24, June 2000

Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining

Sarah F. Frisken Gibson

Abstract— **In Volume Graphics, objects are represented by arrays or clusters of sampled 3D data. A volumetric object representation is necessary in computer modeling whenever interior structure affects an object's behavior or appearance. However, existing volumetric representations are not sufficient for modeling the behaviors expected in applications such as surgical simulation, where interactions between both rigid and deformable objects and the cutting, tearing, and repairing of soft tissues must be modeled in real time. 3D voxel arrays lack the sense of connectivity needed for complex object deformation while finite element models and mass-spring systems require substantially reduced geometric resolution for interactivity and they can not be easily cut or carved interactively.**

This paper discusses a linked volume representation that enables physically realistic modeling of object interactions such as: collision detection, collision response, 3D object deformation, and interactive object modification by carving, cutting, tearing, and joining. The paper presents a set of algorithms that allow interactive manipulation of linked volumes that have more than an order of magnitude more elements and considerably more flexibility than existing methods. Implementation details, results from timing tests, and measurements of material behavior are presented.

Keywords— **volume graphics, volume modeling, physics-based graphics.**

I. Introduction

As computer graphics becomes more widespread and computational power increases, applications such as surgical simulation, computer animation and computer aided design are demanding more and more realism. These applications require physically realistic and/or physically plausible modeling of the dynamics of multi-object collisions, the deformation of complex objects, and the modification of object topology from simulated cutting, sculpting, and fracture. In addition to physical realism, many applications demand interactive feedback, requiring on-line object modification and fast calculation of collision responses and object deformation.

While there has been a substantial amount of work investigating physical interactions between rigid bodies that are represented by object surfaces (e.g. [1], [2], [3]), surface models are not well suited for modeling physics-based object deformation or for modeling arbitrary cutting or sculpting of objects. In fact, whenever the internal object structure is important for the appearance or behavior of a graphical object, a volumetric object representation is necessary. There are three basic classes of volumetric object representations that have been used in computer graphics. These include: 3D sampled data, such as that acquired from tomographic imaging systems or computer simulations; particle systems; and geometric meshes for modeling deformable objects using mathematical techniques such as finite element models (FEM) or mass-spring systems.

Each of these three volumetric representations has its advantages and disadvantages but none is ideally suited for the requirements laid out above. Sampled volumes are generally of high resolution and can be generated from measured data. It is possible to sculpt sampled volumes [4], [5], [6] , to detect collisions between sampled volumes [7], [8] , and to perform limited deformation by warping the volumetric grid [9] . However, sampled volumes lack a sense of connectivity that is necessary for modeling complex tissue deformation or the cutting of volumes into distinct pieces. Particle systems have been used to model objects that are highly deformable and to model separation and joining of such objects [10] but because particle systems lack a representation of internal structure, they are not suitable for modeling most materials.

The geometric meshes used in FEM and mass-spring systems are most promising and there are many examples of such systems that have been applied in computer graphics (e.g. see [11]). In recent work on cloth modeling, Baraff and Witkin introduced a system in which a 6000-node cloth model can be deformed at rates of 2-3 seconds per frame. While the results are impressive and the system is reported to be much faster than competing approaches, it is still too slow for interactive simulation. In surgical simulation, Bro-Nielsen and Cotin et al. have achieved interactive volume deformation of objects with up to several thousand nodes by preprocessing the matrices governing simulation of object deformation [12], [13], [14], [15] . However, because hours of pre-processing are required, arbitrary cuts or other topology changes in the objects can not be made interactively. A small number of papers have discussed or addressed the need for being able to change object topology during object cutting or tearing [16], [13], [17] . Cotin et al use a system in which a relatively small mass-tensor system with 280 vertices (1260 tetrahedra)is embedded into a preprocessed FEM system with 1537 vertices (7039 tetrahedra). The vertices in the mass-tensor system can be interactively cut and carved [14] . While this is an important practical solution for a problem where the action of the surgeon is predictable, it does not solve the problem for more general applications.

This paper presents a *linked volume* representation, in which each element in a sampled volume is explicitly linked to its 6 nearest neighbors. These links are stretched, contracted and sheared during object deformation and deleted or created when objects are cut or joined. In some sense,

Dr. Frisken Gibson can be contacted at MERL - A Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA, 02139. E-mail: gibson@merl.com .

this representation is a hybrid of sampled volumes and geometric mesh representations. However, because of the way interactions are modeled, the linked volume approach is different from the approaches used in FEM or mass-spring systems. In particular, the system is simply stored as a high resolution array of elements, each containing some knowledge of its material properties, its visual properties, its dynamic state, and its links to neighboring elements. All operations are performed locally on the volume elements: if an element is removed, only the links of the 6 neighboring elements need be updated; if a link is removed or added, only two neighbors are affected; and object deformation is not calculated by solving a large system of equations, rather, deformations are propagated from one element to the next via the links, just as a shock wave propagates from one molecule to the next through real material.

This work is motivated by the fact that an interactive system must trade off three things: 1) geometric complexity; 2) the sophistication of the modeling algorithm; and 3) generality, or the ability to cope with on-line changes in shape or topology. Existing approaches using FEM or mass-spring systems have tended to use sophisticated modeling algorithms and relatively low geometric complexity. Interactivity is achieved by preprocessing and/or careful structuring of the system's mathematical representation, resulting in low generality. A linked volume approach uses models with high geometric complexity and generality and achieves interactivity by using low-cost mathematical modeling. This paper presents systems where objects with tens and hundreds of thousands of elements can be interactively deformed and cut or carved, resulting in an order of magnitude more elements to be modeled interactively with considerably more flexibility than existing FEM and mass-spring systems.

The exploration of these ideas have been limited in previous work for a number of reasons. First, linked volumes require a large amount of memory, something that has only recently become available on reasonably-priced computer systems. Second, rendering deformed volumes is extremely slow and has limited the usefulness of high resolution volumes for interactive systems. We currently address this problem by rendering only points or triangles on the surface of the deformed volume. However, we expect that advances in volume rendering hardware will enable fast rendering of deforming volumes within the next few years [18] . Finally, while much can be learned from other work in physics-based graphics with respect to collision detection, dynamic simulation and object deformation, most algorithms cannot be directly applied to linked volumes because of the large numbers of elements in volumetric objects. This paper is intended to demonstrate the potential of the linked volume approach and to provide several specific approaches and algorithms to encourage further work in this area.

The paper is organized as follows. In section II, the linked volume structure is described in detail along with some variations for different applications. In section III a method for detecting collisions between linked volumes is presented along with timing tests and some discussion of alternative methods. In addition, a method for calculating impact forces is proposed. In section IV, a fast method for deforming linked volumes is presented along with tests of the material behavior when this algorithm is used. Limitations and extensions of this approach are also discussed. In section V, algorithms for object cutting, carving, and joining are presented along with implementation details and results. Finally, section VI summarizes contributions of this paper and discusses future directions.

II. Linked-Volume Representation

A. Background

Kaufman, Cohen, and Yagel introduce the field of Volume Graphics in [19]. Volume Graphics deals with the synthesis [20], modeling [5], manipulation [7], and rendering (e.g. [21]) of volumetric objects. Kaufman and his collaborators have shown that many of the visual effects produced by conventional graphics, including object shading, anti-aliasing, inter-object reflectance, and radiosity calculations, can also be performed on volumetric objects [22], [23], [5]. This paper focuses on object manipulation, including object deformation and the modeling of interactions between objects. Prior Volume Graphics work in object manipulation includes haptic interaction with volumetric objects [6], [24], modeling collisions between objects [7], [8], and modeling volumetric object deformation [25]. A linked volume representation was introduced in [26] in reference to surgical simulation.

Because many different terms have been used to describe the components of sampled volumes, the terms used in this paper are defined briefly here and are illustrated in Fig. 1. *Elements* are points within the 3D volume. Element values, such as image intensity, material density, or other material properties, are assumed to be sampled at these points[1]. In this paper, elements are assumed to lie on a (possibly deformed) rectilinear grid. *Cells* are defined to be the volume of space (cubes or polyhedra) between 8 neighboring elements. *Links* are the explicit connections between neighboring elements and, in this paper, up to 6 links are assigned to each element. These are links to the elements' left, right, front, back, top and bottom neighbors.

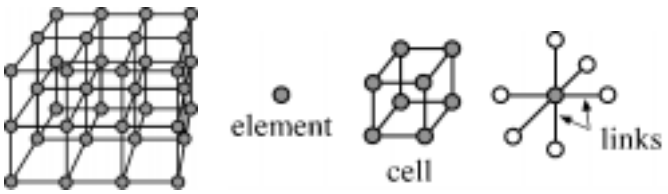


Fig. 1. Components of a sampled volume.

B. Linked Volumes

One of the goals of the linked volume approach is to permit representation of complex geometry in the object

[1]The values can be sampled from a filtered data volume, for example measured samples often represent a value averaged over a finite volume near the sample point.

model. The MRI cross-section in Fig. 2 illustrates the need for geometric complexity. Tissues such as muscle or fat contain a great deal of interior structure, much of it at the resolution of the 3D data. Because this interior structure affects both the local and the global behavior of these tissues, it is important to have an object representation that can incorporate this detail. Linked volumes are constructed at the resolution of available measured or synthetic data by adding explicit links between neighboring elements. When starting from measured data that may contain several objects, the simplest approach, and the approach taken here, is to store each linked object in a separate volume[2]

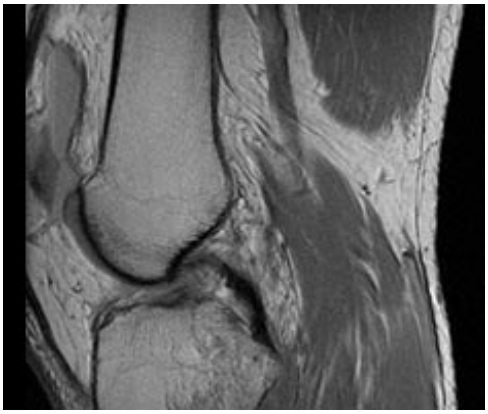


Fig. 2. This cross-section from a 3D Medical Resonance Image of the human knee illustrates the amount of detail available for modeling tissues. Internal structure in the muscle or bone affects the mechanical behavior of these tissues and should be incorporated into models for physically realistic simulation.

There are many possible data structures that can be used to represent linked volumes. Figs. 3 and 4 are two examples. When the element structure of Fig. 3 is used, the object is stored in a 3D array of SimpleLinkedElements, the element's sampled value is an intensity (perhaps from a measured image), and links are encoded into a single byte, in which the lowest 6 bits indicate either the presence or absence of a link to each of the element's 6 neighbors. If a neighbor is present, then it is accessed from the 3D object array using constant index offsets. This structure allows links to be formed and broken and can be used in applications that model cutting, carving, and fracturing but not object deformation.

```
struct SimpleLinkedElementStruct {
        unsigned char intensity;
        unsigned char links;

} SimpleLinkedElement;
```

Fig. 3. Element structure for a simple lined volume that can be cut and carved but not deformed.

The element structure of Fig. 4 does not require a 3D array for storing the object. Instead, elements are stored in an arbitrary list and neighbors are accessed via pointers. This approach is preferred in some circumstances. For example, because this linked list structure does not store empty elements, it may use less memory than a sampled 3D array. A flexible object representation is also desirable in applications such as 3D drawing or painting, where elements are randomly added to the object and the final volume of the object is not known when the object memory is allocated.

```
struct LinkedElementStruct {
        unsigned char r, g, b, a;
        unsigned char type;
        float x, y, z;
        struct LinkedElement *top, *bottom, *right,
                             *left, *front, *back;
} LinkedElement;
```

Fig. 4. Element structure that stores the element's color and transparency, the object type, the element position, and pointers to linked neighbors.

Using the data structure of Fig. 4, cutting and carving are accomplished by setting neighbor pointers to NULL. Joining is performed by setting corresponding pointers of the two elements being joined to point to each other. Deformation or object translation is accomplished by changing the (x,y,z) positions of elements in the object. The element type is used to identify different objects in collision detection and will be discussed in Section III. The r, g, b, and a values illustrate the fact that properties such as color and opacity or other material properties can be stored for each element. The ability to store this information for each element is one of the strengths of a volumetric approach because it allows for the representation of complex, heterogeneous materials. However, there are trade-offs between complexity and object size. The data structure of Fig. 4 contains more than 40 bytes per element and requires a significant amount of memory for reasonably sized objects.

III. Collision Detection

A. Background

Detecting object collisions is a fundamental requirement for a physics-based simulation. Efficient algorithms are essential because in dynamic simulations, collision detection is generally performed more frequently than all other operations. Much progress has been made towards fast detection of collisions between complex polygonal models (e.g. [3], [27]) and several researchers have developed methods for simulating the transfer of energy and momentum between rigid polygonal models upon collision (e.g. [1], [2]). However, because of the large number of elements in volumetric objects, techniques developed for surface-based graphics can not be directly applied.

There has been some work in collision detection using a discrete space representation. A method using an occupancy map, in which collisions can be simply and efficiently detected for relatively large objects was presented in [7] and

[2]Dependencies between objects, such as attachments of a muscle to bone are represented as constraints in the modeling system.

is described in more detail in Section III-B. Related methods can be found in papers by Shinya and Fourge [28] and Uchiki et al. [29]. In addition, He and Kaufman use an octree based method for collision detection of volumetric objects [8]; Greene [30] used a discrete representation of occupied space for guiding the stochastic growth of graphical plants; robotics researchers have used a discrete space representation for obstacle avoidance in path planning (e.g. [31]); and some researchers have investigated haptic rendering of volumetric objects with force-feedback [6], [24].

B. Occupancy Map Method

In its most basic form, the occupancy map method allocates a 3D array of small cells representing the entire volume of virtual space used by the application. The size of each occupancy map cell is similar to the spacing between elements in the volume and each cell can contain a single pointer to an object element. If no element occupies that cell, then the pointer is the NULL pointer. As objects are moved about the virtual space, object elements are mapped into cells in the occupancy map. During the mapping, if a target cell contains a pointer to a different object, then a collision between the two objects is detected. Pointers are stored in the occupancy map so that colliding elements can be easily identified for calculating collision responses. This algorithm is illustrated in 2D in Fig. 5. Pseudocode for the occupancy map method is presented in Fig. 6.

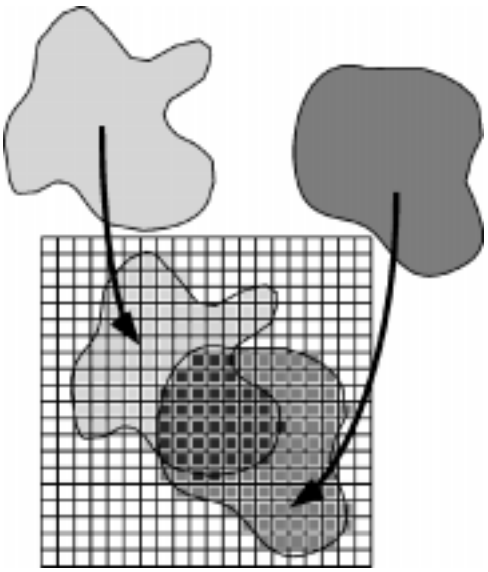


Fig. 5. In the occupancy map method, all object elements are mapped into a regular grid of cells that span the interaction space. Collisions are detected in the dark grey cells where elements of one object are mapped into cells that are already occupied by the other object.

The occupancy map method can be used for both regularly spaced and irregularly spaced volumes as long as the element spacing is comparable to the occupancy map cell size. Fig. 7 shows three frames from an interactive system in which a deformable object is indented in real-time with a probing tool. The collision between the tool

```
SimpleElement object[X][Y][Z];
SimpleElement* occMap[W][H][D];

/* 1) check occupancy map cells */
for(all non-zero object elements) {
   map new element position to occMap cell
   if (cell occupied by another object) {
      detect collision
      reduce step size for backtracking
      exit
      }
   }

/* 2) remove object from occupancy map */
for (all non-zero object elements) {
   map old element position to occMap cell
   if (occMap pointer is equal to element pointer)
      set occMap cell to NULL pointer
   }

/* 3) move object and update occupancy map*/
for (all non-zero object elements) {
   set new element position
   map new element position to occMap cell
   set occMap cell to element pointer
}
```

Fig. 6. The occupancy map algorithm for detecting collisions between moving volumetric objects.

and the deforming object is detected using the occupancy map method. When large deformations are possible, (i.e. when element spacing can change by more than a factor of 2), multi-scale occupancy maps or volume filling methods could be used to build a more accurate occupancy map at the cost of increased complexity and/or reduced speed in collision detection.

Fig. 7. Three still images from an interactive system in which a deformable 3D linked volume object is probed in real-time by a user controlled tool. Collisions between the object and the tool are detected using the occupancy map method and are used to calculate the object deformation. This object has size 40x40x40 and the occupancy map has size 200x200x200. Deformation of the sphere was performed using the algorithm presented in Section IV and the object was rendered by drawing triangles on the surface of the linked volume using OpenGL.

Potential self intersections can be detected in step 1) of the pseudocode of Fig. 6 by noting when an occupancy map cell is already occupied by an element from the *same* object if step 2) is performed before step 1). However, because adjacent object elements may map into the same occupancy map cell (depending on the relative sizes of the occupancy map grid and the element spacing), either the occupancy map cell size must be restricted to be smaller than the minimum distance between neighboring elements or potential self intersections must be ignored when the

colliding elements are near neighbors.

C. Calculating Collision Response

Once a collision between graphical objects is detected, the next step is to determine the response to the collision. There are a number of different responses that might be modeled. For example, in a system where objects are moved interactively but they do not have energy or momentum, the desired effect is usually the prevention of object interpenetration. In systems of rigid objects, collisions result in an exchange of energy and momentum according to physical laws. In systems of non-rigid objects, collisions result in energy exchange, energy dissipation, and energy storage in the form of object deformations. Under some circumstances, graphical objects may be expected to break, tear or fracture upon impact.

When the goal is to prevent object interpenetration, a standard back-tracking algorithm is used to find the point of collision. In back-tracking, objects are moved in relatively large steps towards a desired position. Step sizes are relatively large to ensure interactivity in the simulation but small enough to prevent objects from appearing to leap through each other. If object penetration is detected for a proposed new object position, the object is not moved to that position. Instead, the step size of the moving object is reduced. If a penetration is detected using this new step size, the step size is again reduced. Otherwise, the step size is increased slightly and the new position is investigated. This process is repeated until the maximum allowable step towards the desired position is determined. We have used this approach in several systems. Results from a set of timing tests are reported in section IIIE.

When the goal is to model physically realistic interactions between objects, collision response must also be modeled. Two basic methods are used in computer graphics for calculating collision responses for rigid objects. These are penalty-based methods (e.g. [32]), where restoring forces are introduced to separate penetrating objects, and analytic methods (e.g. [1], [2]), where contact forces are determined analytically from constraints that prevent object inter-penetration and guarantee physically realistic dynamics. While analytic methods are more accurate and numerically stable than penalty-based methods, penalty methods have the following advantages for volumetric objects. First, penalty-based methods have been successfully applied to deformable object models while current analytic approaches are limited to rigid bodies [3]. Second, a tractable solution of an analytic method relies on a relatively small number of contact points. In surface-based graphics, contact points can be generalized to occur at polygon vertices or edges [1], and hence the number of contact points is relatively small for object models represented by a reasonable number of surface polygons. In a volumetric model of the same object, the number of contact points could be several orders of magnitude larger, making analytic solutions less feasible in interactive simulations.

Penalty-based methods apply restoring forces to penetrating vertices of the colliding objects. The restoring forces act to separate the interpenetrating objects during the dynamic simulation. The strength of the restoring force is generally a function of the depth of penetration and the object stiffness. A large stiffness puts a higher penalty on object penetration, but also requires smaller integration times for numerical stability.

A similar penalty-based approach can be used for volumetric objects. Penetrating volume elements are detected using the occupancy map and individual restoring forces on each element are summed. Each restoring force vector is calculated as the product of the material stiffness and the depth of penetration in the direction of the local surface normal. In order to calculate depths of penetration and local surface normals efficiently and accurately, we refer to work discussed in [34], which introduced the use of a distance map for encoding surfaces into sampled volumes for use in high quality shaded volume rendering. Distance maps, which encode the distance to the closest object surface into sampled volume elements, can also be used by penalty methods to determine not only the depth of each penetrating element but also the local surface normal. The pseudocode presented in Fig. 8 outlines this approach. The calculated resultant forces and torques are used in a dynamic simulation of the system.

```
object stiffness = k
set force, F = 0
set torque, T = 0

for (all non-zero object elements) {
    check occupancy map for collisions
    if (collision) {
      get object depth at collision point, depth
      calculate depth gradient at collision point, ∇̂(depth)
      accumulate force, F = F + f = F + k * depth * ∇̂(depth)
      accumulate torque, T = T + t = T + f × r
    }
}
```

Fig. 8. Calculating response forces for colliding volumetric objects.

D. Improving Collision Detection

A number of straight-forward improvements can enhance the speed of the occupancy map method. First, since collisions occur at object surfaces, it may only be necessary to map surface elements or a shell of elements near the surface into the occupancy map. This method is compared with the basic occupancy map method in the following section. Second, instead of maintaining a large occupancy map to represent the entire virtual space, the occupancy map method can be applied in smaller, temporary arrays that span only the volumes of potential overlap between objects in the system. These volumes of potential overlap can be determined quickly using boxes bounding the volumetric objects and exploiting techniques developed in

[3] One exception being the work of Baraff and Witkin [33] who used analytic methods for simulating collisions between flexible bodies with a limited deformation model.

surface-based graphics for fast collision detection between bounding boxes. We have implemented a prototype system to test this approach. Finally, hierarchical data structures can be used to quickly determine which sub-volumes of potentially colliding objects should be written into the occupancy map. He and Kaufman [8] used hierarchical data representations without an occupancy map approach to detect the overlap of volumetric objects. They found that an octree-based representation of static, regularly spaced data yields efficient collision detection for reasonably sized objects. However, such hierarchical methods that rely on significant pre-processing have limitations in systems where objects deform significantly or where elements can be interactively created or destroyed.

E. Experimental Results

We have implemented the basic occupancy map methods for volumetric collision detection in a number of 2D and 3D systems ranging from computer-based jigsaw puzzles and 2D drawing tools to 3D object manipulation. These systems have been implemented in C on either SGI or HP platforms. Most of the systems prevent object interpenetration using back-tracking when object collisions are detected but some preliminary studies using penalty-based methods to calculate collision responses between rigid volumetric objects have been done.

A number of timing tests were made in order to demonstrate the performance of the collision detection algorithm. The tests involved two objects: a voxelized sphere with a radius of 30 voxels located at the center of the interaction space; and a voxelized cube, initially located at the side of the interaction space. While simple geometric objects were used in these tests, collisions between objects of complex geometry would be detected at the same rates as these simple shapes as long as the number of elements in the two sets of objects were the same.

During the test, the voxelized cube was moved in steps of size 5 voxels along a path through the center of the interaction space. Two sets of experiments were performed. In the first, the cube was permitted to penetrate the voxelized sphere and a collision was recorded at each step for each pair of the overlapping elements. In the second experiment, objects were prevented from penetrating each other by combining collision detection with back-tracking. Two sizes of cubes were used in the tests, a cube with dimensions 31x31x31 and a cube with dimensions 15x15x15. The collision detection was performed for two versions of the collision detection algorithm, the first mapping all of the elements of the cube into the occupancy map and the second mapping only surface elements into the occupancy map.

The results from these tests are presented in Table 1. Times are reported in milliseconds. The system was written in C on an SGI Indigo2 without particular effort to optimize code. The reported times are for all operations in each step except rendering. Collision detection timing depends on the degree of overlap between objects. Hence, in the collision detection test, timing is reported for the cube in free space and fully inside the sphere and, in the system for preventing object interpenetration timing is reported for the cube in free space, at initial contact with the sphere, and when the cube is touching the sphere. In addition, because back-tracking is applied as soon as a collision is detected, and because the test system maps object elements into the occupancy map in a fixed element order, the position of the contacting element in the object volume affects the timing. For this reason, in the object penetration tests, the cube is moved towards the sphere from 6 directions, from left to right, right to left, top to bottom, etc. The times reported in Table 1 are the timing results from left to right and the average times (in brackets) for the 6 directions of motion.

These timing tests show that collisions between volumetric objects can be detected at interactive rates for reasonably large objects using a simple collision detection algorithm. When only object surfaces are mapped into the occupancy map, collisions between an object of size 61x61x61 and an object of size 31x31x31 were detected and prevented at rates of approximately 40 Hz using unoptimized code. The collision times increase approximately linearly with the number of elements mapped into the occupancy map. The use of data hierarchies, bounding boxes, and other techniques developed for surface-based graphics will greatly improve the speed of the collision detection.

IV. Object Deformation

A. Background

In many applications, realistic object modeling requires the deformation of soft objects. Volumetric representations have advantages over surface-based models for modeling object deformation because they allow interior structure to impact the physics of object interactions. However, because volumetric objects consist of a large number of elements, most object deformation techniques are computationally intensive. This is especially problematic in simulations where interactivity is required.

The two most common techniques that have been used for modeling the deformation of volumetric objects are FEM and mass-spring systems. Mass-spring systems have been used for facial animation (e.g. [35], [36], [37]), in surgical simulation (e.g. [38]) and to animate graphical objects (e.g. [39], [40]), or cloth (e.g. [41], [42]). FEM is typically used in off-line scientific computing such as the analysis of mechanical structures. However, FEM has also been used in graphical applications for fabric modeling (e.g. [43]), animation (e.g. [44], [45]) and surgical simulation (e.g. [46], [12], [15], [47]).

While mass-spring systems and FEM use volumetric object representations and provide established mathematical techniques for modeling the physical behavior of solid deformable objects, they are computationally demanding. These techniques can be accelerated by a significant reduction in the number of elements in the system [46] and data pre-processing. Bro-Nielsen [15] limits the FEM solution to determine displacements only for surface elements and

TABLE I

Timing for volumetric collision detection. Times, measured in milliseconds, are for moving or attempting to move the voxelized cube with a step size of 5 voxels. The middle column shows times for detecting and recording all of the overlapping points (or overlapping surface points) in the objects as the cube is passed through the center of a large voxelized sphere. The right column shows times for each attempted step when the objects are prevented from penetrating each other.

object (# elements)	**detecting collision points**		**preventing penetration**				
	free space	collision	free space	impact	(avg)	touching	(avg)
large cube (29791)	134	156	133	153	(238)	34.8	(96.9)
small cube (3375)	13.7	15.5	13.8	14.3	(42.5)	0.32	(9.94)
large cube surface (5402)	19.7	23.7	19.6	24.7	(25.4)	8.85	(10.6)
small cube surface (1178)	3.76	4.42	3.79	4.11	(4.74)	0.32	(1.05)

assumes a small number of externally applied forces. Pentland and Williams [48] analyze a given object in a preprocessing step to determine its modes of vibration and calculate deformations as a superposition of these modes. Bro-Nielsen and Cotin [49], [12] pre-calculate responses to infinitesimal forces and deformations for each node in the element and then approximate the global deformation as a linear superposition of these pre-calculated responses.

As discussed above, these techniques for achieving interactivity in deformable object modeling trade off the ability to model complex object geometry and the generality of the system for increased speed. We propose an alternative approach which uses a less accurate mathematical model so that interactivity can be maintained with high geometric complexity and the ability to make arbitrary interactive topological changes to the object. In the following sections, one algorithm that uses this approach is discussed.

B. ChainMail and Elastic Relaxation

Here we review a two-process method for modeling deformation that was originally presented in [25] and provide the results of timing tests and measurements of simulated material behavior. The first process, ChainMail, provides an initial estimate of the new shape that is based on geometric constraints. It guarantees a plausible object shape in one time step even for relatively large deformations, allowing us to achieve interactivity for relatively large numbers of volume elements. The second process relaxes the shape of the approximate deformation over several time steps using elastic relaxation. The behavior of the material is determined both by the geometric constraints and by parameters in the elastic relaxation process.

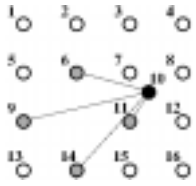
Although inertia and damping are not *explicitly* modeled in this method, modeled objects do exhibit these behaviors. Inertial behavior occurs because the elements in the ChainMail process do not move unless they violate constraints between local neighbors and they move only minimum distances in order to satisfy the violated constraints. Damping occurs because the iterative elastic relaxation process is a closed negative feedback system in which element positions are adjusted in small steps towards an optimal position. When the step size is small, the system is more damped. As the step size – the gain of the closed feedback system – increases, the system becomes critically or under-damped.

B.1 ChainMail

ChainMail uses the linked data structure described in Section II. Each element is linked to its 6 nearest neighbors and disturbances are propagated through the system via these links. During the simulation, if the displacement of an element violates constraints on the links, then the affected neighbors are moved to the closest position where the link constraints are again satisfied. After each element is moved, its links to unmoved neighbors are checked and then corrected if necessary. The disturbance is fully propagated through the volume in one time step, providing a fast approximation of the deformed object shape which satisfies geometric constraints on the link lengths. ChainMail is particularly fast for homogeneous (though possibly anisotropic) materials because each volume element need only be considered at most once and is compared to at most one neighbor.

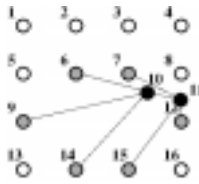
Details of the ChainMail algorithm and proof of the above property are presented in [25]. The basic method is illustrated in Fig. 9, in which the algorithm is applied to a low resolution 2D grid.

The two types of lists that are maintained in ChainMail are shown in the tables on the right side of Fig. 9. These are a list of moved elements and lists of candidates for movement, classified according to whether the list elements are the right, left, top, or bottom neighbors of their sponsoring element. In the diagram, black circles indicate elements that have been moved and grey circles indicate elements that are to be considered for movement. In 9a) element 10 is moved from its equilibrium position. Element 10 is added to the moved list and its 4 neighbors are added to their respective movement candidate lists. Next, each candidate list is processed in order until it is empty. In 9b), the right candidate, 11, is moved to satisfy geometric constraints with respect to its sponsoring element, 10, and then its unmoved neighbors are added to the movement candidate lists. In 9c), the next right candidate is moved and its unmoved neighbors are added to the movement candidate lists. This process is continued until all of the right candidates are exhausted. If an element does not violate constraints with respect to its sponsoring neighbor, it is not moved, and its neighbors are not added to the movement candidate lists. This assures that the disturbance is not propagated farther than necessary and increases the speed



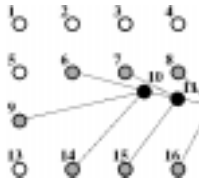
moved	candidates			
	R	L	T	B
10	11	9	6	14

a)



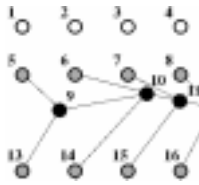
moved	candidates			
	R	L	T	B
10	11	9	6	14
11	12		7	15

b)



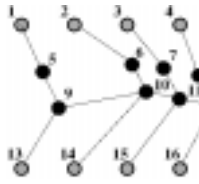
moved	candidates			
	R	L	T	B
10	11	9	6	14
11	12		7	15
12			8	16

c)



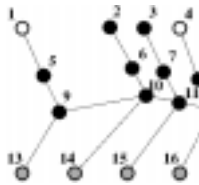
moved	candidates			
	R	L	T	B
10	11	9	6	14
11	12		7	15
12			8	16
9			5	13

d)



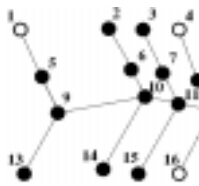
moved	candidates			
	R	L	T	B
10	11	9	6	14
11,12	12		7, 8	15, 16
9			5	13
6, 7, 8			2, 3	
5			1	

e)



moved	candidates			
	R	L	T	B
10	11	9	6	14
11,12	12		7, 8	15, 16
9			5	13
6, 7, 8			2, 3	
5			1	
2, 3				

f)



moved	candidates			
	R	L	T	B
10	11	9	6	14
11,12	12		7, 8	15, 16
9			5	13
6, 7, 8			2, 3	
5			1	
2, 3				
14, 15, 13				

g)

Fig. 9. ChainMail propagated through a 2D object (see text).

of the algorithm.

In 9d), the left list is processed; element 9 is added to the moved list and its unmoved neighbors are added to the movement candidate lists. In 9e) and 9f) the top list is processed and in 9g) the bottom list is processed. Note that elements 1, 4, and 16 are not moved.

Fig. 10 presents pseudocode for the 3D version of ChainMail and Fig. 11 shows two still images from a system with interactive deformation of 3D objects.

```
clear moved list and movement candidate lists
set list indices and counters to zero

move element and add to the moved list
add(top, bottom, left, right, front, back) neighbors of the moved
    element to the (top, bottom, left, right, front, back)
    movement candidate lists

for each element in the front list {
    if (constraints against back neighbor) are violated {
      move element and add to moved list
      add(top, bottom, left, right, front) neighbors of the moved
          element to the (top, bottom, left, right, front)
          movement candidate lists
    }
}
for each element in the back list
    (same as front list but replace front with back)

for each element in the right list {
    if (constraints against left neighbor) are violated {
      move element and add to moved list
      add(top, bottom, right) neighbors of the moved element
          to the (top, bottom, right) movement candidate lists
    }
}
for each element in the left list
    (same as right list but replace right with left)

for each element in the top list {
    if (constraints against bottom neighbor) are violated {
      move element and add to moved list
      add(top) neighbor of the moved element to the
          (top) movement candidate lists
    }
}
for each element in the top list
    (same as top list but replace top with bottom)
```

Fig. 10. Pseudocode for the 3D ChainMail algorithm.

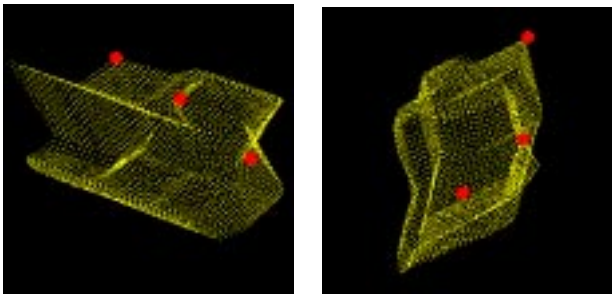


Fig. 11. Two still images from an interactive system in which an object of size 30x30x30 is interactively manipulated and deformed. The object is rendered by drawing surface points using OpenGL.

B.2 Elastic Relaxation

ChainMail produces a deformed shape that satisfies geometric constraints but does not necessarily have an optimal energy configuration. Hence, an elastic relaxation process is applied to locally adjust relative element positions and reduce the system energy. The system energy depends on the distances between object elements. If these distances fall within an optimal range, the system energy is low. When the distances are outside of this range, the system energy is higher. During elastic relaxation, each element position is adjusted sequentially to reduce the system energy in an iterative, closed feedback system. Pseudocode for this process is presented in Fig. 12.

```
for (all non-zero object elements) {
    determine lowest energy position for the element
        relative to its (top, bottom, right, left, front,
        and back) neighbors
    move element towards that position by step size
}
```

Fig. 12. Pseudocode for elastic relaxation.

There are several ways to define an element's optimal position relative to its neighbors for the elastic relaxation step. For example, the optimal position can be simply defined to be the point midway between existing neighbors:

$$(x,y,z)_{opt} = (\frac{1}{N}\sum_{neighbors} x_n, \frac{1}{N}\sum_{neighbors} y_n, \frac{1}{N}\sum_{neighbors} z_n), \quad (1)$$

where N is the number of existing neighbors for the element.

However, this midpoint method causes the object's sides to shrink inwards. Hence, a better expression to calculate the optimal element position is:

$$\begin{aligned}(x,y,z)_{opt} = \quad (\quad & \frac{1}{N}\sum_{neighbors}(x_n - \Delta x_n)_*, \\ & \frac{1}{N}\sum_{neighbors}(y_n - \Delta y_n)_*, \\ & \frac{1}{N}\sum_{neighbors}(z_n - \Delta z_n)_* \quad), \end{aligned} \quad (2)$$

$$\Delta x_n = \begin{cases} -\Delta x & : \quad n = left \\ +\Delta x & : \quad n = right \\ 0 & : \quad \text{all other neighbors} \end{cases}$$

$$\Delta y_n = \begin{cases} -\Delta y & : \quad n = bottom \\ +\Delta y & : \quad n = top \\ 0 & : \quad \text{all other neighbors} \end{cases}$$

$$\Delta z_n = \begin{cases} -\Delta z & : \quad n = back \\ +\Delta z & : \quad n = front \\ 0 & : \quad \text{all other neighbors} \end{cases}$$

where $()_*$ indicates "if the neighbor exists", and Δx, Δy, and Δz are optimal link lengths for left/right, top/bottom, and back/front neighbor pairs. This method is the same as the midpoint method of equation 1 for interior points but prevents surfaces from shrinking excessively.

We have also implemented systems in which two neighboring elements are at equilibrium with respect to each other over a range of link lengths so that elements are only influenced by neighbors if their relative displacement lies outside of this range. Because the object can then reach equilibrium over a range of shapes, these systems behave plasticly.

The material response to deformations also depends on how an element's position is adjusted towards the optimal position. Since the element is moved towards its optimal position in an iterative, closed feedback system, the material behavior is influenced both by the system gain, or the step size towards the optimal position, and the time interval between relaxations. The time constant and damping of the simulated material response can be modified by adjusting the magnitude of the step size in the iterative relaxation and the time interval between relaxations. Experiments in Section IV-C.2 indicate that when the step size is determined as a linear function of the displacement from the optimal position, then the material exhibits a linear stress vs. strain response. When the step size is a non-linear function of displacement from the optimal position, then the material behavior is non-linear. If the step size falls to zero before the optimal position is reached, the material acts as if there is a range of displacements where elements are at equilibrium and the material exhibits plasticity.

C. Experimental Results

ChainMail and an elastic relaxation process have been implemented in C in a number of 2D and 3D systems. Interactive deformation of objects with as many as 50x50x50, or 125,000 elements, have been attained on an SGI Indy. Experiments were performed to test the material behavior of the 2-step process and to determine the speed of the algorithm as a function of the number of elements. Results of these experiments are presented here.

C.1 Timing Experiments

Fig. 13 shows the result of timing tests for ChainMail applied to a 2D system where the number of elements,

the object deformability, and the amount of displacement of the controlled element were varied. The test was applied to objects of the following dimensions: 10x10, 50x50, 100x100, 150x150, 200x200. These results show that the time taken to deform an object will vary at worst linearly with the number of elements in the system. This linearity is achieved because all interactions are local and because elements are moved at most once with each application of both ChainMail and the elastic relaxation process.

In a second experiment, a rigid object, a moderately deformable object (medium), and a very deformable object (loose) were modeled. The medium object allowed link lengths to be compressed to 80% of the optimal length and stretched by a factor of 1.2 from the optimal length. The deformable object allowed link lengths to be compressed to 50% of the optimal length and stretched by a factor of 2.5.

In each experiment, an element located at the center of the object was moved at a constant rate, 100 times around a square path with dimensions: 10x10, 50x50, and 100x100. The figure reports the results of movement around the squares of dimensions 50x50 (small deformation), and 100x100 (large deformation). Rigid objects exhibit the slowest response because all elements in the object are considered and moved for each displacement[4]. However, even in this worst-case scenario, the time required by the ChainMail algorithm increases only linearly with the number of elements in the object. For deformable objects, the response time is better than linear because the deformation often does not have to be propagated to all of the object elements. In a very deformable object, small deformations are propagated to a small number of local neighbors and the deformation is very fast.

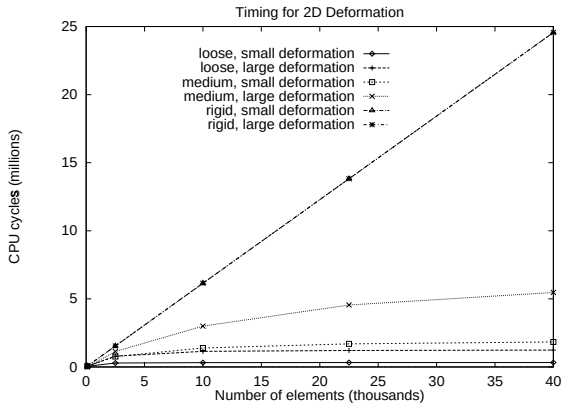


Fig. 13. Results of timing experiments for a 2D system. Objects with 100, 2500, 10000, 22500, and 40000 elements were deformed by moving a control element at the center of the object at a constant rate 100 times around a square of size 50x50 (small deformation) and 100x100 (large deformation). The results for objects with deformability set to rigid, moderately deformable (medium), and very deformable (loose) are presented. The curves for both rigid objects are superimposed. Tests were performed on an SGI Onyx workstation with an R4400 180 MHz processor. On this system, 1 million CPU cycles correspond to approximately 5.6 ms execution time.

C.2 Material Properties

ChainMail and the elastic relaxation process were originally developed for simulating human tissue. Human tissue has a number of complex behaviors that have been reported in the biomechanics literature (e.g. see [50]). One of these behaviors is that the stress in a system loaded at a constant rate of deformation varies non-linearly with the amount of deformation. A second behavior is hysteresis during loading and unloading: the internal forces (or stress) of the material increases at a different rate when a load is applied than it decreases when the load is removed. A third behavior, a process known as relaxation, is that when living tissue is stretched at a constant deformation rate to a given length and then held at that length, the internal stress decreases asymptotically with time towards a lower stress level. In order to test the material properties of the combination of ChainMail and elastic relaxation, a 1D deformable system was implemented. One end of the 1D chain was fixed and the other end could be displaced at a constant rate along the direction of the chain. As described below, the tests show that the combination of ChainMail and elastic relaxation is capable of modeling all three behaviors and hence that it is a good candidate for modeling living tissues.

The combination of ChainMail and elastic relaxation is very general. A variety of functions can be used by the elastic relaxation to determine the internal system forces and to calculate step sizes for local adjustments of element positions. Fig. 14 shows the result of several tests on the 1D chain described above. Two of the curves resulted from applying linear functions in the elastic relaxation to determine internal forces and step sizes and three curves resulted from applying quadratic functions. In practice, a wide variety of analytic and measured behaviors could be attained by using lookup tables to determine forces and step sizes for given link lengths.

The inequality constraints used in the ChainMail algorithm cause objects to behave differently during loading and unloading, resulting in a behavior similar to living tissue. In order to test this hypothesis, ChainMail and a quadratic elastic relaxation function were applied to the 1D chain. The chain was stretched at a constant rate to a given length and then compressed at the same rate while recording the total internal force, measured as a function of link length. As illustrated in Fig. 15, the resultant behavior showed hysteresis in the stress vs. deformation curves that is similar to hysteresis that has been measured in human and animal tissue [50].

The damping that occurs with small step sizes in the elastic relaxation process causes a behavior similar to tissue relaxation as shown in the results in Fig. 16. A 1D chain modeled with ChainMail and elastic relaxation with a quadratic function for calculating internal forces and step sizes was stretched to half of its maximum length and then its endpoints were fixed. The internal forces are plotted as

[4]While it is important for a deformation algorithm to be well behaved for objects that lie in the continuum between rigid and deformable, in practice it is probably not advisable to use a deformable modeling technique to manipulate rigid objects. However, these results for rigid objects can be used to compare this approach with mass-spring methods which do not behave well for stiff objects.

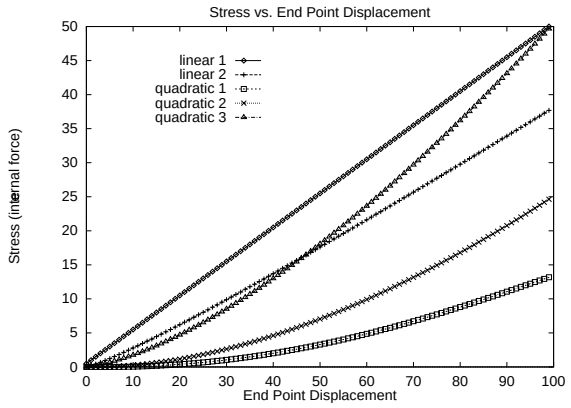


Fig. 14. Measured internal force as a function of deformation for loading at a constant rate. The two linear curves resulted from using a linear function to calculate internal forces and to adjust element positions during elastic relaxation. The three non-linear curves resulted from using a quadratic function to calculate stress and adjust element positions.

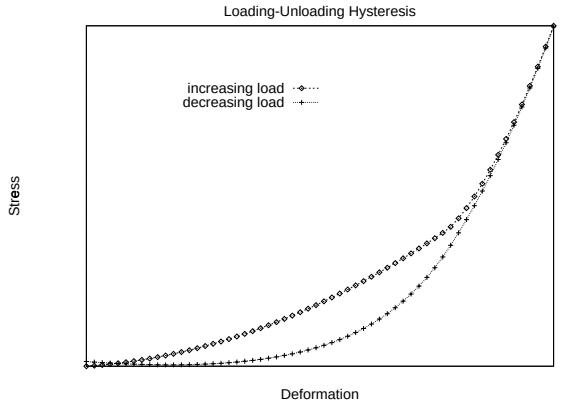


Fig. 15. Stress vs. constant loading and unloading in a 1D chain. The object length is stretched to given point at a constant rate of deformation and then reduced at the same length. As in living tissues, the loading and unloading curves exhibit hysteresis.

a function of time and are shown to decrease asymptotically towards a lower stress level as would be expected in living tissue.

The experiments that have been presented here were designed to test tissue behaviors, rather than to validate the approach for specific materials. For this reason, parameters were not chosen to correspond to the mechanical properties of specific materials or tissues and the measured internal forces are based on heuristic functions (i.e. linear and quadratic curves) rather than known or measured relationships. In general, there are 3 ways to choose material parameters for specific tissues: 1) choose parameters to match the output of this system to predictions of a more rigorous mathematical model such as non-linear, large deformation FEM; 2) choose parameters to match measured tissue responses; and 3) allow surgeons or other specialists to interactively tune the parameters until they are satisfied with the tissue behavior. When the tissue parameters are set with one method, a second method could be used to

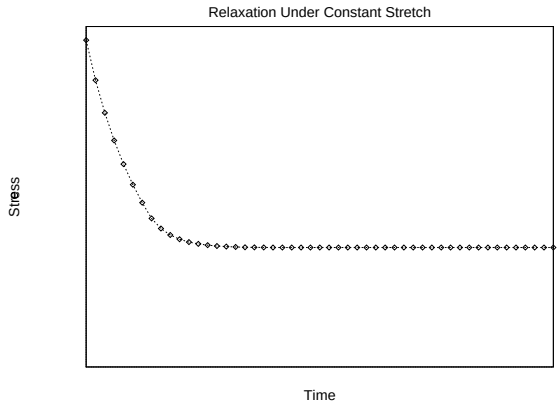


Fig. 16. The result of an experiment in which a 1D chain was stretched to half its maximum length, the length was fixed, and the stress was measured as a function of time. Like living tissue, the material exhibits relaxation: the stress decreases asmyptotically with time towards a lower value.

validate the tissue model as long as the parameter setting technique and the validation technique were independent.

D. Limitations and Extensions to ChainMail

While the algorithm presented in Fig. 10 only handles convex objects, the extension for non-convex objects is relatively simple. As each candidate list is processed, neighbors (other than the sponsoring neighbor) that are not normally added to movement candidate lists are checked to see whether their expected sponsoring element is absent. If so, the neighbor is added to its respective movement candidate list. This is illustrated for an element of the right candidate list in the pseudocode of Fig. 17.

```
for each element in the right list {
    if (constraints against left neighbor) are violated {
        move element
        add element to moved list
        add(top, bottom, right) neighbors
            of the moved element to the
            (top, bottom, right) movement candidate lists

        if (front neighbor exists and front-left neighbor does not exist)
            add front neighbor to front movement candidate list

        if (back neighbor exists and back-left neighbor does not exist)
            add back neighbor to back movement candidate list
    }
}
```

Fig. 17. Pseudocode for a right candidate in an extention of the 3D ChainMail algorithm that models non-convex objects.

The deformation approach presented here has a number of limitations. For example, it does not model volume preservation, a property of many materials and most living tissue. We are investigating a number of ways to add volume preservation, including adding diagonal links between elements, dynamically adjusting ChainMail limits on vertical link lengths based on the stretch of horizontal lengths (or vice-a-versa), and modeling volume preservation in the elastic relaxation step.

A second limitation of the algorithm presented here is that it only models homogeneous materials and a single control point. Both of these issues have been addressed by Schill and Gibson in [51] by ordering lists of movement candidates at each step in the ChainMail algorithm so that the elements with the largest violations on their link lengths are considered first. However, while this approach results in consistent material behavior for heterogeneous materials and for multiple contact or control points, the need to maintain an ordered list of movement candidates makes this extended ChainMail approach significantly slower than the original algorithm. We are continuing to explore these issues.

V. Modifying Object Topology: Cutting, Tearing, Carving, and Joining Volumes

A. Background

Modeling the cutting or tearing of objects with complex structure is a challenging problem for surface-based object models, since object cutting requires the generation of new object surfaces along the cutting path. The problem of clipping a surface-based or geometric object model by an arbitrary 2D plane has been addressed in constructive solid geometry (CSG) (e.g. [52]) and polygon rendering. However, cutting along an arbitrary curved path remains a challenge for surface-based objects. Both determining the intersection of the cutting path with the object and constructing the new cut surface are difficult problems [53]. Related work in CSG provides mathematical techniques for building new surfaces of intersecting solids [54], [55]. However, these methods require the construction of a surface or solid representing the knife path which limits interactivity. In addition, when the cut is made through a surface-based object model that does not contain information about interior structure, then colors, texture, and other features of the cut surface must be fabricated in order to make the cut look realistic [5]. In contrast to the complexity of cutting through surface-based representations, it is relatively straight forward to cut through a linked volumetric object. In addition, interior elements in the volumetric object can be used both to influence the cut path (for example by providing variable resistance to cutting) and to determine the appearance of the new cut surface.

A number of researchers have investigated the sculpting of volumetric objects. Galyean and Hughes [4] represent object material as a 3D array of discrete element values between zero and one, where 1 represents the presence of solid material, 0 represents the absence of material, and values between 0 and 1 represent either partial volume effects or reduced material density. Additive sculpting tools increase element values up to a value of 1 and subtractive tools reduce element values. Element values of the object and tools are filtered with a low-pass filter to reduce aliasing at edges and surfaces. The marching cubes algorithm [56] is used to generate a polygonal surface model for object manipulation and visualization. Wang and Kaufman [5] use a similar sculpting technique but use Volume Rendering for visualization. Avila and Sobierajski [6] added a haptic input device to provide force feedback during sculpting and allow tools to modify 3 different values for each element: material color, density, and an index for material classification. Simulations of NC milling have used similar techniques to model the removal of material by a milling machine (e.g. see [57], [58]).

While these volumetric sculpting techniques have potential applications in volume editing and geometric design, the resultant array of intensity values lack the connectivity required to model the separation of an object into distinct pieces. Pieces cut away from the rest of the volume can not be manipulated as individual objects. In addition, these approaches are not easily extended for sculpting deformable materials.

Bro-Nielsen and Cotin advocate the use of FEM models because in theory they could be used to simulate physically realistic cutting and tearing behavior [49]. Terzopoulos and Fleischer use an FEM object representation to model fracturing or tearing in an off-line simulation of a relatively small 2D deformable mesh in [16]. When stresses at a given node exceed some limit, they zero material property weighting functions at the mesh node to produce a discontinuity that results in fracture behavior. However, accurate modeling of arbitrary cutting through an FEM model would require reworking the equations governing the system, repeating pre-processing steps, and remeshing the model with each intervention in order to provide a higher resolution mesh at high stress points (such as at the knife tip).

B. Carving, Cutting and Tearing

Linked volumes are well suited to interactive modification of object topology. They can be cut, carved and torn by removing elements or by breaking the links between elements. Because linked volumes can have more elements than FEM or mass-spring models, these topological changes can be made at higher resolutions.

During carving, a tool is moved through the occupancy map so that elements encountered by the carving tool can be detected. The encountered elements are then removed and links to the removed elements are deleted from remaining neighbors. In cutting, the tool can remove links as well as elements that are encountered along the path of the tool as it passes through the object, as shown in Fig. 18. Intersections between the knife path and element links are detected by moving the knife volume through the occupancy map and checking for collisions. The occupancy map must be modified slightly so that cells can contain element pointers for cells occupied by an element and pairs of element pointers for cells between linked elements that do not fall into adjacent occupancy map cells. We have used a Bresenham-based line drawing algorithm in a 2D cutting system to determine which occupancy map cells

[5] One way to texture the new surface, when the model is based on 3D image data, is to map the volumetric data onto the cut surface, although this can be costly when the cut path is not planar. Note that this method actually uses a hybrid object model consisting of both the surface model and the volumetric image data.

should be filled with pointers to two elements. If the knife path encounters a cell occupied by an element, the element is removed and appropriate neighbor connections are removed. If the knife passes through a cell indicating a link between two elements, the connection between those two elements is removed in both elements. This algorithm is illustrated in Fig. 19.

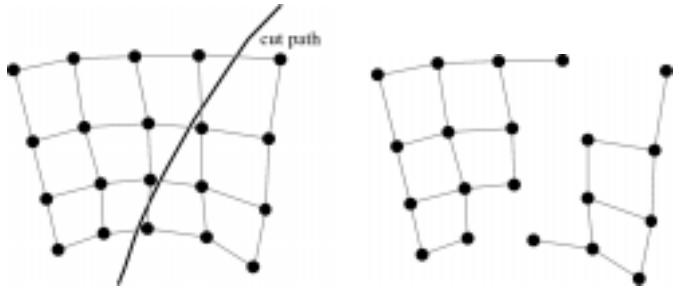


Fig. 18. When cutting volumetric objects, links between elements that lie along the cutting path are removed.

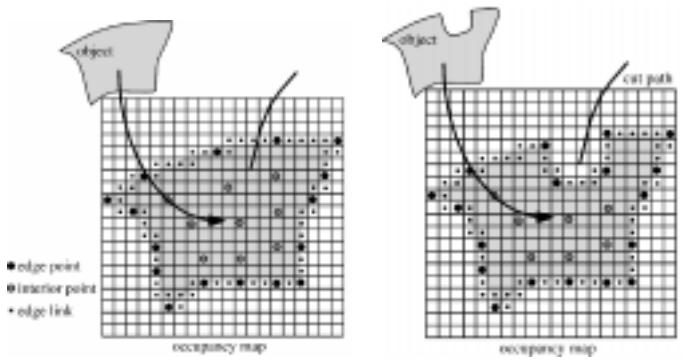


Fig. 19. To model tissue cutting, both object elements and the links between elements are mapped into the occupancy map. When the cut path intersects a cell containing an element, the element and its links to neighbors are removed. When the cut path intersects a link between two elements, then the connection between the two elements is removed. In practice, only links between elements on the object surface need to be mapped into the occupancy map as shown here.

Tearing occurs when the distance between two elements is stretched beyond an allowable limit, for example, when two parts of an object are pulled in opposite directions. When a limit violation between two elements cannot be resolved by moving neighboring elements, the connection between the elements is broken.

C. Joining Object Volumes

For joining objects together, occupancy map cells in the vicinity of the joining tool are searched for object elements that have missing neighbor connections. When such an element is encountered, the occupancy map is searched in a local neighborhood for an element that is missing the complementary neighbor. For example, if a element with a missing right neighbor is encountered in the joining tool path, a region around the element is searched for an element with a missing left neighbor. When complementary neighbors are found, they are then joined by creating the appropriate link. By spiraling outwards from the first element and chosing the first appropriate neighbor, the closest neighbor is selected for joining. The region that is searched for a complementary neighbor is limited by the geometric constraints of ChainMail. This process is illustrated in Fig. 20.

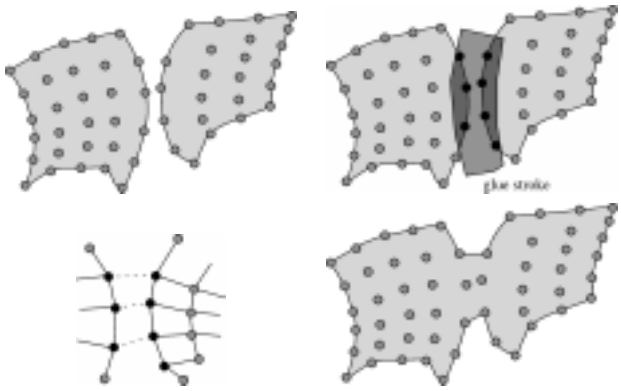


Fig. 20. To join two elements together, occupancy map cells in an area or volume surrounding the joining instrument are searched for elements with missing neighbors. Those elements that have corresponding missing elements are paired. For example, an element missing a right neighbor is paired with an element missing a left element. Finally, paired elements are joined by setting the appropriate neighbor links.

D. Systems for Modifying Object Topology

We have implemented two prototype systems to test the modification of object topology using linked volumes. In the first system, shown in Fig. 21, 2D linked objects can be moved (resulting in object translation), cut, grasped and moved (resulting in object deformation), tacked into place, glued together, and erased interactively using the computer mouse. Pointing and clicking on buttons from a palette switches between these modes. The system was implemented in C, Tcl/Tk, and OpenGL and runs on an SGI platform.

In the second system, we have implemented an interactive 3D system which incorporates the collision detection, deformation, and carving algorithms described in this paper. Because fast volume rendering of deformed volumes is not yet attainable, the system uses OpenGL to render object surface points or polygons that are generated on the fly from the linked volume. Fig. 22 shows several frames from this system.

VI. Discussion and Future Work

Volumetric methods provide a powerful tool for modeling, visualizing, and interacting with graphical objects. In fact, when internal object structure is important for the appearance or behavior of a graphical object, a volume representation is necessary. This paper has presented a linked volume structure and algorithms for modeling interactions between objects such as: collision detection, object defor-

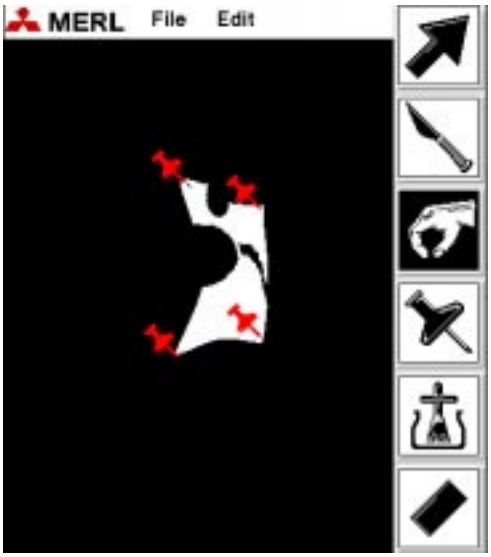


Fig. 21. A 2D system where objects consist of an array of linked elements. By selecting a tool from a palette and manipulating the tool with the computer mouse, objects can be interactively moved, cut arbitrarily, deformed, tacked into place, glued together, and erased.

Fig. 22. Three still images from an interactive prototype system which models 3D object collisions, deformation, and carving. This object has size 20x20x20 and was rendered by drawing triangles on the surface of the linked volume using OpenGL. New triangles are generated interactively during object carving using the link structure.

mation, and the arbitrary carving, cutting and joining of objects, many of which are particularly difficult to model with surface-based graphics. These algorithms have been implemented in C and have been shown to be effective and interactive for reasonably sized volumetric objects.

The results of a number of experiments for testing timing and the material properties resulting from these algorithms have been presented. For example, it has been shown that the ChainMail and elastic relaxation algorithms result in deformation where: the system dynamics exhibit inertial and damping behavior; tissue characteristics can range from rigid to deformable, and elastic to plastic; tissues with non linear stress-deformation curves can be modeled; and, similar to living tissues, materials can exhibit relaxation and hysteresis between loading and unloading.

Although the work that has been presented in this paper has demonstrated the potential and the feasibility of linked volumes, there are many areas that require further investigation. Some examples of these areas are listed here. First, as discussed above, extensions and improvements can be made to the collision detection algorithm and the deformation approach. In addition, better data structures and memory access will improve the efficiency of these methods in modeling, visualization, and haptics. Second, collision response and the modeling of multiple interactions between deformable objects has yet to be carefully investigated. Third, an important step for practical applications will be the experimental verification of simulated material properties by comparing predicted results to experimental data and the output of a rigorous mathematical computation as discussed in IV-D. Finally, hybrid graphics systems that combine both surface-based and volumetric models in rendering and physical modeling are needed so that objects can be represented in an optimal format.

Acknowledgments

Thanks are extended to E. Gibson, H. Lauer, B. Mirtich, R. Perry and several anonymous reviewers for helpful discussions and editorial comments.

References

[1] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1989, Proceedings of SIGGRAPH'89, pp. 19–28.

[2] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1995, pp. 181–188.

[3] M. Lin, *Efficient collision detection for animation and robotics*, Ph.D. thesis, Dept. Electrical Engineering and Computer Science, U. California, Berkeley, 1993.

[4] T. Galyean and J. Hughes, "Sculpting: an interactive volumetric modeling technique," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1991, Proceedings of SIGGRAPH'91, pp. 267–274.

[5] S. Wang and A. Kaufman, "Volume sculpting," in *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1995, pp. 151–156.

[6] R. Avila and L. Sobierajski, "A haptic interaction method for volume visualization," in *IEEE Visualization'96*, R. Yagel and G. Nielson, Eds., 1996, pp. 197–204.

[7] S. Gibson, "Beyond volume rendering: visualization, haptic exploration, and physical modeling of element-based objects," in *Visualization in Scientific Computing*, R. Scateni, J. van Wijk, and P. Zanarini, Eds., pp. 10–24. Springer-Verlag, 1995.

[8] T. He and A. Kaufman, "Collision detection for volumetric models," in *IEEE Visualization'97*, 1997, pp. 27–34.

[9] Y. Kurzion and R. Yagel, "Space deformation using ray deflectors," in *6th Eurographics Workshop on Rendering*, Dublin, Ireland, 1995, pp. 21–32.

[10] M. Desbrun and M-P Gascuel, "Animating soft substances with implicit surfaces," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1995, Proceedings of SIGGRAPH'95, pp. 287– 290.

[11] S. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," Tech. Rep. TR97-19, MERL - A Mitsubishi Electric Research Lab, 1997.

[12] S. Cotin, H. Delingette, N. Ayache, J.M. Clement, V. Tassetti, and J. Marescaux, "Geometrical and physical representations for a simulator of hepatic surgery," in *Medicine Meets Virtual Reality IV*, 1996.

[13] S. Cotin, *Modeles anatomiques deformables en temps-reel*, Ph.D. thesis, INRIA Sophia Antipolis / University de Nice, 1997.

[14] S. Cotin, H. Delingette, and N. Ayache, "Real-time elastic deformations of soft tissues for surgery simulation," *IEEE Transactions on Visualization and Computer Graphics*, 1999, in press.

[15] M. Bro-Nielsen, "Fast finite elements for surgery simulation," in *Medicine Meets Virtual Reality V*, 1997.

[16] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: viscoelasticity, plasticity, fracture," *Computer Graphics*, vol. 22, no. 4, pp. 269–278, 1988.

[17] M. Bro-Nielsen, "Modelling elasticity in solids using active cubes - application to simulated operations," in *Computer Vision, Virtual Reality in Medicine*, 1995, pp. 535–541.

[18] R. Osborne, H. Pfister, H. Lauer, N. McKenzie, S. Gibson, W. Hiatt, and T. Ohkami, "Em-cube: an architecture for low-cost real-time volume rendering," in *SIGGRAPH/Eurographics Workshop on Graphics and Hardware*, 1997, pp. 131–138.

[19] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," *IEEE Computer*, vol. 23, no. 7, pp. 51–64, 1993.

[20] A. Kaufman, "Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1987, Proceedings of SIGGRAPH'87, pp. 171–179.

[21] A. Kaufman, *Volume Visualization*, IEEE Computer Society Press, Los Alamitos CA, 1991.

[22] L. Sobierajski and A. Kaufman, "Volumetric ray tracing," in *Volume Visualization Symposium*, Washington, DC, 1994, pp. 11–18.

[23] L. Sobierajski and A. Kaufman, "Volumetric radiosity," Tech. Rep. Technical Report 94.01.05, Dept. Computer Science, SUNY Stony Brook, 1994.

[24] A. Mor, S. Gibson, and J. Samosky, "Interacting with 3-dimensional medical data: haptic feedback for surgical simulation," in *Phantom Usergroup Workshop*, 1996.

[25] S. Gibson, "3d chainmail: a fast algorithm for deforming volumetric objects," in *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1997, pp. 149–154.

[26] S. Gibson, C.Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, N. McKenzie, A. Mor, S. Nakajima, T. Ohkami, R. Osborne, J. Samosky, and A. Sawada, "Volumetric object modeling for surgical simulation," *Medical Image Analysis*, vol. 2, no. 2, 1998.

[27] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi, "Automatic motion synthesis for 3d mass-spring models," in *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1995, pp. 189–196.

[28] Shinya and Forgue, "Interference detection through rasterization," *J. Visualization and Computer Animation*, vol. 4, no. 2, pp. 132–134, 1991.

[29] T. Uchiki, T. Ohashi, and M. Tokoro, "Collision detection in motion simulation," *Computers and Graphics*, vol. 7, pp. 285–293, 1983.

[30] N. Greene, "Voxel space automata: modeling with stochastic growth processes in voxel space," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1989, Proceedings of SIGGRAPH'89, pp. 175–184.

[31] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg, "Real-time robot motion planning using rasterization computer graphics hardware," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1990, Proceedings of SIGGRAPH'90, pp. 327–335.

[32] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1987, Proceedings of SIGGRAPH'87, pp. 205–214.

[33] D. Baraff and A. Witkin, "Dynamic simulation of non-penetrating flexible bodies," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1992, Proceedings of SIGGRAPH'92, pp. 303–308.

[34] S. Gibson, "Using distance maps for accurate surface representation in sampled volumes," in *IEEE Visualization'98*, 1998.

[35] D. Terzopoulos and K. Waters, "Physically-based facial modeling, analysis, and animation," *J. Visualization and Computer Animation*, vol. 1, pp. 73–80, 1990.

[36] K. Waters, "A muscle model for animating three-dimensional facial expression," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1987, Proceedings of SIGGRAPH'87, pp. 17–24.

[37] Y. Lee, D. Terzopoulos, , and K. Waters, "Realistic modeling for facial animation," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1995, Proceedings of SIGGRAPH'95, pp. 55–62.

[38] R. Koch, M. Gross, F. Carls, D. von Buren, G. Fankhauser, and Y. Parish, "Simulating facial surgery using finite element models," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1996, Proceedings of SIGGRAPH'96, pp. 421–428.

[39] X. Tu and D. Terzopoulos, "Artificial fishes: physics, locomotion, perception, behavior," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1994, Proceedings of SIGGRAPH'94, pp. 43–50.

[40] J. Christensen, J. Marks, and T. Ngo, "Automatic motion synthesis for 3d mass-spring models," *The Visual Computer*, vol. 13, pp. 20–28, 1987.

[41] M. Carignan, Y. Yang, N. Magnenat Thalmann, and D. Thalmann, "Dressing animated synthetic actors with complex deformable clothes," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1992, Proceedings of SIGGRAPH'92, pp. 99–104.

[42] D. Baraff and A. Witkin, "Large steps in cloth animation," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1998, Proceedings of SIGGRAPH'92, pp. 43–54.

[43] J. Collier, B. Collier, G. O'Toole, and S. Sargand, "Drape prediction by means of finite element analysis," *J. of the Textile Institute*, vol. 82, pp. 96–107, 1991.

[44] D. Chen, *Pump it up: computer animation of a biomechanically based model of muscle using the finite element method*, Ph.D. thesis, Media Arts and Sciences, MIT, 1991.

[45] I. Essa, S. Scarloff, , and A. Pentland, "A unified approach for physical and geometric modeling for graphics and animation," in *Eurographics'92*, 1992, vol. 11, pp. 129–138.

[46] 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.

[47] E. Keeve, S. Girod, P. Pfeifle, and B. Girod, "Anatomy-based facial tissue modeling using the finite element method," in *IEEE Visualization'96*, 1996, pp. 21–28.

[48] A. Pentland and J. Williams, "Good vibrations: modal dynamics for graphics and animation," *Computer Graphics*, vol. 23, no. 3, pp. 215–222, 1989.

[49] M. Bro-Nielsen and S. Cotin, "Real-time volumetric deformable models for surgery simulation using finite elements and condensation," in *Eurographics'96*, 1996, vol. 15, pp. 57–66.

[50] Y. Fung, *Biomechanics: mechanical properties of living tissues*, Springer-Verlag, New York, 1993.

[51] M. Schill and S. Gibson, "Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm," in *Proc. Medical Image Computation and Computer Integrated Surgery (MICAI'98)*. October 1998, Springer.

[52] P-G. Maillot, "Three-dimensional homogeneous clipping of triangle strips," in *Graphics Gems*, J. Arvo, Ed. AP Professional, New York, 1991.

[53] M. Bro-Nielsen, D. Helfrick, B. Glass, X. Zeng, and H. Connacher, "Vr simulation of abdominal trauma surgery," in *Medicine Meets Virtual Reality VI*, 1998.

[54] G. Kriezis, N. Patrikalakis, and F. Wolter, "Topological and differential equation methods for surface intersections," *Computer-Aided Design*, vol. 24, pp. 41–55, 1990.

[55] S. Krishnan and D. Manocha, "An efficient surface intersection algorithm based on the lower dimensional formulation," *ACM Transactions on Computer Graphics*, vol. 16, no. 1, pp. 74–106, 1997.

[56] W. Lorensen and H. Cline, "Marching cubes: a high resolution 3d surface construction algorithm," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1989, Proceedings of SIGGRAPH'89, pp. 163–169.

[57] T. van Hook, "Real-time shaded milling display," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1986, Proceedings of SIGGRAPH'86, pp. 15–20.

[58] Y. Huang and J. Oliver, "Nc milling error assessment and tool path correction," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1994, Proceedings of SIGGRAPH'94, pp. 287–294.

[59] B. Naylor, "Sculpt: an interactive solid modeling tool," in *Graphics Interface'90*, 1990, pp. 138–148.

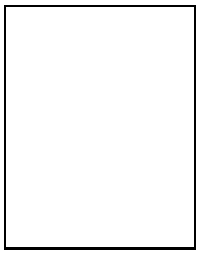
[60] D. Terzopoulos, J. Platt, and K. Fleischer, "Heating and melting deformable models (from goop to glop)," in *Graphics Interface'89*, 1989, pp. 219–226.

[61] S. Wang and A. Kaufman, "Volume sampled elementization of geometric primitives," in *Visualization'93*, 1993, pp. 78–84.

[62] P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transform," in *Computer Graphics Proceedings, Annual Conference Series*. ACM SIGGRAPH, 1994, Proceedings of SIGGRAPH'94, pp. 451–457.

[63] S. Cotin, H. Delingette, and N. Ayache, "Efficient linear elas-

tic models of soft tissues for surgery simulation," Tech. Rep., INRIA, 1998, 3510.



Sarah Gibson received a B.Eng. in Mathematics and Engineering at Queen's University, Canada, in 1985, an M.Sc. in Electrical and Computer Engineering at the University of Wisconsin, Madison, in 1986, and a Ph.D. in Electrical and Computer Engineering at Carnegie Mellon University in 1990. She was a postdoctoral fellow at the Neuman Biomechanics Lab at MIT from 1991-1992, a member of the research faculty at the Robotics Institute at Carnegie Mellon University from 1992-1993, and is currently a Senior Research Scientist at MERL- A Mitsubishi Electric Research Laboratory. Dr. Gibson's interests lie in volume graphics and medical applications, in the areas of volume visualization, physics-based modeling, surgical simulation, and image segmentation.