Techniques for Visualizing 3-Dimensional Manifolds

Michael J. Laszlo
University of Illinois at Chicago
Department of Electrical Engineering and Computer Science

June 1990

Abstract

Computer graphics has long been concerned with representing and displaying surfaces in 3-dimensional space $\Re^3$. We address the questions of representation and display in a higher dimensional setting, specifically, that of 3-manifolds immersed in $\Re^4$. We describe techniques for visualizing the cross-section surfaces of a 3-manifold formed by a cutting hyperplane. The manifold is first triangulated, so that the cross-section may be computed on a per tetrahedron basis. The triangulated manifold is stored in a data structure which efficiently supports calculation of curvature. We expect the techniques, which we have implemented on the Personal IRIS, will support a highly-interactive system when implemented on a fine-grain parallel computer such as the Connection Machine.

1 Introduction

Computer graphics has long been concerned with representing and displaying surfaces in 3-dimensional space $\Re^3$. We address the questions of representation and display in a higher dimensional setting. Three-manifolds, the 3-dimensional version of a surface, are treated by the mathematical branches of topology and geometry. They arise in various scientific disciplines, such as cosmology ("What is the global structure of the universe?").

One may attempt to grasp a 3-manifold in spatial terms, imagining how it curves in n-space or turns back upon itself, yet usually these objects prove difficult to picture. The basic difficulty in visualizing a 3-manifold arises from the fact that mapping a higher-dimensional space to a 2-dimensional screen (or retina) inevitably results in loss of information. To overcome this, one constructs multiple views of a given manifold, usually in the form of a set of still pictures, or an animation, or via an interactive system. Interactively manipulating views of a manifold is probably the best way to appreciate it visually, since the approach corresponds to the way in which we understand an ordinary object of the real world by moving around it or handling it. Unfortunately, the computing demands of a highly interactive system requires compromises, ranging from the use of fairly primitive display techniques, to severe restrictions on the type of manifold that can be viewed.

We describe a technique for constructing and interactively visualizing a 3-manifold $\mathcal{M}$. A view of $\mathcal{M}$ is given by a cross-section, the intersection of $\mathcal{M}$ by some cutting hyperplane. Manifold $\mathcal{M}$ is represented by a complex of tetrahedra, and the cross-section surface(s) is computed on a per tetrahedron basis. The data structure used for the complex of tetrahedra efficiently supports the queries needed to calculate the cross-section surface, as well as curvature of the surface and the manifold itself for the purpose of shading and coloring. We have implemented our techniques on a Silicon Graphics Personal IRIS. Furthermore, these techniques readily parallelize. We are presently implementing them on a Connection Machine, whose fine-grain parallelism is expected to support a highly-interactive system.

In this paper, after first presenting background material, we discuss how a 3-manifold is described, and how

---

Author presently at Department of Mathematics and Computer Science, University of Miami, FL.
This work is supported in part by the National Science Foundation under Grant CCR-8908933.

CH2913-2/90/0000/0342/$01.00 – 1990 IEEE

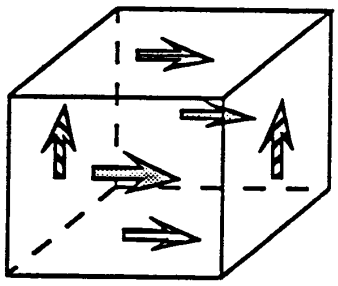

Figure 1: Polyhedral model for the 3-torus $S^1 \times S^1 \times S^1$.

it is decomposed into tetrahedra. We then describe the data structure used to represent the tetrahedrized manifold, and how it supports visualization. We go on to present an example—visualization of a solid torus—and finally conclude with some observations and directions for future research.

2 Background

2.1 Definitions

A *3-manifold* $\mathcal{M}$ is the 3-dimensional version of a surface. Its local topology is that of 3-space $\Re^3$; every neighborhood of arbitrary point $p \in \mathcal{M}$ contains an open set U ($p \in U$) homeomorphic to the open solid ball. Since we will be treating only 3-dimensional manifolds, in this paper we refer to them simply as *manifolds*.

Many manifolds $\mathcal{M}$ can be described by means of a *polyhedral model* P, which consists of a solid polyhedron P in which pairs of facets are identified. The manifold is the *quotient space* formed by P modulo the relation indicated by the facet identifications. An example is depicted by Figure 1. Pairs of facets with like-shaded arrows are identified in such a way that the two arrows coincide. One can imagine (with some difficulty perhaps) deforming P so that all pairs of identified facets may be glued together, "twisting" P as necessary so that the facets mate properly.

A *triangulation* of P is a decomposition $\mathcal{K}$ of P into tetrahedra. $\mathcal{K}$ is a finite set of tetrahedra whose union is P, and for which, if $s \cap t \neq \emptyset$ for $s, t \in \mathcal{K}$, then $s \cap t$ is a vertex, edge or facet of both s and t.

An *immersion* of P in $\Re^4$ is a locally one-to-one, continuous mapping $\phi : P \rightarrow \Re^4$. The immersed manifold may have self-intersection points, points which have at least two pre-images. In section 6, we work with an immersion of the solid 3-torus in $\Re^4$ (see Figure 1). Our particular example is actually an *embedding* since no self-intersection actually occurs.

A *hyperplane* (with respect to $\Re^4$) is a 3-dimensional affine space. Given hyperplane $H \subset \Re^n$, a *cross-section* of imbedded manifold $\mathcal{M}$ is the space $\mathcal{M}_H = \mathcal{M} \cap H$. If non-empty, $\mathcal{M}_H$ is a not-necessarily-connected surface. H is called the *cutting hyperplane*. The notion analogizes that of a plane which slices through a surface in $\Re^3$; the cross-section (the intersection of plane and surface) is a set of zero or more plane curves.

The reader is referred to [ST, Th, We] for an elementary introduction to these topics. The last of these references provides a remarkably pictorial account of 3-manifolds.

2.2 Previous Research

Projection and cross-section are the two most widely used viewing models for forming an image of a manifold. Projection maps the entire manifold to the image plane (up to visibility); however, many different points of the manifold may map to the same pixel *under distinct projectors*, and it is not clear what should be displayed at such pixels. (To see this, consider projection in a setting one dimension lower—projecting a surface in $\Re^3$ to the real line $\Re$. Given point $p \in \Re$, all points of the surface lying in the plane orthogonal to $\Re$ at p project to point p.) The cross-section, defined in the previous subsection, is a surface which may be examined with conventional 3D computer graphics. However, the cross-section ignores those regions of the manifold which do not intersect the cutting hyperplane. Both viewing models result in loss of information [Ro].

The problem is usually overcome by providing multiple views of the same manifold. Multiple views may consist of a small set of static pictures [KBBL], or an animation in which viewing parameters vary with time [No], or stereo views [BrD], or an interactive system which permits a user to vary viewing parameters interactively. Color is often used to correlate the various views. Distinct features of a manifold, such as the facets of a 4-dimensional polyhedron, may be pre-assigned different colors. Alternatively, color may indicate properties of the manifold (such as curvature) which obtain consistently across the different views.

Much recent work in volume visualization proves applicable to the visualization of manifolds. Given scalar valued mapping F defined over connected region $A \subset \Re^3$, we can define function $F' : A \rightarrow \Re^4$ by $F'(x,y,z) = (x,y,z,w)$ where $w = F(x,y,z)$. The iso-valued surface $\{p \in A \mid F(p) = w_0\}$ is precisely the pre-image of the cross-section of $F'(A)$ by cutting hyperplane $w = w_0$. Calculation and display of iso-valued surfaces is treated in [DCH, Le, Sa, LC]. The last reference describes a technique for constructing a polygonal-mesh representation of an iso-valued surface. We employ a similar technique for building a cross-section surface.

3 Triangulating the Polyhedral Model

Solid polyhedron P is triangulated using a two-stage *site generation* approach. First, a finite set S of points (called *sites*) is generated in the interior and boundary of P. Second, a triangulation is constructed over S—that is, the sites of S serve as vertices of the tetrahedra. The site generation approach supports considerable control over local fineness of the triangulation; the triangulation is finer where there is a high concentration of sites.

3.1 Generating Interior Sites

The sites interior to P are distributed according to curvature of $\phi(\mathcal{M})$, where greater curvature requires greater site concentration.

To generate interior sites, we first sample the curvature of $\mathcal{M} = \phi(P)$ at nodes of a regular lattice superimposed on P. To compute curvature, we let $N : P \rightarrow \Re^n$ be the Gauss map for ϕ, which associates with each $p \in P$ the normal vector $N(p)$ at $\phi(p)$. To obtain the curvature at point $p \in P$, let T_p denote the tangent space to ϕ corresponding to point p. The *Weingarten map* $L_p : T_p \rightarrow T_p$ is a linear map defined by $L_p(d\phi_p(v)) = -(\nabla N_1 \cdot v, \nabla N_2 \cdot v, \cdots, \nabla N_n \cdot v)$, where $v \in \Re^3$ is a velocity vector located at p. Here $N = (N_1, N_2, \cdots, N_n)$ is the normal to ϕ corresponding to p, and ∇N_i is the gradient to $N_i = N_i(p)$. $L_p(v)$ measures the rate of change of normal N as one passes through p with velocity v. Counting multiplicities, there are three eigenvalues of L_p, which correspond to the principal curvatures of ϕ at p. The principal curvature of greatest magnitude corresponds to maximum curvature of ϕ at p, and serves as our sample value at p. The *Gauss-Kronecker curvature* of $\mathcal{M}$ at p is equal to the the product of the principal curvatures at p [Th].

The samples obtained at the nodes of the lattice serve as a discrete density function for curvature. The nodes are "unfolded" into a linear sequence, from which a one-dimensional cumulative distribution table is computed. The table is used to generate sites using the Monte Carlo method. When a node of the lattice is selected, a site is generated at random (uniformly) in the vicinity of that node.

3.2 Generating Boundary Sites

Sites in the boundary of P are selected so that pairs of identified (convex) facets $f, f' \in P$ can be mated. Specifically, the 2-dimensional meshes induced in f and f' by the triangulation of P must be congruent.

Let facet $f = (v_1, v_2, \cdots, v_n)$ and $f' = (v'_1, v'_2, \cdots, v'_n)$, and assume that the facets are aligned so that vertices v_i and v'_i mate, for $i = 1, 2, \cdots, n$. To generate matching sites $s \in f$ and $s' \in f'$, we define s and s' by a convex combination of vertices v_i and v'_i, respectively, using the same coefficients for each. That is, having generated random numbers $r_1, r_2, \cdots, r_n$ for which $\sum r_i = 1$, we define $s = \sum r_i v_i \in f$ and $s' = \sum r_i v'_i \in f'$. Our triangulation algorithm ensures that the resulting meshes in f and f' are congruent.

Sites are also generated at each vertex of P, to ensures that the resulting triangulation covers P. In addition, sites are generated along the edges of P to reduce the number of long "sliver" tetrahedra.

3.3 Building the Triangulation

We construct a *Delaunay triangulation* $\mathcal{T}(S)$ over the site set S. The Delaunay triangulation may be defined in various ways. One definition states that a set of four sites $S' \subset S$ determine a tetrahedron of $\mathcal{T}(S)$ if and only if the unique sphere determined by S' (ie. the *circumsphere* of S') contains no site in its interior. If it is the case that no five sites of S are cospherical, then the Delaunay triangulation is unique [PS].

We use the Delaunay triangulation for several reasons. First, the triangulation is more fine in regions of high site concentration. Second, $\mathcal{T}(S)$ covers P exactly (ie. $\cup_{t \in \mathcal{T}(S)} t = P$) if S contains the extreme points of P. Third, tetrahedra tend to be well-proportioned—the ratio of radii of circumsphere to insphere tends to 3 (the ratio of radii in the regular tetrahedron). Such tetrahedra encourage numerical stability in the hyperplane

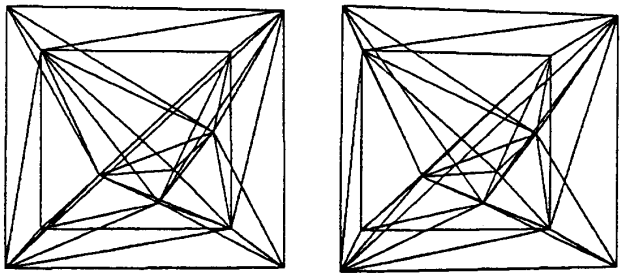

Figure 2: Stereo image of triangulated cube. The site set consists of 12 sites—the 8 extreme vertices of the cube (its corners), and 4 interior sites.

```
delaunay(S)
{
    F ← initial_facet(S);
    while (F ≠ 0) {
        f ←some_element_of(F);
        t ←tetrahedron(f);
        if (t exists) {
            for each facet f' ∈ t
                if (f' ∈ F)
                    delete(f', F);
                else
                    insert(f', F);
        }
    }
}
```

Figure 3: The algorithm *delaunay* for computing triangulation $\mathcal{T}(S)$ of finite site set $S \subset P \subset \Re^3$.

cross-section calculations. Although some long, narrow "slivers" are produced, these tend to be few in number. We hasten to note that this has been our observation, although we know of no formal property possessed by *3-dimensional* Delaunay triangulations in this regard.

A stereo image of a Delaunay triangulation of the cube is presented by Figure 2. To get the 3D effect, hold at arms length, cross your eyes slightly, and focus upon the middle of the three images which form.

We describe the algorithm we use to build a Delauany triangulation over site set S (Figure 3). The algorithm incrementally "grows" a current triangulation, iteratively discovering a new tetrahedron which attaches to the current triangulation. It starts by discovering, and inserting into dictionary $\mathcal{F}$, a facet of the triangulation belonging to the boundary of P. At each step, dictionary $\mathcal{F}$ holds those facets of the current triangulation for which an incident tetrahedron has been sought on exactly one (and not the other) side of the facet. An iteration consists of deleting some facet f from $\mathcal{F}$, then seeking a tetrahedron t incident to f on the side of f not yet examined. If t exists, each of its four facets in turn is deleted from $\mathcal{F}$ if present, otherwise inserted into $\mathcal{F}$. If t does not exist, then facet f lies in the boundary of P and is incident to only one tetrahedron.

In Figure 3, procedure call *tetrahedron(f)* seeks a tetrahedron t_s determined by the three vertices of f and a fourth site s which lies in the side of f not yet examined. That site s is selected for which the radius of the circumsphere of t_s is minimal—this ensures that the interior of the circumsphere contains no sites, so that tetrahedron t_s belongs to the triangulation [AB]. Usually, site s lies near facet f. Our implementation therefore first seeks s in the vicinity of f, then gradually expands the search outward from f if necessary. This is accomplished by a preprocessing stage which divides space into cubes, then drops each site into its appropriate cube. Procedure call *tetrahedron(f)* examines sites in those cubes (roughly) in order of increasing distance from f.

Although the algorithm's worst-case time complexity is $O(|S|^3)$, the use of this search heuristic, together with the fact that the expected size of a triangulation is $O(|S|)$, permits triangulations to be generated over large site sets S fairly quickly. The triangulations with which we regularly work consist on the order of 25000 tetrahedra over about 4200 sites. It is constructed in

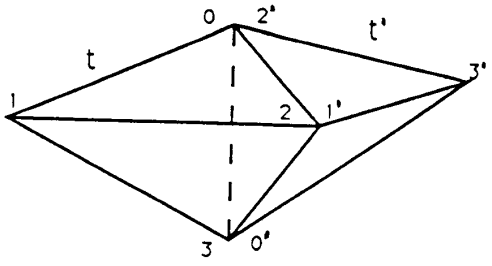

Figure 4: Two tetrahedra t and t' glued at a common facet, with vertex labels. Here $n_t.pmt[1] = (2', 3', 1', 0')$, and $n_{t'}.pmt[3'] = (3, 2, 0, 1)$.

about 12 cpu minutes when an underlying 20^3 grid of cubes is used.

4 The Data Structure

The data structure for the triangulated manifold represents not only the constituent tetrahedra, but also certain relations that hold among them. Specifically, each tetrahedon points to the four tetrahedra adjacent to it. Given an edge e of the triangulation, it is then possible to visit the tetrahedra incident to e in their cyclic order about the edge. This operation is used to smooth-shade cross-section surface $\mathcal{M}_H$. To compute the average normal to a vertex $v \in \mathcal{M}_H$, the tetrahedra incident to the manifold edge to which v belongs can be visited in turn. Each such tetrahedron intersects the cutting hyperplane H in a triangle or quadrilateral, the normal to which contributes to the average normal at v.

The data structure, described by [HW], assigns each tetrahedron t a unique integer identifier, and the four vertices of t distinct labels 0 through 3. The vertex labels are assigned arbitrarily and are local to t—the other tetrahedra sharing a vertex may assign the vertex different labels. Each facet of t is assigned the label of the vertex of t opposite it. Each tetrahedron t is represented by a record n_t, which holds geometrical data (the location of each of t's vertices under the imbedding), as well as topological data which we now describe. Record n_t possesses an array adj of elements $adj[0]$ through $adj[3]$. Element $adj[i]$ contains the identifier of the tetrahedron that shares facet i with t. Record n_t also contains a 4×4 integer array, each row of which represents a permutation of t's facet labels. Imagine tetrahedron t being reflected across the plane containing facet i, so that its facets are made to coincide with those of tetrahedron $adj[i]$. The permutation given by row $pmt[i]$ describes how the facets of t are reflected into the facets of $adj[i]$. Specifically, $pmt[i][j] \equiv k$ when facet j of t coincides with facet k of $adj[i]$ under this reflection across facet i. This is depicted in Figure 4.

```
tnext(α, β)
{
    n ← α.n_t;
    β.n ← n.adj[α.p_3];
    β.p_0 ← n.pmt[α.p_3][α.p_0];
    β.p_1 ← n.pmt[α.p_3][α.p_1];
    β.p_2 ← n.pmt[α.p_3][α.p_3];
    β.p_3 ← n.pmt[α.p_3][α.p_2];
}
```

Figure 5: The procedure *tnext*.

To obtain the cycle of tetrahedra incident to a common edge, we define a tuple (called a *cell-tuple*) $\alpha = (n_t, p_0, p_1, p_2, p_3)$. The cell-tuple consists of a record n_t followed by a permutation of tetrahedron t's vertex labels. It represents the nested set of cells $vertex(\alpha) = p_0$, $edge(\alpha) = p_0p_1$, $facet(\alpha) = p_0p_1p_2$ and $tetra(\alpha) = p_0p_1p_2p_3 = t$. We then define a procedure *tnext* for moving from $t = tetra(\alpha)$ to the tetrahedron around $edge(\alpha)$ which shares $facet(\alpha)$ with t. Procedure *tnext*, when passed cell-tuple α, computes cell-tuple β which represents the next tetrahedron around $edge(\alpha)$; *tnext* may then be applied to β to obtain the *next* tetrahedron about the same edge. The procedure *tnext* is defined in Figure 5.

5 Visualizing the Manifold

5.1 Constructing the Cross-Section

The cross-section is calculated by computing the *cross-section polygon* $t_H = t \cap H$ for each tetrahedron t of (triangulated) manifold $\mathcal{M}$. Polygon t_H is a (possibly degenerate) triangle or quadrilateral. The polygon is

non-empty if and only if at least one of the vertices of t is above or on cutting hyperplane H, and one below. To compute t_H, we assign to each vertex v of t a 1 (0) if v is above or on (below) H. Since each of the tetrahedron's four vertices assumes either of two states, the hyperplane can intersect the tetrahedron in $2^4 = 16$ different ways, ignoring symmetries. The local labels of vertices prescribes how the vertex states are concatenated to form an integer between 0 and 15 inclusive. The integer serves as an index into a table which describes all possible types of intersection. Linear interpolation is used to determine the precise position of the cross-section polygon's vertices. The technique is similar to the "marching cubes" algorithm for constructing iso-valued surfaces from 3-dimensional discrete data [LC].

5.2 Use of Color

Brightness and hue are used to portray distinct properties. Brightness (shading) suggests how cross-section surface $\mathcal{M}_H$ is immersed in the 3-dimensional hyperplane H. The (3-component) normal to cross-section vertex $v \in \mathcal{M}_H$ is obtained by averaging over the normals to the polygons of $\mathcal{M}_H$ which touch v. Since v occurs on some edge of $\mathcal{M}$, procedure *tnext* is employed to visit each tetrahedron incident to the edge. Each such tetrahedron yields a cross-section polygon which touches v, whose normal contributes to the averaged normal at v. Having obtained normals to the vertices of a cross-section polygon of $\mathcal{M}_H$, brightness values are calculated by Lambert's cosine law, and Gouraud shading used to shade the polygon.

The hue of any point $p \in \mathcal{M}_H$ indicates the angle formed between the (4-component) normal to manifold $\mathcal{M}$ (at p) and the (4-component) normal to H. We call this the *orientation angle* of $\mathcal{M}$ with respect to H. To compute hues, we pre-compute the hue at each manifold vertex $v \in \mathcal{M}$. This pre-processing step is done by averaging over the normals to all tetrahedra of $\mathcal{M}$ that touch v, then computing the angle between the averaged normal and the normal to H. (Here we assume all tetrahedra of $\mathcal{M}$ are say right-handed, so that all normals point "outward.") Later, to compute the (4-component) normal to cross-section vertex $v' \in \mathcal{M}_H$, we interpolate between the hues stored at the endpoints of the manifold edge on which v' lies. Having obtained hues at the vertices of a cross-section polygon, the polygon may be Gouraud shaded.

As H is translated in the direction of its normal, cross-section $\mathcal{M}_H$ changes slowly in regions where the orientation angle is approximately $\pi/2$, and rapidly where the orientation angle is close to 0 or to π. $\mathcal{M}_H$ is contracting upon itself in regions where the orientation angle is close to 0, and expanding where the orientation angle is close to π. Hue suggests behavior of the manifold beyond and beneath the current position of the cutting hyperplane. For instance, in highly contractive regions of $\mathcal{M}_H$, the manifold contracts onto itself "just beyond" the current hyperplane. Furthermore, change in hue indicates curvature of the manifold. In regions of $\mathcal{M}_H$ where hue changes rapidly with respect to translation of H, the manifold has high curvature; where hue changes slowly, low curvature.

Although hue and brightness correspond to different properties, use of Gouraud shading to interpolate *both* color components permits the use of fast graphics shading hardware. Note that Gouraud shading is implicitly used to interpolate orientation angle over the entire tetrahedron to which a given cross-section polygon belongs (although the calculation is explicitly carried out only over the polygon).

6 An Example: The Solid Torus

Let us define an immersion $\phi : P \rightarrow \Re^4$ for polyhedral model P (Figure 1), the 3-dimensional torus $S^1 \times S^1 \times S^1$. Where S^1 represents the circle, the 3-torus $\mathcal{M} = \phi(P)$ can be regarded as the product of circle S^1 and 2-torus $S^1 \times S^1$. Let immersion ϕ be given by

$$\begin{aligned} x(\theta,\psi,\alpha) &= \cos\alpha(a + \cos\psi(b + \cos\theta)) \\ y(\theta,\psi,\alpha) &= \sin\psi(b + \cos\theta) \\ z(\theta,\psi,\alpha) &= \sin\theta \\ w(\theta,\psi,\alpha) &= \sin\alpha(a + \cos\psi(b + \cos\theta)) \end{aligned}$$

where $0 \leq \theta, \psi, \alpha < 2\pi$, and $a - 1 > b > 1$ for constants a and b. $\mathcal{M}$ is a circle of 2-tori $\mathcal{M}(\alpha)$ around the yz-plane. Let the x_α-axis be normal to the yz-plane, and form with the x-axis an angle of α radians. Each $\mathcal{M}(\alpha)$ lies in $x_\alpha yz$-space, and occurs as an annulus in Figure 6a, and as a 2-torus in Figure 6b.

We can get an idea of the cross-section surface $\mathcal{M}_H = \mathcal{M} \cap H$ by considering how cutting hyperplane H intersects the 2-tori $\mathcal{M}(\alpha)$. Denote by H_{w_0} the cutting hyperplane of the form $w = w_0$, a translate of xyz-space. Figure 6c depicts the intersection of H_{w_0} with the 2-tori $\mathcal{M}(\alpha_0)$ and $\mathcal{M}(\alpha_1)$. Each intersection

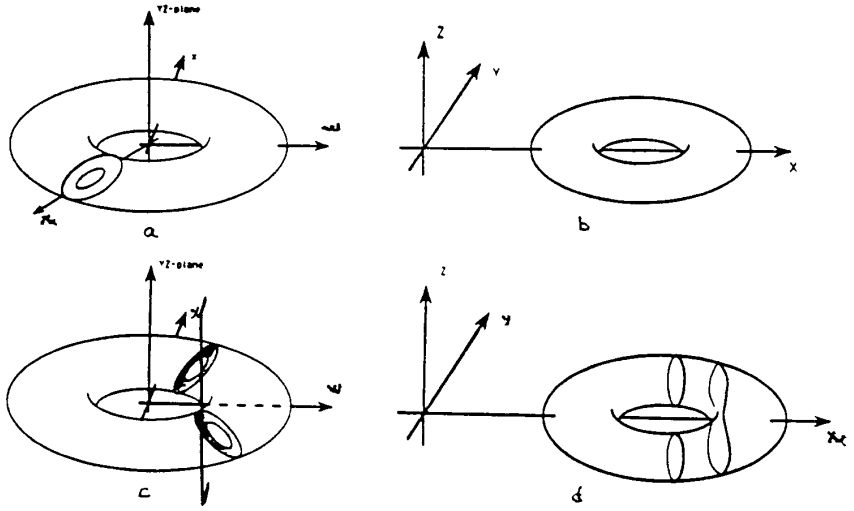

Figure 6: (a) Solid torus $\mathcal{M}$. (b) $\mathcal{M}_\alpha$ in x_α-space. (c) The intersection of H_{w_0} with 2-tori $\mathcal{M}(\alpha_0)$ and $\mathcal{M}(\alpha_1)$. (d) Curves C_0 and C_1 in $x_{\alpha_i}yz$-space, corresponding to the two 2-tori of Figure 6c.

$C_{\alpha_i} = \mathcal{M}(\alpha_i) \cap H_{w_0}$ is a set of zero or more curves in $x_{\alpha_i}yz$-space. In figure 6d, which depicts each $\mathcal{M}(\alpha)$ in a common $x_\alpha yz$-space, C_{α_0} consists of two curves while C_{α_1} is a single curve. As α increases from 0 to 2π, these curves may be "stacked" to form the cross-section surface $\mathcal{M} \cap H_{w_0}$. To stack the curves, one imagines curve(s) C_{α_0} being continuously (and uniformly) deformed across $\mathcal{M}(\alpha)$ until it coincides with C_{α_1}. The cross-section surface resulting from figure 6 is a 2-torus with two handles, pictured in Plate 4.

Plates 1 through 8 display a sequence cross-sections of 3-torus $\mathcal{M}$ as w_0 increases from 0.1 (Plate 1) to 6.6 (Plate 8). The constants $a = 4$ and $b = 2$ are used. The cutting hyperplanes H_{w_0} are all of the form $w = w_0$ for various constants w_0. Stable regions of the cross-section surface (orientation angle $\approx \pi/2$) are colored blue, contractive regions range from blue to magenta to red (as the orientation angle varies from $\pi/2$ to 0), and expansive regions range from blue to cyan to green (as the orientation angle varies from $\pi/2$ to π). The triangulation consists of 39,890 tetrahedra over 6500 sites. Plate 4, in which the cutting hyperplane intersects the most number of tetrahedra (7075), took 18 cpu seconds to produce on a personal IRIS. The other plates took less time to produce; Plate 8, which represents the least number of intersections (438), required about 1 cpu second.

7 Future Research and Conclusion

We are presently implementing these techniques on the Connection Machine, in order to achieve a highly-interactive system. The pre-processing phases (which culminate in the data structure of Section 4) are performed on a sequential computer. The data structure is then downloaded to the Connection Machine. Each tetrahedron record is assigned a virtual processor, capable of computing its cross-section polygon formed by intersection with the current cutting hyperplane. Hue and brightness calculations are achieved by routing messages among the tetrahedra. A cell-tuple serves as the address of a message, and procedure *tnext* is used to compute the address to which a recieved message is to be forwarded.

An important question addresses our choice of the Delaunay triangulation for decomposing the polyhedral model. An imbedded 3-manifold having high curvature (or change in curvature) generally requires a large triangulation to be well-approximated. The data structure might then use excessive memory. In such cases, the Delaunay triangulation might best be replaced by a regular, semi-regular or hierarchical decomposition which can be represented more concisely. What is a good decomposition of space, which uses relatively little memory yet also satisfies the desirable properties (discussed in Subsection 3.3) possessed by the Delaunay triangulation?

Another question concerns the behavior of these techniques in the vicinity of a self-intersection. In such regions, the index of a tetrahedron need not correctly indicate the manifold's behavior within the tetrahedron. Some device is needed to detect and analyze local self-intersections. This is also of concern in regions where distinct connected-components of the cross-section surface are very near one another.

Other questions concern the applicability of these techniques. What is the best way to handle non-orientable manifolds, particularly where two tetrahedra share a facet across which orientation is reversed? Can we characterize the class of manifolds amenable to these methods? Can these techniques be enhanced, or new techniques developed, to treat less-well behaved manifolds? These questions require further research into 3-dimensional topology and geometry, as well as data structure design.

Acknowledgements: The author gratefully acknowledges the support of David Dobkin, Anatoly Libgober, Peter Nelson, Jeff Weeks and the members of the Electronic Visualization Laboratory at UIC.

References

[AB] Avis, D. and B. Bharracharya, "Algorithms for Computing *d*-Dimensional Voronoi Diagrams and their Duals," in *Advances in Computing Research*, Ed. F. P. Preparata, 1, JAI Press, 1983, pp. 159–180.

[AE] Avis, D. and H. ElGindy, "Triangulating Point Sets in Space," Discrete and Comp. Geom., No. 2, 1987, pp. 99–111.

[Ba] Baumgart, B. G., "A Polyhedron Representation for Computer Vision," in *1975 National Computer Conference, AFIPS Conference Proceedings*, vol. 44, AFIPS Press, 1975, pp. 589–596.

[BrD] Brisson, D. W., "Visual Comprehension of *n*-Dimensions," in *Hypergraphics: Visualizing Complex Relationships in Art, Science and Technology*, Ed. D. Brisson, Westview Press, 1978, pp. 109–145.

[BrE] Brisson, E., "Representing Geometric Structures in *d* Dimensions: Topology and Order," *Proc. 5th Symp. on Computational Geometry*, June 1989.

[Ch] Chazelle, B., "Convex Partitions of Polyhedra," SIAM J. Comput. 13(3), pp. 488–507.

[DCH] Dreben, R., L. Carpenter and P. Hanrahan, "Volume Rendering," Computer Graphics, 22(4), August 1988, pp. 65–74.

[DL] Dobkin, D. P. and M. J. Laszlo, "Primitives for the Manipulation of Three-Dimensional Subdivisions," Algorithmica 4(1), Jan. 1989, pp. 3–32.

[EPW] Edelsbrunner, H., F. P. Preparata and D. B. West, "Tetrahedrizing Point Sets in Three Dimensions," Report No. UIUC DCS-R-86-1310, Dept. of Comp. Sci., Univ. of Ill. at Urbana-Champaign, 1986.

[FSB] Feiner, S., D. Salesin and T. Banchoff, "DIAL: A Diagrammatic Animation Language," IEEE Computer Graphics and Applications, 2(7), 1982, pp. 43–56.

[GN] Gallagher, R. and J. Nagtegaal, "An Efficient 3D Visualization Technique for Finite Element Models and Other Coarse Volumes," Computer Graphics, 23(3), July 1989, pp. 185–194.

[GS] Guibas, L. and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," ACM Trans. on Graphics, Vol. 4, No. 2, Apr. 1985, pp. 75–123.

[HM] Hertel, S. and K. Melhorn, "Fast Triangulation of Simple Polygons," Foundations of Comp. Theory, Springer Lect. Notes in Comp. Sci., No. 158, pp. 207–218.

[HW] Hildebrand, M. and J. Weeks, "A Computer Generated Census of Cusped Hyperbolic 3-Manifolds," *Computers and Mathematics*, ed. E. Kaltofen and S. Watt, Springer-Verlag, N.Y., 1989.

[Ho] Ho-Le, K., "Finite Element Mesh Generation Methods: A Review and Classification," Computer Aided Design, 20(1), 1988, pp. 27–38.

[KBBL] Kocak, H., F. Bisshop, T. Banchoff, and D. Laidlaw, "Topology and Mechanics with Computer Graphics: Hamiltonian Systems in Four Dimensions," Advances in Applied Mathematics, Vol. 7, 1986, pp. 282–308.

[Le] Levoy, M, "Display of Surfaces from Volume Data," IEEE Computer Graphics and Applications, 8(3), May 1988, pp. 29–37..

[LC] Lorensen, W. and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics, 21(4), July 1987, pp. 163–169.

[No] Noll, A. M., "Displaying n-Dimensional Hyperobjects by Computer," in *Hypergraphics: Visualizing Complex Relationships in Art, Science and Technology*, Ed. D. Brisson, Westview Press, 1978, pp. 147–158.

[PPP] Pasko, A., V. Pilyugin and V. Pokrovskiy, "Geometric Modeling in the Analysis of Trivariate Functions," Computers and Graphics, 12(3/4), 1988, pp. 457–465.

[PS] Preparata, F. P. and M. I.. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.

[RM] Russell, G. and R. Miles, "Display and Perception of 3D Space-Filling Data," Applied Optics, 26(6), 1987, pp. 973–982.

[Ro] Rossignac, J., "Considerations on the Interactive Rendering of Four-Dimensional Volumes," Chapel Hill Workshop on Volume Visualization Conference Proceedings, May 1989, pp. 67–76.

[Sa] Sabella, P., "A Rendering Algorithm for Visualizing 3D Scalar Fields," Computer Graphics, 22(4), August 1988, pp. 51–58.

[ST] Singer, I., and J. Thorpe, *Lecture Notes on Elementary Topology and Geometry*, Springer-Verlag, 1967.

[Th] Thorpe, J., *Elementary Topics in Differential Geometry*, Springer-Verlag, 1979.

[We] Weeks, J., *The Shape of Space*, Marcel Dekker Inc., 1985.

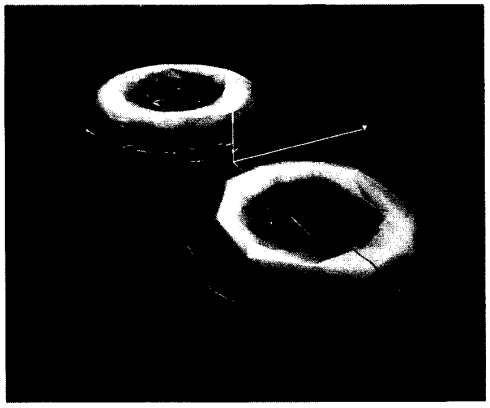

Plate 1
(Color Plate 167, page 487)

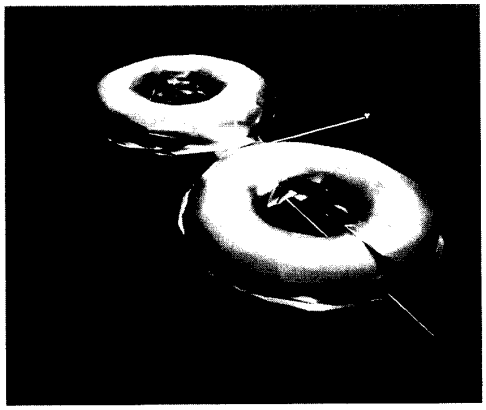

Plate 2
(Color Plate 168, page 487)

Plate 3
(Color Plate 169, page 488)

Plate 4
(Color Plate 170, page 488)

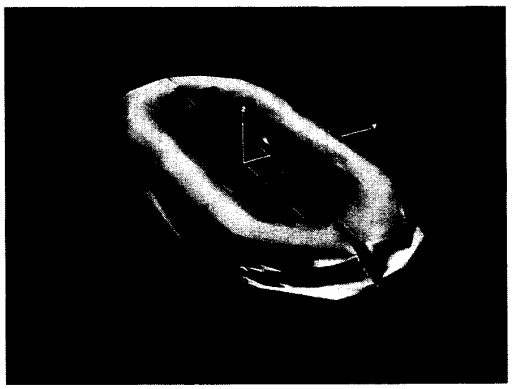

Plate 5
(Color Plate 171, page 488)

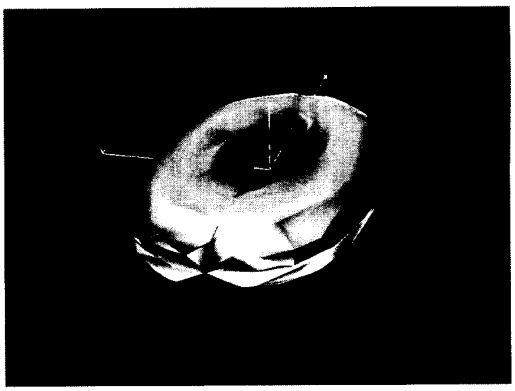

Plate 6
(Color Plate 172, page 488)

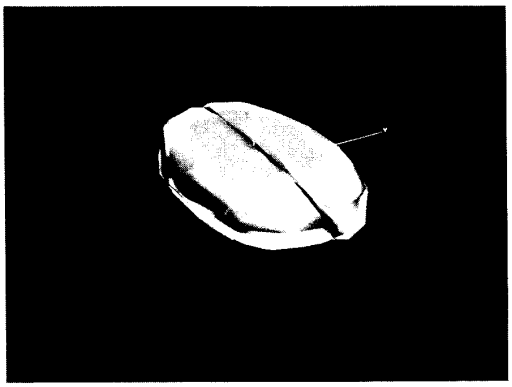

Plate 7
(Color Plate 173, page 488)

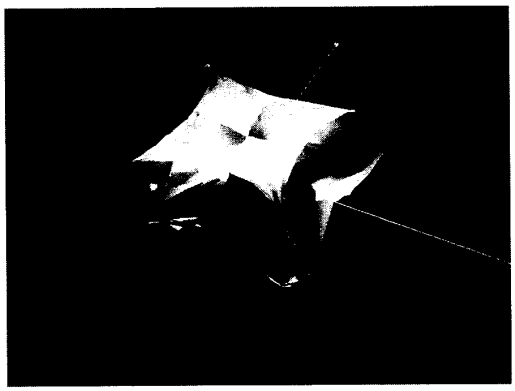

Plate 8
(Color Plate 174, page 488)