4. Manufacturing Isovolumes

Michael Bailey

4.1 Introduction

Displaying a single isosurface provides considerable insight into the distribution of scalar values in a volume. Being able to simultaneously see several isosurfaces provides even more insight. The difference is that a single isosurface displays a point solution. Seeing several isosurfaces also provides "first derivative" information, that is, how fast values are changing in certain regions.

Certainly this insight can be achieved by moving an isovalue slider back and forth very quickly. But, too often, we cannot achieve such dynamics. Either the dataset is too large to be re-displayed at interactive speeds or we are away from our display screens in a presentation environment. The approach described here produces a series of closed isovolumes and then manufactures them, providing a non-volatile display of several isosurfaces. The inspiration for this idea was the common Russian doll set, as shown below:

Figure 4.1. Interlocking Volumes

This is essentially an interlocking isovolume situation. It is easy and intuitive to see how the different layers merge into each other. The display is clear and non-volatile. The question then becomes how to best accomplish this for scientific data.

4.2 Previous Work

Most isosurface work focuses on just producing the surfaces themselves because the end goal is usually a computer graphics display. However, some recent work has focused on producing representations that encompass a complete volume. Tetrahedralizations are useful in volume visualization for display purposes and for determining properties. An excellent overview of triangulations and tetrahedralizations can be found in [1]. An important generalization of the tetrahedralization of a volume is known as the *Interval Volume*, and is described in [2]. An Interval Volume tetrahedralizes the gap between two isosurfaces to create an isovolume.

However, as will be seen in the next section, tetrahedralization produces more information than is necessary for prototype manufacturing. Prototype manufacturing only needs the outer "skin" to be defined. The inner volume is then inferred from that. In [3], the outer skin problem is addressed by creating a closed surface that is topologically equivalent to the outer skin of a sphere. This method starts with a spherical grid in the middle of the volume and then spreads the grid points until they meet the isovalue. This produces good results, but it cannot handle an isovolume that contains passages through it or is in multiple pieces, common occurrences within a single dataset.

4.3 The Manufacturing Process and What Information It Requires

To fabricate the isovolumes, we have been using the UCSD/SDSC TeleManufacturing Facility (TMF) [4,5]. The TMF has made solid freeform fabrication into a routinely-used "3D visualization hardcopy device". This project has connected a Helisys Laminated Object Manufacturing (LOM) [6] and a Z Corporation Z402 [7] system to the Internet. The Z402 and LOM machines are shown below:

Figure 4.2. Z Corporation Z402 and Helisys LOM Machines

These two technologies are ideally suited for manufacturing isovolumes of arbitrary shape. Like all rapid prototyping or solid freeform fabrication processes, these two are *additive* manufacturing methods [8,9,10]. The LOM raw material is a roll of paper, .11 mm (.0044 inches) thick, with heat-activated glue on the underside. For each layer, the paper is automatically rolled into place and laminated to the layers underneath it with heat and pressure. A laser then traces the outline of the part at that level. The Z402 raw material is a powder. The 2D cross-section at each layer is hardened with a binder chemical, spread by means of a laserjet printing head.

When the fabrication is completed, both processes require scrap to be removed, revealing the final part. For LOM, this involves plucking out paper scrap cubes. For the Z402, it involves vacuuming excess powder.

Our machines are front-ended by web-based file submission procedures and automated geometry checking. They are back-ended by web-based cameras that monitor the machines' progress. The project has found that physical models can provide considerable visualization insight in such diverse fields as biochemistry, earth science, fluid dynamics, mathematics, and terrain mapping.

The required input for all solid freeform fabrication machines is the industry-standard STL file. To convert a solid geometry to an STL file requires creating a list of 3D triangles that bound the outer skin of the solid. [8] STL-defined 3D objects must be legal solids, that is, they must be bounded by a complete shell with no holes or cracks. In addition, the triangles of the outer skin must obey the vertex-to-vertex rule: each edge must bound exactly two triangles. Any other edge configuration would result in cracks in the outer surface.

4.4 Forcing the Isovolume to be a Legal Solid

When requesting a graphical isosurface, a single scalar value, S^* is given. A manufacturable isovolume must be a legal solid, which means that it must be continuously bound on all sides. In requesting an isovolume, two scalar values, S_{min}^* and S_{max}^* must be specified.[1]

Turning these two isovalues into a legal solid is a two-step process:

1. Compute each isovalue's corresponding isosurface
2. At the boundaries of the volume, cap the gap between the isosurfaces.

4.5 Triangle "Incremation"

Much of the current graphics and visualization literature is concerned with polygon *decimation*. Polygon decimation seeks to eliminate detail that is perceptually unnecessary, in order to achieve better graphics performance. This technique is especially crucial for isosurfaces of large datasets.

This works well for computer graphics. Computer graphics displays can get away with too little detail through techniques such as smooth shading and dynamics. But, physical solids can play no such trickery. Large polygons that look smooth on a graphics display will create a fabricated surface that looks coarse and "chunky".

[1] If just the inside or just the outside is desired, the value of S_{min}^* can be set to $-\infty$ or the value of S_{max}^* can be set to $+\infty$.

Fortunately, when fabricating isovolumes, display speed is not relevant. Whereas interactive graphics encourages the trading of display quality and accuracy for speed, fabrication encourages the most accuracy and quality regardless of polygon count.

Thus, triangle decimation is not an issue here. In fact, the only such issue is how much we should *increase* the scene detail in pursuit of a quality fabrication. We use the term *triangle incremation* to describe the adding of such scene detail. We have made models composed of over one million 3D triangles, and the count could go quite a bit higher. Thus, in typical volumes, we can apply a considerable amount of polygon incremation before running into manufacturing limits.

4.6 Interpolating within a Marching Cube

The standard isosurface algorithm is Marching Cubes [11,12]. The Marching Cubes algorithm looks at the trilinear behavior of an isosurface in a single cube and renders it as 1-4 polygons. But, in fact, the isovalues within that cube are a smoothly-varying trilinear function. Thus, the real isosurface within that cube is a smooth surface, not a few flat polygons. This surface can be obtained by solving the trilinear equations. Furthermore, it can be made recursive and adaptive, subdividing the smooth isosurface within the cube as much as necessary to achieve a desired accuracy. Such things are allowed to happen when you don't care about polygon count.

Assume the standard Marching Cube, with eight scalar values, S_{ijk}, at the corners. S_{uvw} is the interpolated scalar value somewhere within the cube, with the values of u, v, and w parameterized to lie in the range:

$$0. \le u,v,w \le 1.$$

At an arbitrary (u,v,w), the interpolated scalar value is a function of blending equations and the scalar values at the cube's corners:

$$S_{uvw} = \sum_{i,j,k} B_{ijk} S_{ijk} \qquad (4.1)$$

where the blending functions are:

$B_{000} = (1-u)*(1-v)*(1-w)$
$B_{001} = (1-u)*(1-v)*w$
$B_{010} = (1-u)*v*(1-w)$
$B_{011} = (1-u)*v*w$
$B_{100} = u*(1-v)*(1-w)$
$B_{101} = u*(1-v)*w$

$B_{110} = u*v*(1-w)$
$B_{111} = u*v*w$

Solving these equations for $S_{uvw} = S^*$ along the edges of each face gives us the vertices of the Marching Cubes triangles. We use these triangles as a first step. The problem then becomes one of turning those triangles into a smoother surface that is closer to the actual isosurface that is passing through the cube. To do this, we will recursively subdivide each triangle until it comes within a specified tolerance of the exact isovalue.

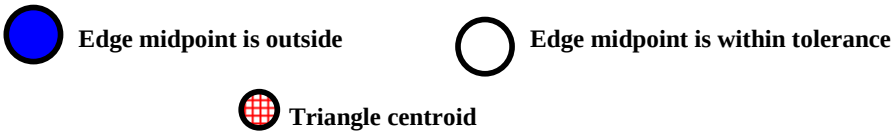
4.7 Recursively Subdividing a Triangle

A legal solid triangulation of an isovolume must not have any cracks where two adjacent surfaces do not precisely meet. (This is often a graphics requirement too, but is sometimes fudged. When the triangles are small, the cracks are less noticeable.) The way to avoid cracks is to be sure that adjacent surfaces share the same edge and vertices. In rapid prototyping terminology, this is called satisfying the "vertex-to-vertex" rule.

Triangles within a single Marching Cube can be made to satisfy the vertex-to-vertex rule by keeping track of common subdivision vertices. But, tracking vertex-to-vertex information between Cubes requires a lot of extra bookkeeping. Our strategy instead involved subdividing a triangle as a function of what is happening at its bounding vertices. Because adjacent Cubes have the same corner vertices, each Cube's subdivision decisions will be the same along the shared boundaries.

The vertex data structure records each vertex's u, v, and w normalized parametric coordinates. This is a convenience in that all other needed quantities (x, y, z, and S) can be interpolated from these parameters as shown in equation (4.1).

The first step in the triangle subdivision is to look at the midpoint of each edge. The midpoint is determined in parametric space. From that (u,v,w), a corresponding scalar value, S_{mid}, is computed. S_{mid} is compared against the required isoalue, S^*. If it is outside a specified tolerance $\boldsymbol{\varepsilon}$, it is flagged as needing to be subdivided. As there are three edge midpoints in a triangle, there are $2^3 = 8$ possible subdivision patterns. The eight possibilities are shown below. Note that there are really only four unique cases, the others being reflections of those four.



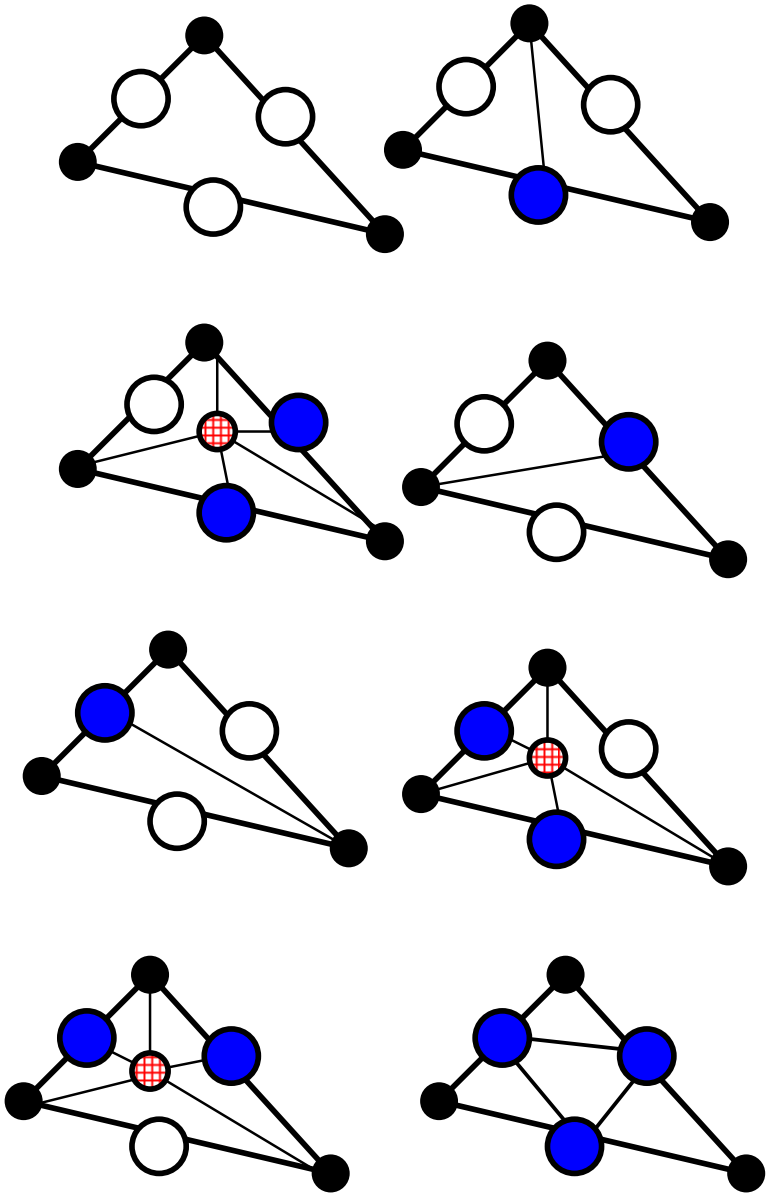


Figure 4.3. Triangle Subdivision Strategies

For the case where no interpolated edge midpoint's scalar value is outside the tolerance to S^*, that triangle can be output as is. For all other cases, the errant

midpoints are "pushed" so that their scalar value is equal to S^* (see the next section for how this is done), and the triangle is subdivided as shown above. Each of the subdivided triangles is then examined according to the same tests and possibly also subdivided. This recursion proceeds until all triangles are within the tolerance of S^*, or until a certain maximum recursion depth is reached.

In the above diagrams, the exact way that the triangles are subdivided is probably not crucial so long as the following rules are obeyed:

1. An edge with a midpoint that is out of tolerance needs to be turned into two edges for two triangles. The rest of this original edge must have the chance to be further subdivided.

2. An edge without a midpoint that is out of tolerance must not be subdivided.

Note that what happens on a particular edge is only dependent on information on its own midpoint, and not on anything that has to do with the other two edges. Thus, adjacent triangles will make the same decision on the shared edge. This is how we can guarantee that no cracks are created in the resulting triangle model.

4.8 Pushing an Arbitrary Point to the Exact Trilinear Isosurface

If we work in parameterized (u,v,w) coordinates, we can slide an arbitrary point within the cube to the isosurface defined by $S = S^*$ by determining how much we need to move in u, v, and w. If the isovalue S_{uvw} at (u,v,w) is not within ε of S^*, we need to push the (u,v,w) point enough to change its isovalue by $\Delta S = S^* - S_{uvw}$. From multivariate calculus,

$$\Delta S = S^* - S_{uvw} = \frac{\partial S}{\partial u}\Delta u + \frac{\partial S}{\partial v}\Delta v + \frac{\partial S}{\partial w}\Delta w \tag{4.2}$$

where $\frac{\partial S}{\partial u}$ is found within an individual Cube by differentiating (4.1):

$$\frac{\partial S_{uvw}}{\partial u} = \sum_{i,j,k} \frac{\partial B_{uvw}}{\partial u} S_{ijk} \tag{4.3}$$

and similarly for $\frac{\partial S}{\partial v}$ and $\frac{\partial S}{\partial w}$.

Equation (4.2) has three unknowns, Δu, Δv, and Δw. Thus, there is no unique (Δu, Δv, Δw) combination that will satisfy (4.2). However, certain combinations will make more sense than others. We chose to push the point (u,v,w) in the direction of the surface normal to the isosurface at that point.

From [11], the surface normal $\vec{N}$ ≡ (nu,nv,nw) is equal to $(\frac{\partial S}{\partial u}, \frac{\partial S}{\partial v}, \frac{\partial S}{\partial w})$.

Letting (Δu, Δv, Δw) = t * (nu,nv,nw) so that the point is pushed along $\vec{N}$, equation (4.2) becomes:

$$\Delta S = t \bullet (nu^2 + nv^2 + nw^2)$$

or,

$$t = \frac{\Delta S}{(nu^2 + nv^2 + nw^2)}$$

so that:

$$\Delta u = \frac{nu \bullet \Delta S}{(nu^2 + nv^2 + nw^2)} \tag{4.4a}$$

$$\Delta v = \frac{nv \bullet \Delta S}{(nu^2 + nz^2 + nw^2)} \tag{4.4b}$$

$$\Delta w = \frac{nw \bullet \Delta S}{(nu^2 + nv^2 + nw^2)} \tag{4.4c}$$

The general algorithm is:

```
-------------------------------------------------
Start with a point (u,v,w) from an edge midpoint
Compute S_uvw from (4.1)
while( |S* − S_uvw| > ε )
{
   Compute (nx,ny,nz) from (4.3)
   ΔS = S* − S_uvw
```

```
    Compute Δu , Δv and Δw from (4.4a-c)
    u = u + Δu
    v = v + Δv
    w = w + Δw
    compute S_uvw from (4.1)
  }
  ------------------------------------------------
```

4.9 Results of a Test Case

An interesting test case is provided in [13]. This paper shows a method of using rational quadric Bezier surfaces as a trilinear interpolant. A single Marching Cube is considered. The scalar data values at the uvw corners are as follows:

Table 4.1. Test Marching Cube

u,v,w	S_{uvw}
0,0,0	0.2
1,0,0	0.9
0,1,0	0.7
1,1,0	0.0
0,0,1	0.9
1,0,1	0.2
0,1,1	0.1
1,1,1	0.9

What makes this test case so interesting is that the isosurface contains a saddle point within thisone cube. This makes the traditional Marching Cubes algorithm produce a particularly jagged set of poygons.

The figures below show the isovolume produced between isovalues of 0.44 and 0.90, using different tolerances. Figures 4.4a-c show the isovolume surfaces. Figures 4.5a-c have shrunk the triangles to show the triangulation decisions made by the algorithm. Figure 4.6 shows a detailed close-up of one region of the triangulation.

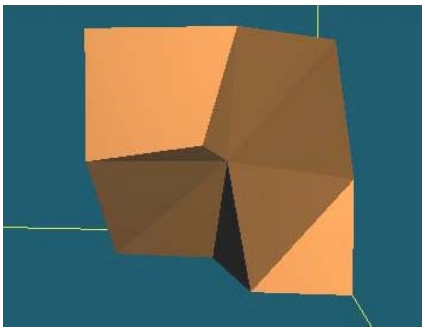


Figure 4.4a. ε = .1, 38 triangles

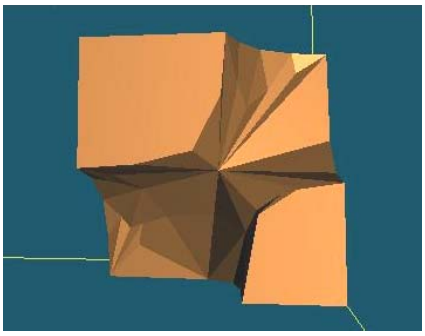


Figure 4.4b. ε = .01, 142 triangles

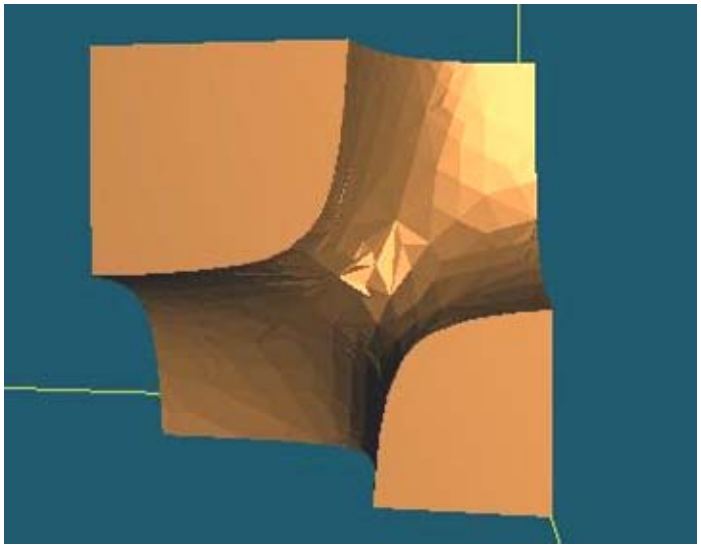


Figure 4.4c. ε = .001, 1152 triangles

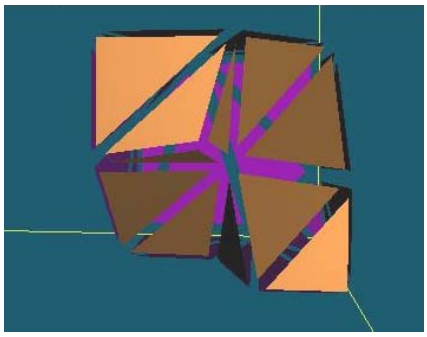


Figure 4.5a. ε = .1, 38 triangles

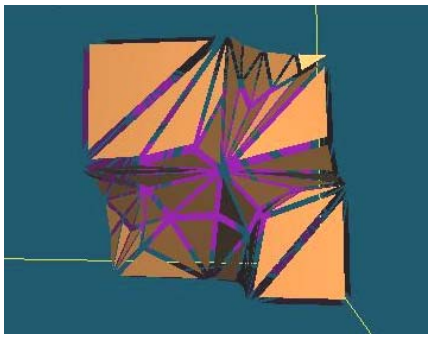


Figure 4.5b. ε = .01, 142 triangles

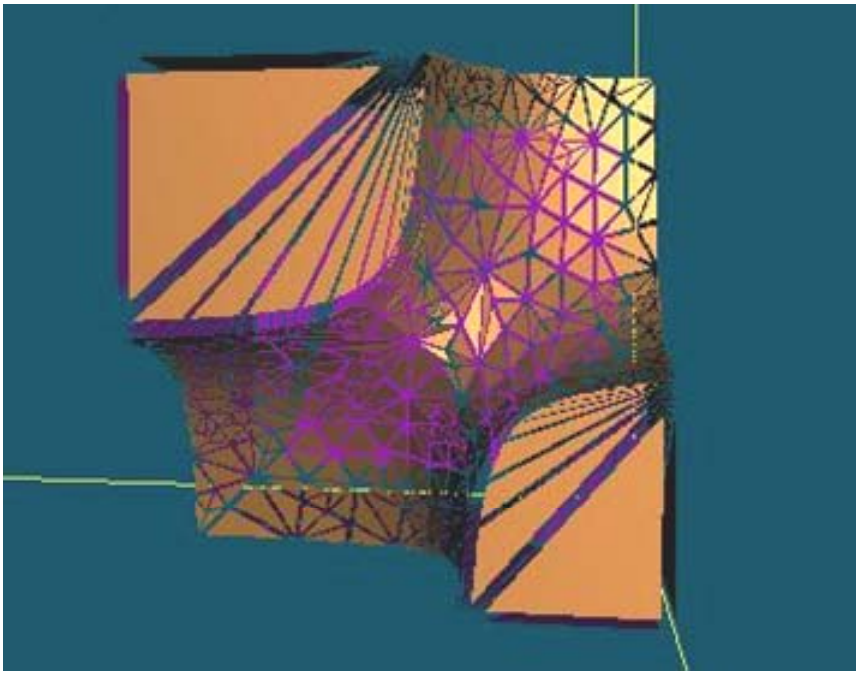


Figure 4.5c. ε = .001, 1152 triangles

Figure 4.6. A Close-up View of the ε = .001 Triangulation

4.10 An Interlocking Example

One of the strengths of this method lies in producing a series of isovolumes with common S^* values at their boundaries. This makes them fit together like the Russian dolls. This example uses a summation of decaying exponentials to seed the scalar values in a volume. The scalar value was assigned by:

$$S(x,y,z) = \sum_{i=1}^{3} A_i e^{-r_i^2}$$

where:

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2$$

As expected, the isosurface shapes were spherical "waves" emmenating from the (x_i,y_i,z_i) points. Several different isovolume ranges were chosen, the STL files were generated, and the parts were fabricated. The results are shown below:

Figure 4.7. Fabricated Interlocking Isovolumes

4.11 Medical Applications

One of the main reasons for this project was to develop better tools for examining medical volume data, as shown in Figures 4.8 and 4.9. Figure 4.8 shows the skull data from the Visualization Toolkit (vtk) [14], turned into a manufactured isovolume. Note in 4.8a how the triangles have been adaptively subdivided, providing more detail in areas of rapid change and less in flatter areas. Figure 4.9 shows a 3D fetal ultrasound isovolume.

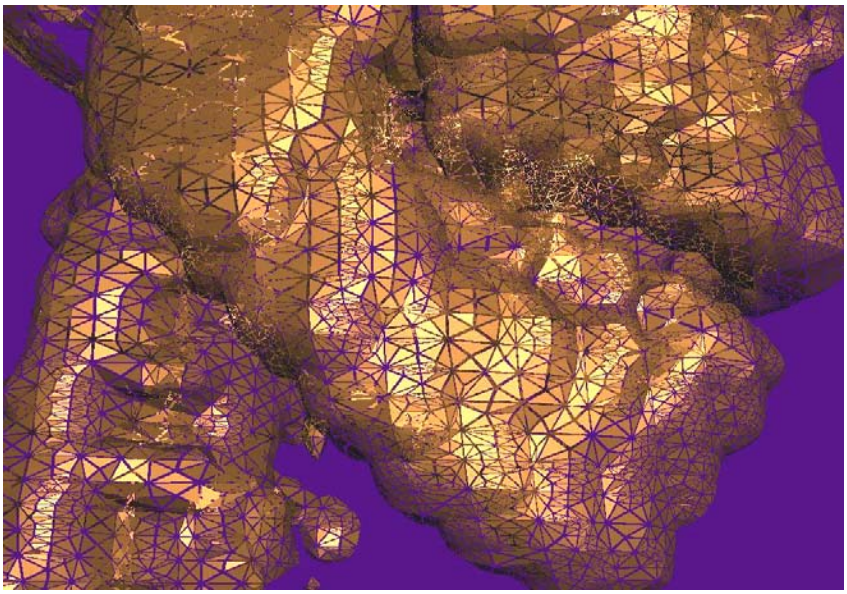


Figure 4.8a: Skull Isovolume Triangles

Figure 4.8b. Manufactured Skull Isovolume

Figure 4.8c. Manufactured Skull Isovolume

Figure 4.9: Fetal Isovolumes

4.12 Conclusions

The results from the examples in Figures 4.7-4.9 show that manufacturing isovolumes are a viable way of examining the relationships between different isosurface shapes by looking at both sides of a single isovolume and by examining multiple isovolumes. It is also a way to transport a volume visualization so it can be seen by multiple viewers without access to a graphics display.

The polygon incremation creates an arbitrarily accurate surface by subdividing only where it is necessary to meet the tolerance. Specifying this tolerance provides a convenient way to trade-off accuracy versus polygon count.

The most obvious benefit of manufacturing isovolumes is that it brings our sense of touch to bear on the task of understanding data that may be too complex to understand by sight alone. Being able to run fingers over and through a physical model conveys extra information that the eye does not even know was missing. We have been surprised how much more is understood about the 3D shape after holding a physical model for even a short amount of time.

There are some less obvious benefits to having a physical model of isovolumes. There appears to be some amount of insight that can be obtained by slightly *twisting* the interlocking pieces as they are being fit together. This imparts a certain sense of spatial derivatives, that is, how much a small physical perturbation changes the isoshape. This is hard to quantify, but everyone who handles these models seems to do this action intuitively. Most likely, we unconsciously do this with items in our everyday life, and have learned to gain insight this way without realizing it.

4.13 Acknowledgements

This work was supported by ONR grant N00014-97-1-0183 and NSF grant MIP-9420099.
The fetal ultrasound data is from Dr. Tom Nelson of the UCSD Medical School.

4.14 Web Page

For more information, see: http://www.sdsc.edu/tmf

4.15 References

1. Gregory Nielson, “Tools for Triangulations and Tetrahedralizations and Constructing Functions Defined over Them”, from Gregory Nielson, Hans

Hagen, and Heinrich Muller, *Scientific Visualization: Overviews, Methodologies, Techniques*, IEEE Computer Society, 1997, pp. 429-525.

2. Gregory Nielson and Junwon Sung, "Interval Volume Tetrahedralization", *Proceedings of IEEE Visualization '97*, pp. 221-228.

3. J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. O'Bara, and M. J. Wozny, "Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data", *Computer Graphics (SIGGRAPH '91 Conference Proceedings)*, Volume 25, July 1991, pp. 217-226.

4. Michael Bailey, "Tele-Manufacturing: Rapid Prototyping on the Internet with Automatic Consistency-Checking," *IEEE Computer Graphics and Applications*, November 1995, pp. 20-26.

5. Kathy Svitil, "A Touch of Science", *Discover* magazine, June 1998, pp. 80-84.

6. Helisys, Inc., http://www.helisys.com , 1999.

7. Z Corporation, http://www.zcorp.com, 1999.

8. Michael Bailey, "The Use of Solid Rapid Prototyping in Computer Graphics and Scientific Visualization," SIGGRAPH Course Notes for *The Use of Touch as an I/O Device for Graphics and Visualization*, 1996.

9. Marshall Burns, *Automated Fabrication*, Prentice-Hall, 1993.

10. Jerome L. Johnson, *A Unified Description of All Free Form Fabrication Technologies Based on Fundamental Principles*, Palatino Press, 1994.

11. W. E. Lorensen and H. E. Cline, "Marching Cubes: A high resolution 3D Surface Construction Algorithm", *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, Volume 21, Number 3, pp. 163-169.

12. Gregory Nielson and Bernd Hamann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes", *Proceedings of IEEE Visualization '91*, pp. 83-91.

13. Bernd Hamann, Isaac Trotts, and Gerald Farin, "On Approximating Contours of the Piecewise Trilinear Interpolant Using Triangular Rational-Quadric Bezier Patches", *IEEE Transactions on Visualization and Computer Graphics*, Vol 3, Number 3, July-Sept 1997, pp. 215-227.

14. Will Schroeder, Ken Martin, and Bill Lorensen, *The Visualization Tooliit: An Object-Oriented Approach to 3D Graphics*, Second Edition, Prentice Hall PTR, 1998.