BOUNDARY DETERMINATION FOR TRIVARIATE SOLIDS

Kenneth I. Joy[1]

Center for Image Processing and Integrated Computing
Department of Computer Science
University of California, Davis

and

Mark A. Duchaineau[2]

Center for Advanced Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 95616

Abstract

The trivariate tensor-product B-spline solid is a direct extension of the B-spline patch and has been shown to be useful in the creation and visualization of free-form geometric solids. Visualizing these solid objects requires the determination of the boundary surface of the solid, which is a combination of parametric and implicit surfaces. This paper presents a method that determines the implicit boundary surface by examination of the Jacobian determinant of the defining B-spline function. Using an approximation to this determinant, the domain space is adaptively subdivided until a mesh can be determined such that the boundary surface is close to linear in the cells of the mesh. A variation of the marching cubes algorithm is then used to draw the surface. Interval approximation techniques are used to approximate the Jacobian determinant and to approximate the Jacobian determinant gradient for use in the adaptive subdivision methods. This technique can be used to create free-form solid objects, useful in geometric modeling applications.

Keywords: splines; boundary surface determination; trivariate B-Spline solids; Jacobian determinant.

[1]Corresponding Author, Department of Computer Science, University of California, Davis, CA 95616-8562, USA; e-mail: joy@cs.ucdavis.edu

[2]Center for Advanced Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA; e-mail: duchaine@llnl.gov

1. Introduction

A trivariate parametric equation of the form

$$\mathbf{p}(u,v,w) = (x(u,v,w), y(u,v,w), z(u,v,w))$$

for $u \in [u_a, u_b]$, $v \in [v_a, v_b]$, and $w \in [w_a, w_b]$, represents a solid model in 3-dimensional space. To render this model in conventional surface-based rendering systems, it is necessary to identify the boundary surfaces that surround the solid.

The solid contains six "boundary faces," each the image of a rectangle representing the boundary of the domain space. These six faces

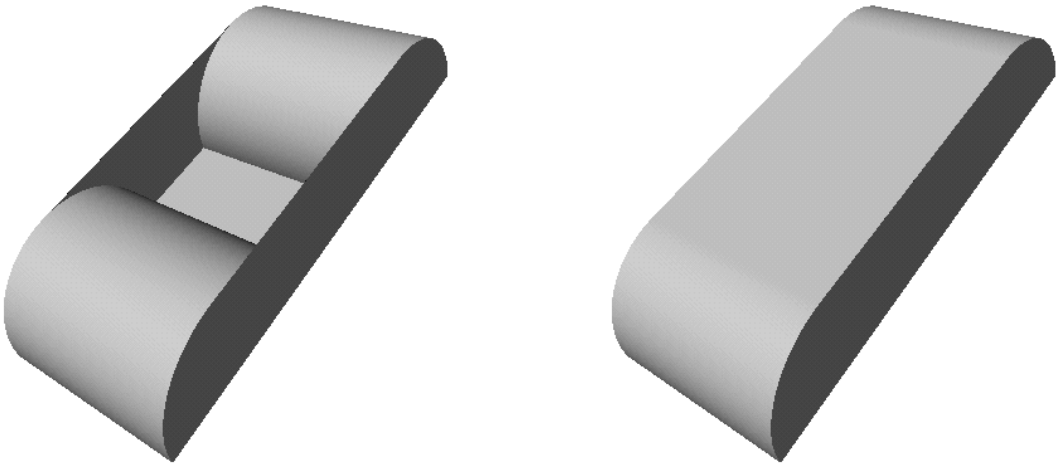
$$\{\mathbf{p}(u_a,v,w) : v \in [v_a,v_b], w \in [w_a,w_b]\}$$
$$\{\mathbf{p}(u_b,v,w) : v \in [v_a,v_b], w \in [w_a,w_b]\}$$
$$\{\mathbf{p}(u,v_a,w) : u \in [u_a,u_b], w \in [w_a,w_b]\}$$
$$\{\mathbf{p}(u,v_b,w) : u \in [u_a,u_b], w \in [w_a,w_b]\}$$
$$\{\mathbf{p}(u,v,w_a) : u \in [u_a,u_b], v \in [v_a,v_b]\}$$
$$\{\mathbf{p}(u,v,w_b) : u \in [u_a,u_b], v \in [v_a,v_b]\}$$

comprise a portion of the boundary of the solid, but in general, they are not sufficient to describe the complete boundary surface. It is straightforward to develop solids for which the boundary faces do not form the complete boundary of the solid. Figure 1 illustrates a solid model defined by

$$\mathbf{p}(u,v,w) = (\cos \pi u + 3w, \sin \pi u, v)\,.$$

This solid represents a half-cylinder that has been swept along a linear path. The parametric boundary faces of the solid are shown, but the top surface is an implicit surface, and not a boundary face. These parametric trivariate solids arise naturally through modeling operations (*e.g.*, sweeping or lofting of bivariate models). It has been shown by Joy [7, 8] that solids formed in this way frequently have the property that their boundary surface is not representable by the boundary face patches.

In this article, we limit ourselves to a discussion of the trivariate tensor-product B-spline solid and methods by which the boundary surface of the solid can be determined. In this case, the boundary face patches are B-spline patches and can be directly calculated. Based upon the Jacobian determinant of the defining function, a robust test is developed that indicates



(a) (b)

FIGURE 1: Images of the boundaries of the domain space are not sufficient to describe the boundaries of the solid. This model was generated by a linear sweep of a half-cylinder, creating a trivariate function. (a) the images of the boundaries in the domain space. (b) the complete solid.

the presence of an implicit boundary surface in a region of the domain space. An adaptive subdivision procedure is generated which splits regions of the domain into cells that (1) do not contain the solid, or (2) cells that may contain the solid. We use a variation of "marching cubes" [11] to generate the implicit surface, fixing up the cracks that may appear in the surface due to the adaptive nature of the algorithm. The union of the face patches together with the implicit boundary surface then give a superset of the boundary surface of the solid.

In Section 2, we review research related to trivariate solid models. Section 3 introduces the trivariate B-spline solid and the mathematical properties of the solid that we require. Definition of the boundary surface of a trivariate solid is discussed in Section 4. Interval methods to approximate the Jacobian determinant over a domain cell are given in Section 5. The adaptive subdivision algorithm that isolates the implicit boundary surface is defined in Section 6. The isosurface generation algorithm is presented in Section 7. Here, we discuss the prevention of cracks that are naturally generated from the adaptive algorithm. Results of the use of the algorithm are shown in Section 8.

2. Related Work

Trivariate B-Spline and Bézier solids have been treated by a number of researchers. Stanton *et al.* [18], Casale and Stanton [3] and Farouki and Hinds [6] all discuss the trivariate form, but avoid the questions of construction of the general boundary surfaces. Lasser [10] discusses the general trivariate B-spline form, the generation of points in the volume and the generation of derivatives for these solids.

Sederberg and Parry [16] utilized the free-form trivariate B-spline solid for deformations. They embed an object in a deformable region of space defined by the trivariate solid such that each point of the object has a unique parameterization that defines its position in the region. The trivariate region is then altered by moving its control points. Using the trivariate form gives great flexibility to the definition of the deformable regions, and gives few parameters (the control points) which can be used to control the deformation.

Joy [7] described a modeling system for trivariate B-spline solids. These solids were rendered by using the parametric face patches where possible. Heuristic methods were developed to render the implicit surface, but these were not robust and did not work in many cases. Joy [8] also generalized his method to sweeping, generating trivariate B-spline solids from the sweeping operations. The trivariate generators of the sweeping operations, however, must be representable by the boundary face patches for this method to work.

Rappaport *et al.* [13] have incorporated physical properties into free-form trivariate solids. They utilize volume preservation and energy measures to limit the flexibility of the deformations of the solids. Reus *et al.* [14] have also treated the trivariate tensor-product solid in a physical manner.

In this paper, we are interested in a boundary-surface description of the trivariate tensor-product B-spline solid. On a surface-based rendering system, we must find the surface that represents the boundary of the solid. We provide a robust method that guarantees the presence of the implicit boundary and renders it to a desired accuracy. To do this, we approximate the Jacobian determinant using interval techniques, and subdivide the domain space to isolate rectilinear cells in the domain that contain the implicit surface. We adapt the marching-cubes algorithm to generate a crack-free surface that represents the complete surface of the solid.

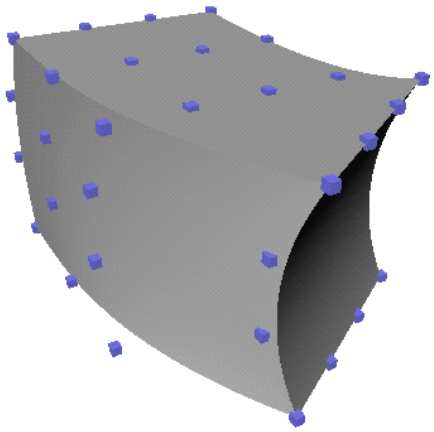


FIGURE 2: A cubic trivariate Bézier solid. The solid is defined by 64 control points, shown in blue. The boundary faces are the images of the boundaries of the domain interval.

3. The Trivariate Tensor-Product B-Spline Solid

The trivariate tensor product B-spline solid is defined by an a set of $(n_1+1)\times(n_2+1)\times(n_3+1)$ control points $\{\mathbf{p}_{i_1,i_2,i_3} \,:\, 0 \le i_1 \le n_1,\ 0 \le i_2 \le n_2,\ 0 \le i_3 \le n_3,\}$ and three sets of knots

$$\{u_0,\ u_1,\ \ldots\ u_{n_1+m_1}\}$$
$$\{v_0,\ v_1,\ \ldots\ v_{n_2+m_2}\}$$
$$\{w_0,\ w_1,\ \ldots\ w_{n_3+m_3}\}$$

where

$$\mathbf{p}(u,v,w) = \sum_{i_1=0}^{n_1}\sum_{i_2=0}^{n_2}\sum_{i_3=0}^{n_3} \mathbf{p}_{i_1,i_2,i_3} N_{i_1,m_1}(u) N_{i_2,m_2}(v) N_{i_3,m_3}(w)$$

for $u \in [u_{m_1-1}, u_{n_1+1}]$, $v \in [v_{m_2-1}, v_{n_2+1}]$ and $w \in [w_{m_3-1}, w_{n_3+1}]$. The products $N_{i_1,m_1}(u)N_{i_2,m_2}(v)N_{i_3,m_3}(w)$ are the trivariate tensor-product B-spline normalized blending functions defined by the knot sequences, and m_1, m_2 and m_3 are the orders of the spline in each of the parametric variables.

The trivariate B-spline provides a robust set of solids for design purposes. One can easily

define many polygonal solids and close approximations to spheres, cylinders, cones and tori (with exact specification if one wishes to use the rational form of the spline). In addition, complex solids can also be defined by using control-point specification with a variety of curve and patch fitting algorithms. The solid can be refined/subdivided by a trivariate adaptation of the a B-spline subdivision algorithm [2, 4]. They satisfy the convex-hull property, so that bounding volumes can be calculated by examining only the control points. The B-spline solids may be refined/subdivided into a set of Bézier solids whose union is the original – similar to the refinement of a B-spline patch into its Bézier components (see [1, 5]).

Partial derivatives of trivariate B-splines functions are also trivariate B-splines. The control points of the partial-derivative B-spline functions $\mathcal{D}_u\mathbf{p} = \frac{\partial}{\partial u}\mathbf{p}$, $\mathcal{D}_v\mathbf{p} = \frac{\partial}{\partial v}\mathbf{p}$ and $\mathcal{D}_w\mathbf{p} = \frac{\partial}{\partial w}\mathbf{p}$ are given by

$$(\mathcal{D}_u\mathbf{p})_{i,j,k} = \frac{m_1 - 1}{u_{i+m-1} - u_i}(\mathbf{p}_{i,j,k} - \mathbf{p}_{i-1,j,k})$$

for $1 \leq i \leq n_1$,

$$(\mathcal{D}_v\mathbf{p})_{i,j,k} = \frac{m_2 - 1}{v_{j+m-1} - v_j}(\mathbf{p}_{i,j,k} - \mathbf{p}_{i,j-1,k})$$

for $1 \leq j \leq n_2$ and

$$(\mathcal{D}_w\mathbf{p})_{i,j,k} = \frac{m_3 - 1}{w_{k+m-1} - w_k}(\mathbf{p}_{i,j,k} - \mathbf{p}_{i,j,k-1})$$

for $1 \leq i \leq n_3$, respectively. These control points are vectors, and each derivative $\mathcal{D}_u\mathbf{p}$, $\mathcal{D}_v\mathbf{p}$, and $\mathcal{D}_w\mathbf{p}$ is of degree one less than the degree of $\mathbf{p}$ in u, v, and w, respectively.

4. The Boundary of a Parametric Solid

We can state a simple result that gives a superset of the topological boundary of a parametric solid.

Theorem 4.1

Given a rectilinear cell $B = [u_a, u_b] \times [v_a, v_b] \times [w_a, w_b]$ and a trivariate B-Spline function $\mathbf{p}(u, v, w)$ defined over B. Then the surface boundary of the solid $\mathbf{p}$ is contained within the union of the boundary faces of the solid over B, and the points where the determinant of the Jacobian of $\mathbf{p}$ over B vanishes.

For a proof of this theorem see [12].

The boundary faces of a trivariate B-Spline solid are each bivariate B-spline patches. Thus the boundary of the solid is a combination of parametric B-spline patches and the isosurface where the determinant of the Jacobian of $\mathbf{p}$ is zero. The determinant of the Jacobian of $\mathbf{p}$ is defined by

$$\begin{aligned}
\mathcal{J}(\mathbf{p}(u,v,w)) &= \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} \\
&= \begin{vmatrix} \frac{\partial x}{\partial u}(u,v,w) & \frac{\partial y}{\partial u}(u,v,w) & \frac{\partial z}{\partial u}(u,v,w) \\ \frac{\partial x}{\partial v}(u,v,w) & \frac{\partial y}{\partial v}(u,v,w) & \frac{\partial z}{\partial v}(u,v,w) \\ \frac{\partial x}{\partial w}(u,v,w) & \frac{\partial y}{\partial w}(u,v,w) & \frac{\partial z}{\partial w}(u,v,w) \end{vmatrix} \\
&= \mathcal{D}_u\mathbf{p}(u,v,w) \cdot (\mathcal{D}_v\mathbf{p}(u,v,w) \times \mathcal{D}_w\mathbf{p}(u,v,w))
\end{aligned}$$

Thus, we have that $\mathcal{J}(\mathbf{p}(u,v,w)) = 0$ if and only if the triple scalar product

$$\mathcal{D}_u\mathbf{p}(u,v,w) \cdot (\mathcal{D}_v\mathbf{p}(u,v,w) \times \mathcal{D}_w\mathbf{p}(u,v,w)) = 0$$

or, equivalently, when the three vectors $\mathcal{D}_u\mathbf{p}(u,v,w)$, $\mathcal{D}_v\mathbf{p}(u,v,w)$, and $\mathcal{D}_w\mathbf{p}(u,v,w)$ are linearly dependent.

Using this theorem, we can state that the boundary faces of a trivariate solid accurately represent the boundary of the solid only if the Jacobian determinant of the defining function does not vanish over its domain. If this determinant does vanish, then the boundary may also be represented by an implicit surface, defined to be the isosurface where $\mathcal{J} = 0$. It is this surface that we wish to calculate.

5. Approximating the Jacobian Determinant

Given a set of unit vectors $\vec{v}_1, \vec{v}_2, ..., \vec{v}_n$, we can bound these vectors by a cone $\mathcal{C}$, defined by an axis $\vec{a}$ and a "spread" angle α, such that the angle between each vector $\vec{v}_i$ and the axis $\vec{a}$ is less than α . A cone gives an "interval" approximation to a set of unit vectors. Each cone can be associated with a region on the unit sphere – the intersection between the cone, with its apex at the origin, and the unit sphere. The construction of a cone that satisfies these properties was described by Sederberg and Meyers [15], or by Kim [9], and an example is shown in Figure 3. For a general set of vectors, with varying lengths, we determine a cone bounding the unit vectors, which are determined by dividing each of the vectors by its length.

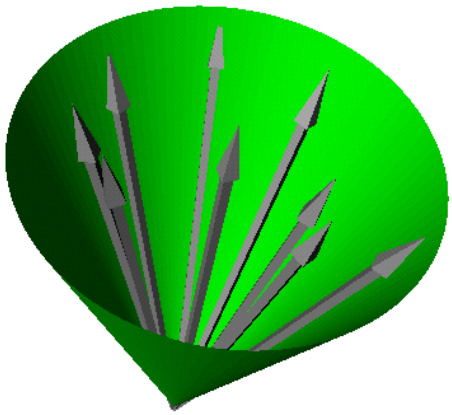


FIGURE 3: A cone approximation to a set of unit vectors.

Given two cones $\mathcal{C}_1$ and $\mathcal{C}_2$, we define the scalar product $\mathcal{C}_1 \cdot \mathcal{C}_2$ to be the interval defining the range of scalar products for pairs of vectors taken from $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. We can also define the cross product of two cones to be the smallest cone surrounding all cross products of vectors from $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively (see [15], and Figure 4).

The convex hull property holds for trivariate B-spline solids, *i.e.*, the solid is contained in the convex hull of its control points. This implies that for a given cell B in the domain, the cones $\mathcal{C}_u$, $\mathcal{C}_v$, and $\mathcal{C}_w$, constructed from the control points of $\mathcal{D}_u\mathbf{p}$, $\mathcal{D}_v\mathbf{p}$, and $\mathcal{D}_w\mathbf{p}$, respectively, bound the range of directions of the respective partials. This implies that a bound on the Jacobian determinant over B is given by

$$\mathcal{J}(\mathbf{p}) = \mathcal{D}_u\mathbf{p} \cdot (\mathcal{D}_v\mathbf{p} \times \mathcal{D}_w\mathbf{p}) \subseteq L\left(\mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)\right) \tag{1}$$

where L is an interval with positive entries[1], defined to be

$$L = [L_{\text{min}}, L_{\text{max}}]$$

[1]We assume none of our vectors have zero length.

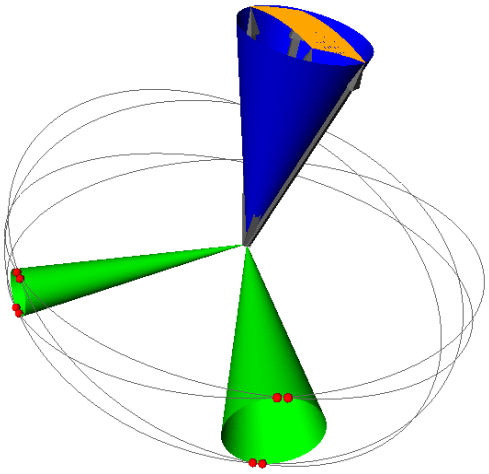


FIGURE 4: The cross-product cone. The blue cone is the smallest cone that surrounds the cross products of vectors in the two green cones. This cone can be directly calculated by bounding the four vectors that form the normals to the four great circles of the unit sphere tangent to the green cones. The yellow region is the actual region spanned by the cross products calculated from vectors in the two cones.

where

$$L_{\text{max}} = \max\left\{|\vec{v}_u||\vec{v}_v||\vec{v}_w| : \vec{v}_u \in \mathcal{C}_u, \vec{v}_v \in \mathcal{C}_v, \text{ and } \vec{v}_w \in \mathcal{C}_w, \right\}, \text{and}$$
$$L_{\text{min}} = \min\left\{|\vec{v}_u||\vec{v}_v||\vec{v}_w| : \vec{v}_u \in \mathcal{C}_u, \vec{v}_v \in \mathcal{C}_v, \text{ and } \vec{v}_w \in \mathcal{C}_w, \right\}.$$

The quantity $L\left(\mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)\right)$ is an interval product, and produces an interval bounding the range of values of $\mathcal{J}(\mathbf{p})$ over B. If L is interval with positive components, it is clear that if $0 \notin \mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)$ then $\mathcal{J}(\mathbf{p}) \neq 0$ in B. Therefore, given a cell B, and a trivariate B-spline solid $\mathbf{p}$ defined over B, we can state that the implicit boundary surface is not contained in B if

$$0 \notin \mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)$$

where $\mathcal{C}_u$, $\mathcal{C}_v$, and $\mathcal{C}_w$ are the bounding cones for $\mathcal{D}_u\mathbf{p}$, $\mathcal{D}_v\mathbf{p}$, and $\mathcal{D}_w\mathbf{p}$, respectively.

6. Adaptively Subdividing the Domain Space

To generate the implicit boundary surface, we adaptively subdivide the domain space, isolating rectilinear cells in the domain where the isosurface lies. We then use an adaptation of the marching-cubes algorithm to find the isosurface (see Lorensen *et al.*[11], or Wyvill and McPheeters [19] for a similar algorithm). We use a priority queue of domain cells, ordered by decreasing values of the widths of the intervals $\mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)$. The full domain space is initially placed on the queue.

When a cell B is removed from the queue, it is subdivided into two pieces B_1 and B_2 via a plane through the center of the cell and parallel to the xy, xz, or yz plane. For each B_i, the cone approximation $\mathcal{C}_u \cdot (\mathcal{C}_v \times \mathcal{C}_w)$ is calculated, yielding an interval. If zero is contained in this interval, the cell is inserted into the queue. If zero is not contained in the interval, the cell is discarded. By this process, we construct a binary tree of cells, keeping the relevant cells in a priority queue. This process is continued until the widths of the intervals of all cells in the queue are less than a prescribed minimum, or the number of cells in the queue reaches a predetermined number.

If the queue becomes empty, then the implicit boundary surface does not exist over the domain space, and the parametric boundary faces represent the boundary surface of the solid.

We have three planes by which we can split each cell. Since we will use a marching-cubes method to generate the isosurface, we would like the isosurface to be relatively flat, relative to the cell size.

The gradient of the Jacobian determinant at a point (u, v, w) is normal to the isosurface through this point. Thus, To accurately predict when a linear approximation of the isosurface is accurate over a box B, we wish to subdivide cells such that the variation of these gradients is minimized.

The gradient of the Jacobian determinant $\nabla\mathcal{J}$ is the vector given by

$$\nabla\mathcal{J}(u,v,w) = \begin{pmatrix} \mathcal{J}_u(u,v,w) & \mathcal{J}_v(u,v,w) & \mathcal{J}_w(u,v,w) \end{pmatrix}$$

where

$$\mathcal{J}_u(u,v,w) = \begin{vmatrix} \mathcal{D}_{uu}\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_{uv}\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_{uw}\mathbf{p}(u,v,w) \end{vmatrix}$$

Similarly,

$$\mathcal{J}_v(u,v,w) = \begin{vmatrix} \mathcal{D}_{uv}\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_{vv}\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_{vw}\mathbf{p}(u,v,w) \end{vmatrix}$$

and

$$\mathcal{J}_w(u,v,w) = \begin{vmatrix} \mathcal{D}_{uw}\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_{vw}\mathbf{p}(u,v,w) \\ \mathcal{D}_w\mathbf{p}(u,v,w) \end{vmatrix} + \begin{vmatrix} \mathcal{D}_u\mathbf{p}(u,v,w) \\ \mathcal{D}_v\mathbf{p}(u,v,w) \\ \mathcal{D}_{ww}\mathbf{p}(u,v,w) \end{vmatrix}$$

We can approximate each of the three quantities on the right side of these equations by Equation (1). This gives an interval for each determinant, and we can sum the three intervals to get a bound on the range of the gradient. This gives an approximation to the gradient over the cell.

If I_u, I_v and I_w are the intervals approximating $\mathcal{J}_u$, $\mathcal{J}_v$, and $\mathcal{J}_w$, respectively, then we examine the three quantities:

$$g_u = width(I_u)(u_b - u_a), g_v \quad = width(I_v)(v_b - v_a), \text{ and} g_w = width(I_w)(w_b - w_a),$$

where $B = [u_a, u_b] \times [v_a, v_b] \times [w_a, w_b]$. We subdivide by a plane parallel to the xy plane if g_w is the maximum, parallel to the xz plane if g_v is the maximum, and parallel to the yz plane if

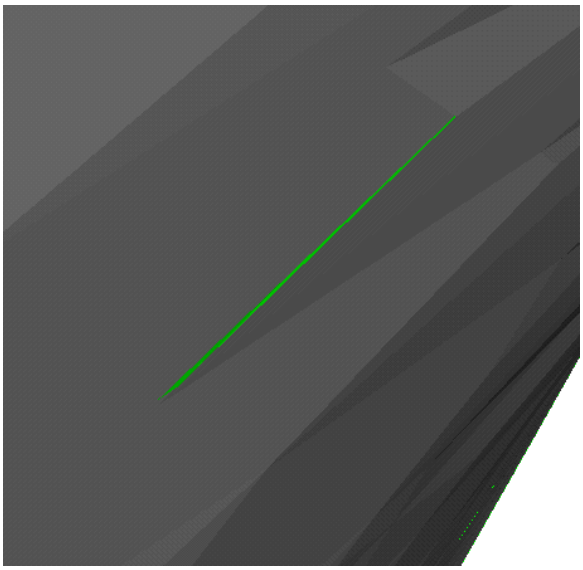


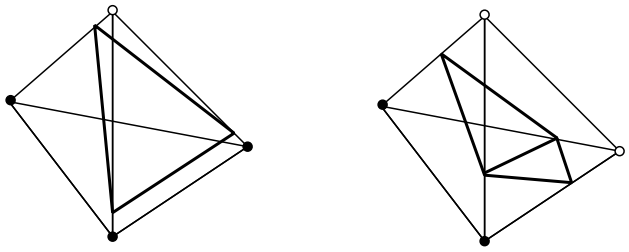
FIGURE 5: A crack in a generated isosurface. The underlying surface was colored green to enhance the crack.

g_u is the maximum. Thus, we subdivide in the direction where the gradient has the maximum variance over the cell, with respect to the cell size..

7. Isosurface Generation

At the finish of the adaptive subdivision process, we have a binary tree of cells. The children of each cell have been created by splitting the parent cell by a plane through the center of the cell and parallel to the xy, xz, or yz plane. The standard way of calculating the isosurface $\mathcal{J}(\mathbf{p}) = 0$ is to use a marching cubes or similar algorithm, see Lorensen *et al.*[11], and Wyvill and McPheeters [19]. However, adaptive subdivision algorithms produce non-uniform cells, and the marching cubes algorithms typically produce cracks in the isosurface when applied to these collections of cells (see Shu *et al.* [17], and Figure 7). Shekhar *et al.*has published an algorithm that patches the cracks by comparing the isolines on abutting cell faces and modifying the isolines on the faces of the smaller cells to correspond to the isolines on the faces of the larger ones.

We use a variation of the marching cubes algorithm that subdivides each cube into tetrahedra. We can easily determine the isosurface in the tetrahedra, as there are only three cases to consider (see Zhao *et al.* [20], and Figure 6). We can also adaptively generate the tetrahedra



(a) (b)

FIGURE 6: Two of the three cases for marching tetrahedra. A type-I tetrahedra, shown in (a), has one vertex that differs in sign from the other three. In this case, one triangle represents the isosurface. A type-II tetrahedra, shown in (b), has two pairs of vertices that differ in sign. In this case, two triangles represent the isosurface. In the third case, not shown, the vertices are all of the same sign and no isosurface exists in the tetrahedron.

representing a cell, such that the generated isosurface does not have cracks.

We first make a pass through our binary tree of cells and identify the cells that can possibly generate cracks. These cells have neighbors that have been subdivided, as shown in Figure 7. For those cells that do not require repair, we split the image of the cell under $\mathbf{p}$ into twelve tetrahedra, as shown in Figure 8. This subdivision has two tetrahedra per face, each connected to the image of the center point of the cell.

For those cells that may generate cracks we can identify two abutting faces, and we split the face of the smaller cell such that the faces of the generated tetrahedra lie in the planes of the tetrahedra generated for the larger cell. Figure 9 illustrates this process. Here the face in the smaller cell is split into three tetrahedra. The three tetrahedra all contain the center point of the smaller cell, and the face points are determined from the larger cell. Three new points are calculated on the edges of the tetrahedra from the larger cell, with Jacobian values interpolated from the values on the larger cell. This insures that the faces of the tetrahedra actually abut, and also that the isolines will be the same – removing the possibilities of cracks.

8. Results

We have implemented this algorithm and used it to generate a number of solids. The algorithms has three major steps: (1) the adaptive subdivision of the cells to isolate the implicit boundary surface; (2) The identification of the cells that could potentially cause cracking; and, (3) the generation of the isosurface.

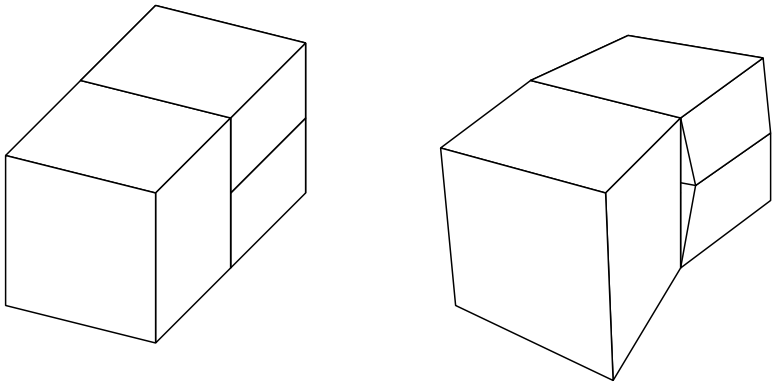


FIGURE 7: With adaptive subdivision, cracks can appear in the resulting isosurface. When two adjoining cells are subdivided differently, as shown in (a), the resulting cells in the range space, shown in (b), will appear to have a crack between the cells.

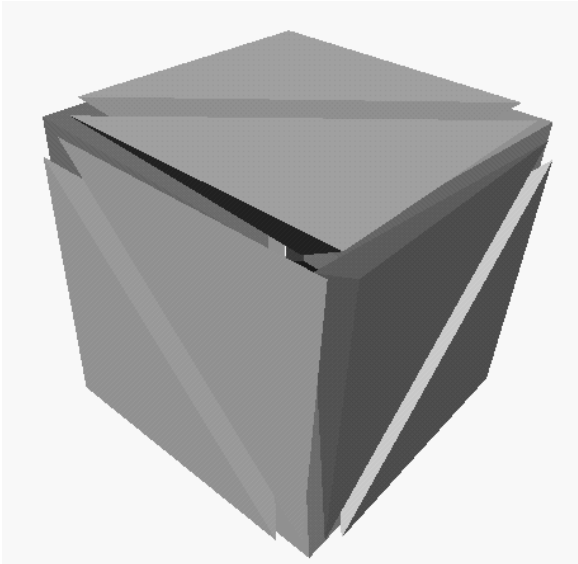
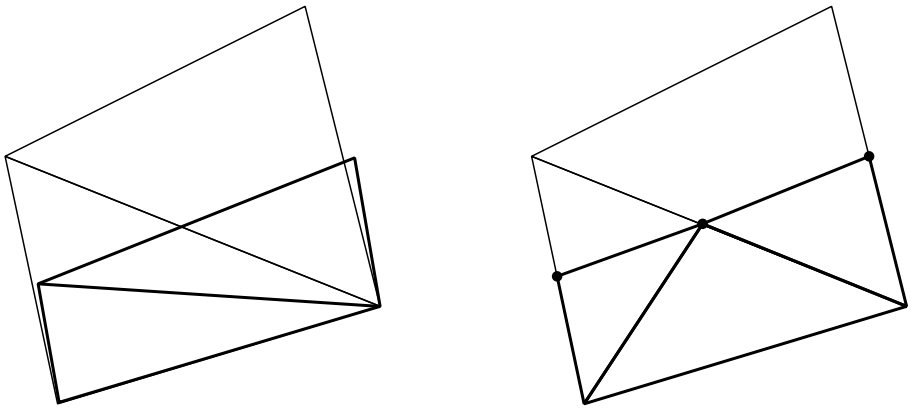


FIGURE 8: Splitting a cell into twelve tetrahedra. This is the default configuration.



(a) (b)

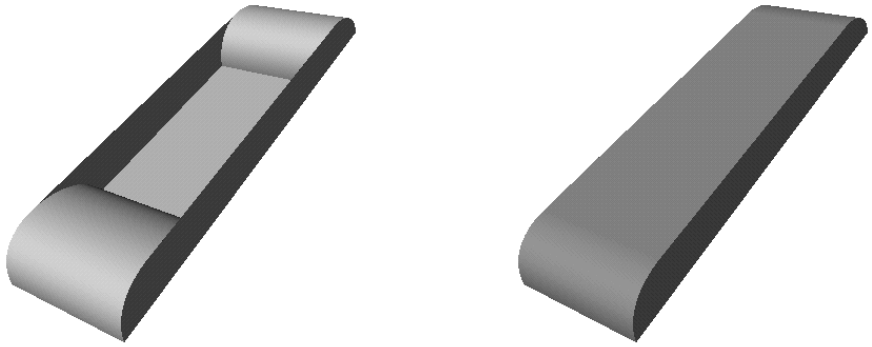
FIGURE 9: Repairing the cracks. The faces of two adjoining cells are shown in (a). The faces are shown with the triangles induced on the face by the tetrahedrazation of the cells. Cracks will appear in the generated isosurface. We can adaptively generate the tetrahedrazation of the smaller cell such that the faces induced by the tetrahedrazation adjoin the faces induced by the tetrahedrazation of the larger cell, shown in (b). This eliminates the cracking problem.

Figure 10 shows a bivariate B-spline patch swept along a linear curve. The resulting function is expressed as a trivariate patch. In this case, the isosurface corresponds to a plane with constant v value in the domain space. Since the defining function $\mathbf{p}$ has $\mathcal{D}_u\mathbf{p} = 0$ and $\mathcal{D}_w\mathbf{p} = 0$, the algorithm adaptively subdivides the domain space only with planes having constant v value. The resulting tree contains only one cell. The algorithm subdivides this cell into 12 tetrahedra, and represents the isosurface with 14 triangles.

The model of Figure 11 uses the bivariate patch from the Figure 10 and rotates the patch 35° as it is swept along a linear path. Four samples of the patch are taken and the trivariate solid is generated by lofting the four bivariate samples. The algorithm produced 900 cells, and generated 5,161 triangles. Figure 12 shows the reduction in the Jacobian interval bound during the operation of the algorithm. Note the smooth reduction of the bounds as the cells are split.

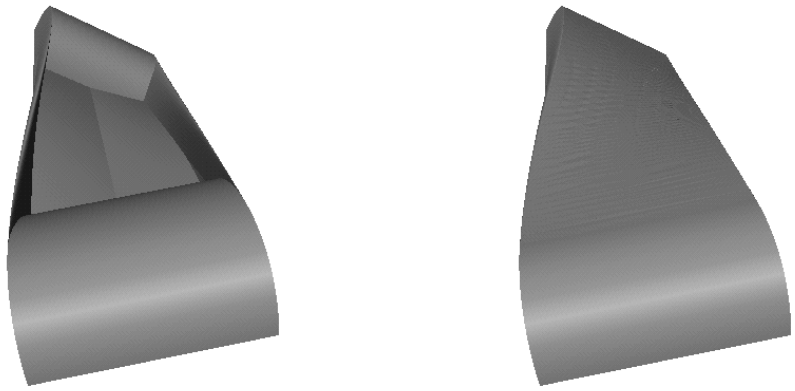
The model of Figure 13 uses a bivariate patch which is swept along another B-spline curve. Again, sections are taken and the trivariate solid is produced by lofting.

We have illustrated these surfaces with Gouraud shading removed so that the reader can see the generated sections of the isosurface. If Gouraud shading is enabled, the surfaces are smoothly rendered.



(a) (b)

FIGURE 10: Sweeping a bivariate patch along a linear curve. The boundary face patches are shown in (a), and the full solid is shown in (b).



(a) (b)

FIGURE 11: Sweeping a bivariate patch along a linear curve with rotation. The boundary face patches are shown in (a), and the full solid is shown in (b).

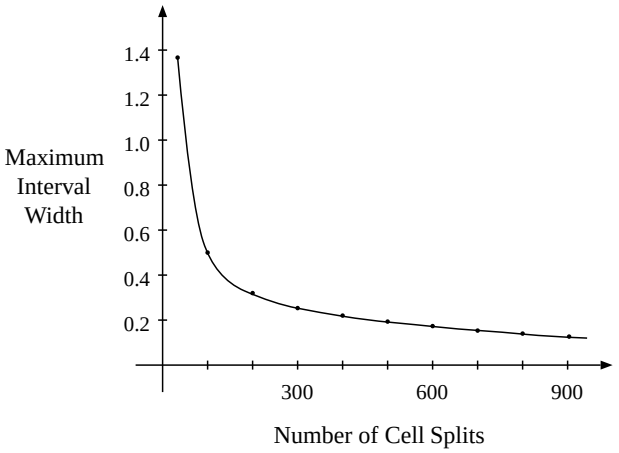
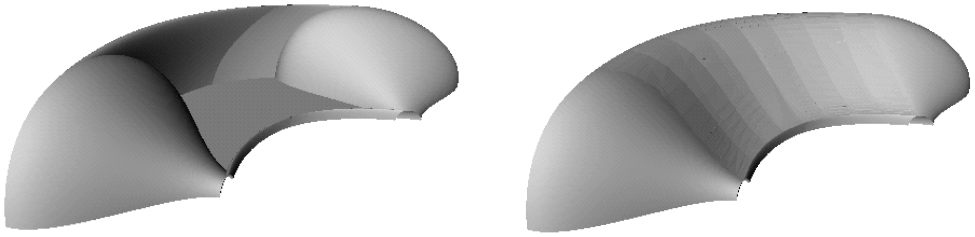


FIGURE 12: Reduction in the width of the Jacobian determinant approximation interval during the operation of the algorithm.



(a) (b)

FIGURE 13: Sweeping a bivariate patch along a curve. The boundary face patches are shown in (a), and the full solid is shown in (b).

9. Conclusions

We present a robust method that calculates the boundary surfaces of a trivariate tensor-product B-spline solid. We define the boundary surface as a combination of the boundary face patches of the solid – images of the two-dimensional boundaries of the domain space – and an implicit boundary surface, defined where the Jacobian determinant of the defining function is zero. We describe robust approximation methods, using cone approximates, to bound the Jacobian determinant over a region in the domain and use these approximates to generate an adaptive marching cubes algorithm that defines the isosurface. We present new methods, based upon tetrahedral decomposition, for patching the cracks in the isosurface generation routine.

Future work in this area is to use these techniques to describe the envelopes of general swept solids. Many of these swept surfaces can be described as a trivariate spline, and the techniques we present can be used. However, more general techniques are required to describe general surface envelopes.

10. Acknowledgments

This work was supported by the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP, the NASA Ames Research Center through an NRA award under contract NAG2-1216, the Lawrence Livermore National Laboratory through an ASCI ASAP Level-2 under contract W-7405-ENG-48 (and B335358, B347878), and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of Silicon Graphics, Inc.

We would like to thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, for their support.

References

[1] BARTELS, R., BEATTY, J., AND BARSKY, B. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Palo Alto, CA, 1987.

[2] BOEHM, W. Inserting new knots into B-spline curves. *Computer-Aided Design 12* (July 1980), 199–201.

[3] CASALE, M. S., AND STANTON, E. L. An overview of analytic solid modeling. *IEEE Computer Graphics and Applications 5*, 2 (Feb. 1985), 45–56.

[4] COHEN, E., LYCHE, T., AND RIESENFELD, R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Comput. Gr. Image Process. 14* (Oct. 1980), 87–111.

[5] FARIN, G. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1993.

[6] FAROUKI, R. T., AND HINDS, J. K. A hierarchy of geometric forms. *IEEE Computer Graphics and Applications 5*, 5 (May 1985), 51–78.

[7] JOY, K. I. Utilizing parametric hyperpatch methods for modeling and display of free-form solids. *Internat. J. Comput. Geom. Appl. 1*, 4 (1991), 455–471.

[8] JOY, K. I. Visualization of swept hyperpatch solids. In *Visual Computing : Integrating Computer Graphics with Computer Vision (Proceedings of Computer Graphics International 92)* (June 1992), T. L. Kunii, Ed., Springer-Verlag, Tokyo, pp. 567–582.

[9] KIM, D.-S. *Cones on Bézier Curves and Surfaces*. PhD thesis, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, June 1990.

[10] LASSER, D. Bernstein-bézier representation of volumes. *Computer Aided Geometric Design 2* (1985), 145–149.

[11] LORENSEN, W. E., AND CLINE, H. E. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (July 1987), 163–169.

[12] O'NEILL, B. *Elementary Differential Geometry*. Academic Press, New York, NY, 1966.

[13] RAPPOPORT, A., SHEFFER, A., AND BERCOVIER, M. Volume-Preserving Free-Form Solids. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (Mar. 1996), 19–27.

[14] REUS, J. F., MISH, K. D., AND JOY, K. I. Mechanical deformations of hyperpatch solids. In *Proceedings of Compugraphics '92* (Dec. 1992), pp. 147–158.

[15] SEDERBERG, T., AND MEYERS, R. Loop detection in surface patch intersections. *Computer Aided Geometric Design 5*, 2 (1988), 161–171.

[16] SEDERBERG, T. W., AND PARRY, S. R. Free-form deformation of solid geometric models. In *Computer Graphics (SIGGRAPH '86 Proceedings)* (Aug. 1986), D. C. Evans and R. J. Athay, Eds., vol. 20, pp. 151–160.

[17] SHU, R., ZHOU, C., AND KANKANHALLI, M. S. Adaptive marching cubes. *The Visual Computer 11*, 4 (1995), 202–217. ISSN 0178-2789.

[18] STANTON, E., CRAIN, L., AND NEW, T. A parametric cubic modelling system for general solids of composite material. *International Journal for Numerical Methods in Engineering 11* (1977), 653–670.

[19] WYVILL, B., MCPHEETERS, C., AND WYVILL, G. Animating soft objects. *The Visual Computer 2*, 4 (1986), 235–242.

[20] ZHOU, Y., CHEN, B., AND KAUFMAN, A. Multiresolution tetrahedral framework for visualizing regular volume data. In *IEEE Visualization '97* (Oct. 1997).