Representation and Extraction of Volumetric Attributes Using Trivariate Splines: A Mathematical Framework

William Martin Elaine Cohen

School of Computing University of Utah
http://www.cs.utah.edu/

Abstract

Our goal in this paper is to leverage traditional strengths from the geometric design and scientific visualization communities to produce a tool valuable to both. We present a method for representing and specifying attribute data across a trivariate NURBS volume. Some relevant attribute quantities include material composition and density, optical indices of refraction and dispersion, and data from medical imaging. The method is independent of the granularity of the physical geometry, allowing for a decoupling of the resolution of the carried data from that of the volume. Volume attributes can be modeled or fit to data.

A method is presented for efficient evaluation of trivariate NURBS. We incorporate methods for data analysis and visualization including isosurface extraction, planar slicing, volume ray tracing, and optical path tracing, all of which are grounded in refinement theory for splines. The applications for these techniques are diverse, including such fields as optics, fluid dynamics, and medical visualization.

Keywords: visualization, trivariate volume, isosurfacing, level sets.

1 Introduction

Volumes have long been important in the fields of scientific and medical visualization. MRI and CAT scanning devices produce a 3-dimensional photograph of the internal state of a subject. In the field of fluid dynamics pressure and velocity are spatially varying quantities whose values are critical to analysis. Likewise, turbidity and pollutant density play a large role in the simulation of atmospheric optics. The ability to deal with volumetric data is clearly a requirement.

True volumetric primitives are encountered less frequently in the field of computer-aided geometric design. Traditionally, boundary representations have been utilized extensively. This certainly makes sense for modeling solids having uniform interior. However, recent advances have led to manufacturing technologies supporting heterogeneous materials. For example, there are now machines with the capability to combine different source materials by percentage. In the area of optical design, lenses with continuously varying index of refraction are coming available — so called GRIN (Gradient Index) lenses. With these advances has come the need to model the interiors as well as the boundaries of objects. Finally, among the engineering community, there is often the desire to perform analyses on the products of design. These tests, such as temperature and stress simulation, generally involve propagation of attributes across an object's interior.

Traditionally, volumetric primitives have been grid based. This has served well among the scientific community, where the data is often regularly spaced along grid lines. This preference may change as adaptive 3D scanning technology becomes more common. Among the geometric design community, NURBS have been the *de facto* primitive of choice. In this article we advocate that a trivariate NURBS model may well serve the needs of both communities.

There are many advantages to the model we propose. First, it decouples geometric representation from attribute representation. This means that complicated geometries with simple attributes, and vice versa, may be represented at the resolution that best suits them. The result may be a large savings in storage and execution time. Furthermore, noise is an important variable in any visualization involving measured data. NURBS generally provide a robust representation for a signal containing moderate noise. Splines are a terse representation. By this, we mean that compared with polygons, or higher dimensional analogues such as voxels, splines generally represent a smooth function with fewer points.

For scientific and medical applications, either shape approximating or interpolating splines may be used with the attribute data, depending on whether a qualitative or more quantitative approach is required. From the CAGD perspective, an extended NURBS representation means that all of the existing algorithms can be applied in the new problem domain. We can consider modeling both the geometry and the attributes carried by the volume. Methods for visualization for design analysis can be borrowed from the visualization community.

We begin with a review of the existing literature and introduce our representation in Section 2. Section 3 deals with techniques for fitting data and modeling shape. Section 4 introduces an efficient method for evaluating trivariates, which is critical for large data sets. In Section 5, we adapt visualization techniques to our aggregate spline representation. We conclude and give future work in Section 6.

2 Background

Trivariate NURBS representations have been considered by a number of researchers. Early, Farouki and Hinds [2] gave a unified approach to curves, surfaces, and volumes. Lasser [5] explored the Bernstein-Bézier volume representation, and extended techniques for evaluation and interpolation to them. In her thesis, Paik [11] explored trivariates for modeling, and in particular, modeling operators, deformations, and animations. Kaufman [4] has combined

modeled objects and measured data within a volume visualization system, along with an algorithm for scan converting tricubic Béziers. Chang, et al. [1] provided a method for rendering volumes using line of sight integration and compositing, and an attribute volume representation similar to the one described here. A sculpting method introduced by Raviv and Elber [13] allows the user to sculpt a three dimensional object by modifying the scalar coefficients of a trivariate NURBS equation. Lee and Park [12] introduce an attribute model whose coefficients are generated from fluid flow data. Finally, Joy and Duchaineau [3] generate a complete representation for the boundary of a trivariate by unioning the faces with an implicit equation based on the Jacobian.

2.1 Trivariate Volumes

A non-uniform rational B-spline (NURBS) volume is a mapping $\mathbf{P}^w : \mathbb{R}^3 \rightarrow \mathbb{P}^3$ that can be formulated as

$$\mathbf{P}^w(u_1,u_2,u_3) = \sum_{i_1=0}^{N_1}\sum_{i_2=0}^{N_2}\sum_{i_3=0}^{N_3}\mathbf{P}^w_{i_1,i_2,i_3}B_{i_1,k_1}(u_1)B_{i_2,k_2}(u_2)B_{i_3,k_3}(u_3)$$

where the superscript w denotes that our formulation produces a point in rational four space. The omission of the superscript, as in $\mathbf{P}$, will denote the function which results from dividing $\mathbf{P}^w$ by its rational coordinate, a mapping $\mathbb{R}^3 \rightarrow \mathbb{R}^3$. The $\{\mathbf{P}^w_{a,b,c}\}$ are the control points $w_{a,b,c}(x_{a,b,c}, y_{a,b,c}, z_{a,b,c}, 1)$ of the $(N_1+1)\times(N_2+1)\times(N_3+1)$ mesh, having basis functions B_{i_j,k_j} of orders k_j with knot vectors $\tau^{\mathbf{j}} = \{\mathbf{u}^j_{i_j}\}_{i_j=0}^{N_j+k_j}$ for $j = 1,2,3$. For notational convenience, we may occasionally assign names to the variables such as u, v, w, where $(u,v,w) = (u_1,u_2,u_3)$.

Such a volume $\mathbf{P}^w$ is defined over the domain $\prod_{j=1}^3[\mathbf{u}^{\mathbf{j}}{}_{k_j-1}, \mathbf{u}^{\mathbf{j}}{}_{N_j+1})$, where we use $\prod$ to denote the Cartesian product. Each non-empty subinterval $\prod_{j=1}^3[\mathbf{u}^{\mathbf{j}}{}_{i_j}, \mathbf{u}^{\mathbf{j}}{}_{i_j+1})$ corresponds to a volume fragment.

The partial derivatives of $\mathbf{P}^w$ are likewise NURBS. Their control points are simply scaled differences of adjacent points in the original control mesh. For example, the control points for $\frac{\partial \mathbf{P}^w}{\partial u_1}$ are given by

$$(D_{u_1}\mathbf{P}^w)_{i_1,i_2,i_3} = \frac{k_1-1}{u^1_{i_1+k_1} - u^1_{i_1+1}}(\mathbf{P}^w_{i_1+1,i_2,i_3} - \mathbf{P}^w_{i_1,i_2,i_3}).$$

The corresponding derivative volume is then given by

$$\frac{\partial \mathbf{P}^w}{\partial u_1}(u_1,u_2,u_3) = \sum_{i_1=0}^{N_1-1}\sum_{i_2=0}^{N_2}\sum_{i_3=0}^{N_3}[(D_{u_1}\mathbf{P}^w)_{i_1,i_2,i_3} B_{i_1+1,k_1-1}(u_1)B_{i_2,k_2}(u_2)B_{i_3,k_3}(u_3)] \quad (1)$$

2.2 Aggregate Data Types

We now provide a representation which incorporates an arbitrary number of volume attributes. Each volume consists of a geometric description $\mathbf{P}^w$ and a set of attribute descriptions $\{\mathbf{A}^w_p\}$. Each attribute is represented as an independent trivariate volume

$$\mathbf{A}^w_p(u_1,u_2,u_3) = \sum_{i_1=0}^{N_{p1}}\sum_{i_2=0}^{N_{p2}}\sum_{i_3=0}^{N_{p3}}\mathbf{A}^w_p{}_{i_1,i_2,i_3}B_{i_1,k_{p_1}}(u_1)B_{i_2,k_{p_2}}(u_2)B_{i_3,k_{p_3}}(u_3)$$

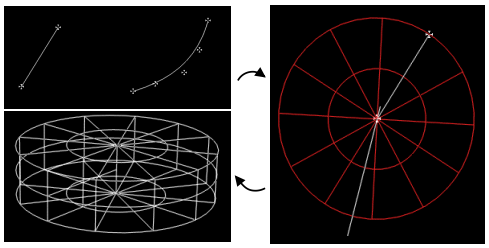

Figure 1: *Design example using aggregate objects.*

All aspects of the representation, e.g. order, dimensions of the control mesh, and knot vectors, are independent of the geometric trivariate representation. The only requirement is that all the volumes share the same parametric domain. In our implementation, we normalize all the domains to the unit cube I^3.

Two advantages of this representation merit mention. First, by decoupling the representation of the geometry from the attributes, and likewise, the attributes from one another, each function may be specified only to its required resolution. As volume data can be large, the so-called "curse of dimensionality," this may result in substantial savings. Furthermore, evaluation times will benefit for functions which are represented at differing orders (degrees). The second advantage of this scheme is that a trivariate NURBS package can be easily extended to include this representation. It does not increase the dimensionality of the points nor does it slow down existing routines.

3 Modeling and Data Fitting

Because the attributes themselves are represented as NURBS volumes, they can be modeled using any of the traditional operators. For example, we can model an attribute curve, extrude it into a surface, and rotate the surface about an axis to obtain a volume. While this might be useful, it can lead to some non-intuitive results, as the values of the attributes are contained in the coordinate values of the resulting objects. It is not immediately clear what meaning those operations have on the attribute functions.

For the purpose of modeling, we have developed a hierarchy of aggregate data types. Aggregate curves contain a geometry curve and an arbitrary number of attribute curves. Similarly, there are aggregate surfaces. At any stage of design, an attribute object may be combined with a geometric object to form an aggregate object. Additional attributes may be added to the aggregate data type at any later stage of design, as well. Once an attribute has been added to an aggregate data type, it becomes carried data. It can still be edited explicitly, but the default target of modification becomes the geometry. If the aggregate data type is mapped one level up the hierarchy, for example, by extruding the geometry curve into a surface, the carried attribute data is also generalized to a higher parametric dimension. By default a ruled object is created from the attribute objects contained in the original aggregate data types. If the geometric modeling operator only took one aggregate parameter, such as ruling or rotation about an axis, each attribute object is ruled with itself. That is, in the absence of additional data, we assume that the attribute varies only along the original parametric dimensions. If the mapping is applied to several aggregate objects, the ruling is applied across the corresponding attribute objects.

Let us consider a brief modeling example. Consider the two curves in Figure 1. The line on the left corresponds to geometry, and the curve on the right represents a particular attribute. Here, the y-coordinate contains the relevant attribute data, and it can be seen that the attribute varies with t in a non-linear way. We form

an aggregate curve from these two curves, and then sweep the geometry about an axis to obtain a disk. Finally, the disk is extruded to form a volume. From the previous discussion, it can be stated that the attribute data varies only radially from the rotation axis in the resulting volume. If we wish to edit the attribute curve, we can do so, and because of the dependency graph used by our modeling environment, all changes will propagate through the structure. Furthermore, we can explicitly request to modify the attribute objects at any level of the hierarchy. Thus if we want to allow variation along other parametric axes, we can do that now, in light of the completed geometric shape. The visualization techniques discussed in Section 5 can provide further information to guide these edits.

The visualization community often produces data by measurement or simulation. In such an instance, data fitting becomes a primary concern. If the qualitative shape is the most important thing, then the data points may simply be used as control points in the trivariate representation. In many cases it is desired that the functional approximation interpolate the data. It is this problem that we turn to address.

If the data points are $\mathbf{c}_{j_1,j_2,j_3}$ the problem is to find the control points $\mathbf{P}_{i_1,i_2,i_3}$ such that

$$\sum_{i_1,i_2,i_3} \mathbf{P}_{i_1,i_2,i_3} B_{i_1}(\tilde{u}^1_{j_1}) B_{i_2}(\tilde{u}^2_{j_2}) B_{i_3}(\tilde{u}^3_{j_3}) = \mathbf{c}_{j_1,j_2,j_3}$$

where $\tilde{u}^1_{j_1}, \tilde{u}^2_{j_2}, \tilde{u}^3_{j_3}$ are particular parameter values that correspond to the region of maximum influence for $\mathbf{P}_{j_1,j_2,j_3}$, called *nodal values* or Greville abscissas. The nodal values in u_1 for knot vector $\tau^1 = \{\mathbf{u}^1_j\}$ are given by

$$\tilde{u}^1_{i_1} = \frac{1}{k_1 - 1} \sum_{j=1}^{k_1 - 1} \mathbf{u}^1_{i_1 + j}$$

There are analogous formulations for $\tilde{u}^2_{i_2}$ and $\tilde{u}^3_{i_3}$.

In similar fashion to [12], we make the following definitions. Let

$$\mathbf{P}_{i_2,i_3}(\tilde{u}^1_{j_1}) \equiv \sum_{i_1=0}^{N_1} \mathbf{P}_{i_1,i_2,i_3} B_{i_1,k_1}(\tilde{u}^1_{j_1}) \quad (2)$$

$$\mathbf{P}_{i_3}(\tilde{u}^1_{j_1}, \tilde{u}^2_{j_2}) \equiv \sum_{i_2=0}^{N_2} \mathbf{P}_{i_2,i_3}(\tilde{u}^1_{j_1}) B_{i_2,k_2}(\tilde{u}^2_{j_2}) \quad (3)$$

Thus, we have

$$\sum_{i_3=0}^{N_3} \mathbf{P}_{i_3}(\tilde{u}^1_{j_1}, \tilde{u}^2_{j_2}) B_{i_3,k_3}(\tilde{u}^3_{j_3}) = \mathbf{c}_{j_1,j_2,j_3}$$

For each fixed j_1, j_2 pair, we can solve $N_3 + 1$ equations for $N_3 + 1$ unknowns to get $\{\mathbf{P}_{i_3}(\tilde{u}^1_{j_1}, \tilde{u}^2_{j_2})\}$. Likewise, for each fixed i_3, j_1 pair, we can solve (3) for $\{\mathbf{P}_{i_2,i_3}(\tilde{u}^1_{j_1})\}$. Finally, for each i_2, i_3, we can solve (2) for $\{\mathbf{P}_{i_1,i_2,i_3}\}$, thereby solving the interpolation problem.

Modeling and data fitting can certainly be combined. For example, suppose the object in Figure 1 represents a lens, with radially varying index of refraction. The attribute curve could have been generated by the data fitting technique discussed above. By importing the curve into our modeling program, it can now be used as a primitive for lens design.

4 Evaluation

Efficient evaluation schemes are critical to the success of any data representation — ever more so as the size of the dataset increases. In this section, we present a rapid evaluation scheme based in refinement.

We evaluate a curve $\mathbf{c}(t)$ with order k and knot vector τ by using refinement to stack $k - 1$ knots (where k is the order of the curve) at the desired parameter value t_*. The refined curve is defined over a new knot vector $\mathbf{t} = \{\mathbf{t_i}\}$ with basis functions $N_{i,k}(t)$ and new control points $\omega_i \mathbf{D}_i$.

Let $t_* \in [t_\mu, t_{\mu+1})$. Then,

$$\begin{aligned} \mathbf{c}(t_*) &= \mathbf{D}_{\mu-\mathbf{k}+\mathbf{1}} \\ \mathbf{c}'(t_*) &= \frac{(k-1)\omega_{\mu-k+2}}{(t_{\mu+1} - t_*)\omega_{\mu-k+1}} [\mathbf{D}_{\mu-\mathbf{k}+\mathbf{2}} - \mathbf{D}_{\mu-\mathbf{k}+\mathbf{1}}] \end{aligned} \quad (4)$$

(see [8]).

Analogously, in order to evaluate a trivariate volume $\mathbf{P}$ having knot vectors τ^u, τ^v, τ^w and orders k_u, k_v, k_w we stack $k_u - 1$, $k_v - 1$, and $k_w - 1$ knots valued u_*, v_* and w_*. If, with regard to the new knot vectors, $u_* \in [u_{\mu_u}, u_{\mu_u+1})$, $v_* \in [v_{\mu_v}, v_{\mu_v+1})$ and $w_* \in [w_{\mu_w}, w_{\mu_w+1})$, then

$$\mathbf{P}(u_*, v_*, w_*) = \mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}} \quad (5)$$

and

$$\begin{aligned} &\mathbf{P}_\mathbf{u}(u_*, v_*, w_*) = \\ &\quad \frac{(k_u - 1)\omega_{\mu_u-k_u+2,\mu_v-k_v+1,\mu_w-k_w+1}}{(u_{\mu_u+1} - u_*)\omega_{\mu_u-k_u+1,\mu_v-k_v+1,\mu_w-k_w+1}} \\ &\quad [\mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{2},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}} - \\ &\quad \mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}}] \end{aligned} \quad (6)$$

$$\begin{aligned} &\mathbf{P}_\mathbf{v}(u_*, v_*, w_*) = \\ &\quad \frac{(k_v - 1)\omega_{\mu_u-k_u+1,\mu_v-k_v+2,\mu_w-k_w+1}}{(v_{\mu_v+1} - v_*)\omega_{\mu_u-k_u+1,\mu_v-k_v+1,\mu_w-k_w+1}} \\ &\quad [\mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{2},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}} - \\ &\quad \mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}}] \end{aligned}$$

$$\begin{aligned} &\mathbf{P}_\mathbf{w}(u_*, v_*, w_*) = \\ &\quad \frac{(k_w - 1)\omega_{\mu_u-k_u+1,\mu_v-k_v+1,\mu_w-k_w+2}}{(w_{\mu_w+1} - w_*)\omega_{\mu_u-k_u+1,\mu_v-k_v+1,\mu_w-k_w+1}} \\ &\quad [\mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{2}} - \\ &\quad \mathbf{D}_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}}] \end{aligned}$$

In order to perform these evaluations, we must first generate the refinement matrices which map the old control points into the new ones. For curves, the refinement problem can be written as $\mathbf{D} = A_u \mathbf{P}$, and for surfaces $\mathbf{D} = A_u \mathbf{P} A_v^T$. However, as the parametric dimensionality increases, this type of notation no longer suffices. We introduce a new notation which generalizes to any number of parametric dimensions.

We define the operator $\otimes^k$ such that

$$\mathbf{D} = A \otimes^k \mathbf{C} \equiv (\mathbf{D})_{c_0,c_1,\ldots,c_{k-1},i,c_{k+1},\ldots c_M} = \sum_j (A)_{i,j} (\mathbf{C})_{c_0,c_1,\ldots,c_{k-1},j,c_{k+1},\ldots c_M}$$

For curves, then

$$\mathbf{D^0} = A_u \otimes^0 \mathbf{C} \equiv (\mathbf{D^0})_i = \sum_j (A_u)_{i,j} (\mathbf{C})_j = A_u \mathbf{C}$$

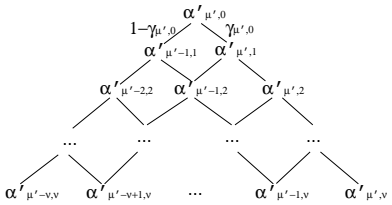

Figure 2: Graph showing how the factor $\gamma_{\mu',0}$ propagates through the recurrence.

Likewise, for surfaces

$$\begin{aligned}\mathbf{D}^{1,0} &= A_v \otimes^1 (A_u \otimes^0 \mathbf{C}) \\ &= A_v \otimes^1 \mathbf{D^0} \\ &= (\mathbf{D}^{1,0})_{i,j} \equiv \sum_l (A_v)_{j,l} (\mathbf{D^0})_{i,l} \\ &= \mathbf{D^0} A_v^T \\ &= A_u \mathbf{C} A_v^T\end{aligned}$$

which is what would be expected. The operator generalizes to n dimensions. For trivariates, the evaluation refinement given at the beginning of this section can then be denoted

$$A_w \otimes^2 (A_v \otimes^1 (A_u \otimes^0 \mathbf{P}^w))$$

Returning to the curve case (4), we are not interested in calculating the full alpha matrix $\mathbf{A}$, but merely rows $\mu - k + 2$ and $\mu - k + 1$, as these are used to generate the points $\mathbf{D}^\omega_{\mu-\mathbf{k}+\mathbf{2}}$ and $\mathbf{D}^\omega_{\mu-\mathbf{k}+\mathbf{1}}$ which are required for point and derivative evaluation.

Suppose $t_* \in [\tau_{\mu'}, \tau_{\mu'+1})$. We can generate the refinement for row $\mu + k - 1$ using a triangular scheme

$$\begin{array}{cc} & \alpha'_{\mu',0} \\ \alpha'_{\mu'-1,1} & \alpha'_{\mu',1} \\ \vdots & \vdots \\ \alpha'_{\mu'-\nu,\nu} \quad \cdots & \alpha'_{\mu',\nu} \end{array}$$

where ν is the number of knots we are inserting and

$$\begin{aligned}\alpha'_{j,1} &= \delta_{j,\mu'} \\ \alpha'_{j,p+1} &= \gamma_{j,p}\alpha'_{j,p} + (1-\gamma_{j+1,p})\alpha'_{j+1,p} \\ \gamma_{j,p} &= (t_* - \tau_{\mu'-p+j-(k-1-\nu)})/d\end{aligned}$$

$\mathbf{A}_{\mu-k+1,j} = \alpha'_{j,\nu}$ for $j = \mu' - \nu, \cdots, \mu'$ and $\mathbf{A}_{i,j} = 0$ otherwise. If n knots exist in the original knot vector τ with value t_*, then $\nu = \max\{k - 1 - n, 1\}$ — that is to say, we always insert at least 1 knot. The quantity ν is used in the triangular scheme above to allow one to skip those basis functions which are trivially 0 or 1 due to repeated knots. As a result of this triangular scheme, we generate basis functions in place and avoid redundant computation of α' values for subsequent levels.

In the refinement scheme we propose, the point on the curve $\mathbf{D}^\omega_{\mu-\mathbf{k}+\mathbf{1}}$ will be a convex blend of the points $\mathbf{D}^\omega_{\mu-\mathbf{k}}$ and $\mathbf{D}^\omega_{\mu-\mathbf{k}+\mathbf{2}}$. The blend factor will be $\gamma_{\mu',0}$. The dependency graph shown in Figure 2 will help to clarify. The factor $\gamma_{\mu',0}$ is introduced at the first level of the recurrence. The leaf terms can be written as

$$\alpha'_{j,\nu} = (1 - \gamma_{\mu',0}) l_{j,\nu} + \gamma_{\mu',0} r_{j,\nu}$$

with $j = \mu' - \nu, \cdots, \mu'$. $\{l_{j,\nu}\}$ and $\{r_{j,\nu}\}$ are those terms dependent on $\alpha'_{\mu-1,1}$ and $\alpha'_{\mu,1}$ respectively. They are the elements of

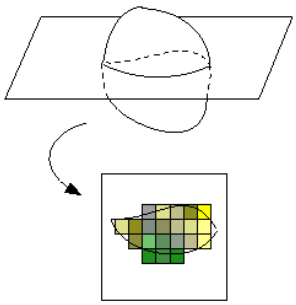

Figure 3: *Planar cut.*

the alpha matrix rows $\mu - k$ and $\mu - k + 2$ with $\mathbf{A}_{\mu-k,j} = l_{j,\nu}$ and $\mathbf{A}_{\mu-k+2,j} = r_{j,\nu}$ for $j = \mu' - \nu, \cdots, \mu'$. We can generate the $\{l_{j,\nu}\}$ by setting $\alpha'_{\mu'-1,1} = 1$ and $\alpha'_{\mu',1} = 0$ and likewise, generate $\{r_{j,\nu}\}$ by setting $\alpha'_{\mu'-1,1} = 0$ and $\alpha'_{\mu',1} = 1$. Thus, $\mathbf{A}_{\mu-k,j}$ and $\mathbf{A}_{\mu-k+2,j}$ can be generated in the course of generating $\mathbf{A}_{\mu-k+1,j}$ at little additional expense.

To produce the desired points for (5) we only need to evaluate

$$\begin{aligned}&\mathbf{D}^\omega_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{w}-\mathbf{k_w}+\mathbf{1}} = \\ &\quad (\mathbf{A_u})_{\mu_u+k_u+1,[\mu'_u-\nu_u\ldots\mu'_u]} \otimes^0 \\ &\quad (\mathbf{A_v})_{\mu_v+k_v+1,[\mu'_v-\nu_v\ldots\mu'_v]} \otimes^1 \\ &\quad (\mathbf{A_w})_{\mu_w+k_w+1,[\mu'_w-\nu_w\ldots\mu'_w]} \otimes^2 \\ &\quad \mathbf{P}^\mathbf{w}_{[\mu'_\mathbf{u}-\nu_\mathbf{u}\ldots\mu'_\mathbf{u}][\mu'_\mathbf{v}-\nu_\mathbf{v}\ldots\mu'_\mathbf{v}][\mu'_\mathbf{w}-\nu_\mathbf{w}\ldots\mu'_\mathbf{w}]}\end{aligned}$$

To calculate (7), we substitute $(\mathbf{A_u})_{\mu_u+k_u+2,[\mu'_u-\nu_u\ldots\mu'_u]}$ for $(\mathbf{A_u})_{\mu_u+k_u+1,[\mu'_u-\nu_u\ldots\mu'_u]}$ in the above expression to obtain $\mathbf{D}^\omega_{\mu_\mathbf{u}-\mathbf{k_u}+\mathbf{2},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1},\mu_\mathbf{v}-\mathbf{k_v}+\mathbf{1}}$, and similarly for $\mathbf{D}^\omega_{\mu_u-k_u+1,\mu_v-k_v+2,\mu_v-k_v+1}$ and $\mathbf{D}^\omega_{\mu_u-k_u+1,\mu_v-k_v+1,\mu_v-k_v+2}$. This can be made quite efficient.

5 Visualization Techniques

5.1 Isosurfacing and Slicing

In this section we provide a unified approach to two common methods of data visualization. An isosurface with respect to a data attribute $\mathbf{A}_p$ is the set of points within the volume having a particular value for $\mathbf{A}_p$. The display of isosurfaces within a volume gives useful information about the variation of $\mathbf{A}_p$. Another visualization technique is to create a planar slice of the volume, and color code it according to a given attribute (see Figure 3).

The isosurfacing problem can be formalized as finding the set of points which obey the equation

$$\begin{aligned}\mathbf{A}_p(\mathbf{u}) &= \frac{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} \mathbf{A}_{p_{i_1,i_2,i_3}} \mathbf{B}_{i_1,i_2,i_3}(u_1,u_2,u_3)}{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} \mathbf{B}_{i_1,i_2,i_3}(u_1,u_2,u_3)} \\ &= \mathbf{A}_* \\ &= \frac{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} \mathbf{A}_* \mathbf{B}_{i_1,i_2,i_3}(u_1,u_2,u_3)}{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} \mathbf{B}_{i_1,i_2,i_3}(u_1,u_2,u_3)}\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{0} &= \frac{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} (\mathbf{A}_{p_{i_1,i_2,i_3}} - \mathbf{A}_*) \mathbf{B}_{i_1,i_2,i_3}(\mathbf{u})}{\sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} \mathbf{B}_{i_1,i_2,i_3}(\mathbf{u})} \\ &= \sum_{i_1,i_2,i_3} w_{p_{i_1,i_2,i_3}} (\mathbf{A}_{p_{i_1,i_2,i_3}} - \mathbf{A}_*) \mathbf{B}_{i_1,i_2,i_3}(\mathbf{u})\end{aligned}$$

If all the weights $w_{p_{i_1,i_2,i_3}}$ are positive, then for a root to be possible on an interval I, the bounding box of the associated control points must contain the origin. In the case of scalar $\mathbf{A}_*$, this means that the difference $(\mathbf{A}_{p_{i_1,i_2,i_3}} - \mathbf{A}_*)$ must change signs.

If we are given a plane $\mathbf{p} = \{\mathbf{x} : (a,b,c,d) \cdot (\mathbf{x},1) = 0\}$, then the points in $\mathbf{P}^w$ which are sliced by $\mathbf{p}$ are given by

$$\begin{aligned} 0 &= \frac{\sum_{i_1,i_2,i_3} w_{i_1,i_2,i_3}(\mathbf{P}_{i_1,i_2,i_3} \cdot (a,b,c,d))\mathbf{B}_{i_1,i_2,i_3}(\mathbf{u})}{\sum_{i_1,i_2,i_3} w_{i_1,i_2,i_3}\mathbf{B}_{i_1,i_2,i_3}(\mathbf{u})} \\ &= \sum_{i_1,i_2,i_3} (w_{i_1,i_2,i_3}\mathbf{P}_{i_1,i_2,i_3}) \cdot (a,b,c,d)\mathbf{B}_{i_1,i_2,i_3}(\mathbf{u}) \end{aligned}$$

In both the case of isosurfacing and planar slicing, we are left with the problem of finding the zeros of a trivariate spline having the form

$$\mathbf{D}(u_1,u_2,u_3) = \sum_{i_1,i_2,i_3} \mathbf{D}_{i_1,i_2,i_3}\mathbf{B}_{i_1,i_2,i_3}(u_1,u_2,u_3).$$

To solve the problem, we first break the $\mathbf{D}$ into Bézier volumes and place these into a search list. For each element in the list, we test whether the control points are within an epsilon box of zero. If so, this volume is added to the root list. If not, we test whether the bounding box of the points contains the origin. If the answer is yes, the volume is subdivided and the resulting volumes are appended to the list. On the other hand, if the answer is no, the volume is discarded. This procedure continues until no further volumes are in the search list.

The result of the outlined procedure is a list of volumes which contain the roots of $\mathbf{D}$. We now have to refine $\mathbf{P}^w$ at the interval values which describe the domains for the volumes in the root list. The resulting geometric volumes may need to be further refined according to a flatness criterion. A polygonal approximation can then be displayed. In the case of planar slicing, the resulting polygons may be colored according to the average isovalue in each volume.

5.2 Tracing Rays

It is a common technique to map scalar values to colors and visualize volumetric data by passing rays through it. Mathematically, this operation can be formulated as the calculation

$$\int_{t_a}^{t_b} \alpha(\mathbf{o}+t\mathbf{d})\mathbf{C}(\mathbf{o}+t\mathbf{d})dt,$$

where $\alpha(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ are the accumulated opacity and color values corresponding the attributes at the point $\mathbf{x}$, and t_a, t_b are the entry and exit points of the ray $\mathbf{o}+t\mathbf{d}$ with the volume $\mathbf{P}(u,v,w)$. It is clear that under energy conservation, $\alpha(\mathbf{x}) \leq 1$. In this section, we provide the machinery for ray tracing volumetric splines.

A trivariate spline $\mathbf{P}(u,v,w)$ is a mapping from the rectangular cell $I_u \times I_v \times I_w$ to $\mathbb{R}^3$. The faces of the domain cell, $\partial I_u \times I_v \times I_w$, $I_u \times \partial I_v \times I_w$, and $I_u \times I_v \times \partial I_w$ map to surfaces in $\mathbb{R}^3$. These surfaces necessarily bound a closed volume, as they share boundaries and are collectively equivalent topologically to a cube. However, as shown in [3], the faces need not enclose the same volume as does $\partial\mathbf{P}(u,v,w)$.

Theorem 1 *Given a rectangular 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, the surface boundary of the solid $\mathbf{P}$ is contained within the union of the faces of the solid over B and the points where the determinant of the Jacobian of $\mathbf{P}$ over B vanishes.*

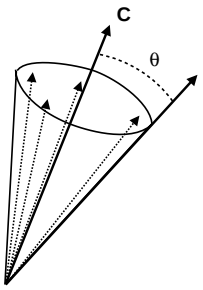

Figure 4: *A bounding cone.*

This theorem was first brought to our attention in [3] and a proof can be found in [10].

To trace a ray through the volume $\mathbf{P}$, we first wish to find the closest point at which the ray $\mathbf{o}+t\mathbf{d}$ contacts the boundary surface $\partial\mathbf{P}$. If a ray is defined as the intersection of two planes $\mathbf{p_1}, \mathbf{p_2}$, where $\mathbf{p_l} = \{\mathbf{x} : (a_l,b_l,c_l,d_l) \cdot (\mathbf{x},1) = 0\}, l = 1,2$, then a ray intersecting $\partial\mathbf{P}$ will satisfy at least one of the following relations:

$$\mathbf{F^k}(s,t) \cdot (a_l,b_l,c_l,d_l) = 0, \text{ for } l = 1,2 \tag{7}$$

or

$$\begin{aligned} \mathbf{P}(u,v,w) \cdot (a_l,b_l,c_l,d_l) &= 0, \text{ for } l = 1,2 \\ \mathcal{J}\mathbf{P}(u,v,w) &= 0 \end{aligned} \tag{8}$$

where $\mathbf{F^k}, k = 1..6$ are the faces of $\mathbf{P}$ and $\mathcal{J}\mathbf{P}$ is the determinant of the Jacobian matrix for $\mathbf{P}$. This is simply a restatement of Theorem 1.

As formulas (7) and (8) are implicit equations, we can apply Newton's method to find the roots, and therefore, the intersection of the ray with $\partial\mathbf{P}$, provided we have a good initial guess. With each face, we generate and store a bounding volume hierarchy using subdivision according to a flatness criteria. This is a preprocessing step. See [8] for a detailed discussion. Boxes which do not intersect the ray are culled, and we apply a Newton's method to (7) using the starting value associated with each of the remaining boxes.

To handle implicit boundaries (8), we store with each volume $\mathbf{P}$ a bounding hierarchy obtained by subdividing $\mathbf{P}$ until each piece is within a maximum volume tolerance. We cull those boxes whose volumes cannot contain a zero Jacobian determinant according to the following method due to [3].

A cone is determined by a normalized axis vector $\hat{\mathbf{C}}$ and a spread angle θ (Figure 4). In what follows, a "$\hat{}$", as in $\hat{\mathbf{v}}$, will denote the normalized form of a vector. Given a set of vectors $\{\mathbf{v_i}\}$, we can fit a bounding cone to them using the following algorithm due to [14]. Set $\hat{\mathbf{C_0}} = \mathbf{v_0}/||\mathbf{v_0}||$, and $\theta_0 = 0$. For each subsequent vector $\mathbf{v_i}$, the angle α between $\mathbf{v_i}$ and $\hat{\mathbf{C}}_{\mathbf{i-1}}$ is given by $\alpha = \cos^{-1}(\hat{\mathbf{v}}_\mathbf{i} \cdot \hat{\mathbf{C}}_{\mathbf{i-1}})$. For $\alpha \leq \theta_{i-1}$, $\hat{\mathbf{C}}_\mathbf{i} = \hat{\mathbf{C}}_{\mathbf{i-1}}, \theta_i = \theta_{i-1}$. Otherwise, we compute an intermediate vector $\mathbf{v_t}$

$$\mathbf{v_t} = \begin{cases} \mathbf{v_{i-1}}, & \text{if } \theta_{i-1} = 0 \\ \cot\theta_{i-1}\sin\alpha\hat{\mathbf{C}}_{\mathbf{i-1}} - \hat{\mathbf{v}}_\mathbf{i}, & \text{otherwise.} \end{cases}$$

We have that

$$\hat{\mathbf{C}}_\mathbf{i} = \frac{\hat{\mathbf{v}}_\mathbf{i}+\hat{\mathbf{v}}_\mathbf{t}}{||\hat{\mathbf{v}}_\mathbf{i}+\hat{\mathbf{v}}_\mathbf{t}||}, \quad \cos\theta_i = \hat{\mathbf{C}}_\mathbf{i} \cdot \hat{\mathbf{v}}_\mathbf{t}$$

We extend the dot product and cross product operators to cones in the following way. Given two cones C_1, C_2, $C_1 \cdot C_2$ is the range of scalar product values for vectors bounded by C_1 and C_2. Analogously, $C_1 \times C_2$ is the cone C_3 which bounds the cross-products

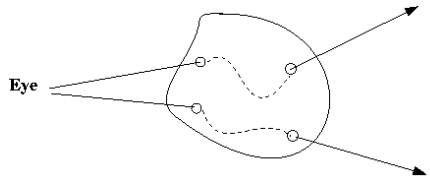

Figure 5: *Ray paths through medium with varying refractive index.*

of vectors bounded by C_1 and C_2. A conservative estimate of the cross product is accomplished by crossing the cone axes, and calculating the spread angle θ_3 via:

$$\theta_3 = \sin^{-1}\left(\frac{\sqrt{\sin^2\theta_1 + 2\sin\theta_1\sin\theta_2\cos\beta + \sin^2\theta_2}}{\sin\beta}\right)$$

where β is the smaller of the angles between the two cone axes [14].

We know that the partial derivatives of a NURBS volume $\mathbf{P}^w$ are again NURBS volumes (1). Consider the cones C_u, C_v, C_w which bound the homogenized ($\mathbb{R}^3$) control points $(\mathcal{D}_u\mathbf{P})_{i_1,i_2,i_3}$, $(\mathcal{D}_v\mathbf{P})_{i_1,i_2,i_3}$, $(\mathcal{D}_w\mathbf{P})_{i_1,i_2,i_3}$, respectively, of the derivative volumes. By virtue of the convex hull property, we have that

$$\mathcal{J}(\mathbf{P}) \equiv \mathcal{D}_u\mathbf{P}\cdot(\mathcal{D}_v\mathbf{P}\times\mathcal{D}_w\mathbf{P}) \subseteq L(C_u\cdot(C_v\times C_w)),$$

where L is an interval of positive values [3]. This implies that $\mathcal{J}\mathbf{P} \neq 0$ in the given volume if $0 \notin C_u\cdot(C_v\times C_w)$. We can remove bounding boxes containing such volumes from consideration. As before, we can also cull those boxes which the ray does not intersect, and apply the Newton iteration to (8) using the start (e.g., average parameter) values stored in the remaining boxes.

The result will be a list of points where the ray $\mathbf{p_0} + \mathbf{v_0}t$ intersects $\partial\mathbf{P}$. For each volume there should be a pair of points: an entry and an exit point. Let the point closest to the ray origin have coordinates $\mathbf{u_1}$, $\mathbf{p_1}$, where $\mathbf{u_1} = (u_1,u_2,u_3)_1^T$. We evaluate the attribute data at the point $\mathbf{u_1}$, and obtain an opacity α_1 and a color $\mathbf{C_1}$. The accumulated color $\mathbf{C_{acc}} = \alpha_1 * \mathbf{C_1}$ and the accumulated opacity $\alpha_{acc} = 1-\alpha_1$.

Now we begin to traverse the volume. Starting from $\mathbf{p_1}$ and traveling a small distance Δt along the vector $\mathbf{v_1}$, we arrive at the point $\mathbf{p}_*$. Since $\mathbf{p}_*$ is close to $\mathbf{p_1}$, we are justified in the approximation $\mathbf{p}_* - \mathbf{p_1} \approx \mathbf{JP}(\mathbf{u_1})(\mathbf{u}_* - \mathbf{u_1})$, where $\mathbf{JP}(\mathbf{u})$ is the Jacobian matrix, $\mathbf{u}_k = (u_1,u_2,u_3)_k^T$ and we know $(u_1,u_2,u_3)_1^T$ from the previous Newton iteration. This leads to the Newton iteration

$$[\mathbf{JP}(\mathbf{u_k})]^{-1}(\mathbf{p}_* - \mathbf{P}(\mathbf{u_k})) + \mathbf{u}_k = \mathbf{u}_{k+1} \tag{9}$$

Note that the functions $\mathbf{P}$ and $\mathbf{JP}$ lack the superscript ω. This denotes a projection into $\mathbb{R}^3$. $\mathbf{P}(\mathbf{u_k})$ is found by dividing $\mathbf{P}^\omega(\mathbf{u_k})$ by its rational coordinate. In similar fashion $\mathbf{JP}(\mathbf{u_k}) = [\mathbf{P_u}(\mathbf{u_k})\ \mathbf{P_v}(\mathbf{u_k})\ \mathbf{P_w}(\mathbf{u_k})]$ is found by computing $\mathbf{P}^\omega_\mathbf{u}(\mathbf{u_k})$, $\mathbf{P}^\omega_\mathbf{v}(\mathbf{u_k})$, and $\mathbf{P}^\omega_\mathbf{w}(\mathbf{u_k})$ and homogenizing each.

From (9) we obtain $\mathbf{u}_*$, which we can again use to calculate the attribute data and corresponding color and opacity functions, $\mathbf{C_2}$ and α_2. We increment the accumulated colors and opacities $\mathbf{C_{acc}} = \mathbf{C_{acc}} + \alpha_{acc} * \alpha_2 * \mathbf{C_2}$ and $\alpha_{acc} = \alpha_{acc} * (1-\alpha_2)$.

This process is repeated until the ray exits the volume. The color of the ray can be written as $\mathbf{C_{acc}} = \sum_{i=1}^{\infty}\alpha_i\mathbf{C_i}\prod_{j=1}^{i-1}(1-\alpha_j)$.

5.3 Optical Path Tracing

An application that sometimes occurs in optics is the desire to trace the path of a ray which is perturbed by a spatially varying refractive index. Some example applications are visualizing atmospheric effects such as thermal clines near the ground, metropolitan pollution, atmospheric perspective, and cutting-edge optical lenses such as GRIN (gradient index of refraction) lenses. The attribute data for such a volume would include a model of the refractive index, $\eta(u,v,w)$, at each point of its interior.

The well-known Snell's law formula at the interface between discrete media is $\eta_0\sin\theta_0 = \eta_1\sin\theta_1$, where θ_0, θ_1 are measured between the normal and the ray. The formula also holds true in a volume with a varying refractive index. The interface in the discrete formula corresponds to the isosurface with constant η in the volume. A ray with direction $\mathbf{v}$ is perturbed with respect to the normal $\mathbf{n}$ of the isosurface which it contacts. Since this normal will by necessity point in the direction of maximum change in η, $\mathbf{n} = \nabla\eta$. It follows that the path of a ray will in general trace a curved path through a medium with continuously varying refractive index.

The gradient $\nabla\eta$ is given by $(\partial\eta/\partial x, \partial\eta/\partial y, \partial\eta/\partial z)$. By the chain rule, we have that

$$\begin{pmatrix} \partial\eta/\partial u \\ \partial\eta/\partial v \\ \partial\eta/\partial w \end{pmatrix} = \begin{pmatrix} \partial x/\partial u & \partial y/\partial u & \partial z/\partial u \\ \partial x/\partial v & \partial y/\partial v & \partial z/\partial v \\ \partial x/\partial w & \partial y/\partial w & \partial z/\partial w \end{pmatrix}\begin{pmatrix} \partial\eta/\partial x \\ \partial\eta/\partial y \\ \partial\eta/\partial z \end{pmatrix}$$

yielding that

$$\nabla\eta = (\mathbf{JP}^T)^{-1}(\partial\eta/\partial u, \partial\eta/\partial v, \partial\eta/\partial w)^T. \tag{10}$$

The method described in section 5.2 may be modified as follows. We calculate the intersection of the ray $\mathbf{p_0} + t\mathbf{v_0}$ with the boundary of the volume $\partial\mathbf{P}$, yielding points $\mathbf{u_1}$ and $\mathbf{p_1}$. We evaluate η and $\mathbf{n_1} = \nabla\eta$ at $\mathbf{u_1}$ to generate a new ray direction $\mathbf{v_1}$. $\mathbf{n_1}$ is reflected, if necessary, so that $\mathbf{n_1}\cdot\mathbf{v_0} < 0$. It must be the case that $\mathbf{v_1}$ is in the plane containing $\mathbf{v_0}$ and $\mathbf{n_1}$. We can generate a vector tangent to the isosurface $\mathbf{t_1} = \mathbf{n_1}\times(\mathbf{v_0}\times\mathbf{n_1})$. As a byproduct of this computation, we obtain $\sin(\theta_0) = ||\mathbf{v_0}\times\mathbf{n_1}||$. The angle between $-\mathbf{n_1}$ and $\mathbf{v_1}$ is given by $\theta_1 = \arcsin(\eta_0\sin(\theta_0)/\eta_1)$, where η_0 is 1 for air. The outgoing ray direction is then given by $\mathbf{v_1} = \sin\theta_1\mathbf{t_1} + \cos\theta_1\mathbf{n_1}$. We walk a distance Δt as before, determining points $\mathbf{u_2}$, $\mathbf{p_2}$, and the perturbation is calculated using the η_1, $\eta_2 = \eta(\mathbf{u_2})$, and $\mathbf{n_2} = \nabla\eta(\mathbf{u_2})$. Note that in the process of calculating (9), we have calculated the Jacobian needed for (10).

As a final note, the stepsize Δt can be made to depend on the gradient $\nabla\eta$. The larger the gradient magnitude, the shorter the distance we can cover without missing something.

5.4 Summary of Ray Tracing

For clarity, we summarize with an algorithm:

```
TraceRay ( Environment env, Ray r, Color C)
  loop
    for each Volume vol in env do
      IntersectBoundary(r,vol,hitlist)
    end for
    if hitlist = NULL then
      return
    end if
    vol = hitlist.closest
    uv = r.hit.uv
    p = r.hit.p
    n = r.hit.normal
    while p ∈ vol do
      r.o = p
      PerturbRayDirection(r,vol,uv)
      ComputeColor(C,vol,uv)
      p = r.o + Δt * r.v
      CalcParametricPoint(r.o,p,uv)
    end while
    r.o = p
    PerturbRayDirection(r,vol,uv)
  end loop
```

6 Conclusion and Future Work

This paper has introduced a framework for representing attribute data orthogonally to geometric data within a trivariate NURBS volume. It extends existing modeling and data fitting techniques to this new representation and presents an efficient algorithm for volume evaluation. In addition, we have incorporated techniques for data visualization such as planar slicing, isosurfacing, ray tracing, and optical path tracing which may serve as invaluable aids to composite design or data analysis.

From the modeling standpoint, there is much yet to be done. Tensor-product surfaces are a deformation of a rectangle, and therefore limited topologically. In the past, trimming curves have provided added flexibility to surface design. In the case of volumes, the problem is decidedly harder, and a practical solution is not yet known. The scheme we prescribe generalizes in a straight-forward manner to subdivision surfaces. On the other hand, greater flexibility leads to a reduction in performance.

Splines lend themselves to multi-resolution methods. For example, modeling can be expedited if changes are made from coarse to fine. As another example, evaluations often only need to be made within a given error tolerance, opening the door to potential savings. We suspect further exploration of these traits will prove fruitful.

7 Acknowledgments

We would like to thank Peter Shirley, Michael M. Stark, and the reviewers for their careful critical review and comments. This work was supported in part by DARPA grant N00014-98-C-0225, the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219), and NSF grants 96-23614, 97-96136, 97-31859, and 98-18344. All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies

References

[1] Y.-K. Chang, A. P. Rockwood, and Q. He. Direct rendering of freeform volumes. *Computer-aided Design*, 27(7):553–558, 1995.

[2] R. Farouki and J. Hinds. A hierarchy of geometric forms. *IEEE Computer Graphics & Applications*, 5(5):51–78, 1985.

[3] Kenneth Joy and Mark Duchaineau. Boundary determination for trivariate solids. *Pacific Graphics '99*, October 1999. Held in Seoul, Korea.

[4] Arie Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):171–179, July 1987. Held in Anaheim, California.

[5] D. Lasser. Bernstein-Bézier representation of volumes. *Computer Aided Geometric Design*, 2(1-3):145–150, 1985.

[6] T. Lyche, E. Cohen, and K. Morken. Knot line refinement algorithms for tensor product B-spline surfaces. *Computer Aided Geometric Design*, 2(1-3):133–139, 1985.

[7] M. M. Madi and D. J. Walton. Modeling and visualization of layered objects. *Computers & Graphics*, 23(3):331–342, June 1999. ISSN 0097-8493.

[8] William Martin, Elaine Cohen, Russell Fish, and Peter Shirley. Practical ray tracing of trimmed NURBS surfaces. *Journal of Graphics Tools*, 5(1):27–52, 2000.

[9] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995. ISSN 1077-2626.

[10] B. O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[11] Karen Lynn Paik. Trivariate splines. Master's thesis, Department of Computer Science, University of Utah, December 1992.

[12] Sangkun Park and Lee Kunwoo. High-dimensional trivariate NURBS representation for analyzing and visualizing fluid flow data. *Computers & Graphics*, 21(4):473–482, July 1997. ISSN 0097-8493.

[13] A. Raviv and G. Elber. Three-dimensional freeform sculpting via zero sets of scalar trivariate functions. *Computer-Aided Design*, 32(8-9):513–526, August 2000. ISSN 0010-4485.

[14] T. Sederberg and R. Meyers. Loop detection in surface patch intersections. *Computer Aided Geometric Design*, 5(2):161–171, 1988.

[15] Peter L. Williams and Nelson Max. A volume density optical model. *1992 Workshop on Volume Visualization*, pages 61–68, 1992.