Rendering of Surfaces from Volumetric Data

June 1, 1987
Doc. 870601
TR87-016

Marc Levoy

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27514

Rendering of Surfaces from Volumetric Data

Marc Levoy

Computer Science Department
University of North Carolina
Chapel Hill, NC 27514

Abstract

A new approach for rendering discrete volumetric data is presented. Surface shading calculations are performed at every voxel using local gradient vectors as surface normals. In a separate step, feature classification operators are applied to obtain partial opacities for every voxel. Operators that detect isovalue surfaces and region boundary surfaces are presented. Independence of shading and classification calculations insures that undistorted visualizations of 3-D shapes are obtained. Non-binary operators insure that small or poorly defined features are not lost. The resulting colors and opacities are digitally composited from back to front along view rays to form an image. The technique exhibits smooth silhouettes and few other aliasing artifacts. Examples from two application areas are given: protein crystallography and medical imaging.

1. Introduction

In classical image synthesis, surfaces are modeled using a variety of geometric primitives such as polygons or curved patches. Rendering consists of converting this database into an array of pixels for viewing on a raster display. There is, however, a growing list of applications for which data is acquired empirically rather than generated synthetically. Typical sources include sensing devices and computer simulations. Unlike synthetic data, acquired data is typically unstructured upon receipt. In this paper, we focus on sampled scalar functions of three spatial dimensions, henceforth referred to as *volumetric data*. In order to use local gradients for shading and classification, we further restrict ourselves to samples drawn from continuous differentiable functions. While this criteria excludes treatment of some phenomena, the class of acceptable functions is sufficiently large to be both useful and interesting.

The currently dominant technique for presenting this data involves fitting surface primitives to the sampled function, then rendering these primitives using classical image synthesis. There are several drawbacks to this approach. Fitting surfaces to acquired data is a hard problem. It is computationally expensive and often requires manual intervention. Low-order geometric primitives are also mediocre reconstruction filters, giving rise to artifacts in the rendered image.

To avoid these problems, researchers have introduced *volumetric rendering* wherein the intermediate surface representation is omitted. Images are formed by directly shading each sample point and projecting it onto the picture plane. Since no data reduction is performed prior to rendering, the selection of appropriate shading parameters becomes critical to the perception of shapes and spatial relationships in the image.

Volumetric rendering is not cheap. The memory required to store a voxel database grows as the cube of its resolution whereas the cost of storing texture maps or images grows only as the square. The computational expense of voxel-by-voxel shading exhibits similar cube-law growth and is not reduced by coherence present in either the data or the image. While rendering costs are high, the technique offers many advantages. The universality of volumetric representations make them readily accessible to a variety of scientific, medical and engineering disciplines. The unstructured nature of voxels facilitates editing, while the simplicity of volumetric rendering algorithms makes them attractive for hardware implementation.

Early work in this area was largely constrained by memory costs. The solution adopted was to threshold the data, reducing grayscale representations to binary representations. [1]. Further reductions were obtained by tracking and storing only surfaces that bounded connected regions [2]. During rendering, voxels were treated as cubes having six polygonal faces. This approach has been termed the *cuberille* model [3].

Spurred by cheaper memories and faster processors, researchers have begun investigating techniques for directly rendering grayscale data. The chief advantage of this approach is its superior shading, achieved by estimating surface normals from local gradient vectors in the grayscale data [4], [5], [6]. In all of these papers, surface normal estimation is performed on classified data rather than on original data. If the classification operators are non-linear, as most useful ones are, this ordering of tasks can distort surface normals and hence shading. Ray tracing has been used to produce geometrically transformed views in [5], [6]. In both cases, rays are traced through original data rather than through derived shades. If an inexpensive filter is used during ray tracing, this ordering can introduce further errors into surface normals and shading. In this paper, shading and classification calculations are independent and both are performed prior to geometric transformations. This greatly reduces shading errors.

Aliasing of silhouette edges remains an area of difficulty in volumetric rendering. Tri-linear interpolation between sample points can be applied to reduce these artifacts during rotation [4], [6], but interpolation in unrotated views trades smoothness for resolution. Supersampling followed by averaging down has also been suggested, but this trades smoothness for computational expense [3]. Assignment of partial opacities and rendering of multiple transparent surfaces is discussed in [5], but shading is computed only at selected voxels along each ray and not used to smooth silhouettes. Researchers at or collaborating with PIXAR Inc. appear to have addressed this problem but their approach has not been published.

2. Rendering method

The problems associated with rendering volumetric data can be illustrated using a simple example. We define a *3-D binary scene* as a continuous function of 3-space consisting solely of zero-regions and one-regions where region boundaries are defined by polygons. Let zero-regions be transparent and one-regions be opaque and of some fixed color. A fully opaque background polygon of a different color is draped behind the scene. If we apply a polygon-clipping hidden-surface algorithm, we produce a continuous 2-D function consisting of visible polygon fragments. This constitutes an exact solution to the visibility of regions in that scene for a single observer position. By filtering this function and sampling it equally finely in two dimensions, we generate a discrete image.

Suppose instead that we filter the original continuous scene and sample it equally finely in three dimensions, producing a discrete volumetric database. Let us assume that samples are drawn on an orthogonal grid having equal spacing in all three directions and that filtering is performed using a box filter of width equal to the sample spacing. In this case, acquisition consists of dicing the scene into abutting cubes or *voxels*. Voxels lying entirely within zero-regions or one-regions are assigned values of zero or one respectively. Voxels lying athwart region boundaries assume intermediate values depending on the boundary's relationship to the voxel. This is called the *partial volume effect (PVE)* [6]. Our goal is to develop a rendering method that, when applied to this sampled function, will match as closely as possible the image obtained by applying hidden-surface removal to the original continuous scene and then sampling.

If we look at a single voxel, its value tells us what percentage of its volume is opaque. It does not, however, tell us what percentage of its profile relative to the view direction is opaque.

Without this information, we cannot guarantee a proper blend of the voxel with its background. When two or more voxels contribute to a single image pixel, interplay between the profiles of the two voxels relative to the view direction determines the outcome. Since we know none of this geometry, we are forced to make reasonable assumptions. The approach taken in this paper is the same one used in 2-D digital compositing [7], [8]; the opaque portion of a voxel is assumed to obscure the transparent and opaque portions of the voxel behind it in equal shares. We effectively treat each voxel as a semi-transparent object containing no sub-voxel geometry but an opacity equal to the known coverage percentage. This technique insures smooth silhouettes and minimizes aliasing artifacts in general.

Figures 1 and 2 summarize the method. We begin with a left-handed 3-axis coordinate system. Some non-zero portion of the all-positive octant is filled with discrete samples of a continuous scalar function. Samples are evenly spaced in all three directions to form a 3-D orthogonal grid. Since both of the example datasets considered in this paper contain physical density as a function of position, we refer to the value of a voxel as its density. Using a shading model described in section 4, the density $d(\mathbf{x}_i)$ of each voxel $\mathbf{x}_i = (x_i, y_j, z_k)$ is used to compute a color $c_\lambda(\mathbf{x}_i)$, $\lambda = r,g,b$. Using one of the feature classification procedures described in section 3, each density is also used to compute an opacity $\alpha(\mathbf{x}_i)$. Assume that all values range between 0 and 1 with $\alpha = 0$ being transparent and $\alpha = 1$ being opaque. By projecting all voxels perpendicularly toward the $z = 0$ plane, we form an orthographic image. Assuming that x and y spacing of 2-D pixels and 3-D voxels match and that the grids are aligned, a pixel in the image corresponds to a row of voxels in the dataset. A fully opaque background of color c_{bkg} and opacity $\alpha_{bkg} = 1$ is draped behind the dataset at $z_k = zmax + 1$. We compute the color $C_\lambda(\mathbf{u}_i)$ of each pixel $\mathbf{u}_i = (u_i, v_j)$ by digitally compositing in back-to-front order using the well-known transparency algorithm:

```
for $i = xmin, \ldots, xmax$,
    for $j = ymin, \ldots, ymax$,
        for $k = zmax, \ldots, zmin$,                                    (1)
            $C_\lambda(u_i,v_j) = c_\lambda(x_i,y_j,z_{k+1})(1 - \alpha(x_i,y_j,z_k)) + c_\lambda(x_i,y_j,z_k)\alpha(x_i,y_j,z_k)$.
```

In practice, shading need not be computed for voxels whose opacity is 0, but many voxels along each view ray will have non-zero opacities.

3. Feature classification

In addition to insuring smooth silhouettes, the mapping from acquired data to opacity performs classification, enchancing selected features while suppressing others. While the mapping from data to color can also serve this function, as exemplified by pseudo-coloring, a hardwired shading model has been used. This artificially limits the capabilities of our rendering method but enables us to focus attention on a single classification parameter - voxel opacity.

We further limit ourselves to one family of classification functions - those that detect and render surfaces in the data. Surfaces have several characteristics that make them useful for visualizing volumetric data:

(1) Our world is dominated by opaque surfaces. We are thus better trained to evaluate the shapes and spatial relationships of objects from shading on their bounding surfaces than from scattering of light through their interiors.

(2) The directional shading used to render surfaces accentuates even slight changes in their orientation. Surface shading can therefore convey subtle detail.

(3) The human visual system can usually distinguish directionally-variant shading from albedo-variant shading, allowing us to use surface color, shininess and texture to carry additional information [9].

3.1. Rendering of isovalue surfaces

We will first consider the detection and rendering of surfaces defined by points of equal value in grayscale scenes. The driving problem for this study was protein crystallography but the method has wider application.

Determining the structure of large molecules is a difficult problem. The method most commonly used is *ab initio* interpretation of electron density maps, which represent the averaged density of a molecule's electrons as a function of position in 3-space. These maps are obtained from X-ray diffraction studies of crystallized samples of the molecule. Figure 9 shows four slices from a discretized electron density map. Each whitish cloud represents a single atom. The figure is described in greater detail in section 6.

Current methods for presenting this 3-D data include stacks of isodensity contours, ridge lines arranged in 3-space so as to connect local density maxima [10], and basket meshes representing selected isodensity surfaces [11]. Ideally, raster visualizations of isovalue surfaces should meet the following criteria:

(1) Surface interiors should be smoothly shaded.

(2) Surface silhouettes should be free from faceting or aliasing.

(3) Semi-transparent surfaces should be allowed.

(4) Multiple concentric surfaces should be allowed.

Criteria (4) is a hard one. One obvious classification procedure is to make voxels having densities greater than some threshold opaque. This produces 3-D regions of opaque voxels the outermost layer of which is the desired isodensity surface. Unfortunately, this solution prevents display of multiple concentric surfaces. Using a density window in place of a threshold does not solve the problem. If the window is too narrow, holes appear. If it too wide, display of multiple surfaces is constrained.

The classification procedure employed in this study consists of reconstructing, mapping and resampling the volumetric database. The procedure will be outlined first in one dimension, then extended to higher dimensions. We start with a continuous differentiable density function $d(x)$ as shown in figure 3. Data acquisition consists of pre-filtering this function and sampling it to obtain discrete densities $d(x_i)$. We begin classification by post-filtering the samples to form a reconstructed density function $\hat{d}(x)$. A mapping $\beta = \beta(d)$ from density to opacity is then applied, resulting in a continuous opacity function $\alpha(x) = (\hat{d} \circ \beta)(x) = \beta(\hat{d}(x))$. This function is pre-filtered and resampled to yield discrete opacities $\alpha(x_i)$. These opacities are passed to the digital compositing algorithm described in section 2.

Within this general framework, we have considerable latitude in the selection of filters and mappings. The choices shown here are based on empirical trials and represent only one possible implementation. Specifically, reconstruction is implemented using the first-degree Taylor polynomial

$$\hat{d}(x) = d(x_i) + \frac{d(d(x))}{dx}(x_i)(x - x_i)$$

where the first derivative is approximated using the operator

$$\frac{d(d(x))}{dx}(x_i) \approx d(x_{i+1}) - d(x_i).$$

This method was selected because it is inexpensive and localized, requiring only two neighboring density samples to form each approximation. Mapping is implemented using the delta function

$$\beta(d) = \alpha_s \delta(d - d_s)$$

such that selected density d_s is mapped to selected opacity α_s while all other densities are mapped to an opacity of 0. The continuous opacity function is then given by

$$\alpha(x) = \alpha_s \delta(\hat{d}(x) - d_s).$$

This effectively detects each time the reconstructed density function $\hat{d}(x)$ crosses selected density d_s, producing a spike of opacity α_s at that x. The pre-filter preceding resampling is implemented by spatially convolving the continuous opacity function with a Bartlett window of radius r, which is given by

$$g(x) = \begin{cases} 1 - \frac{|x|}{r} & \text{if } |x| \leq r \\ 0 & \text{otherwise.} \end{cases}$$

This filter is also inexpensive and localized. If we assume a resampling interval s equal to the filter radius r, each opacity $\alpha(x_i)$ depends only on density values $d(x_i)$ and $d(x_{i+1})$. This can be verified by combining the above expressions, applying the convolution and solving for $\alpha(x_i)$.

We extend this method to three dimensions by using $d(\mathbf{x}_i)$ in place of $d(x_i)$ and the magnitude of the gradient vector $|\nabla d(\mathbf{x}_i)|$ in place of the derivative $\frac{d(d(x))}{dx}(x_i)$. We also replace the 1-D Bartlett window with a 3-D spherically isotropic linear ramp. By ignoring the orientation of the gradient vector, we introduce additional error into the reconstruction, but its effect was empirically determined to be negligible, at least for the spatially invariant mapping and spatially isotropic pre-filter used here. The entire procedure is summarized by the expression

$$\alpha(\mathbf{x}_i) = \alpha_s \begin{cases} 1 & \text{if } |\nabla d(\mathbf{x}_i)| = 0 \text{ and } d(\mathbf{x}_i) = d_s \\ 1 - \frac{s}{r} \left| \frac{d_s - d(\mathbf{x}_i)}{|\nabla d(\mathbf{x}_i)|} \right| & \text{if } |\nabla d(\mathbf{x}_i)| > 0 \text{ and } d(\mathbf{x}_i) - |\nabla d(\mathbf{x}_i)| \leq d_s \leq d(\mathbf{x}_i) + |\nabla d(\mathbf{x}_i)| \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

where the gradient vector is approximated using the operator

$$\nabla d(\mathbf{x}_i) = \nabla d(x_i, y_j, z_k) \quad (3)$$

$$\approx \left[d(x_{i+1}, y_j, z_k) - d(x_i, y_j, z_k),\ d(x_i, y_{j+1}, z_k) - d(x_i, y_j, z_k),\ d(x_i, y_j, z_{k+1}) - d(x_i, y_j, z_k) \right].$$

A graph of $\alpha(\mathbf{x}_i)$ as a function of $d(\mathbf{x}_i)$ and $|\nabla d(\mathbf{x}_i)|$ for a typical value of d_s is shown in figure 4.

If more than one isodensity surface is to be displayed in a single image, they can be classified separately and their opacities combined using equation (1). Alternatively, given density values d_{s_i}, $i = 1, \ldots, n$, $n \geq 1$ and opacities α_{s_i}, we can use equation (2) to compute $\alpha_{s_i}(\mathbf{x}_i)$, then apply

$$\alpha_{tot}(x_i) = 1 - \prod_{i=1}^{n} (1 - \alpha_{v_i}(x_i)). \tag{4}$$

Since reconstruction is based on a first-degree polynomial, high second derivatives in the original density function result in poor reconstructions and aliasing in the image. Moving to a higher-order polynomial considerably increases computational expense. Fortunately, the time and ensemble averaging inherent in X-ray diffraction tends to mitigate these artifacts. They can be reduced still further, at the expense of some resolution, by loading a 2-D lookup table with samples of $\alpha(x_i)$ and blurring the table slightly. A sample image is shown in figure 10. Although no transparency was used here, two concentric isodensity surfaces are visible where clipped by the dataset boundaries.

3.2. Rendering of region boundary surfaces

We will next consider the detection and rendering of surfaces bounding regions of constant density in grayscale scenes. The driving problem for this study was medical imaging, specifically, display of computed tomography (CT) data.

From a densitometric point of view, the human body is a complex arrangement of biological tissues each of which is fairly homogeneous and of predictable density. Clinicians are mostly interested in the boundaries between tissues, from which the sizes and spatial relationships of features can be inferred. A second argument for rendering surfaces is that physicians know what the surface of an organ looks like - they see it during surgery. While interior surfaces are obviously invisible when viewed from outside the body, not to mention lacking any illumination, it doesn't require much imagination to accept visualizations in which surfaces are rendered transparently and with hypothetical illumination.

The currently dominant method for presenting these surfaces involves forming a mesh of polygons from contours drawn or computed (usually with some manual intervention) on each slice [12]. Approaches based on volumetric rendering were surveyed in the introduction. Ideally, raster visualizations of region boundary surfaces should meet criteria (1) through (4) from section 3.1. In addition, the occurrence of false negatives - missed surfaces, and false positives - artifactual surfaces, should be minimized.

As before, criteria (4) is a hard one. To render region boundaries opaquely without also making enclosed regions opaque, we must isolate voxels lying on boundaries from voxels lying wholely within regions. Specifically, we wish to detect voxels lying on transitions between some fixed tissue type A having known density d_A and an arbitrary number of other tissues types B_i, $i = 1, \ldots, n$, $n \geq 1$ having unknown densities d_{B_i}. For all such voxels, we must first deduce i, then compute a parameter t, $0 \leq t \leq 1$ indicating where this voxel lies along the transition from d_A to d_{B_i}. Given t, we may construct a mapping in which opacity rises linearly from 0 to some user-selected value α_{v_i} as t rises from 0 to .5, then falls again to 0 as t continues from .5 to 1. This yields an anti-aliased rendering of the boundary between regions of tissue types A and B_i. Proper presentation of these boundaries require that transitions between tissue type A and tissue types B_i appear identical for all i. Since the d_{B_i}'s differ, the mapping from density d_{B_i} to parameter t is different for each i. Unfortunately, determination of i and hence t based on density alone is not possible. This problem is discussed in the context of 2-D area filling in [13]. In particular, it corresponds to the case of trying to fill an image containing three or more colors that are co-linear in rgb-space.

These considerations suggest that the classification procedure developed for isovalue surfaces will not be applicable here. The technique actually employed in this study is based on the principle of matched detectors [14]. In particular, we empirically develop an operator that classifies voxels based on both density and gradient vector magnitude. We start by defining for each pair of tissue types a mapping in which t rises linearly from 0 to 1 as density $d(\mathbf{x}_i)$ transits from d_A to d_{B_i}. We then map t to α_{v_i} as described above. We finally define a composite function $\alpha_1(\mathbf{x}_i)$ equal to the convex hull of these n opacity functions as shown in the top graph of figure 5. In most cases, voxels having densities that fall between d_A and d_{B_i} but low gradient vector magnitudes do not belong to transitions we are detecting but to interiors of unrelated regions. Voxels having gradient vector magnitudes greater than the difference between densities d_A and d_{B_i} almost certainly do not belong to transitions we are detecting and may be similarly discarded. We encode these criteria by defining, for each pair of tissue types, a separate mapping in which opacity rises gradually from 0 to 1 as gradient vector magnitude $|\nabla d(\mathbf{x}_i)|$ rises from 0 to the maximum possible difference $|d_A - d_{B_i}|$, then falls sharply to 0 for greater magnitudes. Ramps were used in accordance with the principle followed throughout this paper of avoiding binary mappings. Linear ramps were used for simplicity in this implementation and slopes were determined by trial. We then define a second composite function $\alpha_2(\mathbf{x}_i)$ equal to the convex hull of these n functions as shown in the bottom graph of figure 5. Finally, we multiply the two composite opacity functions together. For a single pair of tissue types of density d_A and d_B, $d_A \neq d_B$, the procedure is summarized by the expression

$$\alpha(\mathbf{x}_i) = \alpha_v \begin{cases} \left[1 - 2\left|\dfrac{d_A - d(\mathbf{x}_i)}{d_A - d_B} - \dfrac{1}{2}\right|\right]\left[\dfrac{|\nabla d(\mathbf{x}_i)|}{|d_A - d_B|}\right] & \text{if } (d_B \le d(\mathbf{x}_i) \le d_A \text{ or } d_A \le d(\mathbf{x}_i) \le d_B) \\ & \quad \text{and } |\nabla d(\mathbf{x}_i)| \le |d_B - d_A| \\ \left[1 - 2\left|\dfrac{d_A - d(\mathbf{x}_i)}{d_A - d_B} - \dfrac{1}{2}\right|\right]\left[2 - \dfrac{k|\nabla d(\mathbf{x}_i)|}{|d_A - d_B|}\right] & \text{if } (d_B \le d(\mathbf{x}_i) \le d_A \text{ or } d_A \le d(\mathbf{x}_i) \le d_B) \\ & \quad \text{and } |d_B - d_A| \le |\nabla d(\mathbf{x}_i)| \le (1 + 1/k)\ |d_B - d_A| \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where the gradient vector is approximated using the operator

$$\begin{aligned} \nabla d(\mathbf{x}_i) &= \nabla d(x_i, y_j, z_k) \qquad (6) \\ &\approx \left[\frac{1}{2}d(x_{i+1},y_j,z_k) - d(x_{i-1},y_j,z_k),\ \frac{1}{2}d(x_i,y_{j+1},z_k) - d(x_i,y_{j-1},z_k),\ \frac{1}{2}d(x_i,y_j,z_{k+1}) - d(x_i,y_j,z_{k-1})\right] \end{aligned}$$

As in section 3.1, this implementation is inexpensive and localized, each opacity $\alpha(x_i)$ depending only on density values $d(x_j)$, $j = i-1, \ldots, i+1$. A graph of $\alpha(\mathbf{x}_i)$ as a function of $d(\mathbf{x}_i)$ and $|\nabla d(\mathbf{x}_i)|$ for typical values of d_A, d_B and the constant k is shown in figure 6.

If surfaces bounding more than one region type A are to be displayed in a single image, they can be classified separately and their opacities combined using equation (4). A set of sample images is shown in figure 11. All four images were computed from the same dataset. The top and bottom pairs differ only in parameters of the classification procedure. The left and right pairs represent views taken at right angles to each other. The horizontal bands through the patient's teeth are artifacts due to scattering of X-rays from dental fillings and are present in the acquired data. The bands across her forehead and under her chin are gauze bandages used to immobilize her head during scanning. Her skin and nose cartilage are rendered transparently over the bone surface in the two lower views but are only faintly visible in the figure due to the dynamic range limitations of photographic printing.

4. Shading calculations

Using the rendering method presented in section 2, the mapping of acquired data to color does not participate in the classification operation. Accordingly, a shading model was selected that provides satisfactory rendering of smooth surfaces at a reasonable cost. It is not the main point of the paper and is presented mainly for completeness. The model chosen is due to Phong, but extended to include linear depth-cueing. This formulation is adopted from [15]:

$$c_\lambda(\mathbf{x}_i) = c_{a\lambda}k_{a\lambda} + \frac{c_{p\lambda}}{r+k}\left[k_{d\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{L}) + k_{s\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{H})^n\right] \qquad (7)$$

where

$c_\lambda(\mathbf{x}_i)$ = λ'th component of color of voxel $\mathbf{x}_i$, $\lambda = r,g,b$,

$c_{a\lambda}$ = λ'th component of color of ambient illumination,

$c_{p\lambda}$ = λ'th component of color of parallel light source,

$k_{a\lambda}$ = ambient reflection coefficient for λ'th color component,

$k_{d\lambda}$ = diffuse reflection coefficient for λ'th color component,

$k_{s\lambda}$ = specular reflection coefficient for λ'th color component,

n = exponent used to approximate highlight,

r, k = terms used in linear approximation of depth-cueing,

$\mathbf{N}(\mathbf{x}_i)$ = surface normal of voxel $\mathbf{x}_i$,

$\mathbf{L}$ = normalized vector in direction of light source,

$\mathbf{H}$ = normalized vector in direction of maximum highlight.

Since a parallel light source is used, $\mathbf{L}$ is a constant. Furthermore,

$$\mathbf{H} = \frac{\mathbf{V}+\mathbf{L}}{2}$$

where

$\mathbf{V}$ = normalized vector in direction of viewer.

Since an orthographic projection is used, $\mathbf{V}$ and hence $\mathbf{H}$ are also constants. Finally, the surface normal $\mathbf{N}$ is given by

$$\mathbf{N} = \frac{\nabla d(\mathbf{x}_i)}{|\nabla d(\mathbf{x}_i)|}.$$

There are numerous methods for estimating the gradient vector $\nabla d(\mathbf{x}_i)$. A general discussion of gradient operators can be found in any image processing textbook [14]. The selection of an operator depends on the frequency spectra of the data being rendered and the features being sought. Since the gradient vector is used in both the shading and opacity calculations, efficiency considerations prompted the use of the same operator for both tasks. Different operators were used, however, for the two different data types, as seen by comparing equations (3) and (6). These operators were selected empirically and reflect the frequency characteristics of each type of data. It is worth noting that these operators are always applied to unclassified densities $d(\mathbf{x}_i)$. By separating estimation of surface normals from classification, we insure an undistorted visualization of shapes.

5. Geometric transformations

There are two principle reasons for applying geometric transformations: data preparation and animation. Data preparation consists of correcting for undesirable attributes of the acquisition process - geometric attributes in this case. For example, sampling grids may be non-orthogonal, spacing between samples may vary as a function of position in 3-space, or voxel aspect ratios may not be cubic. In this paper, we limit data preparation to aspect ratio correction, which we implement by interpolating additional samples in one or more directions. Real-time or pre-computed motion sequences are an invaluable aid to comprehension of volumetric data. We implement this capability using 3-axis rotation followed by orthographic projection. We note that data preparation is performed only once while animation requires labor to produce each frame.

Numerous methods for applying geometric transformations to 2-D textures may be found in the literature. An excellent survey is contained in [16]. Incorporation of these techniques into rendering pipelines that include visibility and shading calculations has also been addressed [17]. These same issues apply to transformation of volumetric data. Even restricting ourselves to the rendering method presented in this paper, there are many choices to be made:

(1) Transformations can be applied either before or after shading and classification, i.e. to acquired data or to derived shades and opacities.

(2) Transformed values can be accumulated in new datasets or composited immediately to form images.

(3) Transformations can be applied in object-order - looping through input samples, or in image-order - looping through output samples.

(4) Transformations can be applied separately in each of three orthogonal directions or simultaneously.

Let us consider the problem of interpolation. The rendering method presented in section 2 assumes that sample spacing is even in all three directions and matches image sample spacing in the two directions parallel to the picture plane, at least for the unrotated case. If acquired data does not meet these requirements, interpolation must be performed. Phong has demonstrated that better surface shading is obtained by interpolating normals than by interpolating intensities, as noted in [15]. Even better shading is obtained by interpolating the geometry from which normals are computed. This implies that we should interpolate densities rather than gradients or colors. The same principle applies to any non-linear operator including the feature classification procedures of section 3. This supports our decision to interpolate densities. The disadvantage of this approach is that the gradient detectors we use to estimate normals are sensitive to discontinuities in the first derivative of interpolated data. We can mitigate this problem by employing a filter that is second-order or higher.

Rotation, unlike interpolation, is best applied near the end of the rendering pipeline. By precomputing colors and opacities, most of the hard work need only be done once; expensive shading models and classification procedures can be employed without incurring per-frame costs. Furthermore, since gradient vectors have already been estimated, we can rotate the data using an inexpensive first-order filter without introducing errors into the shading or classification. The disadvantage of this method is that shading becomes fixed; light sources appear to travel around with the data as it rotates and highlights are incorrect. Since most visualizations produced using volumetric rendering are of imaginary or invisible phenomena anyway, observers are seldom troubled by this effect.

If the specular component of the shading model is reduced, a convincing simulation of walking around an immobile dataset illuminated by fixed light sources is obtained.

Figure 7 shows how each geometric transformation fits into the rendering pipeline. Interpolation is performed by spatially convolving the acquired data $d_0(\mathbf{x}_i)$ with a 3-D separable cubic B-spline filter kernel. The results are stored in a second dataset $d_1(\mathbf{x}_i)$. Shading and classification are performed using equations (2) through (7), resulting in a color $c_\lambda(\mathbf{x}_i)$ and opacity $\alpha(\mathbf{x}_i)$ at each voxel. These are stored in two additional datasets. Rotation and orthographic projection are incorporated into the digital compositing algorithm by casting parallel rays into the data at an angle as shown in figure 8. Samples are taken at evenly spaced intervals along each ray working from the background plane towards the eye. Since an orthographic projection is used, sample coordinates can be efficiently calculated using differencing. At each sample, a color and opacity are computed using tri-linear interpolation from the nearest eight neighbors. The results are composited into the color $C_\lambda(\mathbf{u}_i)$ of the current ray using equation (1).

Interpolation was used in the production of all images in this paper while only figures 10 and 12 include rotation. Note that although the inexpensive filter results in some blurring in figure 12, classification and shading are comparable to that of the orthogonal views in figure 11 since they are computed prior to transformation. The ray tracing method described here also supports translation and scaling. Although this is not in accordance with the principle of interpolating densities rather than colors, images larger than the stored dataset can be produced simply by casting more rays through the pre-computed colors and opacities. This expedient was used, in addition to interpolation during data preparation, for figures 10 through 12.

6. Implementation and results

The techniques presented in this paper were implemented in the C language under UNIX bsd 4.2. The timings given below are for a VAX 11/785 having sufficient physical memory to prevent paging.

The dataset used in the crystallography study is from the protein Cytochrome B5. The lengthy process of acquiring an electron density map is beyond the scope of this paper and readers are referred to [18]. Our map was obtained from the crystallographers as a 49 x 31 x 66 sample grid representing a 46 x 29 x 62 angstrom cube. A 20 x 20 x 20 sample portion of this map was extracted and expanded to fill a 113 x 113 x 113 voxel dataset. Interpolation was performed using the 3-D separable cubic B-spline described in section 5 and took 6 hours. Figure 9 shows four slices spaced 10 voxels apart in this expanded dataset. Classification and shading were applied as described in sections 3.1 and 4 and required 30 minutes total. The resulting shades and opacities were stored in an intermediate dataset as shown in figure 10. Rotation, projection and compositing were performed as described in section 5 and took 4 hours, yielding the image in figure 10. Some scaling was also included in the projection operation, producing a 400 x 400 pixel image. Without scaling, the image would have been 200 x 200 pixels and would have required 30 minutes to render.

The dataset used in the medical imaging study is from a cadaver and was obtained as a 256 x 256 x 113 sample grid. Care was taken during scanning to insure that all slices were parallel, evenly spaced, and completely covered the anatomy being scanned. For a general discussion of CT scanning and reconstruction techniques, the reader is referred to [19]. The acquired grid was expanded to fill a 256 x 256 x 226 voxel dataset by interpolating in one direction only, which took 90 minutes. The four images in figure 11 were produced by classifying, shading and immediately compositing in either the x or y direction as shown in figure 1 and required 5 hours per image. In a separate trial, shading and classification were performed without compositing, resulting in an intermediate dataset as shown in figure 10. This took 4 hours. Rotation, projection and compositing were performed as described in section 5 and took 8 hours, yielding the image in figure 12.

Some scaling was included in this projection, producing a 512 x 512 pixel image. Without scaling, the image would have been 400 x 400 pixels and would have required 4 hours to render.

Worst-case storage requirements for this implementation occur after shading and classification have been applied but before geometric transformation. Densities, colors and opacities are represented as 8-bit values throughout. Therefore, 28 megabytes were required to store the 256 x 256 x 226 colors and opacities computed during the medical imaging study.

7. Conclusions

Since volumetric rendering techniques work from sampled data rather than geometric primitives, they are necessarily approximate. Errors in visibility and classification do occur, as does aliasing. This paper has presented a technique for reducing these artifacts by following several guidelines. Firstly, all voxels participate in the rendering of any image. Secondly, feature classification operators are continuous rather than binary. Thirdly, shading and classification calculations are independent, particularly with regard to estimation of surface normals.

A number of improvements to the current implementation can be proposed. More work is needed on reconstruction methods for isovalue surfaces, especially with respect to resolving closely spaced multiple concentric surfaces. By pre-computing and storing second or higher derivatives, very accurate approximations can be made at only modest increases in cost. Derivative operators that are more resistant to noise should also be investigated. For region boundary surfaces, more work is needed on design of matched filters. Detection of boundaries between soft tissues is harder than air-tissue or bone-tissue boundaries due to the inhomogeneity of these features and the narrow range of densities they occupy in CT scans. Studies using magnetic resonance imaging (MRI) data, which exhibits better distinctions between soft tissue features, should be conducted. On a general note, a unified approach to classification of surfaces - one that includes both isovalue and region boundary surfaces, would be desirable.

Surface shading is only one approach to the visualization of volumetric data. Shading models based on scattering of light from semi-transparent media are also possible. Color and texture can be added to represent such variables as gradient magnitude. Visualizing discrete vector functions of 3-space is still largely unexplored. Visualizations combining acquired and synthetic data also hold much promise. For example, it might be useful to superimpose ball-and-stick molecular models onto electron density maps or medical prosthesis devices onto CT scans. In order to obtain correct visibility, a true 3-D merge of the acquired and synthetic data must be performed. One possible solution is the *rgbαz* buffer presented in [20]. Another is to scan-convert the synthetic geometry directly into the acquired density map and render the ensemble.

The prospects for real-time or near real-time rotation of volumetric data are encouraging. By pre-computing shades and opacities and storing them in intermediate 3-D datasets, we simplify the volumetric rendering problem to one of transforming two values per sample point and digitally compositing the results. One promising technique for speeding up these transformations is to apply a 3-pass version of the 2-pass texture mapping technique presented in [21]. By filtering separately in each of three orthogonal directions, computational expense and algorithmic complexity are reduced. This further suggests that hardware implementations might be feasible. A recent survey of architectures for rendering voxel data is given in [22]. One imagines a general-purpose scene animation machine that can rotate synthetic scenes of arbitrary geometric complexity in real-time with anti-aliasing, although with fixed shading, provided that the scene can be scan-converted into a volumetric database of colors and opacities. By coupling pre-computed surface normals with shading hardware, movable light sources could also be supported.

Acknowledgements

The author wishes to thank Profs. Frederick P. Brooks Jr., Turner Whitted, Henry Fuchs, and Steven M. Pizer for their encouragement on this paper, and Andrew S. Glassner, Lawrence M. Lifshitz, John M. Gauch, Lee Westover, and Mark Harris for enlightening discussions and advice. This work was supported by NIH grant RR02170.

References

[1] Herman, G.T. and Liu, H.K., "Three-Dimensional Display of Human Organs from Computer Tomograms," *Computer Graphics and Image Processing*, Vol. 9, No. 1, January, 1979, pp. 1-21.

[2] Artzy, E., Frieder, G., and Herman, G.T., "The Theory, Design, Implementation and Evaluation of a Three-Dimensional Surface Detection Algorithm," *Computer Graphics and Image Processing*, Vol. 15, No. 1, January, 1981, pp. 1-24.

[3] Chen, L., Herman, G.T., Reynolds, R.A., and Udupa, J.K., "Surface Shading in the Cuberille Environment," *IEEE Computer Graphics and Applications*, Vol. 5, No. 12, December, 1985, pp. 26-32.

[4] Hohne, K.H. and Bernstein, R., "Shading 3D-Images from CT Using Gray-Level Gradients," *IEEE Transactions on Medical Imaging*, Vol. MI-5, No. 1, March, 1986, pp. 45-47.

[5] Schlusselberg, Daniel S. and Smith, Wade K., "Three-Dimensional Display of Medical Image Volumes," *Proceedings of the 7th Annual Conference of the NCGA*, Anaheim, CA, May, 1986, Vol. III, pp. 114-123.

[6] Goldwasser, Samuel, "Rapid Techniques for the Display and Manipulation of 3-D Biomedical Data," *Tutorial presented at 7th Annual Conference of the NCGA*, Anaheim, CA, May, 1986.

[7] Wallace, Bruce A., "Merging and Transformation of Raster Images for Cartoon Animation," *Computer Graphics*, Vol. 15, No. 3, August, 1981, pp. 253-262.

[8] Porter, Thomas and Duff, Tom, "Compositing Digital Images," *Computer Graphics*, Vol. 18, No. 3, July, 1984, pp. 253-259.

[9] Robertson Phillip K. and O'Callaghan, John F., "The Application of Scene Synthesis Techniques to the Display of Multidimensional Image Data," *ACM Transactions on Graphics*, Vol. 4, No. 4, October, 1985, pp. 247-275.

[10] Williams, Thomas Victor, *A Man-Machine Interface for Interpreting Electron Density Maps*, PhD thesis, University of North Carolina, Chapel Hill, NC, 1982.

[11] Purvis, George D. and Culberson, Chris, "On the Graphical Display of Molecular Electrostatic Force-Fields and Gradients of the Electron Density," *Journal of Molecular Graphics*, Vol. 4, No. 2, June, 1986, pp. 89-92.

[12] Fuchs, H., Kedem, Z.M, and Uselton, S.P., "Optimal Surface Reconstruction from Planar Contours," *Communications of the ACM*, Vol. 20, No. 10, 1977, pp. 693-702.

[13] Fishkin, Kenneth P. and Barsky, Brian A., "A Family of New Algorithms for Soft Filling," *Computer Graphics*, Vol. 18, No. 3, July, 1984, pp. 235-244.

[14] Castleman, Kenneth R., *Digital Image Processing*, Prentice-Hall, 1979.

[15] Foley J.D., Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982, p. 575-584.

[16] Heckbert, Paul, "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, Vol. 6, No. 9, September, 1986, pp. 56-67.

[17] Cook, Robert L., Carpenter, Loren and Catmull, Edwin, "The Reyes Image Rendering Architecture," *Computer Graphics*, July, 1987 (to appear).

[18] Glusker, Jenny Pickworth and Trueblood, Kenneth N., *Crystal Structure Analysis*, Oxford University Press, 1985.

[19] Ekstrom, Michael P., *Digital Image Processing Techniques*, Academic Press, 1984.

[20] Duff, Tom, "Compositing 3-D Rendered Images," *Computer Graphics*, Vol. 19, No. 3, July, 1985, pp. 41-44.

[21] Catmull, Ed and Smith, Alvy Ray, "3-D Transformations of Images in Scanline Order," *Computer Graphics*, Vol. 14, No. 3, July, 1980, pp. 279-285.

[22] Kaufman, Arie, "Voxel-Based Architectures for Three-Dimensional Graphics," *Proceedings of the IFIP 10th World Computer Congress*, Dublin, Ireland, September, 1986, pp. 361-366.

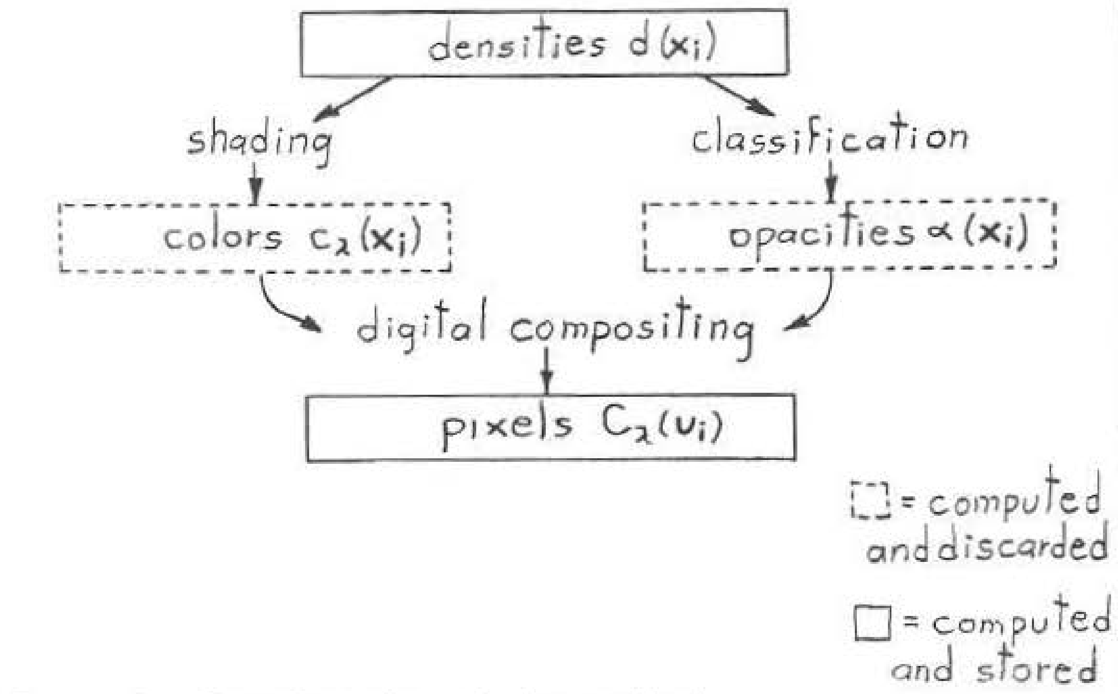

Figure 1 - Overview of rendering method

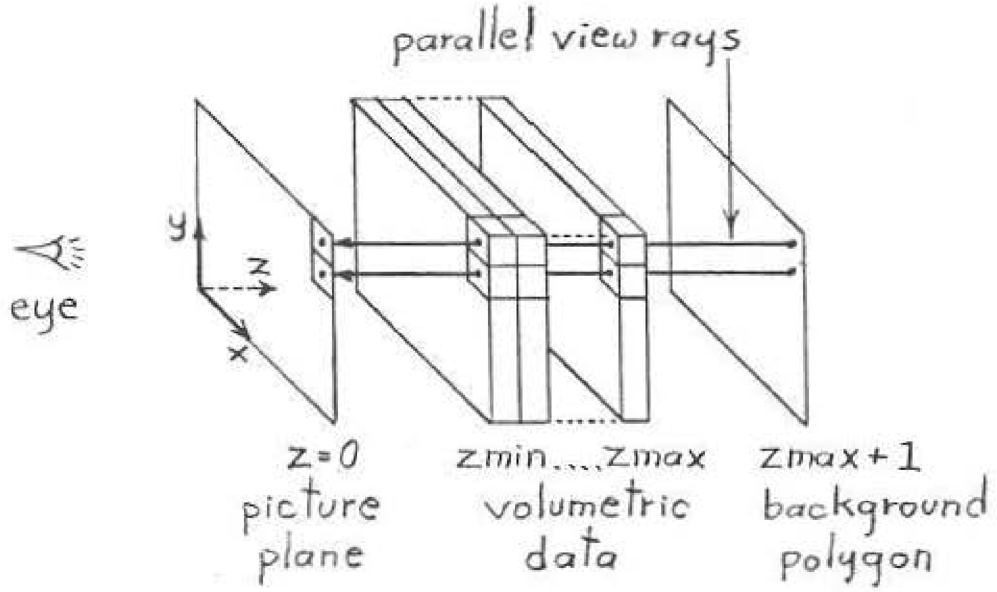

Figure 2 - Setup for digital compositing step

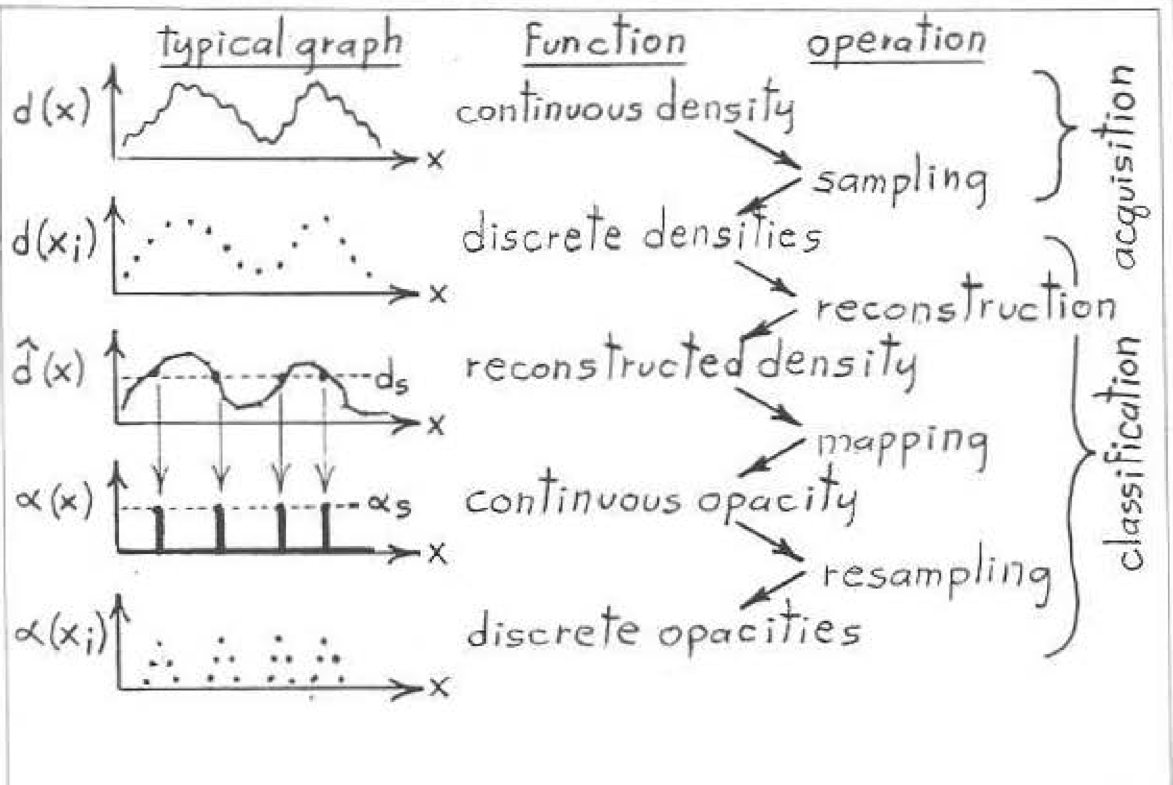

Figure 3 - Procedure for classifying isovalue surfaces

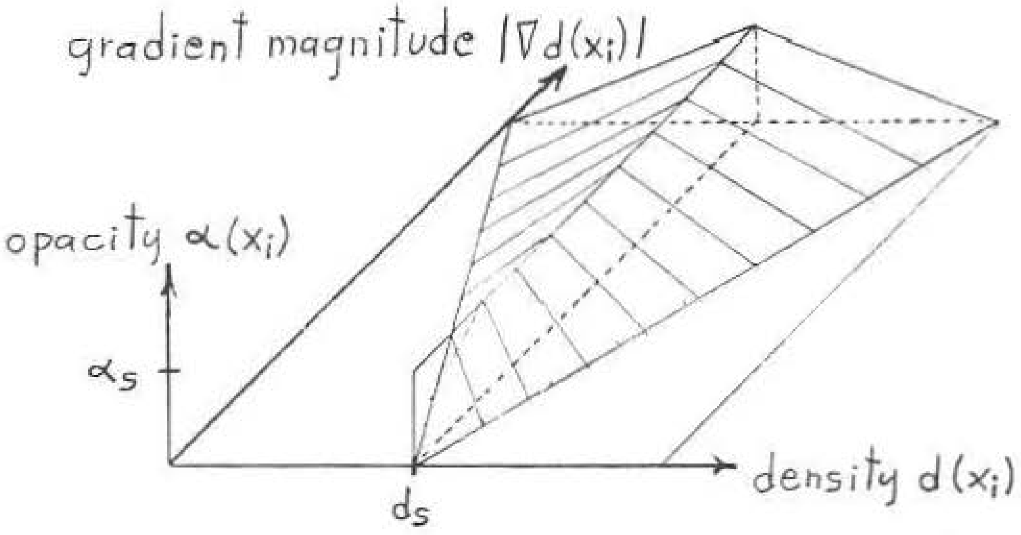

Figure 4 - Isovalue surface classification mapping

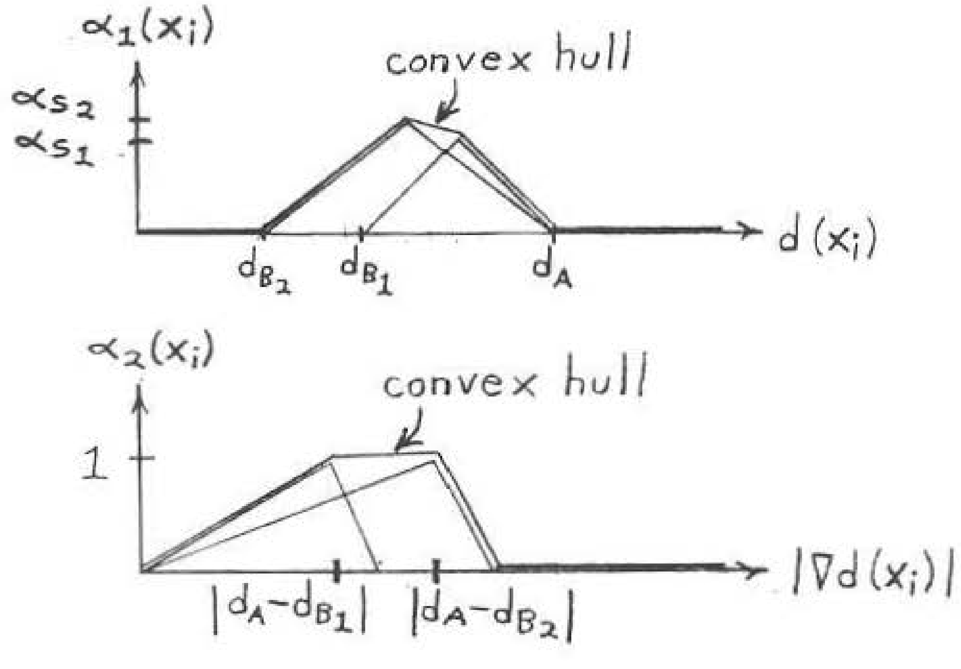

Figure 5 - Procedure for classifying boundary surfaces

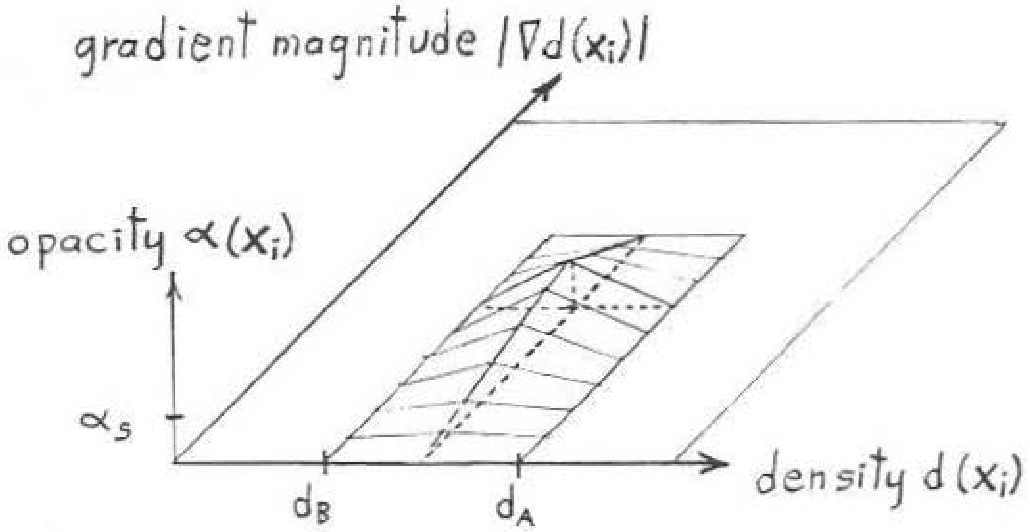

Figure 6 - Boundary surface classification mapping

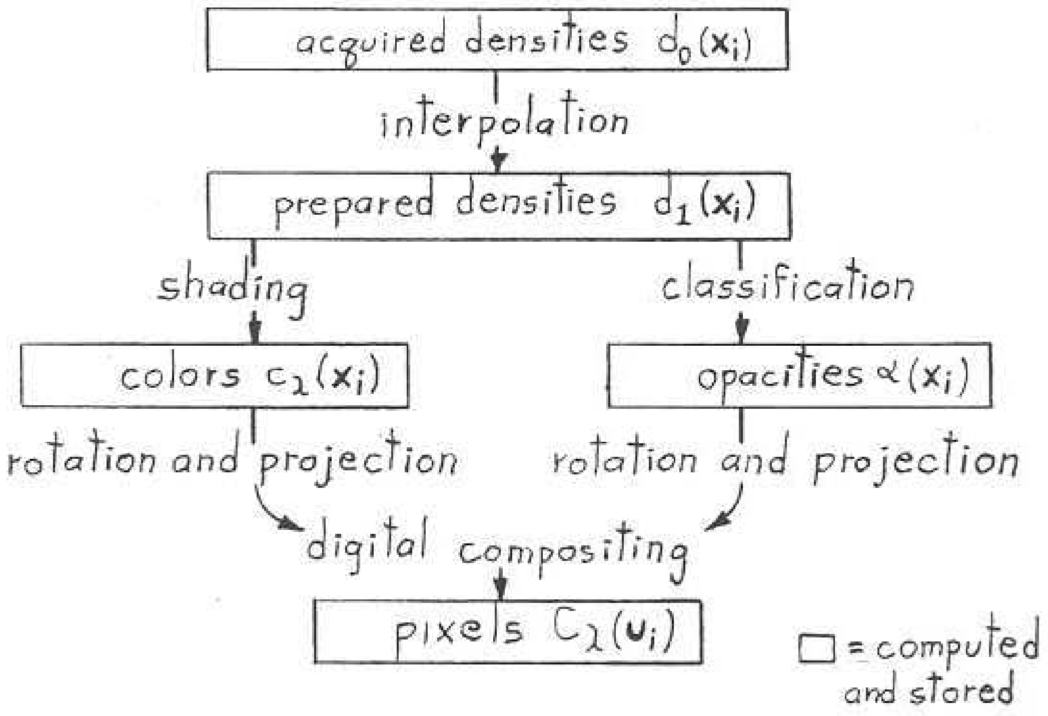

Figure 7 - Overview of geometric transformations

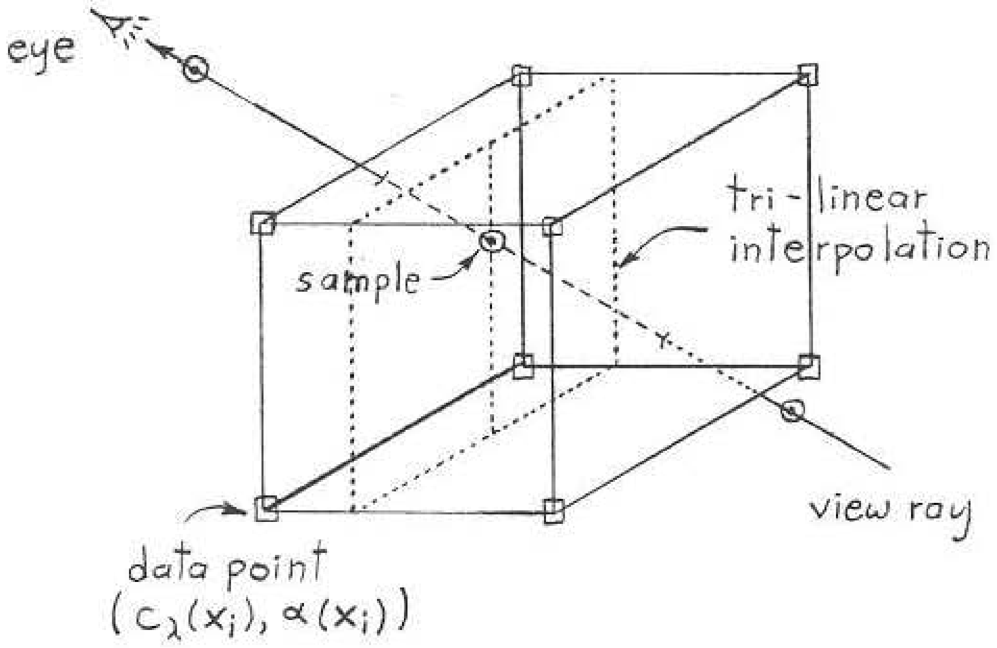

Figure 8 - Setup for ray tracing step

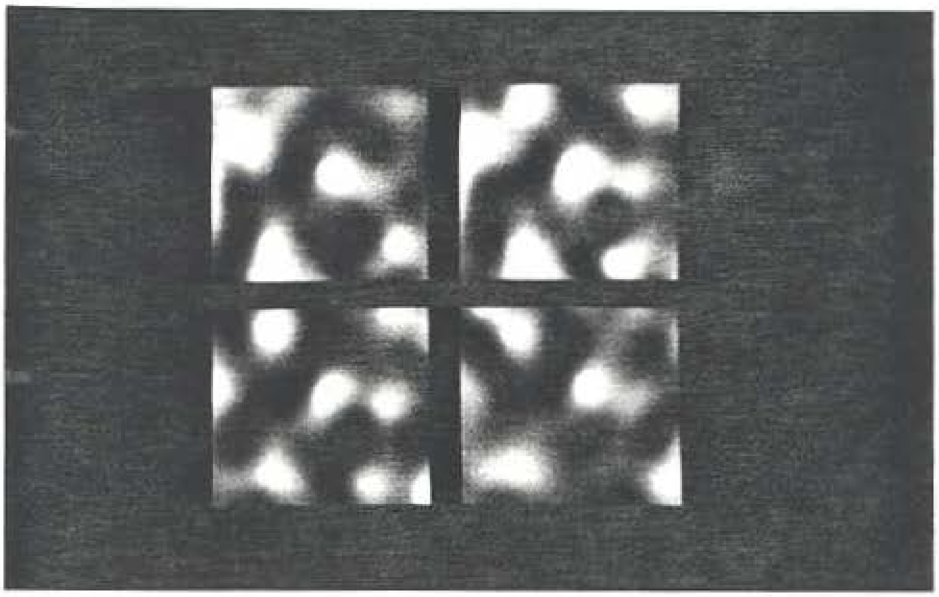

Figure 9 - Slices from electron density map

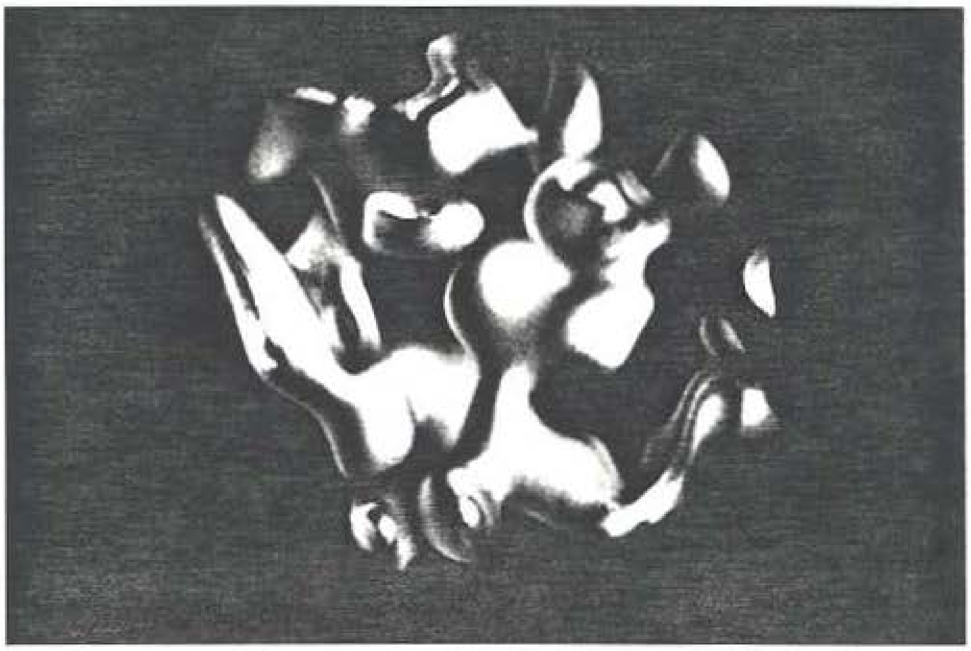

Figure 10 - Isovalue surfaces from electron density map

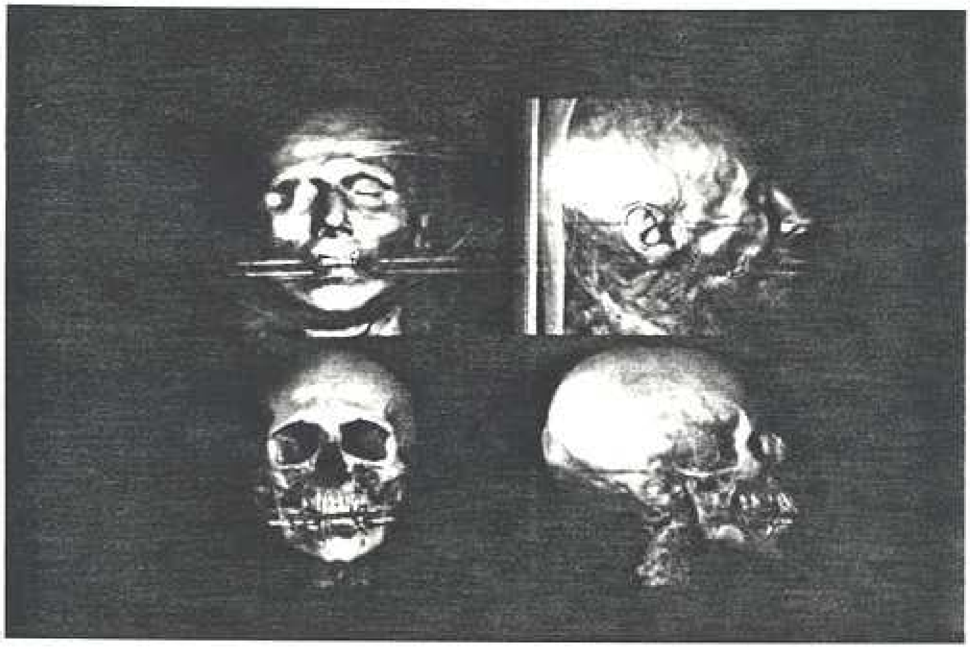

Figure 11 - Boundary surfaces from CT data

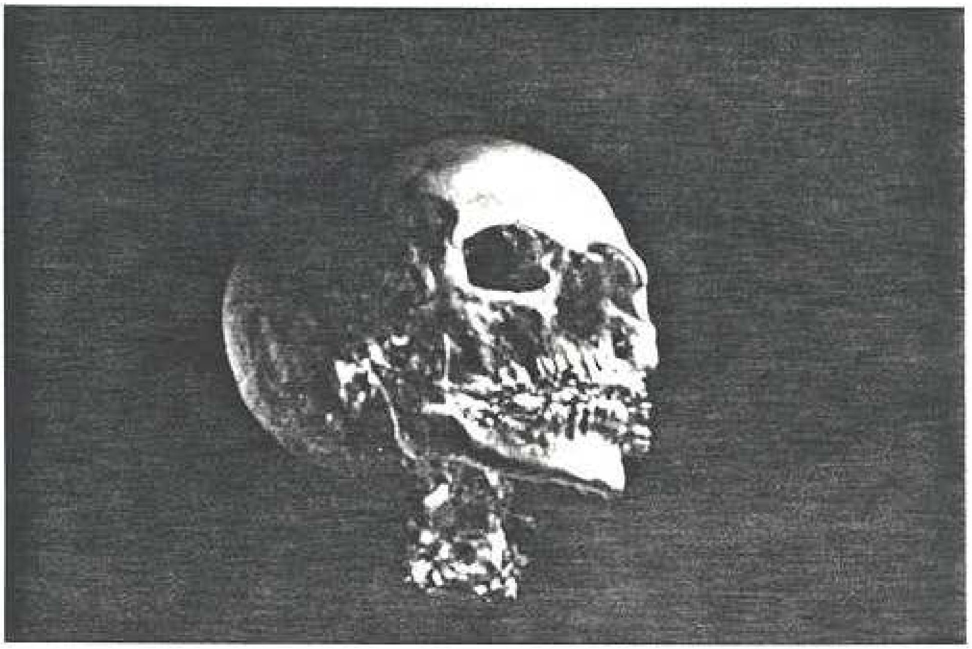

Figure 12 - Geometric transformation of surfaces from CT data