Rendering of Surfaces from Volumetric Data

TR87-036

June, 1987 (TR87-016-Revised November, 1987)

Marc Levoy

The University of North Carolina at Chapel Hill
Department of Computer Science
Sitterson Hall, 083A
Chapel Hill, NC 27599-3175

Rendering of Surfaces from Volumetric Data

Marc Levoy

June, 1987
(revised November, 1987)

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

Abstract

A new technique 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. Non-binary decision functions insure that small or poorly defined features are not lost. Independence of shading and classification calculations insures that undistorted visualizations of 3-D shapes are obtained. The resulting colors and opacities are digitally composited from back to front along view rays to form an image. The technique is simple and fast, yet produces surfaces exhibiting smooth silhouettes and few 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 in which data is represented as arrays of samples rather than as geometry. Typical sources include sensing devices and computer simulations. In this paper, we focus on samples of scalar functions of three spatial dimensions, henceforth referred to as *volumetric data.*

The currently dominant technique for presenting this data involves fitting surface primitives to the sampled function, then rendering these primitives using classical image synthesis [Fuchs77, Purvis86]. Fitting surfaces to volumetric data is a hard problem. It is computationally expensive and prone to errors. Low-order geometric primitives are also mediocre reconstruction filters, giving rise to artifacts in the rendered image. The visual impact of these errors can be reduced by employing surface primitives equal in size to the spacing between data samples [Lorensen87] but rendering the hundreds of thousands of polygons produced using such a technique becomes a significant expense.

To avoid these problems, researchers have introduced *volumetric rendering* wherein the intermediate surface representation is omitted. Images are formed by directly shading each sample and projecting it onto the picture plane. In the graphics literature, volumetric rendering has been used to display clouds by [Csuri79, Blinn82, Kajiya84]. Its use to display surfaces has been developed primarily in the medical field, where it has a long history.

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 [Herman79].

Further reductions were obtained by tracking and storing only surfaces that bounded connected regions [Artzy81]. During rendering, voxels were treated as cubes having six polygonal faces. This approach has been termed the *cuberille* model [Chen85].

Spurred by cheaper memories and faster processors, researchers have begun investigating techniques for directly rendering grayscale data [Hohne86, Schlusselberg86, Goldwasser86, Trousset87, Hohne87]. The chief advantage of this approach is its superior shading, achieved by estimating surface normals from local gradient vectors in the grayscale data. In all of these papers, shading is applied only to voxels lying on the surface of interest. If binary decision functions are used to find these surfaces, distortions in perceived surface configuration can result. The use of non-binary decision functions reduces these artifacts. Strict independence of shading and classification calculations eliminates them entirely. Both techniques are used in this paper.

Ray tracing has been used to produce geometrically transformed views in [Schlusselberg86, Goldwasser86, Hohne87]. Where sampling issues have been treated [Goldwasser86], rays are traced through original data prior to estimation of surface normals. If an inexpensive filter is used during resampling, errors in normals and hence surface shading can result. In this paper, geometric transformations are applied after shading, eliminating this source of artifacts.

Aliasing of surface silhouettes remains an area of difficulty in volumetric rendering. Supersampling followed by averaging down has been suggested [Chen85] but the computational expense of this solution is high. Interpolation between acquired samples is used in [Goldwasser86] but only for geometrically transformed views. Researchers at or collaborating with PIXAR Inc. appear to have addressed this problem [Smith87] but no details of their approach have been published.

2. Rendering method

The rendering method used in this study combines aspects of ray tracing [Whitted80] with *volumetric compositing*, a 3-D extension of image compositing [Wallace82, Porter84, Duff85]. It differs from classical ray tracing in that the data is represented as an array of samples rather than as geometry. This obviates any need to find ray-object intersections. In addition, multiple reflections and refractions are not handled, eliminating any need to trace rays recursively. The differences between volumetric and image compositing are more subtle and will be discussed later in the paper.

The method is summarized in figure 1. We begin with an array of acquired values $f_0(\mathbf{x}_i)$ at voxel locations $\mathbf{x}_i = (x_i, y_j, z_k)$. The first step is data preparation which may include correction for non-orthogonal sampling grids in protein crystallography, correction for patient motion artifacts in medical imaging, histogram modification and interpolation of additional samples. The output of this step is an array of prepared values $f_1(\mathbf{x}_i)$. This array is used as input to the shading model described in section 4, yielding an array of colors $c_\lambda(\mathbf{x}_i)$, $\lambda = r,g,b$. In a separate step, the array of prepared values is used as input to one of the classification procedures described in section 3, yielding an an array of opacities $\alpha(\mathbf{x}_i)$. Rays are then cast into these two arrays according to the current viewing parameters. For each ray, a vector of colors $c_\lambda(\mathbf{x}_{\bar{i}})$ and opacities $\alpha(\mathbf{x}_{\bar{i}})$ is computed by sampling the voxel database at K evenly spaced locations $\mathbf{x}_{\bar{i}} = (x_{\bar{i}}, y_{\bar{j}}, z_{\bar{k}})$ along the ray and tri-linearly interpolating from the colors and opacities in the eight voxels closest to each sample location as shown in figure 2. Finally, a fully opaque background of color $c_{bkg,\lambda}$ is draped behind the dataset and the interpolated colors and opacities are merged with each other and with the background by compositing in back-to-front order to yield a single color $C_\lambda(\mathbf{u}_{\bar{i}})$ for the ray and, since only one ray is cast per image pixel, for the pixel location $\mathbf{u}_{\bar{i}} = (u_{\bar{i}}, v_{\bar{j}})$ as well.

The compositing calculations referred to above are simply linear interpolations. Specifically, the color $C_{out,\lambda}(\mathbf{u}_{\bar{i}})$ of the ray as it leaves each sample location is related to the color $C_{in,\lambda}(\mathbf{u}_{\bar{i}})$ of the ray as it enters and the color $c_\lambda(\mathbf{x}_{\bar{i}})$ and opacity $\alpha(\mathbf{x}_{\bar{i}})$ at that sample location by the well-known transparency formula

$$C_{out,\lambda}(\mathbf{u}_{\bar{i}}) = C_{in,\lambda}(\mathbf{u}_{\bar{i}})(1 - \alpha(\mathbf{x}_{\bar{i}})) + c_\lambda(\mathbf{x}_{\bar{i}})\alpha(\mathbf{x}_{\bar{i}})$$

Solving for pixel color $C_\lambda(\mathbf{u}_{\bar{i}})$ in terms of the vector of sample colors $c_\lambda(\mathbf{x}_{\bar{i}})$ and opacities $\alpha(\mathbf{x}_{\bar{i}})$ gives

$$C_\lambda(\mathbf{u}_{\bar{i}}) = C_\lambda(u_{\bar{i}},v_{\bar{j}}) = \sum_{\bar{k}=0}^{K} \left[c_\lambda(x_{\bar{i}},y_{\bar{j}},z_{\bar{k}})\alpha(x_{\bar{i}},y_{\bar{j}},z_{\bar{k}}) \prod_{\bar{m}=\bar{k}+1}^{K} (1 - \alpha(x_{\bar{i}},y_{\bar{j}},z_{\bar{m}})) \right] \tag{1}$$

where $c_\lambda(x_{\bar{i}},y_{\bar{j}},z_0) = c_{bkg,\lambda}$ and $\alpha(x_{\bar{i}},y_{\bar{j}},z_0) = 1$.

3. Feature classification

Using the rendering method presented above, the mapping from acquired data to opacity performs the essential task of classification, enhancing selected features while suppressing others. Although color can also be used to classify data, as exemplified by pseudo-coloring, a hardwired shading model has been used here. 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.

3.1. Rendering of isovalue surfaces

We will first consider the detection 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. Current methods for presenting this data include stacks of isodensity contours, ridge lines arranged in 3-space so as to connect local density maxima [Williams82], and basket meshes representing selected isodensity surfaces [Purvis86].

One obvious way to generate raster visualizations of isovalue surfaces is to opaquely render all voxels having values greater than some threshold. This produces 3-D regions of opaque voxels the outermost layer of which is the desired isovalue surface. Unfortunately, this solution prevents display of multiple concentric semi-transparent surfaces, a very useful capability. Using a 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.

Figure 3 outlines the classification procedure employed in this study. It consists of five steps represented by the five numbered rows in the figure. The operator for each step is listed first and the resulting function is listed second. Graphs of each function for a typical data sample are shown on the right. In order to simplify the diagrams, functions of one dimension are used. Extension to three dimensions is straightforward and will be discussed later.

Beginning from the top of the figure and working downward, data acquisition consists of filtering a continuous scalar function $f(x)$, which we call our *input function*, to yield $\tilde{f}(x)$, then sampling to obtain discrete values $f(x_i)$. Ignoring data preparation for the moment, we can take this database to be the starting point for our classification procedure. For each discrete value, we form a reconstruction $\hat{f}(x)$ defined over a small interval surrounding the sample location x_i. A mapping $D = D(f)$ from input value to physical density is then applied, resulting in the continuous density function $D(x) = (\hat{f} \circ D)(x) = D(\hat{f}(x))$. We may call this intermediate result our *surface definition* *function*. Since the rendering method presented in section 2 operates on discrete opacities, we filter $D(x)$ to yield $\tilde{D}(x)$, sample to obtain discrete densities $D(x_i)$ and finally convert density to opacity using a mapping $\alpha = \alpha(D)$, resulting in discrete opacities $\alpha(x_i) = (D \circ \alpha)(x_i) = \alpha(D(x_i))$.

The physical model underlying our surface definition function is opaque particles suspended in a transparent medium where $D(x)$ gives the number of particles per unit volume. Using this model, the mapping from discrete density to discrete opacity is given by the exponential attenuation relation [Johns83]

$$\alpha(D(x_i)) = 1 - e^{-\mu\tau}$$

where the linear attenuation coefficient μ is set to density $D(x_i)$ and the optical depth τ is set to the interval s between samples.

Within this general framework, we have considerable latitude in the selection of operators. The choices shown here are based on empirical trials and represent only one possible implementation. Specifically, the reconstruction in step 1 is implemented using the first-degree Taylor polynomial of $f(x)$ near x_i:

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

for $x_i - r \leq x \leq x_i + r$ where the derivative is approximated using the operator

$$\frac{d(f(x))}{dx}(x_i) \approx \frac{1}{2}\left[f(x_{i+1}) - f(x_{i-1})\right]$$

This method was selected because it is inexpensive and localized, requiring only three samples to form each linear approximation. The radius r is some small integer multiple of the sampling interval s. Appropriate values for r are discussed later. The mapping in step 2 is implemented using the delta function

$$D(f) = D_v \delta(\hat{f} - f_v)$$

such that input value f_v is mapped to density D_v while all other values are mapped to a density of 0. This effectively detects each time the reconstructed function $\hat{f}(x)$ crosses f_v, producing a spike of density D_v at that x. The filter in step 3 is implemented by spatially convolving the resulting density function with a Bartlett window of radius r, which is given by

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

This filter is also inexpensive and localized. Since its non-zero extent exactly matches the interval over which each Taylor approximation is defined, the sampling in step 4 produces discrete densities $D(x_i)$ each of which depend only on input values $f(x_{i-1})$, $f(x_i)$ and $f(x_{i+1})$.

We extend this method to three dimensions by using $f(\mathbf{x}_i)$ in place of $f(x_i)$ and the magnitude of the gradient vector $|\nabla f(\mathbf{x}_i)|$ in place of the derivative $\frac{d(f(x))}{dx}(x_i)$. We also replace the 1-D Bartlett window with a 3-D spherically symmetric linear ramp. The replacement of a signed derivative by an unsigned vector magnitude works because the filter kernel is isotropic. Making the appropriate substitutions in each of the above expressions and solving for $\alpha(\mathbf{x}_i)$ gives

$$\alpha(\mathbf{x}_i) = 1 - e^{-D(\mathbf{x}_i)s} \tag{2}$$

where

$$D(\mathbf{x}_i) = D_v \begin{cases} 1 & \text{if } |\nabla f(\mathbf{x}_i)| = 0 \text{ and } f(\mathbf{x}_i) = f_v \\ 1 - \dfrac{s}{r} \left| \dfrac{f_v - f(\mathbf{x}_i)}{|\nabla f(\mathbf{x}_i)|} \right| & \text{if } |\nabla f(\mathbf{x}_i)| > 0 \text{ and } f(\mathbf{x}_i) - \dfrac{r}{s}|\nabla f(\mathbf{x}_i)| \le f_v \le f(\mathbf{x}_i) + \dfrac{r}{s}|\nabla f(\mathbf{x}_i)| \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

and the gradient vector is approximated using the operator

$$\nabla f(\mathbf{x}_i) = \nabla f(x_i, y_j, z_k)$$

$$\approx \left[\frac{1}{2}\left[f(x_{i+1},y_j,z_k) - f(x_{i-1},y_j,z_k)\right], \frac{1}{2}\left[f(x_i,y_{j+1},z_k) - f(x_i,y_{j-1},z_k)\right], \frac{1}{2}\left[f(x_i,y_j,z_{k+1}) - f(x_i,y_j,z_{k-1})\right] \right]. \quad (4)$$

A graph of $D(\mathbf{x}_i)$ as a function of $f(\mathbf{x}_i)$ and $|\nabla f(\mathbf{x}_i)|$ for typical values of f_v, D_v, s and r is shown in figure 5.

If more than one isovalue surface is to be rendered in a single image, they can be classified separately and their densities added. Specifically, given input values f_{v_n}, $n = 1, \ldots, N$, $N \ge 1$, densities D_{v_n} and filter radii r_n, we can use equation (3) to compute $D_n(\mathbf{x}_i)$, then apply

$$D_{tot}(\mathbf{x}_i) = \sum_{n=1}^{N} D_n(\mathbf{x}_i) \quad (5a)$$

to obtain a single density value for use in equation (2). Alternatively, we can use equations (2) and (3) to compute opacities $\alpha_n(\mathbf{x}_i)$ and combine them using the relation

$$\alpha_{tot}(\mathbf{x}_i) = 1 - \prod_{n=1}^{N} (1 - \alpha_n(\mathbf{x}_i)). \quad (5b)$$

3.2. Rendering of region boundary surfaces

We will next consider the detection of surfaces bounding regions of constant value 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 anatomical features can be inferred. The currently dominant method for presenting these surfaces involves forming a mesh of polygons that either connects contours drawn on each slice [Fuchs77] or follows a selected isovalue surface [Lorensen87]. Approaches based on volumetric rendering were surveyed in the introduction.

It is not clear that isovalue surfaces are well suited to the display of medical data. The reason can be explained briefly as follows. Given an input function containing two tissue types A and B having values f_{v_A} and f_{v_B} where $f_{v_A} < f_{v_B}$, data acquisition will produce voxels having values $f(\mathbf{x}_i)$ such that $f_{v_A} \le f(\mathbf{x}_i) \le f_{v_B}$. Thin features of tissue type B may be represented by regions in which all voxels bear values less than f_{v_B}. Indeed, there is no threshold value greater than f_{v_A} guaranteed to detect arbitrarily thin regions of type B and use of thresholds close to f_{v_A} are as likely to detect noise as signal.

Figure 4 outlines the procedure employed in this study. It is similar to the method described in section 3.1, having one additional operation between steps 4 and 5 and differing in the choice of operators for the other steps. Working again in one dimension, the reconstruction in step 1 is implemented using the step function

$$\hat{f}(x) = \begin{cases} f_{v_A} & \text{if } x < x_i + s\left[\frac{1}{2} - \frac{f(x_i) - f_{v_A}}{f_{v_B} - f_{v_A}}\right] \\ f_{v_B} & \text{otherwise.} \end{cases}$$

for $x_i - \frac{s}{2} \leq x \leq x_i + \frac{s}{2}$. Taken in isolation, this function is correct only if the derivative $\frac{d(f(x))}{dx}(x_i)$ is positive but it produces correct results for derivatives of either sign when used in conjunction with the isotropic filter in step 3. The effect of this reconstruction is to estimate the x-location of the region boundary that gave rise to discrete value $f(x_i)$. The mapping in step 2 is implemented using the linear expression

$$D(f) = D_v(f - f_{v_A})$$

such that input value f_{v_A} is mapped to a density of 0 and f_{v_B} is mapped to density D_v, thus displaying tissues of type A transparently and tissues of type B opaquely or semi-opaquely. The filter in step 3 is implemented by spatially convolving the resulting density function with a box filter of radius $\frac{s}{2}$, which is given by

$$g(x) = \begin{cases} 1 & \text{if } |x| \leq \frac{s}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Since the non-zero extent of this filter is equal to the sampling interval s, the sampling in step 4 produces discrete densities $D(x_i)$ each of which depend only on input value $f(x_i)$.

In order to allow multiple concentric semi-transparent surfaces, we must render region boundaries opaquely without also making enclosed regions opaque. This necessitates an additional classification operation. To avoid artifacts when confronted with small or poorly defined regions, we should use a continuous decision function. This additional step, numbered 4.5 in the figure, is implemented using the edge enhancement operator

$$D_2(x_i) = D(x_i)\left|\frac{d(f(x))}{dx}(x_i)\right|$$

where the derivative is approximated as described in section 3.1. Since resolution is an important consideration in medical imaging, the narrower asymmetric approximation

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

may be substituted if the data is relatively noise-free or has been pre-smoothed.

Applying the final conversion to opacity and extending the method to three dimensions as described in section 3.1, then combining expressions and solving for $\alpha(x_i)$ gives

$$\alpha(x_i) = 1 - e^{-D_2(x_i)s} \tag{6}$$

where

$$D_2(\mathbf{x}_i) = |\nabla f(\mathbf{x}_i)|\, D_v \begin{cases} 1 & \text{if } f(x_i) > f_{v_B} \\ \dfrac{f(x_i) - f_{v_A}}{f_{v_B} - f_{v_A}} & \text{if } f_{v_A} \le f(x_i) \le f_{v_B} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and the gradient vector is approximated using equation (4) or the operator

$$\nabla f(\mathbf{x}_i) = \nabla f(x_i, y_j, z_k)$$
$$\approx \left[f(x_{i+1}, y_j, z_k) - f(x_i, y_j, z_k),\ f(x_i, y_{j+1}, z_k) - f(x_i, y_j, z_k),\ f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_k) \right]. \tag{8}$$

A graph of $D_2(\mathbf{x}_i)$ as a function of $f(\mathbf{x}_i)$ and $|\nabla f(\mathbf{x}_i)|$ for typical values of f_{v_A}, f_{v_B} and D_v is shown in figure 6.

If the acquired data contains more than two types of tissues, it can still be rendered correctly providing that certain adjacency criteria are met. Specifically, given an input function $f(\mathbf{x}_i)$ containing N tissue types having values f_{v_n}, $n = 1, \ldots, N$, $N \ge 1$ such that $f_{v_m} < f_{v_{m+1}}$, $m = 1, \ldots, N-1$, no tissue of value $f_{v_{n_1}}$ may touch any tissue of value $f_{v_{n_2}}$, $|n_1 - n_2| > 1$. If these criteria are met, the $N-1$ types of region boundaries each consisting of the pair of tissue values f_{v_m} and $f_{v_{m+1}}$ can be classified separately and their densities added or opacities combined as described in section 3.1. Violation of these rules leads to voxels that cannot be unambiguously classified as belonging to one type of region boundary or another and hence cannot be rendered correctly.

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 [Phong75]:

$$c_\lambda(\mathbf{x}_i) = c_{p,\lambda} k_{a,\lambda} + \frac{c_{p,\lambda}}{k_1 + k_2 d(\mathbf{x}_i)} \left[k_{d,\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{L}) + k_{s,\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{H})^n \right] \tag{9}$$

where

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

$c_{p,\lambda}$ = 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,

k_1, k_2 = constants used in linear approximation of depth-cueing,

$d(\mathbf{x}_i)$ = perpendicular distance from picture plane to voxel location $\mathbf{x}_i$,

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

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

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

Since a parallel light source is used, **L** is a constant. Furthermore,

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

where

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

Since an orthographic projection is used, **V** and hence **H** are also constants. Finally, the surface normal is given by

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

There are many ways to estimate the gradient vector $\nabla f(\mathbf{x}_i)$. 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 (4) and (8). It is worth noting that these operators are always applied to unclassified function values $f(\mathbf{x}_i)$. By separating estimation of surface normals from classification, we insure an undistorted visualization of shapes.

5. Discussion

5.1. Computational complexity

One of the strengths of the rendering method presented in this paper is its low computational expense. Since intermediate results are stored at various stages along the rendering pipeline, the cost of producing a new image depends on which parameters are changed. Let us consider some typical cases.

Given input value $f(\mathbf{x}_i)$ and gradient magnitude $|\nabla f(\mathbf{x}_i)|$, application of feature classification to yield opacity $\alpha(\mathbf{x}_i)$ can be implemented with one lookup table reference. This implies that if we store gradient magnitudes for all voxels, computation of new opacities following a change in classification parameters entails only generation of a new lookup table followed by one table reference per voxel.

The cost of computing new colors $c_\lambda(\mathbf{x}_i)$ following a change in observer direction **V**, light source direction **L** or other shading parameter is more substantial. Effective rotation sequences can be produced, however, using a single set of colors. The visual manifestation of fixing the shading is that 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 an unchanging light source is obtained.

The most efficient way to produce a rotation sequence is to hold both colors and opacities constant and alter only the direction in which rays are cast. If we assume a square image n pixels wide and use orthographic projection, in which case sample coordinates can be efficiently calculated using differencing, the combined cost of ray tracing, tri-linear interpolation and compositing to compute n^2 pixel colors $C_\lambda(\mathbf{u}_I)$ is only $3Kn^2$ additions and $4Kn^2$ linear interpolations where K is the number of samples computed along each ray. This modest expense can be reduced further by suppressing tri-linear interpolation of colors and compositing for samples whose opacity is determined to be zero.

5.2. Image quality

Although the notation used in equation (1) has been borrowed from image compositing, the analogy is not exact and the differences are fundamental. The alpha values used in [Wallace82, Porter84, Duff85] are a function of viewpoint-dependent coverage, which in turn tells us what percentage of a pixel's area is covered when an opaque 3-D geometry is projected in a given viewing direction onto the picture plane. The alpha values used in equation (1) are a function of viewpoint-independent coverage, which tells us what percentage of a voxel's volume is covered by an opaque 3-D geometry but not what percentage of its profile relative to the view direction is covered. In other words, by using compositing to render a volumetric database, we implicitly assume that voxels contain no geometric detail; the covered portion of a voxel is assumed to obscure the covered and uncovered portions of the voxel behind it in equal shares regardless of viewing direction. We are essentially rendering voxel-sized cubes of semi-transparent gel having homogeneous color and having opacity derived from the known coverage percentage.

Since our method does not consider sub-voxel geometry, image quality is limited by the number of viewing rays. In the current implementation, we cast one ray per pixel. Improvements can be obtained by super-sampling and averaging down but cost rises as the cube of the number of samples in each direction. The solution used in this paper is to blur the opacities prior to compositing. Shading, which carries most of the 3-D shape information, is left untouched.

Although filtering 3-D geometry is a valid method of anti-aliasing, it is not equivalent to filtering its 2-D projection. Specifically, both occur prior to image sampling but the former occurs prior to visibility calculations while the latter occurs after. To illustrate the distinction, let us consider the simple case of two spheres of identical size but different colors placed one behind the other relative to our eye. If an analytic hidden-surface removal algorithm is applied, then the resulting 2-D projection is filtered, the closer sphere will entirely obscure the farther one. If the 3-D object geometry is filtered first, producing blurry spheres, then the same analytic hidden-surface algorithm is applied, the color of the farther sphere will leak through and appear as a halo around the closer sphere. For the datasets studied in this paper, these halo effects are either not seen because suitable alignments seldom occur or they occur but are not visually objectionable.

The decision to reduce aliasing at the expense of of resolution arises from two conflicting goals: producing artifact-free images and keeping rendering costs low. In practice, the slight loss in image sharpness might not be disadvantageous. Given the approximate nature of surface reconstruction operators, it is not clear that the accuracy afforded by more expensive visibility calculations is useful. Indeed, the sharpness of surface silhouettes in [Fuchs77, Lorensen87] can be misleading. Blurry silhouettes have less visual impact, but they reflect the true imprecision of the surface reconstruction process.

The implementation of opacity blurring depends on the feature classification procedure employed. For isovalue surfaces, some blurring of opacities can be obtained by selecting filter radii r_n that exceed the sampling interval s. A value of $2s$ was used to produce figure 8. Since each opacity value depends only on input values in a small neighborhood surrounding the sample location, this method only works if the data is already fairly smooth. Fortunately, the time and ensemble averaging inherent in X-ray crystallography generates suitable data. Application of global filtering during data preparation can be used to obtain further smoothing if necessary but the blurred database should only be used as input to the classification procedure so as not to adversely affect shading. For region boundary surfaces, some blurring of opacities is inherent in the filtering applied in step 3 but, since each opacity value depends only on a single input value, the results are satisfactory only if the data is already smooth. Application of global filtering during data preparation can be used if necessary.

An interesting byproduct of this approach is that the level of blurring can be selected on an object-by-object basis. This capability allows us to take advantage of the fact that the human visual system is strongly distracted by coherent aliasing but readily overlooks an equivalent amount of noise. Specifically, for large surfaces of gradual curvature where aliasing caused by point sampling would be coherent and hence visually objectionable, strong blurring can be applied. For small features where aliasing would be incoherent and minimally distracting, less blurring is required. A judicious application of adaptive filtering has the potential to squeeze more useful information out of a given dataset than global blurring.

Another issue that affects image quality is the order of operations within the rendering pipeline. For example, since many data preparation tasks are geometric in nature, the opportunity exists to incorporate them into the ray tracing step. There are strong arguments, however, for applying them beforehand. Phong has observed that better specular highlights are obtained by interpolating normals than by interpolating intensities [Phong75]. The underlying reason is that specular reflection is a non-linear function of surface orientation. The same principle applies to the non-linear relationship between input function value and opacity. This suggests that we interpolate the acquired data prior to both shading and classification. A 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. Since data preparation is performed only once, the additional expense is amortized over all images produced using that dataset. Rotations, unlike interpolation, are best applied near the end of the rendering pipeline. Since gradient vectors have already been estimated, we can rotate the data using an inexpensive first-order filter without introducing objectionable errors into the shading or classification.

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 11/780 having sufficient physical memory to prevent paging.

The dataset used in the crystallography study is from the protein Cytochrome B5. It was initially acquired 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 interpolated to fill a 113 x 113 x 113 voxel dataset in 6 hours using a 3-D separable cubic B-spline. Figure 7 shows four slices spaced 10 voxels apart in this expanded dataset. Each whitish cloud represents a single atom. Classification and shading were applied as described in sections 3.1 and 4 and required 30 minutes total. Ray tracing and compositing were performed as described in section 2 and took 4 hours, yielding the image in figure 8. Some scaling was also included in the projection operation, producing a 400 x 400 pixel image. Although this is not in accordance with the principle discussed earlier of interpolating during data preparation rather than after shading and classification, it was used here as an expedient. Without scaling, the image would have been 200 x 200 pixels and would have required 30 minutes to render. The presence of the dataset boundaries within the field of view gives rise to artifacts in some parts of the image, but the clipping of isovalue surfaces that occurs along these boundaries has been found to provide a useful shape cue.

The dataset used in the medical imaging study is of a cadaver and was acquired as a 256 x 256 x 113 sample grid. This was expanded to fill a 256 x 256 x 226 voxel dataset by interpolating in one direction only, which took 90 minutes. Each of the images in figure 9 represents a different set of classification and shading parameters. For each image, colors and opacities were computed as described in sections 3.2 and 4. This step required 4 hours per image. Ray tracing and compositing were then performed as described in section 2, taking 2 hours per image. 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 semi-transparently over the bone surface in the lower-right image. Figure 10 was produced from the colors and opacities already computed for the lower-left image in figure 9. Some scaling was included in this projection, producing a 512 x 512 pixel image in 8 hours. 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 ray tracing. 28 megabytes were required to store the 256 x 256 x 226 8-bit 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 classification and visibility do occur, as does aliasing. The use of compositing introduces further approximations, hence more errors. The visual impact of these errors can be reduced by following several guidelines. Firstly, all voxels participate in the rendering of any image. Secondly, any decision functions used during feature classification are continuous rather than binary. Thirdly, shading and classification calculations are independent, particularly with regard to estimation of surface normals.

A number of improvements can be made to the current implementation. For isovalue surfaces, more work is needed on reconstruction methods, 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, successful reconstruction depends on the accuracy of our model of the input function and the acquisition process. Specifically, the use of a linear mapping from $f(x_i)$ to x in step 1 is based on the assumption that tissues are homogeneous, their transitions are step functions and bandlimiting was performed prior to acquisition using a box filter of radius $\frac{s}{2}$. Excessively gradual transitions in voxel value, whether due to fluctuations in tissue density, gradual transitions between tissues or the use of wide filter kernels during bandlimiting will result in incorrect estimation of region boundary locations and excessively blurry renditions. For typical CT data, our model of tissues and tissue transitions is generally correct. For typical medical scanners, our model of the aquisition process is nearly correct in the z-direction (between slices) but only approximate in the x and y directions (within a slice). The characteristics of specific scanners should be measured and incorporated into the classification procedure.

Another issue related to region boundary surfaces is the rendering of tissues not meeting the adjacency criteria described in section 3.2. This includes most internal soft tissue organs. One possible approach would be to employ user interaction and/or automatic scene analysis to isolate sets of voxels that meet the adjacency criteria. Since the user or algorithm is not called upon to define surface geometry but merely to isolate regions of interest, this approach promises to be easier and to produce better images than techniques involving surface fitting.

Although this paper focuses on rendering of surfaces, the method extends readily to rendering of semi-transparent interstitial volumes. 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 geometric 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. 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 [Duff85]. Another is to scan-convert the geometry directly into the acquired database and render the ensemble. A third is to incorporate classical rendering of the geometry directly into the volumetric rendering algorithm.

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 geometrically transforming two values per voxel and 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 [Catmull80]. 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 [Kaufman86]. 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. Turner Whitted, Henry Fuchs, Steven M. Pizer and Frederick P. Brooks Jr. for their encouragement on this paper, and John M. Gauch, Andrew S. Glassner, Lawrence M. Lifshitz, John M. Gauch, Lee Westover and Mark Harris for enlightening discussions and advice. The comments of Nelson Max on an early version of this paper were also helpful. The electron density map used in this study was obtained from Jane and Dave Richardson of Duke University and the CT scan was provided by the Radiation Oncology Department at North Carolina Memorial Hospital. This work was supported by NIH grant RR02170.

References

[Artzy81] 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.

[Blinn82] Blinn, James F., "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces," *Computer Graphics,* Vol. 16, No. 3, July, 1982, pp. 21-29.

[Catmull80] Catmull, Edwin and Smith, Alvy Ray, "3-D Transformations of Images in Scanline Order," *Computer Graphics,* Vol. 14, No. 3, July, 1980, pp. 279-285.

[Chen85] 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.

[Csuri79] Csuri, C., Hackathron, R., Parent, R., Carlson, W. and Howard, M., "Towards an Interactive High Visual Complexity Animation System," *Computer Graphics*, Vol. 13, No. 2, August, 1979, pp. 289-299.

[Duff85] Duff, Tom, "Compositing 3-D Rendered Images," *Computer Graphics*, Vol. 19, No. 3, July, 1985, pp. 41-44.

[Fuchs77] 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.

[Goldwasser86] 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.

[Herman79] 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.

[Hohne86] 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.

[Hohne87] Hohne, K. H., Riemer, M. and Tiede, U., "Viewing Operations for 3D - Tomographic Gray Level Data," *CAR '87 conference proceedings.*

[Johns83] Johns, Harold Elford and Cunningham, John Robert, *The Physics of Radiology*, Charles C. Thomas, 1983.

[Kajiya84] Kajiya, James T. "Ray Tracing Volume Densities," *Computer Graphics*, Vol. 18, No. 3, July, 1984, pp. 165-174.

[Kaufman86] Kaufman, Arie, "Voxel-Based Architectures for Three-Dimensional Graphics," *Proceedings of the IFIP 10th World Computer Congress*, Dublin, Ireland, September, 1986, pp. 361-366.

[Lorensen87] Lorensen, William E. and Cline, Harvey E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 4, July, 1987, pp. 163-169.

[Phong75] Bui-Tuong, Phong, "Illumination for Computer-Generated Pictures," *Communications of the ACM*. Vol. 18, No. 6, June, 1975, pp. 311-317.

[Porter84] Porter, Thomas and Duff, Tom, "Compositing Digital Images," *Computer Graphics*, Vol. 18, No. 3, July, 1984, pp. 253-259.

[Purvis86] 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.

[Schlusselberg86] 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.

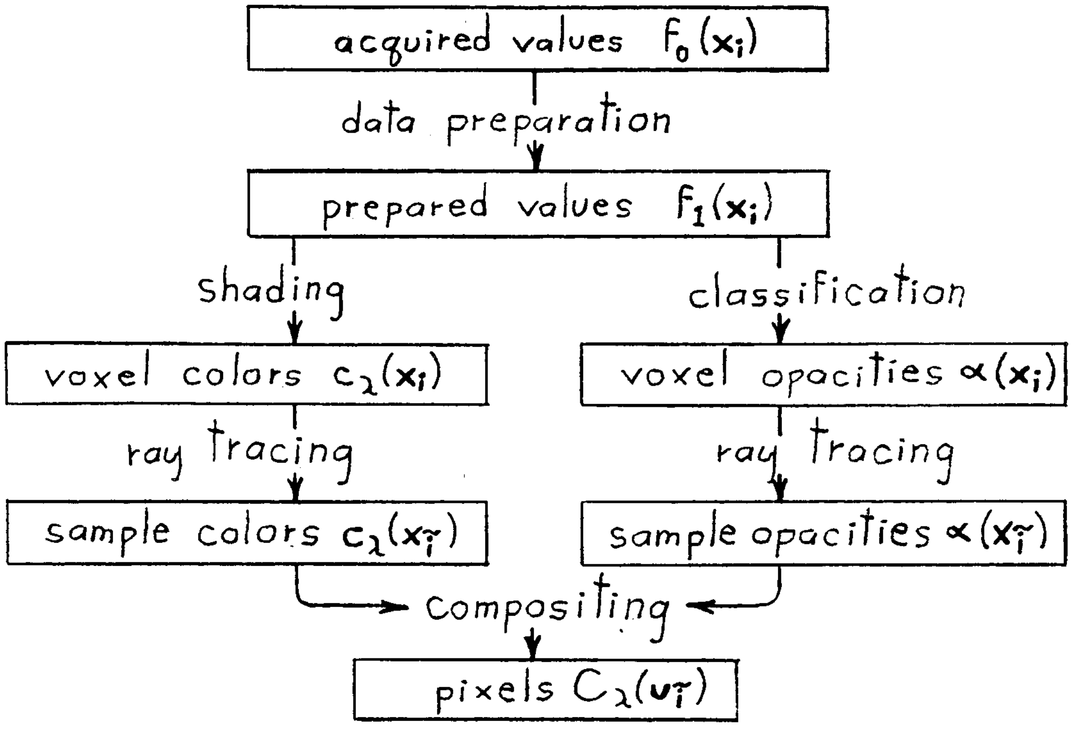
[Smith87] Smith, Alvy Ray, "Volume graphics and Volume Visualization: A Tutorial," Technical Memo 176, PIXAR Inc., San Rafael, California, May, 1987.

[Trousset87] Trousset, Yves and Schmitt, Francis, "Active-Ray Tracing for 3D Medical Imaging," *EUROGRAPHICS '87 conference proceedings*, pp. 139-149.

[Wallace82] Wallace, Bruce A., "Merging and Transformation of Raster Images for Cartoon Animation," *Computer Graphics*, Vol. 15, No. 3, August, 1981, pp. 253-262.

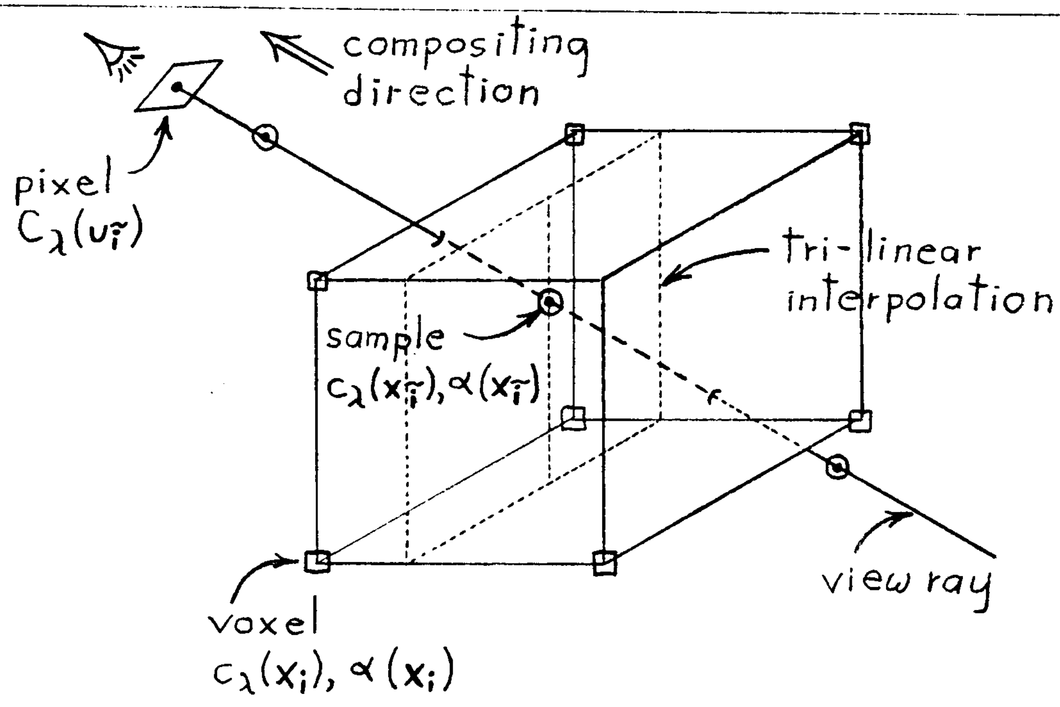
[Williams82] Williams, Thomas Victor, *A Man-Machine Interface for Interpreting Electron Density Maps*, PhD thesis, University of North Carolina, Chapel Hill, NC, 1982.

[Whitted80] Whitted, Turner, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, Vol. 23, No. 6, June, 1980, pp. 343-349.



acquired values $f_0(x_i)$
data preparation
prepared values $f_1(x_i)$
shading
classification
voxel colors $c_\lambda(x_i)$
voxel opacities $\alpha(x_i)$
ray tracing
ray tracing
sample colors $c_\lambda(x_{\tilde{i}})$
sample opacities $\alpha(x_{\tilde{i}})$
compositing
pixels $C_\lambda(u_{\tilde{i}})$

Figure 1 - Overview of rendering method



compositing direction
pixel $C_\lambda(u_{\tilde{i}})$
tri-linear interpolation
sample $c_\lambda(x_{\tilde{i}}), \alpha(x_{\tilde{i}})$
view ray
voxel $c_\lambda(x_i), \alpha(x_i)$

Figure 2 - Ray tracing and compositing steps

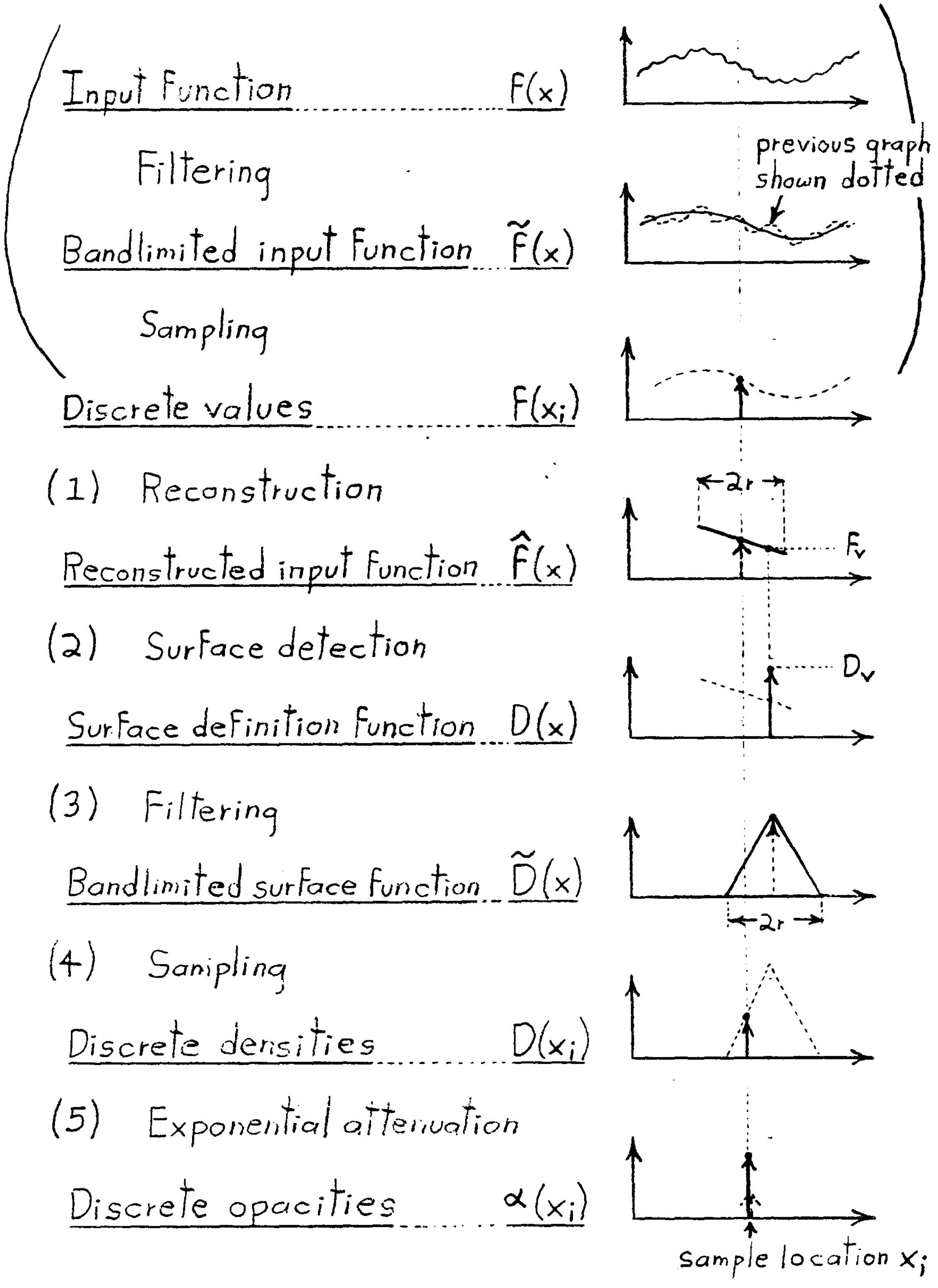

Input Function F(x)
Filtering
previous graph shown dotted
Bandlimited input Function $\tilde{F}(x)$
Sampling
Discrete values $F(x_i)$
(1) Reconstruction
2r
F_v
Reconstructed input Function $\hat{F}(x)$
(2) Surface detection
D_v
Surface definition Function D(x)
(3) Filtering
Bandlimited surface Function $\tilde{D}(x)$
2r
(4) Sampling
Discrete densities $D(x_i)$
(5) Exponential attenuation
Discrete opacities $\alpha(x_i)$
sample location x_i

Figure 3 - Procedure for classifying isovalue surfaces

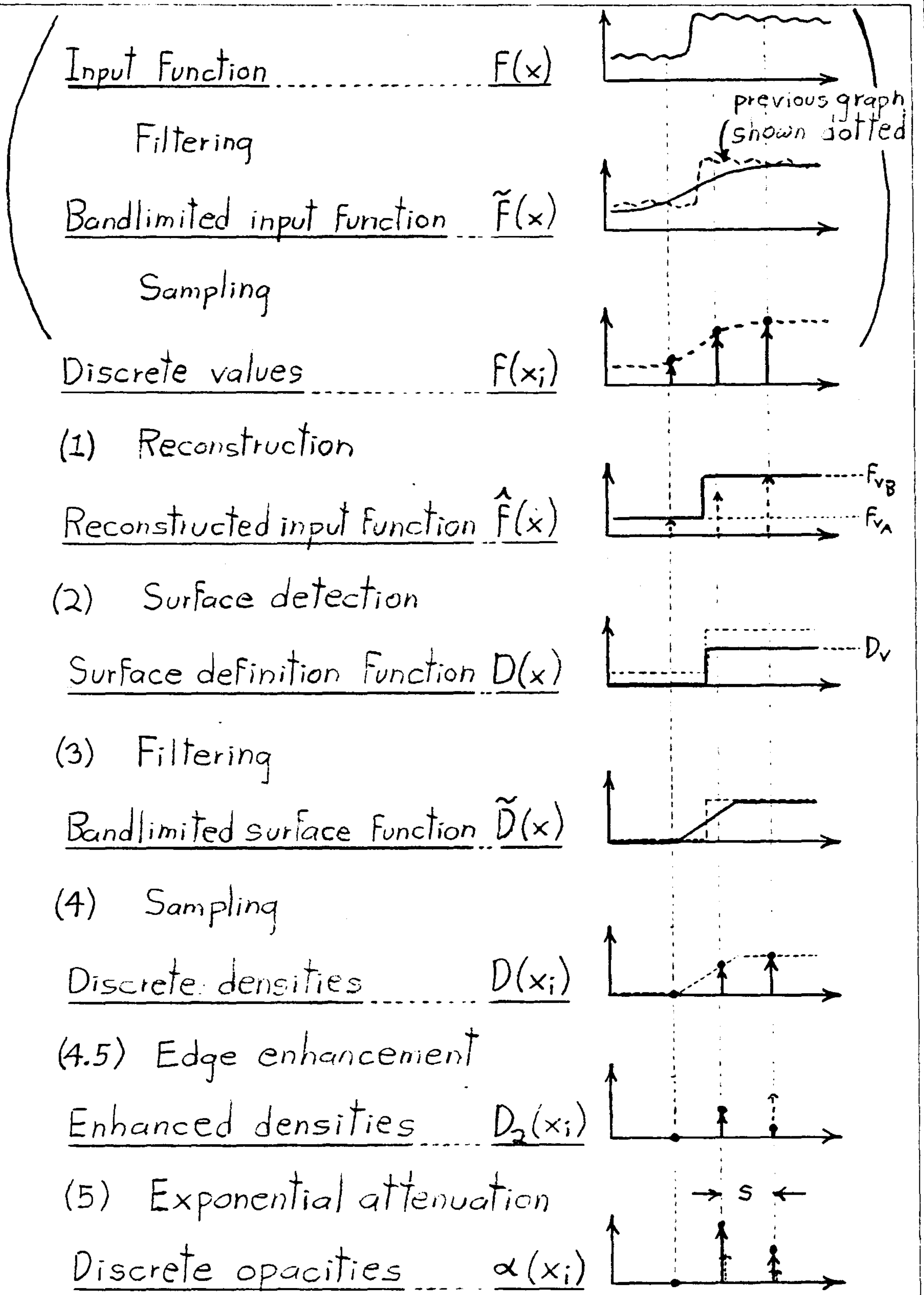
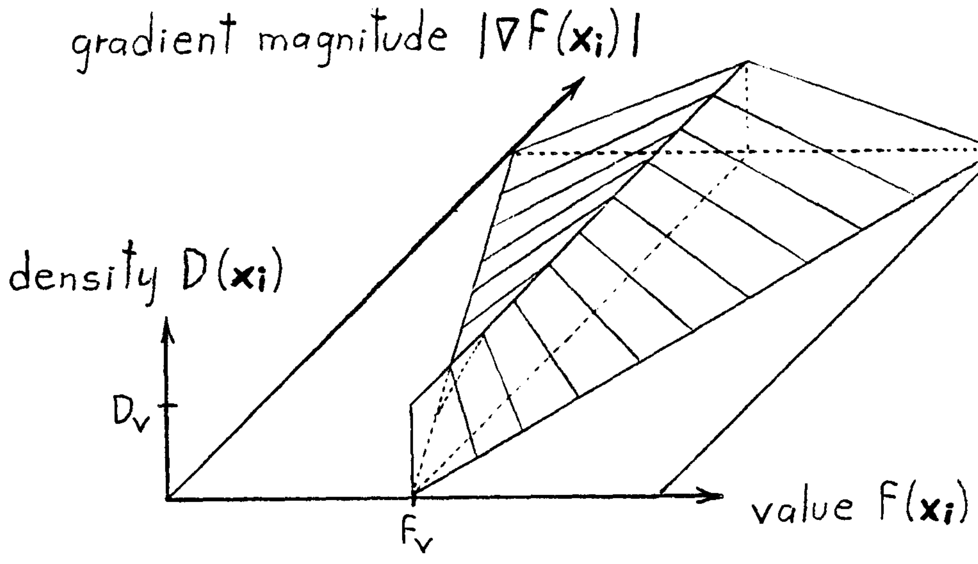
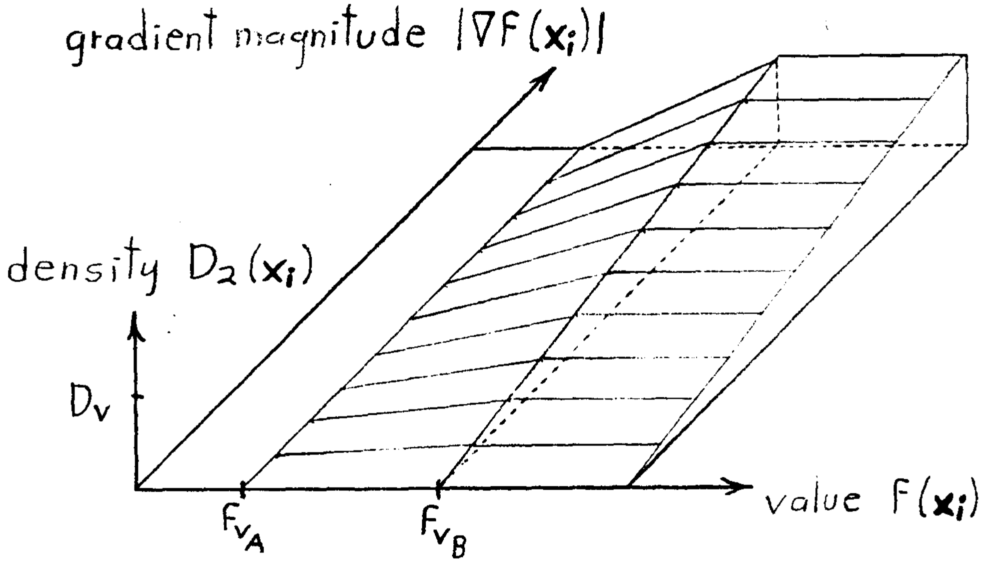

Input Function $F(x)$
Filtering
previous graph (shown dotted)
Bandlimited input function $\tilde{F}(x)$
Sampling
Discrete values $F(x_i)$
(1) Reconstruction
Reconstructed input function $\hat{F}(x)$
F_{v_B}
F_{v_A}
(2) Surface detection
Surface definition function $D(x)$
D_v
(3) Filtering
Bandlimited surface function $\tilde{D}(x)$
(4) Sampling
Discrete densities $D(x_i)$
(4.5) Edge enhancement
Enhanced densities $D_2(x_i)$
(5) Exponential attenuation
s
Discrete opacities $\alpha(x_i)$

Figure 4 - Procedure for classifying region boundary surfaces



gradient magnitude $|\nabla F(\mathbf{x}_i)|$
density $D(\mathbf{x}_i)$
D_v
F_v
value $F(\mathbf{x}_i)$

Figure 5 - Isovalue surface classification mapping



gradient magnitude $|\nabla F(\mathbf{x}_i)|$
density $D_2(\mathbf{x}_i)$
D_v
F_{v_A}
F_{v_B}
value $F(\mathbf{x}_i)$

Figure 6 - Region boundary surface classification mapping

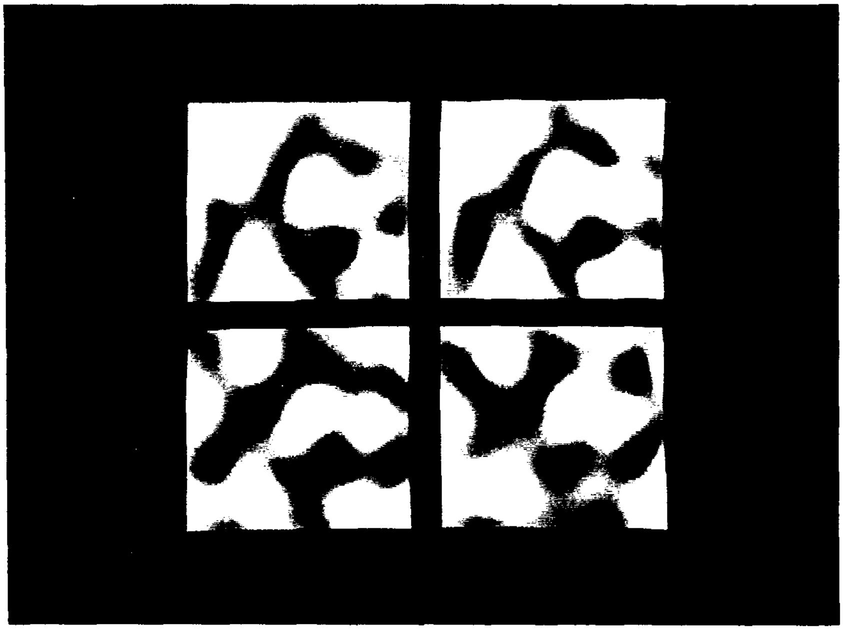


Figure 7 - Slices from electron density map

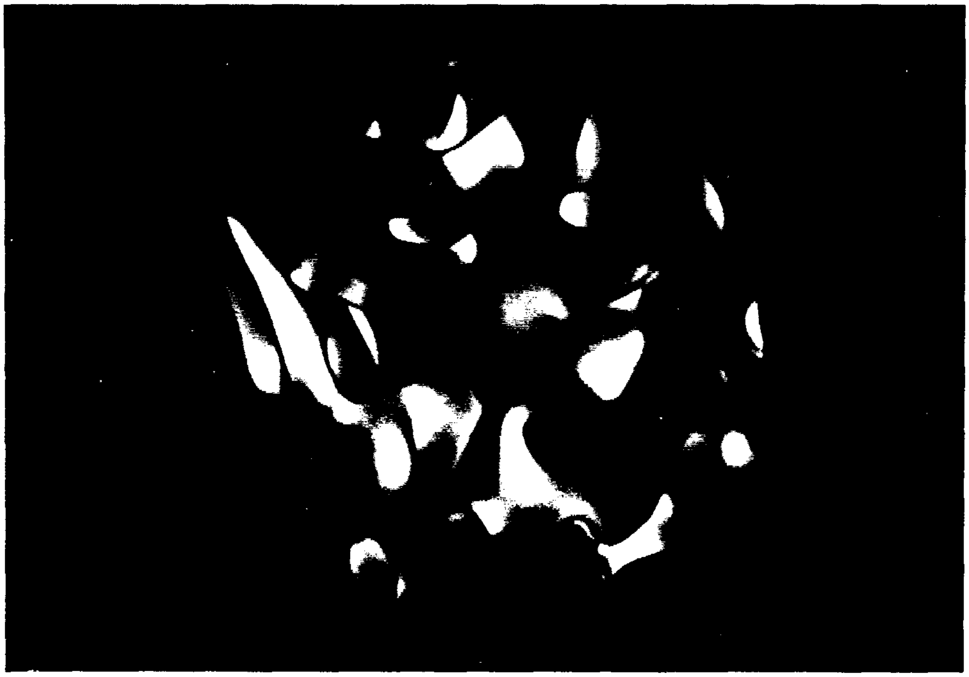


Figure 8 - Isovalue surface from electron density map

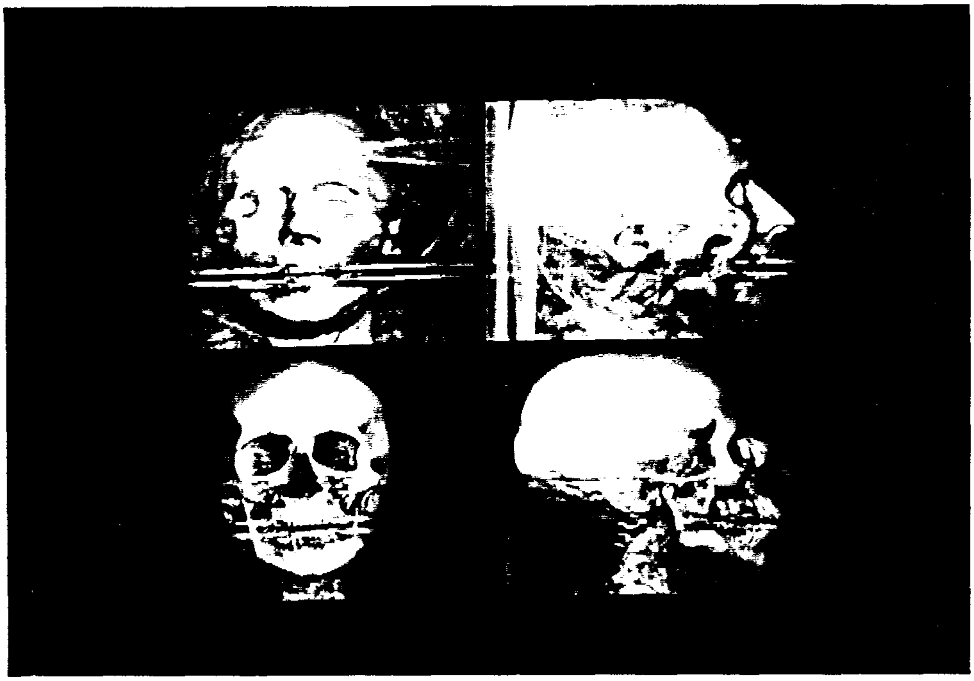


Figure 9 - Region boundary surfaces from CT data

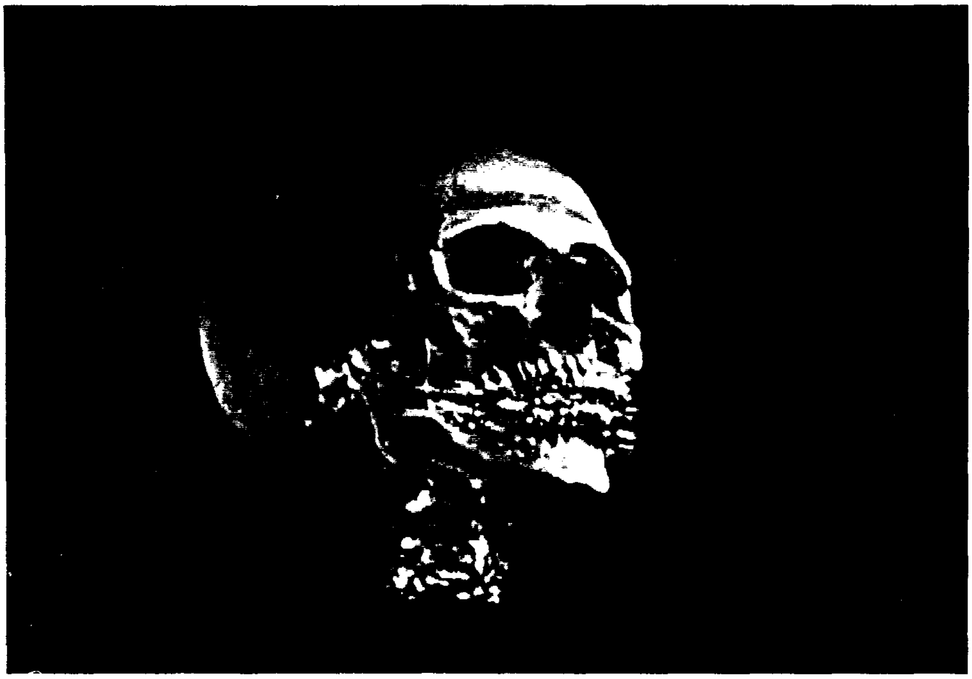


Figure 10 - Rotated version of figure 9, lower-left image