Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering

Gordon Kindlmann [*] and James W. Durkin [†]

Program of Computer Graphics
Cornell University

Abstract

Although direct volume rendering is a powerful tool for visualizing complex structures within volume data, the size and complexity of the parameter space controlling the rendering process makes generating an informative rendering challenging. In particular, the specification of the transfer function — the mapping from data values to renderable optical properties — is frequently a time-consuming and unintuitive task. Ideally, the data being visualized should itself suggest an appropriate transfer function that brings out the features of interest without obscuring them with elements of little importance. We demonstrate that this is possible for a large class of scalar volume data, namely that where the regions of interest are the boundaries between different materials. A transfer function which makes boundaries readily visible can be generated from the relationship between three quantities: the data value and its first and second directional derivatives along the gradient direction. A data structure we term the *histogram volume* captures the relationship between these quantities throughout the volume in a position independent, computationally efficient fashion. We describe the theoretical importance of the quantities measured by the histogram volume, the implementation issues in its calculation, and a method for semi-automatic transfer function generation through its analysis. We conclude with results of the method on both idealized synthetic data as well as real world datasets.

1 Introduction

1.1 The Task of Finding Transfer Functions

Transfer functions make a volume dataset visible by assigning renderable optical properties to the numerical values which comprise the dataset. The most general transfer functions are those that assign opacity, color, and emittance [12]. Useful renderings can often be obtained, however, from transfer functions which assign just opacity, with the color and brightness derived from simulated lights which illuminate the volume according to some shading model. We use the term *opacity functions* to refer to this limited subset of transfer functions. During the rendering process, the sampled and interpolated data values are passed through the opacity function to determine their contribution to the final image. Since the opacity function does not normally take into account the position of the region being rendered, the role of the opacity function is to make opaque those data values which consistently correspond, across the whole volume, to features of interest. This paper addresses only the problem of setting opacity functions, as this is a non-trivial yet manageable problem whose solution is pertinent to more general transfer function specification issues.

[*] Address: Department of Computer Science, University of Utah, Salt Lake City, UT 84102. Email: gk@cs.utah.edu.

[†] Address: Program of Computer Graphics, Cornell University, Ithaca, NY 14853. Email: jwd@graphics.cornell.edu.

Finding a good transfer function is critical to producing an informative rendering, but even if the only variable which needs to be set is opacity, it is a difficult task. Looking through slices of the volume dataset allows one to spatially locate features of interest, and a means of reading off data values from a user-specified point on the slice can help in setting an opacity function to highlight those features, but there is no way to know how representative of the whole feature, in three dimensions, these individually sampled values are. User interfaces for opacity function specification typically allow the user to alter the opacity function by directly editing its graph, usually as a series of linear ramps joining adjustable control points. This interface doesn't itself guide the user towards a useful setting, as the movement of the control points is unconstrained and unrelated to the underlying data. Thus finding a good opacity function tends to be a slow and frustrating trial and error process, with seemingly minor changes in an opacity function leading to drastic changes in the rendered image. This is made more confusing by the interaction of other rendering parameters such as shading, lighting, and viewing angle.

1.2 Direct Volume Rendering of Boundaries

A significant assumption made in this paper is that the features of interest in the scalar volume are the boundary regions between areas of relatively homogeneous material[1]. For instance, this is often true of datasets from medical imaging. But if the goal is to render the boundaries of objects, why use direct volume rendering, and not isosurface rendering? Although this question itself deserves investigation, it is widely accepted that direct volume rendering avoids the binary classification inherent in isosurface rendering — either the isosurface passes through a voxel or not [11]. To the extent that an object's surface is associated with a range of values, an opacity function can make a range of values opaque or translucent. This becomes especially useful when noise or measurement artifacts upset the correlation between data value and material type.

As a quick illustration of this, consider a dataset generated from limited angle tomography [6], where there are often streaks and blurriness in the data caused by the unavailability of projections at some range of angles. This type of data is studied in the Collaboratory for Microscopic Digital Anatomy[19], an ongoing project aimed at providing remote, networked access to sophisticated microscopy resources. Fig. 1 shows two renderings of a mammalian neuron dataset, using the same viewing angle, shading, and lighting parameters, but rendered with different algorithms: a non-polygonal ray-cast isosurface rendering and a shear-warp direct volume rendering produced with the Stanford VolPack rendering library [10]. Towards the bottom of the direct volume rendered image, there is some fogginess surrounding the surface, and the surface itself is not very clear. As can be confirmed by looking directly at slices of the data itself, this corresponds exactly to a region of the

[1] We use *boundary* to refer, not to an infinitesimally thin seperating surface between two areas of disparate data value, but to the thin region wherein the data value transitions from one material value to the other.

dataset where the material boundary is in fact poorly defined. The surface rendering, however, shows as distinct a surface here as everywhere else, and in this case the poor surface definition in the data is manifested as a region of rough texture. This can be misleading, as there is no way to know from this rendering alone that the rough texture is due to measurement artifacts, and not a feature on the dendrite itself.

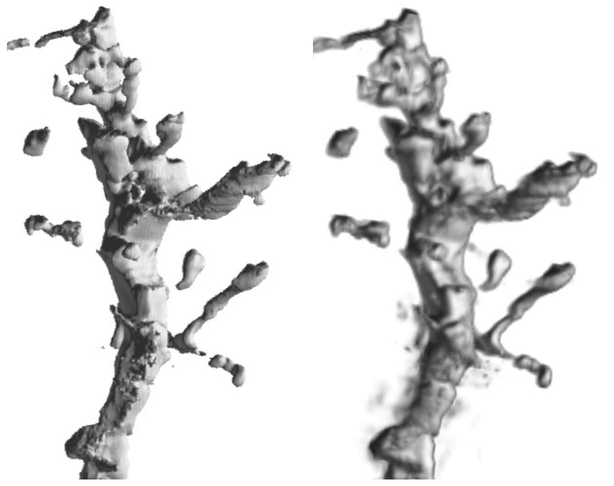

(a) Isosurface Rendering (b) Direct Volume Rendering

Figure 1: Two renderings of a spiny dendrite from a cortical pyramidal neuron. The volume dataset was reconstructed from images of a 2 micron thick section acquired with an intermediate high voltage electron microscope at the National Center for Microscopy and Imaging Research, San Diego, California, using single-tilt axis tomography. Specimen kindly provided by Prof. K. Hama of the National Institute for Physiological Sciences, Okazaki, Japan.

2 Related Work

Two methods have been proposed for assisting the user in the exploration of possible transfer functions. He et al. [7] use genetic algorithms to breed a good transfer function for a given dataset. Judging from small thumbnail renderings, the user picks desirable transfer functions from an automatically generated population, until the iterative process of image selection and transfer function intercombination converges. Alternatively, the system can run automatically by using some user-specified objective function (entropy, energy, or variance) to evaluate rendered images. Marks et al. [13] address the problem of "parameter tweaking" in general, with applications including light placement for rendering, motion control for articulated figure animation, as well as transfer functions in direct volume rendering. The goal is to create a visual interface to the complex parameter space by using an image difference metric to arrange renderings from a wide variety of transfer functions into a "design gallery", from which the user selects the most appealing rendering. While both of these methods reportedly succeed in finding useful transfer functions, and while they both allow the user to inspect the transfer function behind a rendering, the systems are fundamentally designed for finding good renderings, not for finding good transfer functions. Both processes are entirely driven by analysis of rendered images, and not of the dataset itself. Rather than having an high-level interface to *control* the transfer function, the user has to *choose* a transfer function from among those randomly generated, making it hard to gain insight into what makes a transfer function appropriate for a given dataset.

Other visualization tools have been described which are more driven by the data itself. Bergman et al. [3] describe a perceptually informed rule-based method for colormap selection which takes into account the data's spatial frequency characteristics and the purpose of the visualization. Closer to the goal of the current paper is the contour spectrum, described by Bajaj et al. [1], which helps the user find isovalues for effective isosurface volume visualizations of unstructured triangular meshes. By exploiting the mathematical properties of the mesh, important measures of an isosurface such as surface area and mean gradient magnitude can be computed with great efficiency, and the results of these measurements are integrated into the same interface which is used to set the isovalue. By providing a compact visual representation of the metrics evaluated over the range of possible isovalues, the user can readily decide, based on their rendering goals, which isolevel to use. The importance to the current paper is that the contour spectrum is a good example of how an interface can use measured properties of the data to guide the user through the parameter space controlling the rendering.

3 Ideal Boundary Characterization

3.1 Boundary Model

Since our particular goal is the visualization of material boundaries, we have chosen a model for what constitutes an ideal boundary and developed methods around that. We assume that at their boundary, objects have a sharp, discontinuous change in the physical property measured by the values in the dataset, but that the measurement process is band-limited with a Gaussian frequency response, causing measured boundaries to be blurred by a Gaussian. Fig. 2 shows a step function representing an ideal boundary prior to measurement, the Gaussian which performs the band-limiting by blurring, and the resulting measured boundary (prior to sampling). The resulting curve happens to be the integral of a Gaussian, which is called the *error function* [9]. Actual measurement devices band-limit, so they always blur boundaries somewhat, though their frequency response is never exactly a Gaussian, since this has infinite support. Although certain mathematical properties of the Gaussian are exploited later, we have not found the inexact match of real-world sampling to the Gaussian ideal to limit application of our technique. A final assumption made for the purposes of this analysis is that the blurring is isotropic, that is, uniform in all directions. Again, our methods will often work even if a given dataset doesn't have this characteristic, but results may be improved if it is pre-processed to approximate isotropic blurring.

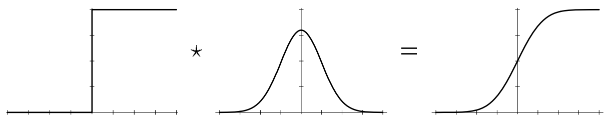

(a) Step function (b) Gaussian (c) Measured boundary prior to sampling

Figure 2: Boundaries are step functions blurred by a Gaussian.

3.2 Directional Derivatives along the Gradient

Although it was suggested in Section 1.2 that isosurfaces are not always sufficient for visualizing objects in real world volume data, the method presented in this paper still indirectly employs them as an indicator of object shape. That is, based on the mathematical property that the gradient vector at some position always points perpendicular to an isosurface through that position, we use the gradient vector as a way of finding the direction which passes perpendicularly through the object boundary. Even though isosurfaces don't always conform to the local shape of the underlying object, if we average over the whole volume, the gradient vector does tend to point perpendicular to the object boundary. We rely on the statistical properties of the histogram to provide the overall picture of the boundary characteristics.

The directional derivative of a scalar field f along a vector $\mathbf{v}$, denoted $\mathbf{D_v} f$, is the derivative of f as one moves along a straight path in the $\mathbf{v}$ direction. This paper studies f and its derivatives as one cuts directly through the object boundary — moving along the gradient direction — in order to create an opacity function. Because the direction along which we're computing the directional derivative is always that of the gradient, we employ a mild abuse of notation, using f' and f'' to signify the first and second directional derivative along the gradient direction, even though these would be more properly denoted by $\mathbf{D}_{\widehat{\nabla f}} f$ and $\mathbf{D}^2_{\widehat{\nabla f}} f$, where $\widehat{\nabla f}$ is the gradient direction. We treat f as if it were a function of just one variable, keeping in mind that the axis along which we analyze f always follows $\widehat{\nabla f}$, which constantly changes orientation depending on position. Fig. 3 shows how the gradient direction changes with position to stay normal to the isosurfaces of a simple object.

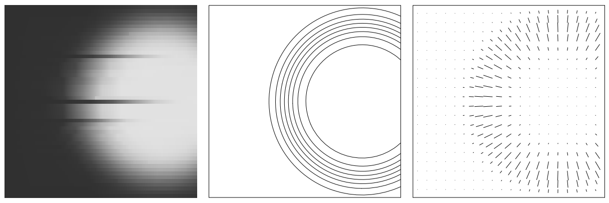

(a) $f(\mathbf{x})$ (b) Isolevels of f (c) ∇f

Figure 3: ∇f is always normal to f's isosurfaces.

Fig. 4 analyzes one segment of the cross-section of this same object. Shown are plots of the data value and the first and second derivatives as one moves across the boundary. Because of band-limiting, the measured boundary is spread over a range of positions, but an exact location for the boundary can be defined with either the maximum in f', or the zero-crossing in f''. Indeed, two edge detectors common in computer vision, Canny [4] and Marr-Hildreth [14], use the f' and f'' criteria, respectively, to find edges.

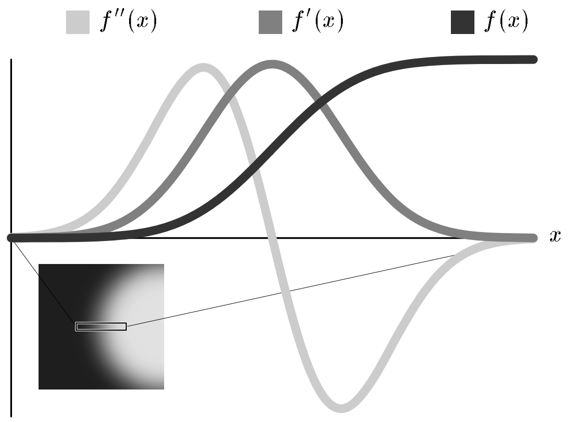


Figure 4: Measuring f, f', and f'' across boundary.

3.3 Relationship Between f, f', and f''

As our goal is to find functions of data value which highlight boundary regions, our problem is rather different than that addressed by edge detectors. Because the opacity function will be applied throughout the volume irrespective of position, we must locate the boundary not in the spatial domain, but in the range of data values. In contrast, edge detectors locate boundaries in the spatial domain, independent of data values. Yet, we still want to borrow from computer vision the notion that boundaries are somehow associated with a maximum in f' and/or a zero-crossing in f''. To see how this is possible, consider just the relationship between f and f'. As both of these are functions of position, they can be plotted with a three-dimensional graph, as in Fig. 5. The three-dimensional curve can be projected downward to form the plot of data value versus position, and projected to the right to show first derivative versus position. Projecting the curve along the position axis, however, eliminates the position information, and reveals the relationship between data value and first derivative. Because the data value increases monotonically, there is a (non-linear) one-to-one relationship between position and data value, so the first derivative f', which had been a function of *position* x, can also be expressed as a function of *data value* f. This is what the third projection in Fig. 5 depicts.

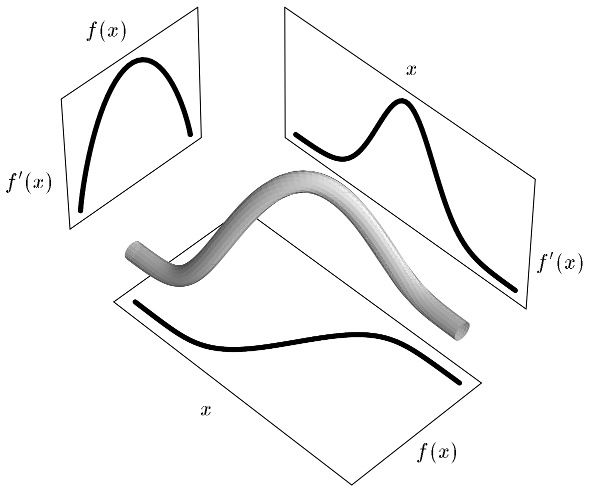


Figure 5: f, f' and position x.

The same projections can be done for data value and its second derivative, as seen in Fig. 6. Projecting the curve downward or to the right produces the graphs of data value or second derivative versus position (first seen in Fig. 4), while projecting along the position axis reveals the relationship between data value and its second derivative.

Finally, having "projected out" position information, one can make a three-dimensional graph of the first and second derivatives as functions of *data value*, as seen in Fig. 7. The significance of this curve is that it provides a basis for automatically generating opacity functions. If a three dimensional record of the relationship between f, f' and f'' for a given dataset contains curves of the type shown in Fig. 7, we can assume that they are manifestations of boundaries in the volume. With a tool to detect those curves and their position, one could generate an opacity function which makes the data values corresponding to the middle of the boundary (indicated with cross-hairs in Fig. 7) the most opaque, and the resulting rendering should show the detected boundaries. Short of that, one could use a measure which responds to some specific feature of the curve (say, the zero crossing in f'') and base an opacity function on that. This is what the current paper seeks to do.

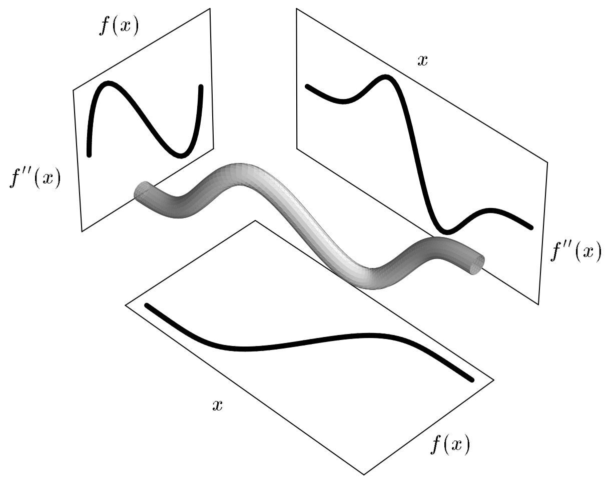


Figure 6: f, f'' and position x.

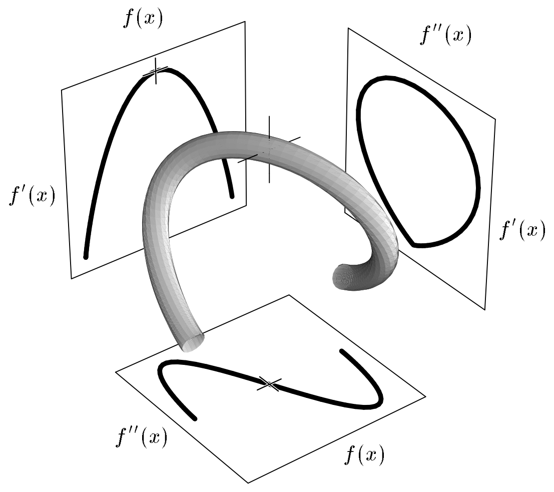


Figure 7: The underlying relationship of f, f', and f''.

ing *where* the boundary was in the idealized dataset. For instance, Fig. 4 was produced with knowledge of where to place a path so as to cross through the boundary. In the case of real volume data, however, the positions of the boundaries are not known, but the same relationship between f, f' and f'' needs to be revealed by some measurement technique.

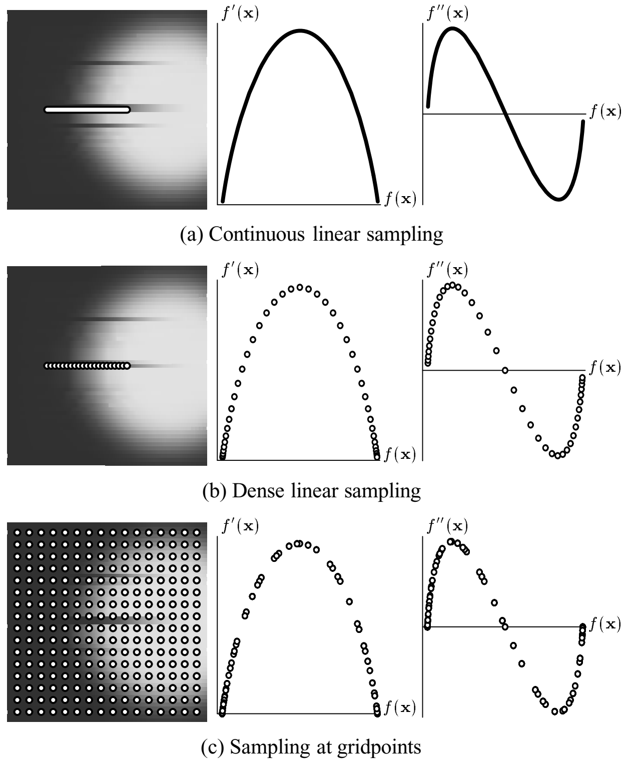


Figure 8: Sampling the boundary: from continuous to discrete.

4 The Histogram Volume

4.1 Histogram Volume Structure

To measure the relationship between the data value and its derivatives described in the last section, we use a three-dimensional histogram we term a *histogram volume*. There is one axis for each of the three quantities f, f', and f'', and each axis is divided into some number of (one-dimensional) bins, causing the interior volume to be divided into a three-dimensional array of bins. The histogram volume has two defining characteristics:

1. Each bin in the histogram volume represents the combination of a small range of values in each of the three variables f, f', and f''.

2. The value stored in each bin signifies the number of voxels in the original volume within that same combination of ranges of these three variables.

4.2 Histogram Volume Creation

Fig. 7 illustrated the position-independent relationship between f, f', and f'' that characterized an ideal boundary. To find that relationship, however, we afforded ourselves the luxury of first knowing *where* the boundary was in the idealized dataset.

It is sufficient to measure f, f', and f'' at each point of a uniform lattice. Fig. 8(a) shows a boundary being sampled continuously to produce smooth graphs of f' and f'' versus f. In Fig. 8(b), the sampling is along the same path, but is now discrete. The smooth graphs have been replaced by scatterplots, but the sequence of measurements traces out the same curves as before, indicating that discrete sampling and the resulting scatterplots are sufficient to illuminate the important relationships between f and its derivatives. Finally, in Fig. 8(c), the boundary is sampled everywhere on a uniform grid. Though now the points are distributed differently — many more hits have accumulated along where f' and f'' are near zero — the scatterplots trace out the save curves as before. By sampling everywhere, we no longer require knowledge of boundary location, and the "global" derivative characteristics of the boundary have been measured. This is precisely the sort of information relevant to opacity function generation.

The approach taken in this paper is to measure f and its directional derivatives exactly once per voxel, at the original sample points of the dataset. One might be concerned that sampling merely at the original data points is not a sufficient sampling density to produce the curves seen in Figs. 7 and 8. However, with real volume data this will not be a problem, since the band-limiting in data acquisition assures there will always be some blurring, and since the boundaries of real objects tend to assume a variety of positions and orientations relative to the sampling grid.

4.3 Implementation

One implementation issue in the histogram volume creation is how many bins to use. There is a trade-off between storage and processing requirements versus having sufficient resolution in the histogram volume to discern the patterns from which we generate the opacity functions. In our experiments, good results were obtained with histogram volumes of sizes between 80^3 and 256^3 bins, though there is no reason that the histogram volumes need to have equal resolution on each axis. Also, we have found it sufficient to use only 8 bits to represent the values in the histogram volume, scaling and/or clipping the number of hits to the range 0–255 if necessary.

A more subtle issue is what range of values to include along each axis. Obviously, for the data value axis, the full range should be included, since we intend to capture all the values at which boundaries might occur. But along the axes for first and second derivative, it makes sense to include something less than the full range. Since derivative measures are by nature sensitive to noise, including the full range of derivative values in the histogram volume may cause the important and meaningful sub-range of values to be compressed to a smaller number of bins, thereby hampering the later step of detecting patterns in the histogram volume. We do not have an *a priori* knowledge of the meaningful ranges of derivatives values, so currently the derivative value ranges are set with an educated guess. This is a matter in need of further research and automation.

The most significant implementation issue is the method of measuring the first and second directional derivatives. The first derivative is actually just the gradient magnitude. From vector calculus [15] we have:

$$\mathbf{D}_{\mathbf{v}} f = \nabla f \cdot \mathbf{v}, \tag{1}$$

thus

$$\mathbf{D}_{\widehat{\nabla f}} f = \nabla f \cdot \widehat{\nabla f} = \nabla f \cdot \frac{\nabla f}{\|\nabla f\|} = \|\nabla f\|. \tag{2}$$

Unfortunately there is no similarly compact formula for $\mathbf{D}^2_{\widehat{\nabla f}} f$, the second directional derivative along the gradient direction. Twice applying Eqn. 2 gives:

$$\begin{aligned} \mathbf{D}^2_{\widehat{\nabla f}} f &= \mathbf{D}_{\widehat{\nabla f}}(\|\nabla f\|) = \nabla(\|\nabla f\|) \cdot \widehat{\nabla f} \\ &= \frac{1}{\|\nabla f\|} \nabla(\|\nabla f\|) \cdot \nabla f \end{aligned} \tag{3}$$

Or, using the Taylor expansion of f along $\widehat{\nabla f}$ [5] gives:

$$\mathbf{D}^2_{\widehat{\nabla f}} f = \frac{1}{\|\nabla f\|^2} (\nabla f)^{\mathrm{T}}\, \mathbf{H} f\, \nabla f \tag{4}$$

where $\mathbf{H}f$ is the Hessian of f, a 3×3 matrix of second partial derivatives of f [15]. Alternatively, we can use the Laplacian $\nabla^2 f$ to approximate $\mathbf{D}^2_{\widehat{\nabla f}} f$:

$$\mathbf{D}^2_{\widehat{\nabla f}} f \approx \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{5}$$

The approximation is exact only where the isosurfaces have zero mean surface curvature [5].

These three expressions for $\mathbf{D}^2_{\widehat{\nabla f}} f$ each suggest different implementations for the second derivative measure. Although this would benefit from more detailed study, we can still make useful observations regarding the comparative merits of each. While we've found the Hessian method to be the most numerically accurate, the others have proven sufficiently accurate in practice to make them appealing by virtue of their computation efficiency. The Laplacian computation is direct and inexpensive, but the most sensitive to quantization noise. The gradient of the gradient magnitude (Eqn. 3) is better, and its computational expense is lessened if the gradient magnitude has already been computed everywhere for the sake of volume rendering (e.g., as part of shading calculations).

By measuring the derivatives only at the original data points, the calculation of the first and second partial derivatives required in the above expressions is greatly facilitated by the use of discrete convolution masks applied at the data points; we have used standard central differences. Thus, our task is somewhat distinct from the usual problem of derivative measurement in volume rendering, where a primary concern is continuity of the derivative between sample points to allow for correct shading of interpolated data values [2, 16].

The general algorithm for creating the histogram is straightforward:

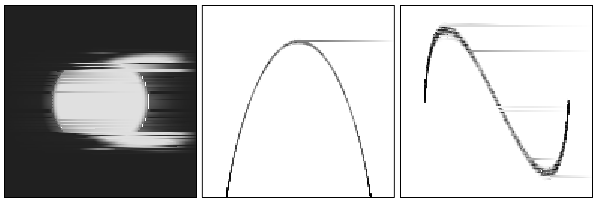
1. Initialize the histogram volume to all 0's.
2. Make one pass through the volume looking for the highest values of f' and f'', and the lowest value of f''; assume 0 for the lowest value of f'. Set ranges on the histogram volume axes accordingly.
3. On a second pass through the volume,
   - 3a. Measure f, f', and f'' at each voxel,
   - 3b. Determine which bin in the histogram volume corresponds to the measured combination of f, f', and f'', and
   - 3c. Increment the bin's value.

4.4 Histogram Volume Inspection

It is possible to gain some insight into the object boundaries of the original dataset by simple visualization of the histogram volume after it has been calculated. One may be tempted to simply volume render the histogram volume from arbitrary views, but this usually turns out to be unrevealing due to the speckled and noisy nature of the histogram volume. A better way is to use summed-voxel of projections the histogram volume, projecting along either the f' or the f'' axis, to produce scatterplots of f'' versus f or f' versus f, respectively. This allows testing of the premise in Section 3.3 — if there are boundaries in the original dataset that conform to the boundary model, there should be curves like that of Fig. 7 in the histogram volume.

Fig. 9 shows cross-sections and scatterplots for two synthetic datasets. The curves in the scatterplots are exactly the form seen in Fig. 8, and one can see an important property of the histogram volume — for each pair of materials that share a boundary, there is a curve in the histogram volume. This property is again visible in Fig. 10, wherein various computed tomography datasets are being analyzed. Though the scatterplots are noisier, it clear that the histogram volume is successfully capturing information about the materials and their boundaries.

It should be noted that a related technique has been used in computer vision for feature identification. Panda and Rosenfeld [18] use two-dimensional scatterplots of data value and gradient magnitude to perform image thresholding for night vision applications. They, however, do not assume a boundary model, instead limiting their analysis of the scatter plot to identifying particular distributions within regions of low and/or high gradient magnitude.


(a) Cylinder

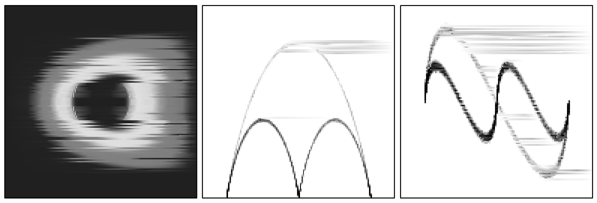

(b) Nested Cylinders

Figure 9: Dataset Slice, f' versus f, and f'' versus f.

5 Opacity Function Generation

5.1 Mathematical Boundary Analysis

In order to develop a method for opacity function generation that uses our boundary model and the information stored in the histogram volume, it is helpful to look at the equation we have used to describe the ideal boundary data value as a function of position:

$$v = f(x) = v_{min} + (v_{max} - v_{min})\frac{1 + \mathrm{erf}(\frac{x}{\sigma\sqrt{2}})}{2} \tag{6}$$

Just as we have taken "position" to always be along an axis pointing in the gradient direction, we define 0 to always be the position of the inflection point in the boundary. v_{min} and v_{max} are the data values of the materials on either side of the boundary. As $\mathrm{erf}()$ ranges from -1 to 1, v ranges from v_{min} to v_{max}. The parameter controlling the amount of boundary blurring is σ. The first and second derivatives of f are as follows:

$$f'(x) = \frac{v_{max} - v_{min}}{\sigma\sqrt{2\pi}}\exp(-\frac{x^2}{2\sigma^2}) \tag{7}$$

$$f''(x) = -\frac{x(v_{max} - v_{min})}{\sigma^3\sqrt{2\pi}}\exp(-\frac{x^2}{2\sigma^2}) \tag{8}$$

Our choice of boundary parameterization means that $f'(x)$ is a Gaussian, with σ being the usual standard deviation. Since the Gaussian has inflection points at $\pm\sigma$, this is where $f''(x)$ attains its extrema. The same positions can serve as artificial delimiters for the extent of the boundary — we *define* the "thickness" of the boundary to be 2σ. Note that the thickness of a boundary can be recovered if the maximum values of f' and f'' are known:

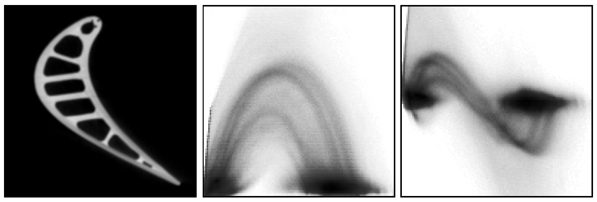
$$\frac{f'(0)}{f''(-\sigma)} = \sigma\sqrt{e} \tag{9}$$

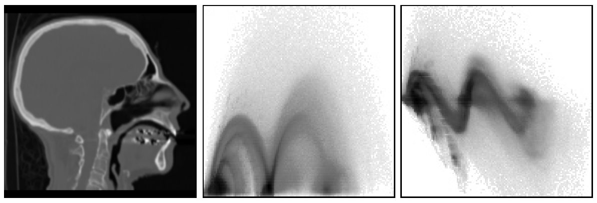
More importantly, once σ is known, we can recover the position x knowing only the values of f' and f'':

$$\frac{f''(x)}{f'(x)} = -\frac{x}{\sigma^2} \tag{10}$$

5.2 Opacity functions of data value

Before using Eqn. 10 as the basis for opacity function generation, we define some important functions of data value: $g(v)$ is the average first directional derivative of f over all the positions $\mathbf{x}$ at which $f(\mathbf{x}) = v$, and $h(v)$ is likewise the average second directional derivative at value v. These two functions can be obtained from the histogram volume by slicing it at data value v, and finding the centroid of the scatterplot of f' and f'' at that value. The f' axis coordinate of the centroid is $g(v)$, and the f'' axis coordinate is $h(v)$.


(a) Turbine Blade


(b) Head

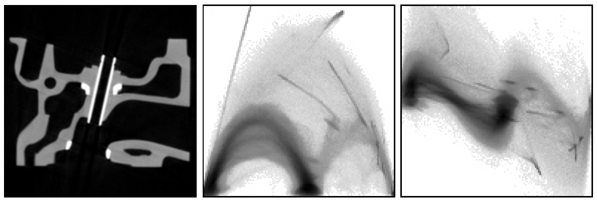

(c) Engine Block

Figure 10: Dataset Slice, f' versus f, and f'' versus f.

Knowing $g(v)$ and $h(v)$ for all v, one can find the ratio of their maxima to recover σ with Eqn. 9, assuming that g attains its maxima at $f(0)$, and that h attains its maxima at $f(-\sigma)$. With this information, we define a mapping $p(v)$ from data value to an approximate position along a boundary:

$$p(v) = \frac{-\sigma^2 h(v)}{g(v)} \approx \frac{-\sigma^2 f''(f^{-1}(v))}{f'(f^{-1}(v))} = x \tag{11}$$

Roughly speaking, $p(v)$ tells us on which side of the nearest boundary a data value v tends to fall. For values closer to v_{min}, $p(v)$ will be negative; for values closer to v_{min}, $p(v)$ will be positive. In practice, we have found it useful to modify Eqn. 11 to account for the fact that the gradient magnitude at the interior of materials is rarely exactly zero. Knowing how it differs from zero is again a matter of experience, but assuming one can find a g_{thresh} which is higher than the ambient gradient magnitude, Eqn. 11 is re-formulated, with a slight loss of mathematical accuracy, as

$$p(v) = \frac{-\sigma^2 h(v)}{\max(g(v) - g_{thresh}, 0)} \tag{12}$$

The user supplies the last piece of information needed: a function $b(x)$ we term the *boundary emphasis function*, which maps from position along a boundary to opacity. As the intent is to make only boundaries visible in the rendering, $b(x)$ should be non-zero only near 0. For this reason, we have not been especially careful to prevent $p(v)$ from attaining infinite values due to a low $g(v)$; such a data value v should not contribute to the final image. With $b(x)$, the user can directly control whether rendered boundaries will appear thick or thin, sharp or fuzzy, and the proximity of the rendered boundary to the object interior. The final opacity function $\alpha(v)$ is then defined as

$$\alpha(v) = b(p(v)) \tag{13}$$

Instead of exploring the parameter space of all possible opacity functions, the user explores the parameter space of $b(x)$ and lets the information from the histogram volume, embodied in $p(v)$, constrain the search to those opacity functions which display object boundaries. Defining opacity as a function of position within a boundary then becomes a more intuitive task than defining opacity as a function of data value, as there is a more predictable relationship between changes made to the boundary emphasis function and the corresponding change in rendered results. As there is a potential for inaccuracy in the calculation of σ from Eqn. 9, the user may need to experiment with different scalings in the domain of $b(x)$. Fig. 11 shows how the choice of boundary emphasis function affects the opacity function and the rendered image, for a synthetically created dataset containing two concentric spheres at distinct data values. It should be stressed that the user does not set the location of the peaks in $\alpha(v)$, since this is determined by the information in $p(v)$, but the user can influence the location, as well as the width, height, and shape of the peaks. This is the main benefit of this method: if the histogram volume has successfully captured information about the boundaries in the dataset, the user enjoys high-level control over the character of the rendered boundaries without having to worry about the exact specification of $\alpha(v)$. Yet, the $\alpha(v)$ so generated is sensible enough that it could be edited by hand if desired. For example, since this technique will attempt to make *all* boundaries opaque, a useful supplement to the interface would be a feature which allows supression of the peaks in $\alpha(v)$ for one or more boundaries.

Even though we have made some strong assumptions about the boundary characteristics in the volume dataset, the technique described here typically works well even if the material boundaries are not "ideal". Essentially, by taking the ratio of the second and first derivatives, and by having $b(x)$ assign opacity to positions around zero, we are more apt to make opaque those data values associated with both low second derivatives and high first derivatives. Or, even if $p(v)$ is not a perfect indicator of "position relative to boundary", the sign change in f'' around its zero-crossing affords us some control over whether we want to emphasize regions closer to or further from the object's interior. Fig. 12 on the accompanying colorplate shows a rendering of an MRI dataset which does not have ideal boundaries but for which this technique still works.

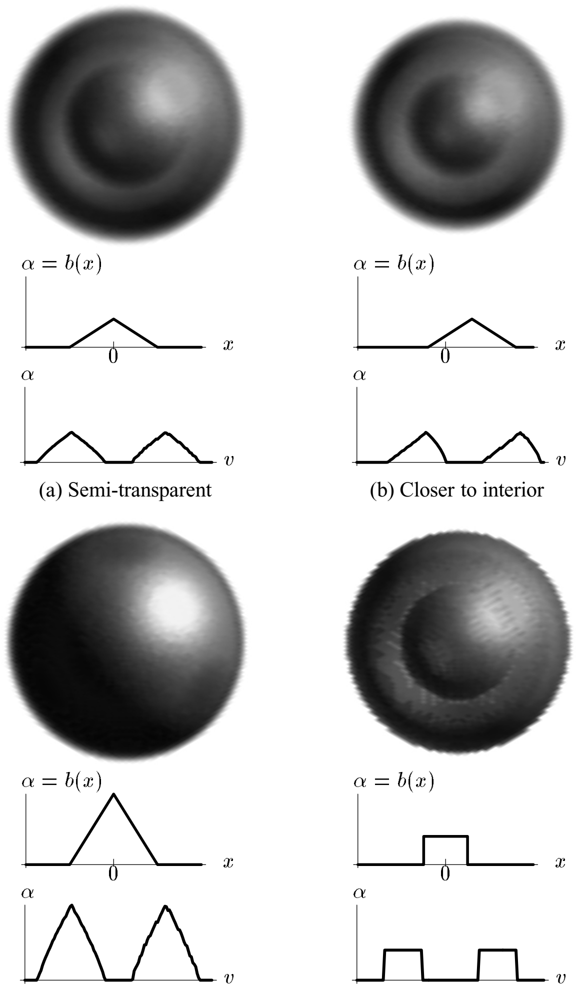


Figure 11: Relationship between $b(x)$, $\alpha(v)$, and the rendered result.

5.3 Opacity functions of data value and gradient magnitude

So far the opacity functions under consideration have assigned opacity based on data value alone. Higher quality renderings can sometimes be obtained, however, if the opacity is assigned as a function of both data value and gradient magnitude. Defining these two-dimensional opacity functions by hand is especially challenging because there are even more degrees of freedom than in one-dimensional, value-based opacity functions. Fortunately, the ideas presented so far easily generalize to allow semi-automatic generation of two-dimensional opacity functions.

Analogous to the definition of $h(v)$, we define $h(v,g)$ to be average second derivative over all locations where the data value is v and the gradient magnitude is g. We similarly define a new position function, and from that an opacity function:

$$p(v,g) = \frac{-\sigma^2 h(v,g)}{\max(g - g_{thresh}, 0)} \tag{14}$$

$$\alpha(v,g) = b(p(v,g)) \tag{15}$$

σ is calculated as before; the ratio of the extremum of the average first and second derivatives. The benefit of this kind of opacity function is that it can distinguish between boundaries that have overlapping ranges of values. For instance, the nested cylinders volume in Fig. 9 and the engine block volume in Fig. 10 each have one boundary which overlaps the two other boundaries in data value, spanning from the higher of the two material values to the background value. Selectively rendering this single boundary is impossible with a value-based opacity function, but because the boundary has a distinct curve in the plot of data value versus first derivative, it is possible to create an opacity function which selects only the voxels comprising this boundary. As it did in the case of value-based opacity functions, the technique presented here will generate two-dimensional opacity functions which make all detected boundaries opaque; a simple "lasso" tool could then be used to select different regions in the two-dimensional opacity function to render one boundary at a time. In Fig. 13 on the colorplate, the feet of the female Visible Human CT dataset [17] are rendered with four different two-dimensional opacity functions. Using a modification of an automatically generated opacity function, one rendering shows almost exclusively the registration cord laced around the body prior to scanning.

The lower right rendering in Fig. 13 demonstrates another advantage of two-dimensional opacity functions — the ability to accurately render the surface of a material which attains a wide range of data values, as is the case for the bone tissue in this same CT scan. Different parts of the bone surface are more radio-opaque than others, leading to a wide range of data values associated with bone, which in turn causes a wide range of gradient magnitudes within the boundary region between bone and soft tissue. Knowing the average second derivative for each location in (v, g) space, we can make opaque only those voxels near the middle of the boundary (near the zero-crossing in f''), regardless of the bone data value. As is visible in the opacity function generated with Eqn. 14, this implies that as gradient magnitude increases, there is an upward shift in the data values which should be made most opaque. This kind of careful opacity assignment is not possible with a simple value-based opacity function, though it is reminiscent of the two-dimensional opacity functions described by Levoy [11]. Although space does not permit a detailed comparison between our approach and Levoy's, the main difference is that (ideally) the measured first and second derivative information serves to constrain the opacity function generation so as to only show boundaries, while in Levoy's method the user still has to experiment to find the right parameter settings.

6 Conclusions and Future Work

We have shown that semi-automatic generation of opacity functions is possible for datasets where the regions of interest are boundaries between materials of relatively constant data value. The histogram volume structure presented here captures information about the boundaries present in the volume and facilitates a high-level interface to opacity function creation. The user controls which portions of the boundary are to be made opaque, without having to know the data values that occur in the boundary.

Given that boundaries in the volume are always manifested by a curve of a particular shape in the histogram volume, it makes sense to apply computer vision object recognition techniques to the histogram volume. We are investigating the feasibility of using the Hough transform to detect the curves in the histogram volume and measure their intensity [8]. Also, it may be possible to adapt the methods to non-scalar data, such as comes from multi-echo MRI. Finally, as mentioned before, we are interested in performing perceptual studies to validate the claim that direct volume rendering can, unlike isosurface rendering, accurately convey surface quality or measurement uncertainty to the viewer.

7 Acknowledgements

This work was supported in part by grants from the National Science Foundation (ASC 93-18180) to Mark Ellisman, Donald Greenberg, and Sid Karin, from the National Institutes of Health National Center for Research Resources (RR 04050) to Mark Ellisman, and by the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC 89-20219). We gratefully acknowledge equipment grants from Hewlett-Packard, on whose workstations this research was conducted. Turbine blade dataset courtesy of GE Corporate Research and Development.

References

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The Contour Spectrum. In *Proceedings Visualization '97*, pages 167–173, 1997.

[2] Mark J. Bentum, Barthold B.A. Lichtenbelt, and Tom Malzbender. Frequency Analysis of Gradient Estimators in Volume Rendering. *IEEE Trans. Visualization and Computer Graphics*, 2(3):242–254, 1996.

[3] Lawrence D. Bergman, Bernice E. Rogowitz, and Lloyd A. Treinish. A Rule-based Tool for Assisting Colormap Selection. In *Proceedings Visualization '95*, pages 118–125, 1995.

[4] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[5] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*, chapter 4: Edge Detection. MIT Press, Cambridge, MA, 1993.

[6] Joachim Frank. *Electron Tomography*, pages 205–213. Plenum, New York, 1992.

[7] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proceedings Visualization '96*, pages 227–234, 1996.

[8] J. Illingworts and J. Kittler. A Survey of the Hough transform. *IEEE Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[9] Granio A. Korn and Theresa M. Korn. *Mathematical Handbook for Scientists and Engineers*, page 820. McGraw-Hill, New York, 1968.

[10] Philip Lacroute and Marc Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform. In *ACM Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 451–458, July 1994.

[11] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, 8(5):29–37, 1988.

[12] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi. *Introduction to Volume Rendering*, chapter 4. Prentice-Hall, New Jersey, 1998.

[13] J. Marks, B. Andalman, P.A. Beardsley, and H. Pfister et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *ACM Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 389–400, August 1997.

[14] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B 207:187–217, 1980.

[15] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*, chapter 2.6, 4.2. W.H. Freeman and Company, New York, 1996.

[16] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. A Comparison of Normal Estimation Schemes. In *Proceedings Visualization '97*, pages 19–25, 1997.

[17] National Library of Medicine (U.S.) Board of Regents. Electronic imaging: Report of the Board of Regents. U.S. Department of Health and Human Services, Public Health Service, National Institutes of Health. NIH Publication 90-2197, 1990.

[18] Durga P. Panda and Azriel Rosenfeld. Image Segmentation by Pixel Classification in (Grey Level Edge Value) Space. *IEEE Trans. on Computers*, 27(9):875–879, September 1978.

[19] S.J. Young, G.Y. Fan, D. Hessler, S. Lamont, T.T. Elvins, M. Hadida, G. Hanyzewski, J.W. Durkin, P. Hubbard, G. Kindlmann, E. Wong, D. Greenberg, S. Karin, and M.H. Ellisman. Implementing a Collaboratory for Microscopic Digital Anatomy. *Supercomputer Applications and High Performance Computing*, 10(2/3):170–181, 1996.