Multi-Dimensional Transfer Function Design

Sangmin Park*
University of Texas at Austin

Chandrajit Bajaj†
University of Texas at Austin

Abstract

Direct volume rendering can be accomplished through correct transfer functions that convert data values to visual properties such as transparency and color. Multi-dimensional transfer functions that have additional axes of gradient magnitude and the second derivative as well as intensity, are useful for the visualization of the features that intensity ranges are overlaped, but gradient magnitudes are distinguishable. However, multi-dimensional transfer functions are hard to control by hand, since correct combination of data values of each axis should be searched to find a correct voxel role in terms of boundary by try and error in multi-dimesional space. In other words, some voxels exactly on a boundary should be totally opaque and others supporting the boundary should have proper transparencies based on the distance from the boundary. In this paper, we present how to decide the voxel role that is fixed once a volume dataset is generated and suggest a multi-dimensional transfer function that minizes user interactions through the pre-defined voxel roles.

Keywords: Volume Rendering, Transfer Function

1 Introduction

One of the main advantages of direct volume rendering is to visualize volume data without generating geometric structures. The direct volume rendering simply converts each volume element (or voxel) to visual properties such as opacity and color and composes the properties into an image. The converting work is usually done by transfer functions. Good transfer functions produce good images that visuzlize interesting structures in volume data while removing uninteresting area. However, it is known that finding good transfer function is one of the hardest problems in the direct volume rendering area.

The multi-dimensional transfer functions of direct volume rendering are precisely the mathematical functions of several parameters such as intensity, gradient magnitude and the second derivative. Since transfer functions that have more than one domain are hard to control by hand, many people still use intensity-based one-dimensional (1D) transfer functions. The survery of [Kindlmann 2002] reviews many types of transfer functions and describes the reasons why transfer function generation is not a trivial work. It also explains that transfer functions can be generalized by increasing the function's domain. [Kniss et al. 2001] shows the advantage of multi-dimensional transfer functions against 1D transfer functions. Even though multi-dimensional transfer functions are hard to control, the functions are still useful for separately visulizing several features that intensity ranges are overlapped, but other data such as gradient magnitude are distinguishalbe.

When multi-dimensional transfer functions are designed, intensity has been worked as the first axis perfectly. The second axis of transfer functions has been gradient magnitude in many researches such as [Levoy 1988], [Kindlmann and Durkin 1998], [Kniss et al. 2001], [Kniss et al. 2002], and [Kniss et al. 2003] without any doubt for more than a decade, since it has been believed that some region of volume data with high gradient magnitude is likly to have a boundary (or a feature) that should be visualized. As the third axis, the directional second derivative along gradient direction is used in [Kniss et al. 2001], [Kniss et al. 2002], and [Kniss et al. 2003].

When we consider only two-dimensional (2D) transfer functions with intensity and gradient magnitude, the two parameters do not represent boundary information directly. If we add one more axis of the directional second derivative to the 2D transfer function, then it is more promising to visualize boundaries, but a boundary and surrounded voxel information of the boundary still hide behide the three axes. [Kindlmann and Durkin 1998] suggests a semi-automatic method to find boundaries in the 2D space of intensity and gradient magnitude through histogram volume inspection. [Kniss et al. 2001] developed a convenient interface with manipulation widgets to search boundaries in 3D space of intensity, gradient magnitude and the directional second derivative. However, the methods suggest a way to search boundaries using gradient magnitude and the second derivative that have no direct boundary information.

Once a scalar volume dataset such as a medical volume dataset is generated, the role of each voxel of the volume dataset is fixed in terms of boundary. In other words, some of the voxels are exactly on a boundary, some are a part of the boundary with some thickness and others are not related to any boundary. If we know the role of each voxel, then transfer function generation will be more intuitive and easier. In this paper, we present a method to decide voxel roles in terms of boundary and sugget another data values for the second axis and the third axis for multi-dimensional transfer functions.

The remainder of the paper is organized as follows. In the next section, we review related work. In section 3, we present a method to decide a voxel role in volume data in terms of boundary. Based on the voxel role, we suggest new opacity functions in section 4. In section 5, The suggested opaicy functions are implemented using modern graphics hardware and the rendering results are explained. Finally, we make conclusions and suggest a couple of future work in section 6.

*e-mail: smpark@cs.utexas.edu
†e-mail: bajaj@cs.utexas.edu

2 Related Work

[Levoy 1988] suggested two-dimensional transfer functions of intensity and gradient magnitude and suggested a way to visualize multiple semi-transparent surfaces. After the research, multi-dimesional transfer functions have been the function of intensity and gradient magnitude. [Kindlmann and Durkin 1998] suggested semi-automatic generation of transfer functions. They still used two-dimensional transfer functions of intensity values and gradient magnitudes. The semi-automatic generation algorithm uses second derivatives to automatically compute boundary thickness at some intensity value and uses a linear function to assign alpha values to the thickness. [Kniss et al. 2001] suggested three-dimensional transfer functions and interactive interface widgets. The survey of [Kindlmann 2002] explains many kinds of transfer functions, but all multi-dimensional transfer functions are the functions of intensity, gradient, and the second derivative.

Distance map have been used for volume rendering in [Gibson 1998a] and [Gibson 1998b]. For the purpose of binary segmented volume data visualization, the distance map method was developed. This reasearch gave us a hint on the multi-dimensional trasnfer function design, since the distance map has an interesting character like that: the zero-value iso-surface of the distance map yields the object surface.

3 Boundary Detection and Voxel Roles

Gradient magnitude has been used for the second axis of transfer functions without any doubt, since it contains the rate of chage of values and allows the distinction between homogeneous regions and transition regions [Kindlmann 2002]. Even though most boundaries are in the transition regions that have high gradient magnitude, gradient magnitude itself does not represent boundaries clearly. In other words, some region that have relatively low gradient magnitude can be a boundary, while high gradient region cannot be a boundary. The second derivative as one of the axes of transfer functions is promising to find boundaries, since the zero-crossing locations of it can be boundaries. However, it still has false boundaries at the local minima of the gradient magnitude. We belive that the reasons make transfer function generation hard. In this section, we exploit boundary detection techniques and suggest another data values for the second axis of transfer functions.

3.1 Boundary Detection

Many boundary detection (or edge detection) algorithms have been developed in the image processing and pattern recognition areas. One of the traditional edge detection techniqes is Canny's method that finds the local maxima along the gradient direction [Canny 1983]. The most common edge detection schemes include three operations: differentiation, smoothing and edge labeling [Ziou and Tabbone 1998].

First, differentiation is the computation of the derivatives to identify edges. We compute gradient vectors represented by $\bigtriangledown f$ using the central difference operator and the magnitude of the gradient vector is $\parallel \bigtriangledown f \parallel$. The normalized gradient vector is computed as following:

$$\vec{n} = \frac{\bigtriangledown f}{\parallel \bigtriangledown f \parallel} \tag{1}$$

The frequently used second-order derivative operators are the Laplacian operator and the directional second-order derivative. In this paper, we compute and use the directional second-order derivative along gradient direction. The operator is definded by:

$$\frac{\partial^2 f}{\partial^2 \vec{n}} = \vec{n} \cdot \bigtriangledown \parallel \bigtriangledown f \parallel \tag{2}$$

Second, for the smoothing purpose, a bilateral filter that smoothes data values while preserving edges [Tomasi and Manduchi 1998], is applied to gradient magnitude and the directional second derivative. A bilateral filter consists of a domain filter, $W_d(\vec{x})$, and a range filter, $W_r(\vec{x})$, at a voxel, $\vec{x} = \begin{pmatrix} i & j & k \end{pmatrix}^T$:

$$W_d(\vec{x}) = \exp\left[-\tfrac{(\vec{x}-\vec{y})^T(\vec{x}-\vec{y})}{\sigma_d^2}\right], \quad \vec{y} \in N_{\vec{x}} \tag{3}$$

and

$$W_r(\vec{x}) = \exp\left[-\tfrac{(f(\vec{x})-f(\vec{y}))^2}{\sigma_s^2}\right], \quad \vec{y} \in N_{\vec{x}}, \tag{4}$$

where $N_{\vec{x}}$ is the set of neighbors of $\vec{x}$. The domain filter, $W_d(\vec{x})$, computes the weights of each voxel based on spatial distance at $\vec{x}$ and the range filter, $W_r(\vec{x})$, measures the "photometric" similarity between a function value at $\vec{x}$ and the neighborhood function values of center $\vec{x}$. The bilateral filter that combines the domain and range filters and generates a new function value, $\widetilde{f}(\vec{x})$, is described as follows:

$$\widetilde{f}(\vec{x}) = \frac{\sum_{\vec{y} \in N_{\vec{x}}} W_d(\vec{x}) W_r(\vec{x}) f(\vec{x})}{\sum_{\vec{y} \in N_{\vec{x}}} W_d(\vec{x}) W_r(\vec{x})} \tag{5}$$

Finally, edge labeling is to identify authentic edges while suppressing false edges produced by the reasons of noise and non-maximum high gradient magnitude values. Since we asssume that volume data have reasonably high signal-to-noise ratio and some noise that can be accumulated in the first and second derivative computations is reduced by a bilateral filter, we consider only removing the "phantom edges" (defined in [Clark 1989]) or non-maximum gradient magnitude among the zero-crossing locations of the directional second derivative. To detect the zero-crossing locations of the directional second derivative in 3D space, we compute the derivative values with Equation (2) at every voxel and we compose a cube that has the eight directional second derivative values at each vertex. If the eight values of each vertex does not have the same signs, then the cube contains the zero-crossing locations.

The survey, [Ziou and Tabbone 1998], shows several ways to remove the phantom edges, but the edges can be distinguised easily by climbing the gradient magnitude values along gradient direction, when we decide voxel roles. In the next sub-section, we present how to decide the roles of each voxel in terms of boundary and how to remove the phantom edges.

3.2 Distance as a Voxel Role

Since the direct volume rendering visualizes volume data without generating any geometric structure, we do not need to search the complete surfaces of the zero-crossing locations of the directional second derivatives. Instead of searching the surfaces, it is enough to find proper transparencies for all voxels. If we know the roles of each voxel in terms of boundary, then we can assign correct transparencies to the voxels easily based on the voxel roles. In other words, if a voxel is exactly on a boundary, then the voxel should be totally opaque, while a voxel that is a part of the boundary with some thickness should have proper transparency.

In this paper, we define voxel roles with distance to an authentic edge. The distance is Euclidean distance along gradient direction in 3D space. To compute a distance from a voxel to an authentic edge, two rays are shot to the both directions of the positive and negative gradient at every voxel location. The two values of gradient magnitude and the directional second derivative are interpolated with the tri-linear interpolation method at every sampling locations along both positive and negative gradient directions. If the gradient magnitude values decrease at the sampling locations of one of the gradient directions, then the ray that has decreasing gradient magnitude values is stopped to go further and only the other direction ray keeps going until it hit a zero-crossing locations of the directional second derivative. Since gradient direction is perpendicular to the edge orientation [Ziou and Tabbone 1998], if we follow the gradient directions of a voxel that is close to a boundary, we easily find a zero-crossing locations or a boundary in 3D space.

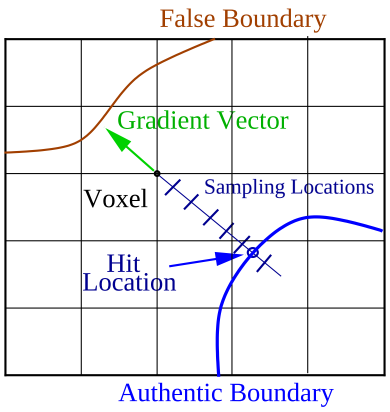


Figure 1: The Voxel Role or Distance Computation by Sampling from a Voxel to an Authentic Edge along the Gradient Direction of the Voxel

Figure 1 shows a 2D example on computing a distance from a voxel to an authentic edge along the negative gradient direction. Ideally, the two values of gradient magnitude and the directional second derivative are changed like Figure 2 along both positive and negative gradient directions.

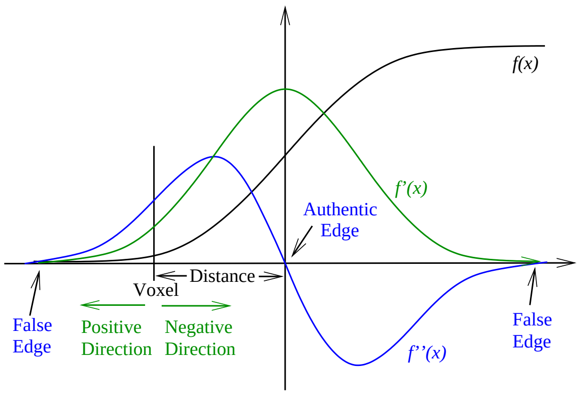


Figure 2: The Authentic and False Edges and The Relations of f, f', and f''

The sampling locations are represented by $\vec{x}_s(t) = \vec{x} + t\frac{\nabla f(\vec{x})}{\|\nabla f(\vec{x})\|}$, where $-d_{max} < t < d_{max}$. If the signs of the two directional second derivative values are changed at the two consecutive sampling locations of t_1 and t_2 as following, $f''(\vec{x}_s(t_1)) \times f''(\vec{x}_s(t_2)) < 0$, then we compute the exact zero-crossing location with the bisection method, [Buchanan and Turner 1992]. Experimentally, we decide the sampling interval and d_{max} such as $L_{min}/5$ and $L_{min} \times 15$ respectively, where $L_{min} = Min($ Width of a Voxel, Height of a Voxel, Depth of a Voxel $)$.

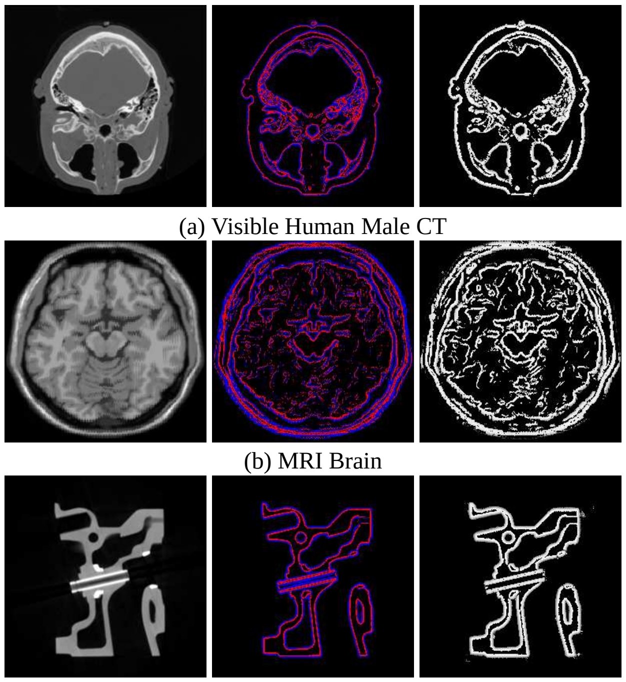


Figure 3: Volume Data Slice(left), the Directional Second Derivative(middle) and Distance(right): The red and blue colors of the middle images represent the negative and positive values of the directional second derivative respectively.

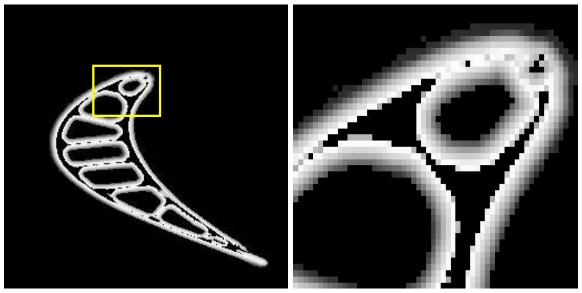


Figure 4: Turbin Blade Distance Image: The right-hand side image is the enlarged picture of the left image of the yellow box.

Figure 3 shows the results of the distance and the directional second derivative computation as well as each dataset slice. The positive and negative second derivative values are colored with blue and red respectively. Therefore, the zero-crossing locations of the second derivative are between the two colors. To denote the distance values through images, we linearly flip the distance values. For example, if the distance values at $d(\vec{x})$ ranges from 0 to d_{max}, then we simply compute the following formula for each pixel value of Figure 3 (right) and Figure 4.

$$D(\vec{x}) = \frac{d_{max} - d(\vec{x})}{d_{max}} \times 255 \qquad (6)$$

4 Multi-Dimensional Transfer Functions

In this section, we suggest the opacity functions of the distance from a voxel to an authentic edge and the 2D opacity functions of intensity and gradient magnitude at hit locations. The two kinds of opacity functions are multiplied to generate the final opacities of each voxel.

4.1 Opacity Functions of Intensity

Once we decide each voxel role in terms of boundary, the transparency of each voxel can be generated easily based on the role or the distance that is computed in the previous section. We suggest three different opacity functions, linear, concave nonlinear, and convex nonlinear functions as Figure 5. The linear opacity function is to map the flipped distance and the concave and convex nonlinear funcions use the *n*-th power of the distance as following:

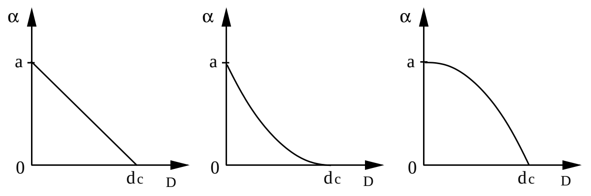
$$\alpha_d(d) = Max(-\frac{a \times d}{d_c} + a, \quad 0), \qquad (7)$$

$$\alpha_d(d) = \begin{cases} \frac{a}{d_c^n}(\mid d - d_c \mid)^n & \text{if } d < d_c \\ 0 & \text{others} \end{cases}, \qquad (8)$$

and

$$\alpha_d(d) = Max(-\frac{a \times d^n}{d_c^n} + a, \quad 0), \qquad (9)$$

where $0 \leq d_c \leq d_{max}$, $0 \leq a \leq 1$, and $n > 1$. Equation 7, 8 and 9 are linear (Figure 5 (a)), concave nonlinear (Figure 5 (b)), and convex nonlinear (Figure 5 (c)) opacity functions respectively. The location of the three opacity functions are controled with d_c and a and n dominates the shape of the nonlinear functions.



(a) Linear Map (b) Concave Nonlinear (c) Convex Nonlinear

Figure 5: Alpha Maps

Since the opacity function, $\alpha_d(d)$, is computed based on only distance, we define another opacity function of intensity like $\alpha_u(v)$ that is controled by a user. The final opacity value is computed by the multiplication of the two opacity functions as following, $\alpha_u(v) \times \alpha_d(d)$. The opacity function has both of user control and automatic opacity generation. While a user turn on some range of the intensity values with $\alpha_u(v)$ by assigning a totally opaque value such as 1, $\alpha_d(d)$ automatically generate the opacities of each voxel with the alpha map of Figure 5. $\alpha_u(v)$ also provides the ramps of traditional 1D transfer functions.

4.2 Gradient Magnitude for the Second Axis

Most multi-dimensional transfer functions have gradient magnitude for the second axis, while intensity works as the first axis. The 2D transfer functions of intensity and gradient magnitude are more powerful than 1D transfer functions to visualize separately several features that have the overlapped intensity ranges, but the distinguishable gradient magnitude ranges. However, as the dimension of transfer functions is extended to 2D space, it is much harder to control by hand. One of the main reasons is that, to visualize an interesting feature or a boundary, while we decide the intensity range of the feature, we also need to search the gradient magnitude range of the feature. In other words, we need to search the combined ranges of intensity and gradient magnitude.

Since direct volume rendering assumes that most boundaries of volume data have some thickness, all voxels that are in a thick boundary (or boundary voxels) should be visualized with proper transparencies, while iso-surfacing extracts very thin surfaces at the exact locations of a boundary. Therefore, the gradient magnitude range of a feature can vary from a relatively small value to a big value in a thick boundary. Even though all voxel's opacities of a volume dataset are decided by some pre-processing, it is not a trivial work to collect boundary voxels through searching a gradient magnitue range.

To reduce the gradient magnitude range searching work, we replace each voxel's gradient magnitude value with the interpolated gradient magnitude value at the hit location. When we compute the distance as in Figure 1, the gradient magnitude value at the hit location is interpolated by the tri-linear interpolation method. If we use the interpolated gradient magnitude at hit location as the second axis, then it will make the searching work easier, since we only have to consider the gradient magnitude range at a boundary.

The 2D opacity function of intensity and gradient magnitude is represented as $\alpha_u(v,g)$, where g represents gradient magnitude of each voxel at the hit location. Each voxel's opacity is finally decided by $\alpha_u(v,g) \times \alpha_d(d)$. The 2D opacity function, $\alpha_u(v,g)$, is controled by a user like [Kniss et al. 2001]'s function, but the opacity values of each voxel can be 1 always, since the opacity funciton of distance, $\alpha_d(d)$, generates alpha values. A user only have to select some regions to be visualized in the 2D space of intensity and gradient magnitude.

The 3rd column of Figure 6 shows the graphs of intensity and gradient magnitude at the hit location that is described in Figure 1. The each slice of Figure 3 is used for the graphs. We can easily recognize that each blob of the 3rd column graphs of Figure 6 represents a boundary and the blobs are usually located in the local maxima of gradient magnitude (the 1st column graphs of Figure 6) or in the zero-crossing locations of the second derivative (the 2nd column graphs of Figure 6).

5 Implementation and Results

We have implemented a 3D texture-based volume renderer using nVidia graphics cards such as GeForce3, 4, and FX. Since the cards provide at least four 3D multi-textures and dependent texture reads with register combiners, the mult-dimensional transfer functions can be implemented on a PC equipped with those graphics cards. As far as we use a single PC, the maximum resolution of the loadable volume data is governed by the texture memory size of those graphics cards. To visualize a large size volume dataset, we also have implemented a 3D texture-based parallel volume rendering

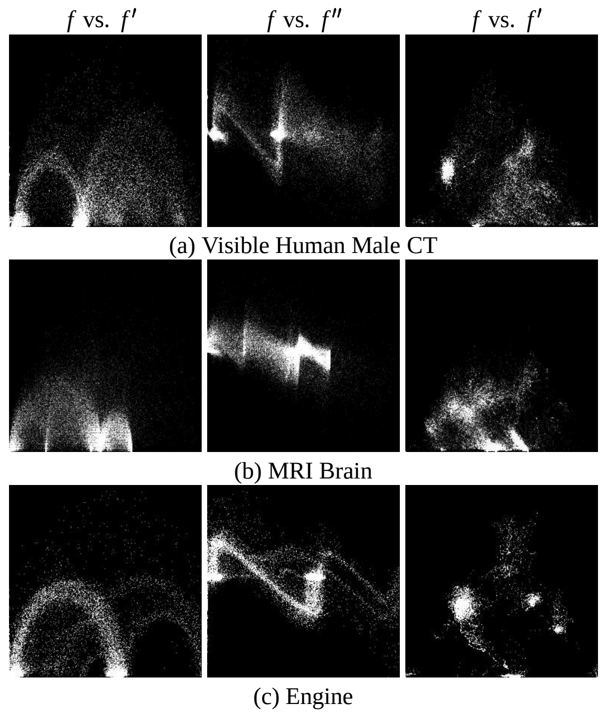


Figure 6: The Graphs of f vs. f' (1st Column), f vs. f'' (2nd Column), and f vs. f' (3rd Column): The f and f' values of the 3rd column graphs are interpolated at the hit locations of the zero-crossing second derivative. Each slice of (a), (b), and (c) of Figure 3 is used to generate these graphs.

tool, but we skip the detail description, since the topic is beyond this paper,

Figure 7 shows the rendering pipe line in a nVidia GeForce card. In the rendering pipeline, dependent texture is used for implementing the opacity function, $\alpha_u(v, g)$ that is controled by a user and assigning colors to each voxel with the color function of v and g. For the shading purpose, diffuse is computed by the dot product of a light vector, L, and a normal vector, N. The normal vectors of voxels are stored in a 3D texture memory with the RGB format. The dot product of the half vector, $H = (L+V)/2$, and a normal vector, N, computes the specular contribution. To avoid too wide specular area, we compute the 8th power of the dot product, $(H \cdot N)^8$, with the register combiners.

The volume rendering pipeline requires the six components, RGB normal, intensity (v), gradient magnitude (g), and distance (d), for each voxel. If we want to visualize a 256^3 volume dataset, (=16 Mbytes), then we need at least $256^3 \times 6$, (=96 Mbytes) texture memory. To reduce the texture memory requirement, our implementation relies on hardware assisting texture compression that is one of the ARB OpenGL extensions. Especially, we use the s3tc texture compression format [NVIDIA 2004] that is provided by the nVidia graphics cards. Since the format treats a 4×4 block as a single pixel, the compression ratio is $1/16$. In our implementation, the s3tc compression format is used for only normal vectors, and the rendering results are reasonably acceptable without hurting rendering performance a lot, even though it is a lossy compression method. If we use the texture compression format, then the loadable maximum resolution of a graphics card that has 128 Mbyte texture memory is $256 \times 256 \times 512$.

We generated several images to test the distance-based multidimen-

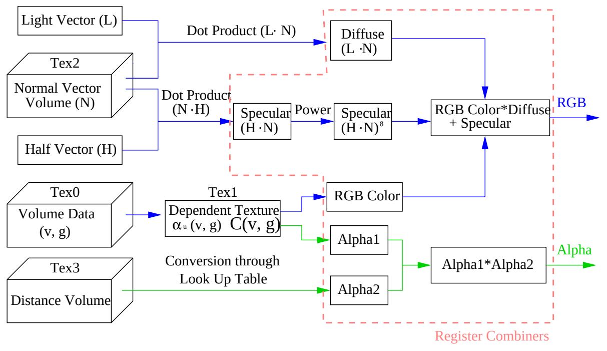


Figure 7: Rendering Pipeline in a nVidia GeForce Card: Three texture volume datasets feed into the register combiners and a RGB color and an alpha value are computed with the register combiners. The solid lines represent data flow and processes and the dashed line indicate the register combiners.

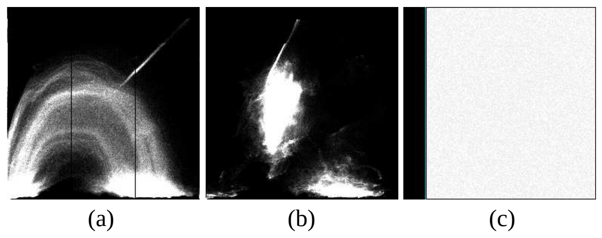


Figure 8: The Turbine Blade Graphs and an Alpha Map: (a) is the general graph of f vs. f', (b) is the graph of f vs. f' at Hit Locations and (c) is the alpha map, $\alpha_u(v, g)$, defined by a user

sional transfer functions with several alpha maps. First, all voxels are turned on by assigning 1 to all voxels that are in the white region of Figure 8 (c) and we adjust the alpha map parameters of a and n and fixed d_c like Figure 9. When we want to visualize thick and opaque boundaries of a volume dataset, then the convex alpha map of Figure 9 (c) will be usefull, while the concave alpha maps of Figure 9 (c) - (e) are useful to make boundaries thinner. Figure 9 shows that simply assigning small alpha values to each voxel (a) is different from making boundary thinner (e).

Since the gradient magnitude value of a voxel is replaced with the value at the hit location of the zero-crossing second derivative, it is easy to decide a gradient magnitude range. Figure 10 shows the advantange of the f vs. f' graph at the hit locations, (b), against the general f vs. f' graph, (a). You can see a feature that has a wide gradient magnitude range from the top to the bottom of Figure 10 (a) and the range is mixed with other features. However, the feature is easily distingushable in Figure 10 (b). Therefore, the proposed multi-dimensional transfer function can reduce user interaction.

One of the main reasons that make transfer function geneneration hard is a huge number of degrees of freedom. Even though we consider only 1D transfer functions, each control point has the two degrees of freedom in intensity and transparency axes. However, in the proposed multi-dimensional transfer functions, the number of degrees of freedom is reduced very much, since the alpha values of each voxel are decided by the alpha function, $\alpha_d(d)$, automatically. Plus, since the shape and location of $\alpha_d(d)$ can be controled by the three parameters, a, d_c, and n, a user can adjust the transparency effect.

6 Conclusions and Future Work

In this paper, we presented the multi-dimensional transfer functions of intensity, gradient magnitude at hit location, and distance to the zero-crossing locations of the second derivative. Multi-dimensional transfer functions are better than 1D transfer functions to visualize separately some features that have overlapped intensity ranges, but distinguishable gradient magnitude ranges. However, the traditional multi-dimesional transfer functions are hard to control by hand, since the functions have an enormous number of degrees of freedom. The proposed multi-dimensional transfer functions can reduce the number of degrees of freem radically, since the alpha values of all voxels are decided automatically with the three parameters, a, d_c, and n based on distance to the zero-crossing location of the second derivative. Plus, the gradient magnitude range searching time can be saved, since each voxel's gradient magnitude values is replaced with the gradient magnitude value of the hit location.

Since we use 2D dependent texture for the alpha function, $\alpha_u(v,g)$, and distance is converted to transparancy by a look up table that is computed with $\alpha_d(d)$, the two alpha functions are separated in the rendering pipeline. Therefore, we only can use the same function, $\alpha_d(d)$, for all v and g. We need to modity the rendering pipeline to implement 3D dependent texture or a similar thing. Another improvement of this paper is to show how much bilateral filters can improve Canny's edge detector. Originally, Canny's edge detector is combined with Gaussian filters.

7 Acknowledgments

This work was supported in part by NSF grants ACI-0220037, CCR-9988357, EIA-0325550, a UT-MD Anderson Whitaker grant, and a subcontract from UCSD 1018140 as part of the NSF-NPACI project, Interaction Environments Thrust.

References

BAJAJ, C. L., PASCUCCI, V., AND SCHIKORE, D. R. 1997. The contour spectrum. In *Proceedings of the 1997 IEEE Visualization Conference*, 167–173.

BUCHANAN, J. L., AND TURNER, P. R. 1992. *Numerical Methods and Analysis*. McGraw-Hill, Inc.

CANNY, J. 1983. Finding edges and lines in images. Tech. rep.

CLARK, J. J. 1989. Authenticating edges produced by zero-crossing algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 1, 43–57.

FRANCIS, J. J., AND DE JAGER, G. 2003. The bilateral medial filter. In *Proceedings of the Fourteenth Anuual Symposium of the Pattern Recognition Association of South Africa*.

GIBSON, S. F. F. 1998. Constrained elastic surface nets: generating smooth surfaces from binary segmented data. In *Medical Image Computation and Computer Assisted Surgery*.

GIBSON, S. F. F. 1998. Using distance maps for accurate surface representation in sampled volume. In *Volume Visualization Symposium*, IEEE.

KINDLMANN, G., AND DURKIN, J. W. 1998. Semi-automatic generation of transfer functions dor direct volume rendering. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*.

KINDLMANN, G. 2002. Multidimensional transfer functions for interactive volume rendering: Design, interface, interaction. *SIGGRAPH Course Notes 8*, 3.

KNISS, J., KINDLMANN, G., AND HANSEN, C. 2001. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the Conference on Visualization*.

KNISS, J., KINDLMANN, G., AND HANSEN, C. 2002. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3, 270–285.

KNISS, J., PREMOŽE, S., IKITS, M., LEFOHN, A., HANSEN, C., AND PRAUN, E. 2003. Gaussian transfer functions for multi-field volume visualization. In *IEEE Visualization 2003*.

LEVOY, M. 1988. Display of surfaces from volume data. *Computer Graphics and Applications 8*, 5, 29–37.

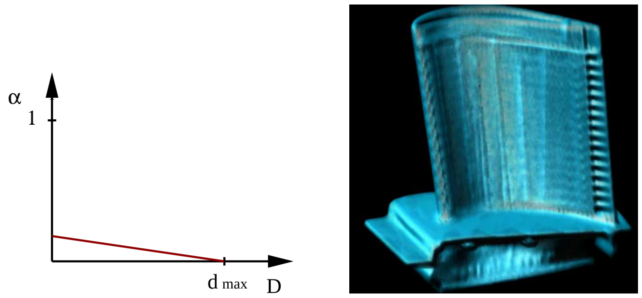
NVIDIA. 2004. *OpenGL Extension Specifications*. NVIDIA Corporation, March. http://developer.nvidia.com/page/home.

PFISTER, H., LORENSEN, B., SCHROEDER, W., BAJAJ, C., KINDLMANN, G., AND PFISTER, H. 2001. The transfer function bake-off. In *IEEE Computer Graphics and Applications*, 16–22.

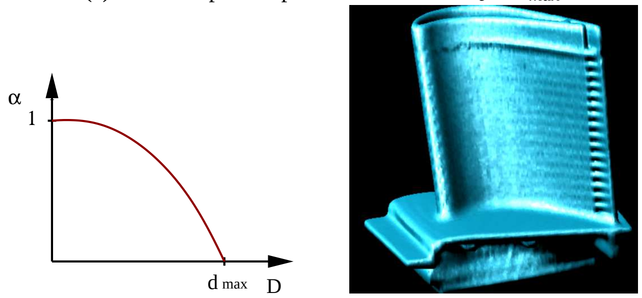
TENGINAKAI, S., LEE, J., AND MACHIRAJU, R. 2001. Salient iso-surface detection with model-independent statistical signatures. In *Proceedings IEEE Visualization*, 231–238.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*.

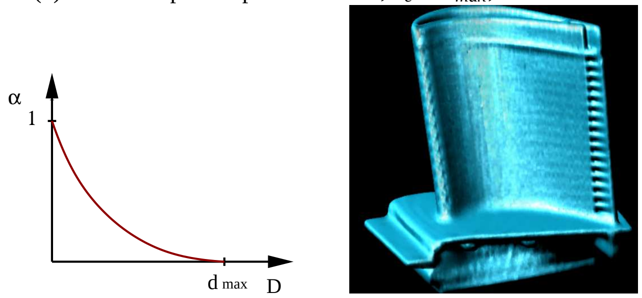
ZIOU, D., AND TABBONE, S. 1998. Edge detection techniques - an overview. *Pattern Recognition and Image Analysis 8*, 4, 537–554.



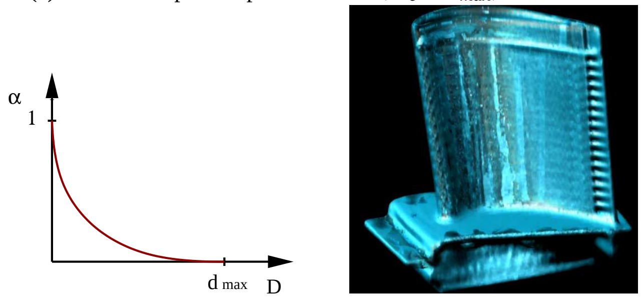
(a) Linear Alpha Map with $a = 0.1$ and $d_c = d_{max}$



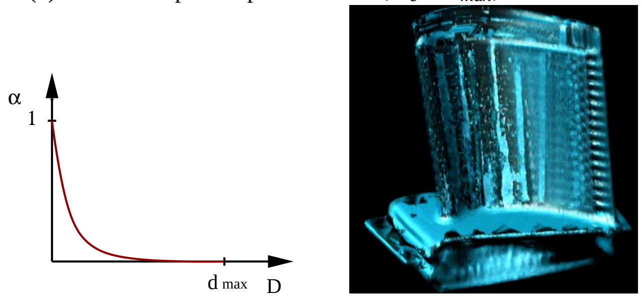
(b) Convex Alpha Map with $a = 1.0$, $d_c = d_{max}$, and $n = 3.0$



(c) Concave Alpha Map with $a = 1.0$, $d_c = d_{max}$, and $n = 3.0$

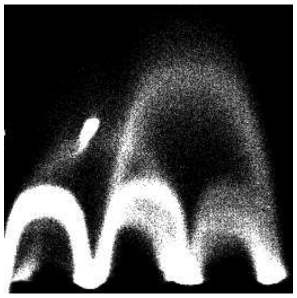


(d) Concave Alpha Map with $a = 1.0$, $d_c = d_{max}$, and $n = 5.0$

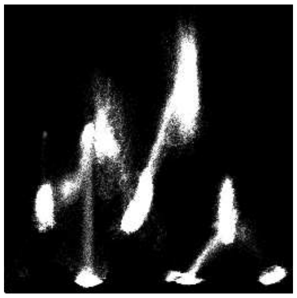


(e) Concave Alpha Map with $a = 1.0$, $d_c = d_{max}$, and $n = 7.0$

Figure 9: Turbine Blade Rendering with several Alpha Maps

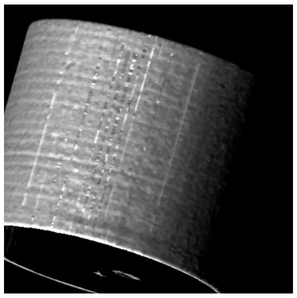


(a) General f vs. f'

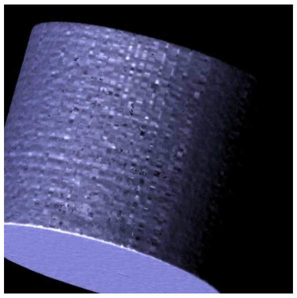


(b) f vs. f' at Hit Locations

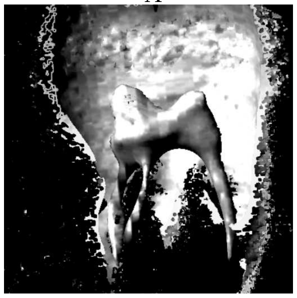
Figure 10: The Tooth Dataset Graphs



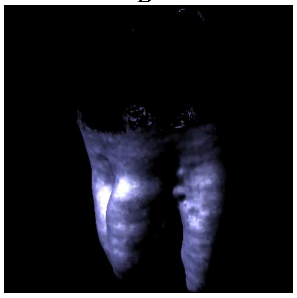
A



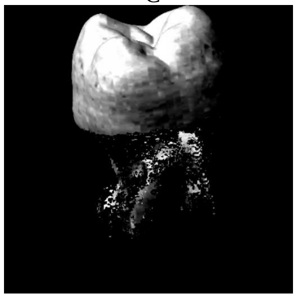
B



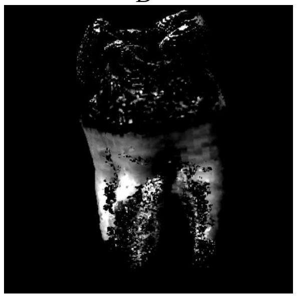
C



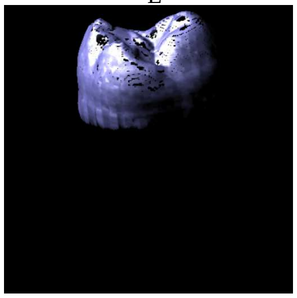
D



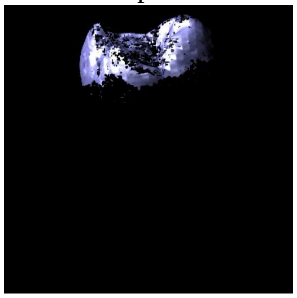
E



F



G



H

Figure 11: Rendered Images of the Tooth Dataset: Each image of A - H is generated by using the alaph maps, A - H, of Figure 12 and the convex alpha map of Figure 9 (b) is used for $\alpha_d(d)$

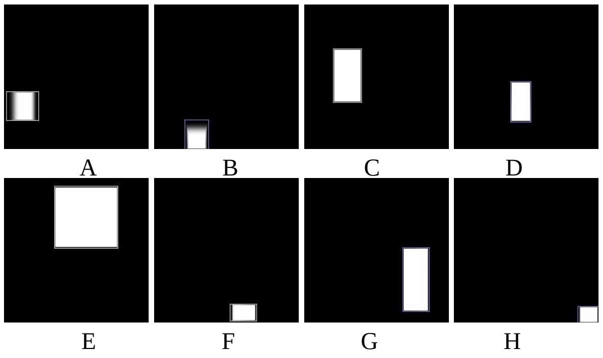
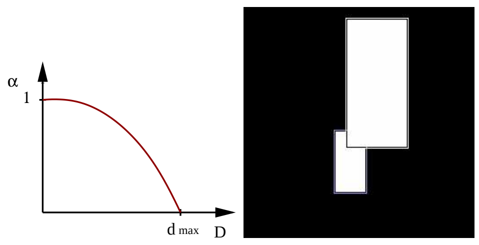
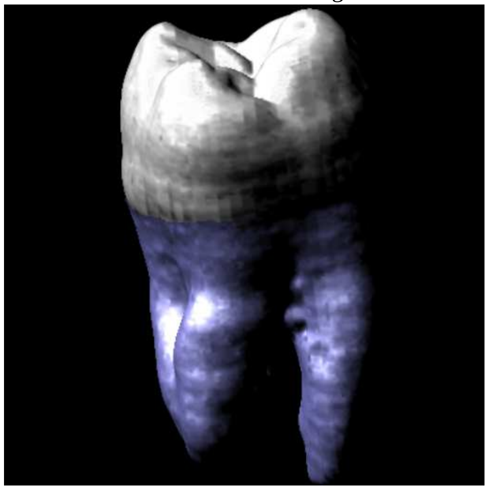


Figure 12: Alpha Maps, $\alpha_u(v,g)$, for the Tooth Images of Figure 11

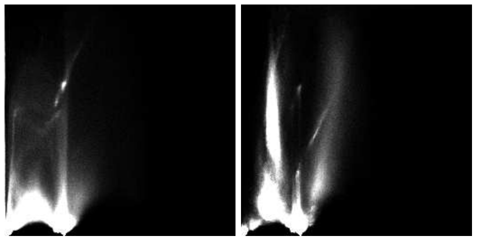


(a) User Defined Opacity Function in 2D Space of Intensity and Gradient Magnitude: Gradient magnitude is replace with the value at the hit locations of the zero-crossing second derivative

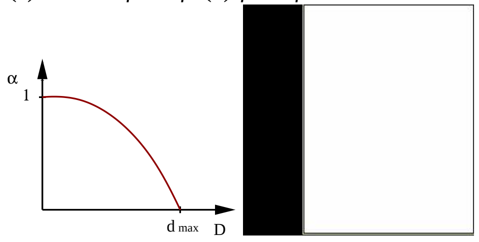


(b) Rendered Image: The two alpha functions are used to generate this image, $\alpha_u(v,g)$ of (a) and $\alpha_d(d)$ of Figure 9 (b). Two different regions are rendered at the same time.

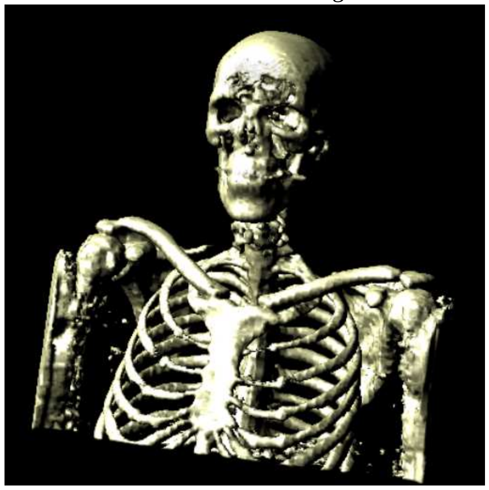
Figure 13: Tooth Image



(a) General f vs. f' (b) f vs. f' at Hit Locations



(c) User Defined Opacity Function in 2D Space of Intensity and Gradient Magnitude: Gradient magnitude is replace with the value at the hit locations of the zero-crossing second derivative



(d) Rendered Image: The two alpha functions are used to generate this image, $\alpha_u(v,g)$ of (a) and $\alpha_d(d)$ of 9 (b).

Figure 14: Visible Human Male CT