Department of
Computer Science

An Adjustable Gradient Filter for Volume Visualization

Michael E. Goss

Technical Report CS-93-105

April 13, 1993

Colorado State University

An Adjustable Gradient Filter for Volume Visualization

Michael E. Goss
Computer Science Department
Colorado State University
Fort Collins, CO 80523

March 30, 1993

Abstract

Volume visualization techniques extract and display structures from three dimensional arrays of sample values representing physical objects. The shading of surfaces of the structures to simulate natural lighting makes the features of these structures visible in images. An important step in the image rendering process is the determination of the orientation of an extracted surface in the form of a gradient vector at each point on the surface. This gradient vector is then used as a surface normal vector for shading calculations. In addition, some volume visualization techniques used the gradient vector for classification of structures contained in the volume. A common method for estimation of the gradient vector is the center difference, which also acts as a fixed-response low pass filter, smoothing details in the image.

This paper presents an improved method for computation of the gradient. The center difference method is extended to allow control of sensitivity to fine details in the source data. The amount of smoothing performed by the gradient computation can be tuned to filter out high frequency noise and aliased energy while retaining as much as possible of the valid frequency content which may be lost using the conventional center difference.

1 Introduction

Volume imaging processes such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) produce a three dimensional array of sample values (voxels). Each voxel typically represents a density measure of some physical property (for example, values resulting from CT represent density of tissue which absorbs X-rays). The volume visualization process attempts to classify the voxels by tissue or material type and produce renderings of structures composed of particular tissues or materials.

A number of different methods have been described in recent literature for classifying and displaying volume data. One common thread in many of these methods is the way in which shading is performed to generate images. The goal is typically to display the best approximation possible to a surface (detected by some classification method), shading the surface using the

standard Gouraud [Gou71] or Phong [Bui75] shading models. The surface normal vectors required by these shading models are usually found using gradient vectors calculated from the voxel values (sometimes known as the *gray-level gradient* [T+90]). These gradient vectors are perpendicular to the estimated surface of constant density, and so can be used as surface normal vectors.

The most commonly used method for approximation of the volume gradient vector is the *center difference*. For voxel value $v_{i,j,k}$ at location i, j, k in the volume, the three components of the gradient vector $\vec{g}$ are computed by taking the scaled difference along each axis of the voxels on either side of the point of interest (the division by the distance 2 is frequently omitted since the length of the gradient vector is normalized before performing shading calculations):

$$\begin{aligned} g_x &= \frac{v_{i+1,j,k} - v_{i-1,j,k}}{2} \\ g_y &= \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2} \\ g_z &= \frac{v_{i,j,k+1} - v_{i,j,k-1}}{2} \end{aligned}$$

The surface normal $\vec{n}$ used for shading is the unit vector in the direction of $\vec{g}$

$$\vec{n} = \frac{\vec{g}}{|\vec{g}|}$$

The volume visualization algorithms which use this method can be divided into two broad classes. One class of algorithm extracts surface information as a 3D model, which can then be rendered using conventional rendering techniques. Lorensen, Cline, et. al. [LC87, CLL+88] use the center difference calculation to compute the gradient vector for shading of models extracted from volume data: polygonal models constructed from a volume in the "marching cubes" algorithm and point primitives in the "dividing cubes" algorithm.

The other broad class of volume visualization algorithms generate pixel values directly from the source data without an intermediate geometric model. Levoy describes a ray-casting method [Lev88, Lev90] which uses the center difference to compute gradient vectors used for classification as well as for shading. Westover [Wes90] uses the the same shading techniques for some images with a forward (voxel to pixel) mapping rendering technique. Drebin, Carpenter, and Hanrahan [DCH88] use the center difference to compute gradient vectors for both classification and shading in a voxel to pixel projection method.

Höhne, Tiede, et. al. [H+90, T+90] also use a technique originated by Zucker and Hummel [ZH81] which computes a weighted average of center differences along each axis and the diagonals closest to that axis. The use of additional off-axis data reduces sensitivity to noise and aliasing, although it is computationally expensive and produces additional smoothing which may or may not be desired.

Cohen et. al. [CKBB90] consider using a larger neighborhood with a depth-buffer (Z-buffer) based gradient calculation. In this case increasing the neighborhood size using the depth-buffer *congradient shading* technique will smooth the picture while a smaller neighborhood will be more sensitive to small changes on the surface.

It is possible to use a larger neighborhood for gradient calculation to increase sensitivity to small changes on a surface. To see how this may be done, the gradient calculation will be examined as a digital filtering process. The properties of the center difference as a filter will be analyzed, and

it will be shown that it is possible to design a filter with greater sensitivity to small changes. This filter will be designed to have adjustable sensitivity to high frequencies which can be reduced in the presence of high frequency noise or aliasing.

2 Gradient Calculation Under Ideal Conditions

Under ideal conditions, volume data would be sampled from a source which contained no noise, and had been filtered before sampling to remove any frequencies above half the sampling frequency. Under these ideal conditions, this low pass filtered source data could be perfectly reconstructed from the samples (in practice, of course, noise and aliasing are always present to some degree).

Ideal low pass filtering produces a continuous function in three dimensions. For such a function, the components of the gradient vector are the partial derivatives in the direction of each primary axis. The first derivative (slope) of a one dimensional cross section in a particular direction can therefore be used to calculate the component of the gradient vector in that direction.

For a one–dimensional analog function in space $x_a(t)$ of continuous variable t (for example, a cross section of the original data from which a volume data set was constructed) samples can be taken at some regular interval T to produce the sequence $x(n) = x_a(nT)$ for integer n. In order to correctly reconstruct the original function $x_a(t)$ from samples $x(n)$, the sample spacing T must satisfy the relationship $T < \frac{1}{2F}$, where F is the highest frequency present in $x_a(t)$ ($2F$ is the *Nyquist* sampling rate) [OS75].

For samples of a periodic function with period NT, the Discrete Fourier Transform (DFT) allows a finite sequence of samples $x(n)$ of length N to be transformed from the space domain to the frequency domain using the relationship

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{2\pi i k n}{N}} \qquad \left(-\frac{N}{2} \leq k \leq \frac{N}{2}\right)$$

where $X(k)$ (k is an integer) is a sample of the frequency spectrum of $x_a(t)$ at frequency $\frac{k}{NT}$, and $i = \sqrt{-1}$.

The inverse of the DFT (IDFT) transforms frequency samples $X(k)$ back to the space domain samples $x(n)$:

$$x(n) = \frac{1}{N} \sum_{k=-N/2}^{N/2} X(k) e^{\frac{-2\pi i k n}{N}} \qquad (0 \leq n \leq N)$$

Given the frequency samples $X(k)$ and the IDFT relationship, the original function $x_a(t)$ can be reconstructed:

$$x_a(t) = \frac{1}{N} \sum_{k=-N/2}^{N/2} X(k) e^{\frac{-2\pi i k t}{NT}}$$

The slope (first derivative) can also be reconstructed from the frequency samples; differentiating both sides of the above gives:

$$\frac{d}{dt} x_a(t) = \frac{1}{N} \sum_{k=-N/2}^{N/2} X(k) \, \frac{-2\pi i k}{NT} \, e^{\frac{-2\pi i k t}{NT}}$$

Define a filter $H(k) = \frac{-2\pi ik}{NT}$ and then define $G(k) = X(k)H(k)$; the above then becomes

$$\frac{d}{dt}x_a(t) = \frac{1}{N}\sum_{k=-N/2}^{N/2} G(k)e^{\frac{-2\pi ikt}{NT}}$$

Sample this slope function at the same locations as the samples of $x_a(t)$ to get slope samples $g(n)$ corresponding to each function sample $x(n)$; it can be seen from the form of this result that $g(n)$ can be generated using the IDFT of G:

$$\begin{aligned} g(n) &= \frac{d}{dt}x_a(nT) \\ &= \frac{1}{N}\sum_{k=-N/2}^{N/2} G(k)e^{\frac{-2\pi ikn}{N}} \qquad (0 \le n \le N) \end{aligned}$$

For the more common case of non-periodic data, a good approximation can be obtained by applying a *leveling* procedure to compute the slope [GP93]. The leveling procedure subtracts a line having the mean slope of the samples $\frac{(x(N)-x(0))}{(N-1)T}$, finds the slope g of this leveled function, and then adds the mean slope to each $g(n)$.

The frequency domain filter $H(k)$ defines the frequency response for the ideal slope calculation. The magnitude is linearly proportional to the frequency $\frac{k}{NT}$, indicating that the slope calculation is very sensitive to high frequencies. Since in real data the high frequencies are most affected by noise and aliasing, it is necessary to control this sensitivity.

3 Analysis of the Center Difference Method

The center difference method estimates the slope of a one-dimensional function along each axis using only the two samples adjacent to the point at which the slope is desired.

For a one–dimensional signal x at point n the center difference is

$$c_x(n) = \frac{1}{2}x_{n+1} - \frac{1}{2}x_{n-1}$$

This is a convolution of samples $x(n)$ with the three element sequence

$$h_c(n) = \left[\frac{1}{2}, 0, -\frac{1}{2}\right] \qquad (n = -1, 0, 1)$$

The performance of this calculation in accurately estimating the slope can be analyzed by examining the frequency response of this filter. The DFT of the sequence gives the frequency response, which is

$$\begin{aligned} H_c(k) &= \frac{1}{2}\left(e^{-2\pi ik/N} - e^{2\pi ik/N}\right) \\ &= -i\sin 2\pi\frac{k}{N} \qquad \left(-\frac{N}{2} \le k \le \frac{N}{2}\right) \end{aligned}$$

Figure 1 compares the frequency response of this filter with the frequency response of the ideal slope

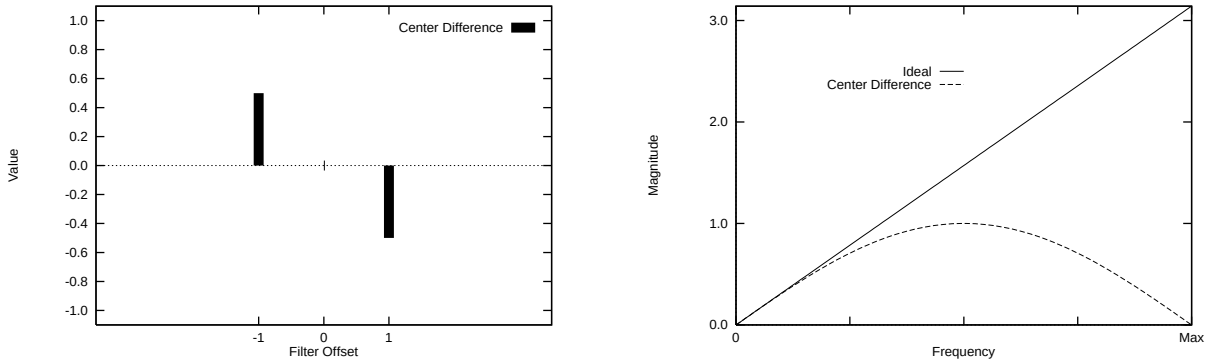


Figure 1: Center Difference Filter and Frequency Response Compared to Ideal

filter (DFT method) discussed above. While the center difference provides a good approximation of the ideal at lower frequencies, it shows a rapid fall off in response in the higher frequencies, corresponding to loss of response to fine detail in the resulting slope values.

This fall off at higher frequencies has some desirable properties. Volume data (along with many other types of data) often contains noise and aliased energy. The ratio of this spurious energy to the actual signal is usually highest in the upper frequencies, where the sensitivity of the ideal filter is greatest. By attenuating the higher frequencies, artifacts caused by noise and aliasing can be reduced.

The undesirable property of the center difference is that the frequency response is fixed. For data with little noise or aliasing, the center difference still removes fine detail in the data, resulting in a blurred appearance.

4 An Adjustable Gradient Filter

A gradient filter which could be adjusted in sensitivity to high frequencies would retain the desirable properties of the center difference without undesired blurring. A simple spatial filter for gradient computation can be generated by truncation of the (infinite duration) impulse response of an ideal gradient filter. For a filter with M non–zero elements (even M, with $M/2$ elements to each side of the 0 center), this results in the filter:

$$h_t(n) = \begin{cases} \frac{\cos \pi n}{n} & \text{for } n \neq 0 \text{ and } -\frac{M}{2} \leq n \leq \frac{M}{2} \\ \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Note that } \frac{\cos \pi n}{n} = \begin{cases} -\frac{1}{n} & \text{for } n \text{ odd} \\ \\ \frac{1}{n} & \text{for } n \text{ even} \end{cases}$$

The convolution with the voxel data along a single axis direction (for example, in the x direction) is computed by

$$\sum_{m=-M/2}^{M/2} h_t(-m) v_{i+m,j,k}$$

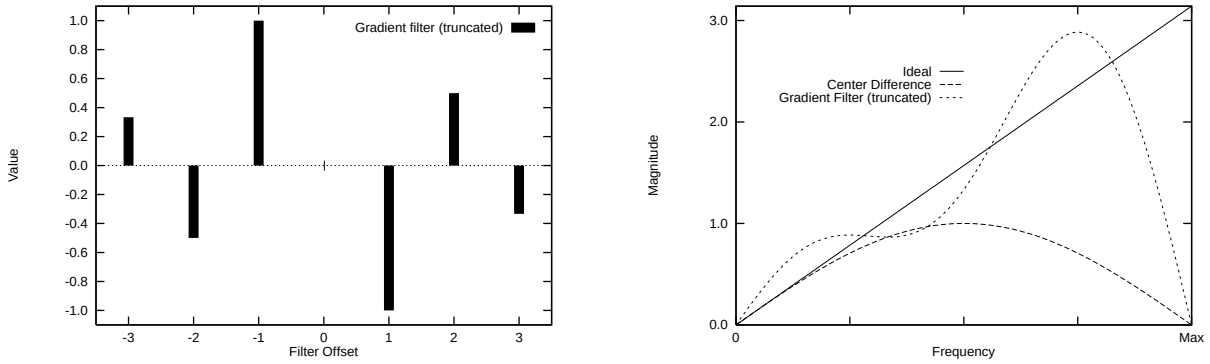


Figure 2: Truncated Gradient Filter with Frequency Response

The resulting filter and its frequency response are shown in Figure 2 for the 7-element $(M = 6)$ filter $[0.333, -0.5, 1, 0, -1, 0.5, -0.333]$. Ideal and center difference frequency responses are shown for comparison.

The large ripples throughout the frequency range due to the abrupt truncation of the filter must be dealt with in order to use this filter. Multiplying the filter by a window function (a function which gradually tapers off toward 0 at the edges) can be used to reduce the effects of truncation of the filter. The *Kaiser window* [Ant79] is particularly useful since it has an adjustable parameter α (alpha) which controls how quickly it tapers off to zero at the edges. A Kaiser window of N samples with parameter α is computed as

$$w_\alpha(n) = \frac{I_0(\beta)}{I_0(\alpha)} \qquad \left(-\frac{N}{2} \leq n \leq \frac{N}{2}\right)$$

where $I_0(x)$ is the order 0 modified Bessel function [PFTV88] of x, and β is calculated from α as

$$\beta = \alpha\sqrt{1 - \left(\frac{2n}{N-1}\right)^2}$$

Kaiser windows with values of α near 0 approximate a rectangular window, while increasing values of α smooth the transition to 0 at the edges of the window. When the Kaiser window is multiplied by the gradient filter, larger values of α reduce ripple in the frequency response at the cost of attenuation of the higher frequencies. Figure 3 shows Kaiser windows computed with three different values of α. Figure 4 shows the filters which result from multiplication of the truncated gradient filter by Kaiser windows with α values of 4.0, 8.0, and 16.0. At $\alpha = 4.0$, the ripples in the frequency response have been almost completely smoothed out, but the response to high frequencies is still quite good compared to the center difference (shown for comparison).

By adjusting the Kaiser window parameter α, the response to higher frequencies can be adjusted to any degree desired. As can be seen in Figure 4, as the value of α increases, the response of the filter to high frequencies decreases, approaching the response of the center difference. If the noise and aliasing characteristics of the data can be estimated, a value of α can be selected to optimize the gradient filter response. If sufficiently fast processing hardware is available, the rendering software could even allow a user to interactively adjust the value of α to produce a satisfactory image.

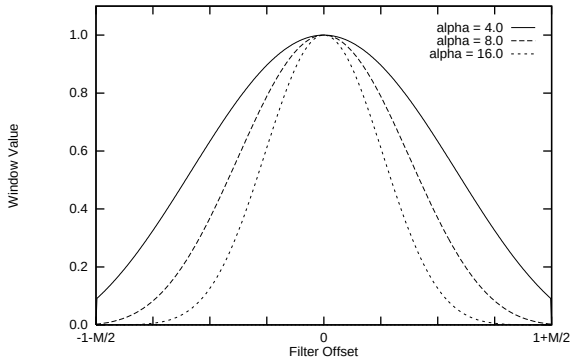


Figure 3: Kaiser windows with different α values

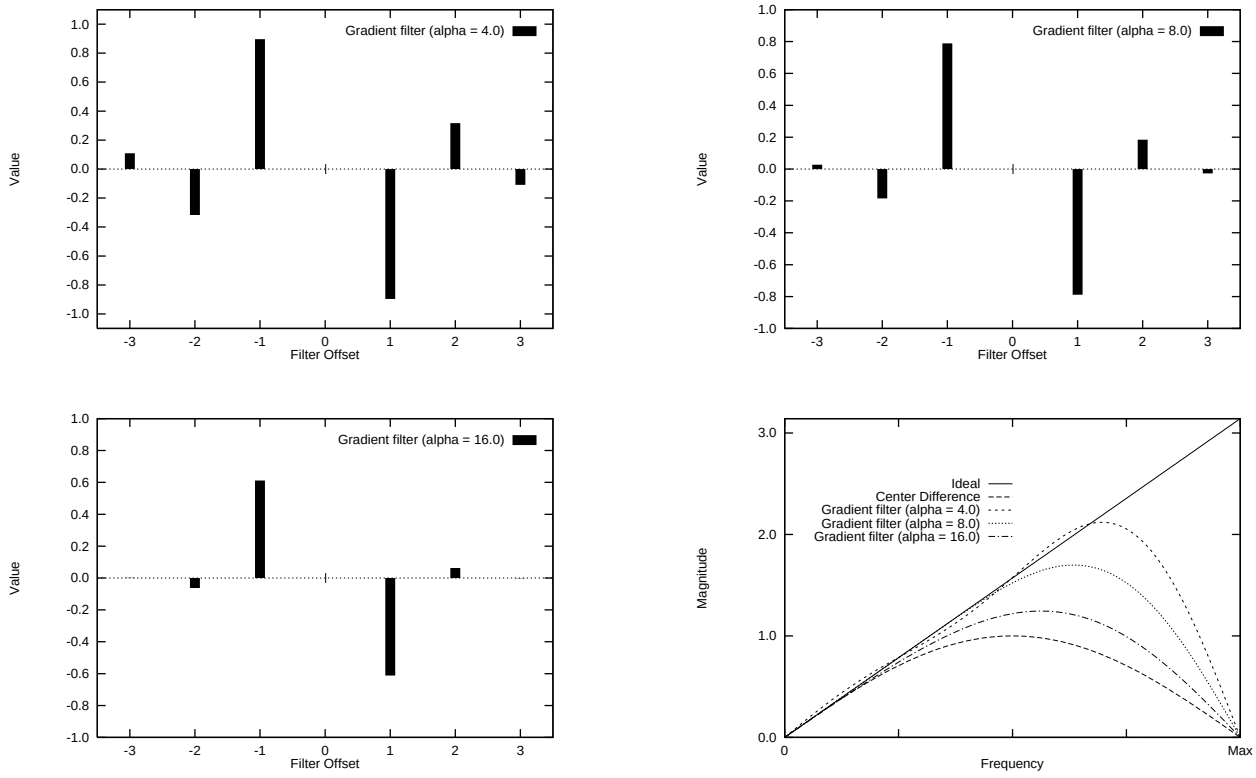


Figure 4: Windowed Gradient Filters with Frequency Responses

5 Results

Figures Img–1 through Img–3 show images generated from two volume data sets. `MRbrain` is a $256 \times 256 \times 109$ voxel magnetic resonance image of a head with the brain surface exposed. `CThead` is a $256 \times 256 \times 113$ voxel CT image of a head. Both source data sets come from the University of North Carolina Volume Rendering Test Dataset, Volume II.

The images in the figures were generated using ray casting techniques based on [Lev88], but the effect of the gradient filter on the level of detail visible in the images should be similar using other volume visualization techniques. The images were rendered at a resolution of 400×400 pixels, using the Phong shading model.

The upper two images in Figure Img–1 show approximately a quarter of the `MRbrain` data set towards the top and back of the head, while the lower two images show close-up views generated using the same rendering parameters. In the images on the left the center difference was used for gradient computations. A 7 element gradient filter was used to generate the images on the right. The filter was generated with $\alpha = 4.0$, giving filter values $[0.1086, -0.3167, 0.8964, 0, -0.8964, 0.3167, -0.1086]$. In the images on the right, the greater response to high frequencies of the 7 element gradient filter results in sharper contrast and better visibility of the smaller features on the surface when compared to the images on the left.

Figure Img–2 shows a similar comparison with images generated from the `CThead` data set. Because the bone surface is smoother (and therefore has fewer high frequency components), the differences are not as marked as in the `MRbrain` images, but some improvement can still be seen.

Figure Img–3 shows images from `MRbrain` with high frequency filtering at levels in between those of the images in Figures Img–1(c) and (d). The upper image (a) was generated with $\alpha = 8.0$, giving gradient filter values $[0.0276, -0.1845, 0.7888, 0, -0.7888, 0.1845, -0.0276]$. The lower image (b) was generated with $\alpha = 16.0$, giving gradient filter values $[0.0018, -0.0631, 0.6116, 0, -0.6116, 0.0631, -0.0018]$. These two images show how the value of α can be used to adjust the level of high frequencies to which the gradient filter responds. The higher α value in (b) results in a controlled blurring of the image by reduction of the high frequencies.

6 Conclusions

Frequency response is an important consideration in calculation of the gradient of volume data. The fixed frequency response of the center difference gradient computation method prevents rendering of high frequency detail. Use of the adjustable gradient filter described in this paper provides the flexibility to tune the gradient calculation in order to provide any desired balance between sensitivity to high frequency detail present in the data and attenuation of high frequency noise and aliasing which may also be present.

7 Acknowledgements

This research was supported in part by a Faculty Research Grant from Colorado State University, and by equipment donations from IBM and Hewlett-Packard. Most of the rendering software used was written by Scott Bowman and Anthony Schwickerath. Thanks to the University of North

Carolina at Chapel Hill (and especially Henry Fuchs) for supplying the volume rendering test data sets.

References

[Ant79] Andreas Antoniou. *Digital Filters: Analysis and Design*. McGraw–Hill, New York, 1979.

[Bui75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[CKBB90] Daniel Cohen, Arie Kaufman, Reuven Bakalash, and Samuel Bergman. Real time discrete shading. *The Visual Computer*, 6(1):16–27, February 1990. Also in [Kau91].

[CLL+88] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter. Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics*, 15(3):320–327, May/June 1988. Also in [Kau91].

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, August 1988. SIGGRAPH Conference Proceedings.

[Gou71] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, June 1971.

[GP93] Michael E. Goss and Ivor P. Page. Normal vector generation for sampled data using fourier filtering. *The Journal of Visualization and Computer Animation*, 4(1):33–49, January-March 1993.

[H+90] Karl Heinz Höhne et al. 3D visualization of tomographic volume data using the generalized voxel model. *The Visual Computer*, (6):28–36, 1990.

[Kau91] Arie Kaufman, editor. *Volume Visualization*. IEEE Computer Society Press, 1991.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. SIGGRAPH Conference Proceedings.

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[Lev90] Marc Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, 10(2):33–40, March 1990.

[OS75] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice–Hall, Englewood Cliffs, NJ, 1975.

[PFTV88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge UP, Cambridge, England, 1988.

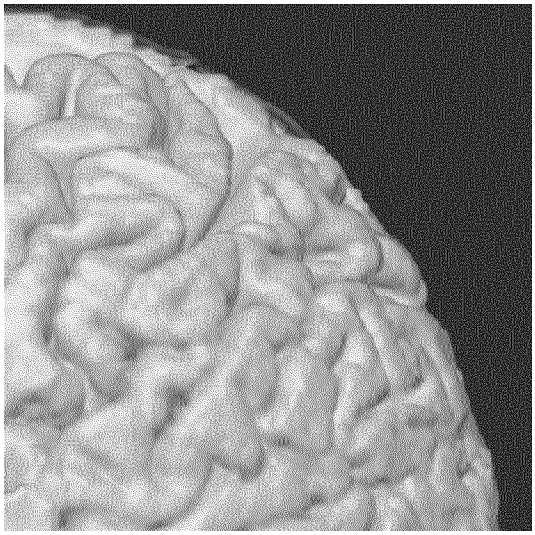
[T+90] Ulf Tiede et al. Investigation of medical 3D–rendering algorithms. *IEEE Computer Graphics and Applications*, 10(2):41–53, March 1990.

[Wes90] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, August 1990. SIGGRAPH Conference Proceedings.

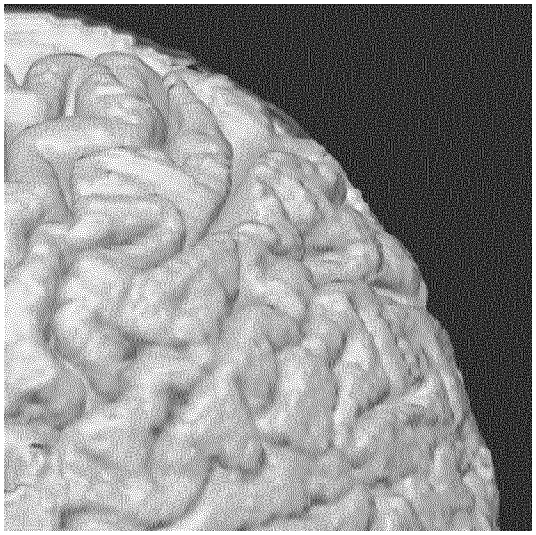
[ZH81] Steven W. Zucker and Robert A. Hummel. A three-dimensional edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(3):324–331, May 1981. Also in [Kau91].

Note: the images on the following pages were converted to a black and white bitmap format for printing on a standard 300dpi PostScript laser printer. This process unavoidably degrades image quality. Gray scale images corresponding to the figures are available in X11 window dump format (.xwd) via anonymous ftp from site `beethoven.cs.colostate.edu` in directory `pub/TechReports/1993/tr-105-images`. The compressed image files are named after the figure numbers; for example, figure Img–2(c) would be named `Img2c.xwd.Z`. The following commands will work on most Unix/X11 systems to display an image (for example, `Img2c.xwd.Z`):

```
uncompress Img2c.xwd.Z
xwud -in Img2c.xwd
```



(a) Center Difference

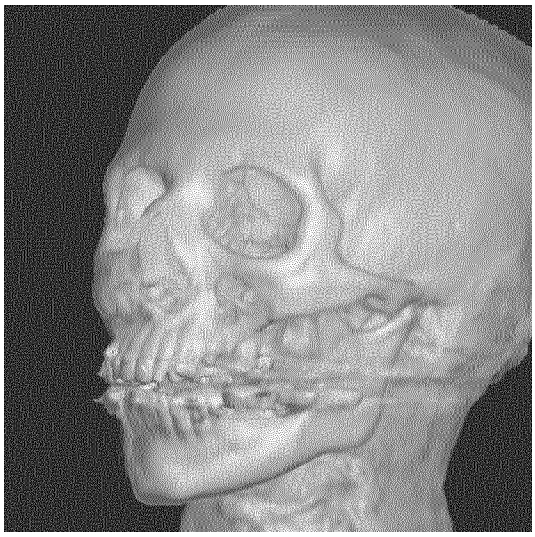


(b) 7 Element Filter

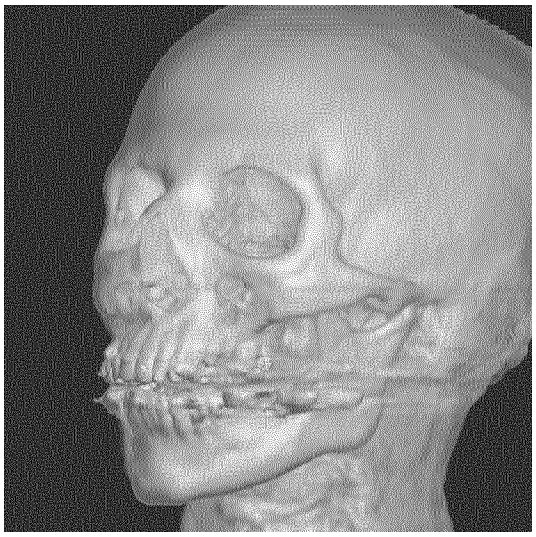
(c) Center Difference

(d) 7 Element Filter

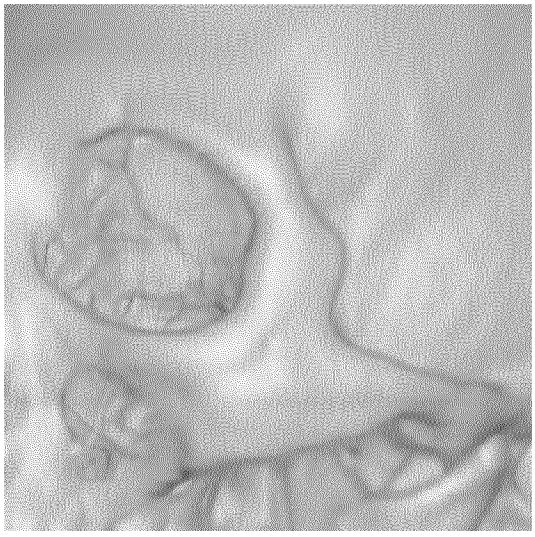
Figure Img–1: `MRbrain` rendered using two different gradient computation techniques



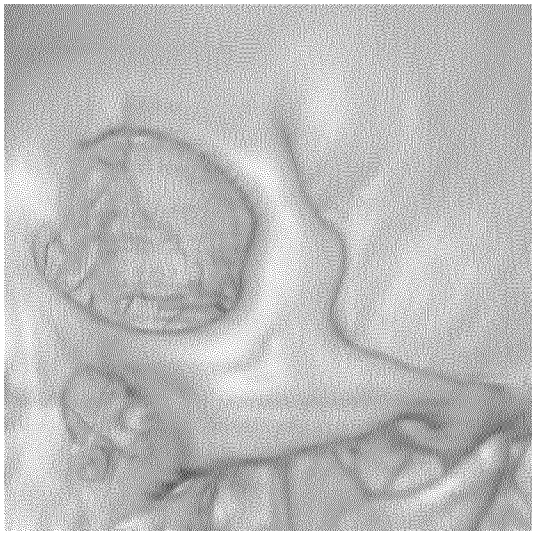
(a) Center Difference



(b) 7 Element Filter



(c) Center Difference



(d) 7 Element Filter

Figure Img–2: `CThead` rendered using two different gradient computation techniques

(a) α = 8.0

(b) α = 16.0

Figure Img–3: `MRbrain` rendered with 7 element gradient filter, varying α values