Frequency analysis of gradient estimators in volume rendering

final version

Mark J. Bentum
University of Twente
Dept. of Electrical Engineering
Lab. for Network Theory
P.O. box 217
7500 AE Enschede
The Netherlands
phone : +31-53-892673
fax : +31-53-334701
email : mark@nt.el.utwente.nl

Barthold B.A. Lichtenbelt and Tom Malzbender
Hewlett Packard Laboratories
Visual Computing Department
1501 Page Mill Road
Palo Alto, CA 94304
United States
phone : +1-415-8576760
fax : +1-415-8523791
email : barthold@hpl.hp.com
email : tom_malzbender@hpl.hp.com

November 8, 1995

Keywords: Volume rendering, volume visualization, gradient filters

Abstract

In volume rendering gradient information is used to classify and color samples along a ray. In this paper we present an analysis of the theoretically ideal gradient estimator and compare some commonly used gradient estimators with the ideal estimator. A new method is presented to calculate the gradient on an arbitrary position, using the derivative of the interpolation filter as the basis for the new gradient filter. As an example we will discuss the use of the derivative of the cubic spline. A comparison will be made showing the differences with other methods. Computational efficiency can be realized since parts of the interpolation computation can be leveraged in the gradient estimation.

1 Introduction

Visualizing a given three dimensional dataset can be done by surface rendering algorithms, e.g. Marching Cubes [8] or by direct volume rendering algorithms, e.g. raycasting [7] or splatting [19]. For direct volume rendering methods the voxel intensity and the gradient magnitude is often used to shade and classify the dataset. For surface rendering techniques the gradient is used to estimate the surface normal. This normal is needed in order to shade the polygons that form the surface.

The shading of objects in natural environments provides cues to the three dimensional structure and position of the objects. In volume rendering the local gradient is often used as an approximation to the surface normal for use in a shading model. Phong shading[11], is widely used in surface and direct volume rendering algorithms.

In volume rendering the dataset to be rendered has to be classified. Classification calculates an optical density value for each voxel in the dataset, called opacity. Opacity is calculated to accentuate surfaces or the boundaries between different materials. Opacities are typically calculated using either voxel intensities or a combination of voxel intensities and gradient information.

Once the dataset is classified and the opacity and color is calculated, the data has to be rendered. The color and opacity values are composited to achieve the final two dimensional projection. Several rendering techniques are known. The most important distinction between the techniques is the order in which the voxels are processed to create an image, image-order or object-order algorithms. Examples of image-order algorithms are raycasting [7] and Sabella's method [14]. Examples of object-order algorithms are the splatting algorithm [18], V-buffer algorithm [16] and the Slice Shearing algorithm [4].

It is possible to calculate color and opacity as a preprocessing step, yielding two new voxel datasets, a color and an opacity dataset. During the rendering step the color and opacity on sample points (generally not on voxel positions) are calculated by interpolation. This may reduce the quality of the image, and an alternative is to calculate the color and opacity during rendering on the sample positions. This has some consequences for the complexity of the algorithm.

This paper consists of 9 sections. In section two ideal interpolation will be briefly discussed. Understanding interpolation is essential to understand the ideas in the rest of the paper. Section three discusses the notion of perfect gradient estimation. Then in section four some well known gradient estimators are discussed and analyzed with respect to the perfect gradient estimator. Section five discusses the frequency analysis of gradient estimators. Section six proposes a different view to designing a gradient estimator by using the derivative of the interpolation function as the gradient estimator. An example which uses the cubic spline is given. Section seven discusses the implementation of this example in volume rendering.

Section eight gives the results of using this example and section nine discusses the proposed methods.

2 Ideal Interpolation

The process of interpolation is one of the fundamental operations in digital signal processing and computer graphics. The purpose of interpolation is to calculate intermediate values of a continuous signal $f(x,y,z)$ from a discrete signal. In volume rendering interpolation is used to calculate the values on the sample positions along rays, since it is unlikely these points will be positioned on gridpoints.

According to the sampling theorem [15] a signal can be reconstructed exactly by the ideal interpolation function if it is bandwidth-limited and sampled at the Nyquist rate, or higher. The ideal interpolation function is the *sinc* function

$$r(x)_{ideal} = \mathrm{sinc}(x) = \frac{\sin x}{x} \tag{1}$$

This ideal filter removes all replicates of the frequency spectrum introduced by sampling the original continuous function by multiplying with a block function in the frequency domain.

In computer graphics one has a set of discrete points, an image or volume. This set of points is obtained by sampling the continuous function $f(x,y,z)$ in the three dimensional case. One has to be careful and realize that the set of discrete points might not represent the continuous function $f(x,y,z)$ one wishes to process because of undersampling (sampled at a rate lower than the Nyquist rate) or because this continuous function is not bandwidth limited. If this is the case this set of discrete points of course still can be manipulated and interpolated. The *sinc* is still the perfect reconstruction filter, but it will reconstruct a continuous function $f'(x,y,z)$ that is (slightly) different from the one originally sampled.

Given a set of discrete points, the *sinc* is indeed the best interpolation function. This *sinc* function, however, is defined over an infinite spatial interval, and can therefore not be used as an interpolation function. Other methods must be used, such as the nearest neighbor, linear and higher order interpolation functions [1],[6],[9],[10].

3 Ideal Gradient Estimation

The gradient, or normal, of a surface is the partial derivative of the surface with respect to all three directions. Given a function $f(x, y, z)$ the gradient is

$$\nabla f(x, y, z) = \left(\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}, \frac{\delta f}{\delta z} \right) \tag{2}$$

In volume rendering, a 3-D dataset consists of discrete samples of $f(x, y, z)$ called voxels. If this function $f(x, y, z)$ is not known, which in general is the case, the gradient has to be calculated using these voxels.

Gradient estimation can be analyzed in the same way as interpolation. When the gradient is needed at a location other than a given voxel point, some kind of reconstruction filter has to be used to estimate the derivative (in each direction) of $f(x, y, z)$. Compare this to interpolation which estimates $f(x, y, z)$ itself at a sample point.

Since the gradient is the partial derivative of the original function $f(x, y, z)$ and ideal interpolation with the *sinc* will reconstruct that function, the gradient can be reconstructed exactly by using the derivative of the *sinc* as a reconstruction kernel.

In one dimension the ideal gradient reconstruction filter is

$$\begin{aligned} \frac{d}{dx}\left(\frac{\sin \pi x}{\pi x}\right) &= \frac{\pi x \pi \cos(\pi x) - \sin(\pi x)\pi}{\pi^2 x^2} \\ &= \frac{\cos \pi x}{x} - \frac{\sin \pi x}{\pi x^2} = \frac{\cos(\pi x) - \mathrm{sinc}(\pi x)}{x} \end{aligned} \tag{3}$$

This result is consistent with the results found in [13].

In order to analyze the filter of equation 3 we will look at its frequency response. The Fourier Transform of the *sinc* function is a block signal in the interval $[-\pi, \pi]$. Using the derivative theorem for Fourier Transforms [3] we find that the Fourier Transform of the derivative of the *sinc* is $j\omega$ times the Fourier Transform of the *sinc* itself. This results in a constant slope in the frequency domain for the ideal gradient filter in the interval $[-\pi, \pi]$. See Figure 1.

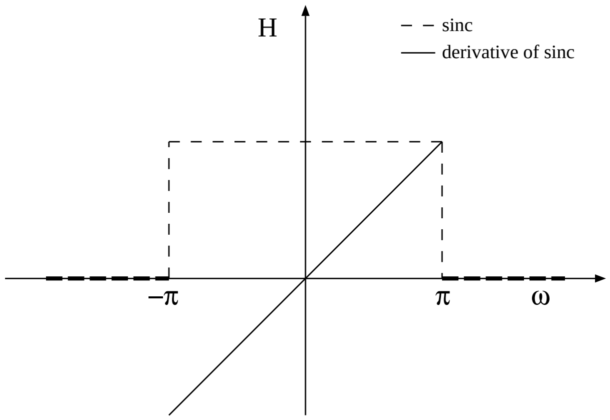

Figure 1: *Frequency response of the ideal gradient estimator and of the sinc function.*

4 Some commonly used Gradient Estimators

Several methods exist to estimate a local gradient in volume datasets. The gradient of a surface to be visualized is an important value, since both the shading and opacity values may depend on the gradient of the surface.

One of the most commonly used methods in volume rendering calculates the gradient with a 6 neighborhood function. This is also referred to as the central difference method. The gradient at voxel (x, y, z) is calculated as follows

$$\begin{aligned} g(x,y,z) = \nabla f(x_i, y_j, z_k) \quad = \quad & \frac{1}{2}(f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k)), \\ & \frac{1}{2}(f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k)), \\ & \frac{1}{2}(f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1})) \end{aligned} \tag{4}$$

Alternatively, all 26 neighbors of a voxel can be used to calculate the gradient. This is usually a better estimation of the gradient, but takes more time to calculate. Another disadvantage is that additional smoothing is introduced. See [5] and [12]. The Zucker-Hummel 3-D edge operator [21] is one example of a 26 point gradient estimator.

For small objects in the dataset even a 6 voxel neighborhood gradient estimator may be too large. An algorithm which adapts to the thickness of the object was proposed in [12]. It selects between 3-6 adjacing voxels to compute the gradient. For the x component of the gradient it works in the following way: If the voxel value at (i, j, k) is larger (smaller) than the value of the neighbors at $(i-1,j,k)$ and $(i+1,j,k)$, it takes the gradient between the voxel itself and the the neighbor with the lower (higher) gray value; otherwise it takes the gradient between both neighbors. The same method is used for the y and z components of the gradient vector. This principle can be generalized to all 26 neighborhoods. This methods has been called adaptive gray-level gradient estimation.

Another adaptive method is proposed in [20]. The authors propose to segment the volume first into small contexts which do not contain any discontinuities. Then for each voxel in each context the gradient is estimated. The gradients will not vary greatly within a context. The disadvantage is that the method relies on a segmentation step and a pre and postfiltering step besides the gradient estimation itself.

Bosma et al. [2] propose a slightly different method than the central difference method. They calculate the gradient using two neighboring values. In that case the gradient is calculated in between the original voxel positions. Estimating the gradient on sample positions requires a linear interpolation between the closest gradient values. We refer to this method as the intermediate difference method.

Mathematically the intermediate and central difference methods can be described as follows.

Suppose we have the values $f(-1), f(0), f(1), f(2)$. Using the central difference method, gradients are calculated on voxel positions as follows

$$g(0) = \frac{f(1) - f(-1)}{2} \tag{5}$$

$$g(1) = \frac{f(2) - f(0)}{2} \tag{6}$$

Using the intermediate difference method, gradients are calculated in between voxel positions

$$g(-0.5) = f(0) - f(-1) \tag{7}$$

$$g(0.5) = f(1) - f(0) \tag{8}$$

If we want to know the gradient on position x with x between 0 and 0.5, the estimation with the central difference method equals

$$
\begin{aligned}
g(x) &= (1-x)g(0) + xg(1) \\
&= (1-x)\frac{1}{2}(f(1)-f(-1)) + x\frac{1}{2}(f(2)-f(0)) \\
&= \frac{1}{2}(f(1)-f(-1)) + x(\frac{f(-1)}{2} - \frac{f(0)}{2} - \frac{f(1)}{2} + \frac{f(2)}{2})
\end{aligned} \tag{9}
$$

While the intermediate difference method results in

$$
\begin{aligned}
g(x) &= (1-(\frac{1}{2}+x))g(-0.5) + (1-(\frac{1}{2}-x))g(0.5) \\
&= (\frac{1}{2}-x)g(-0.5) + (\frac{1}{2}+x)g(0.5) \\
&= \frac{1}{2}(f(1)-f(-1)) + x(f(-1) - 2f(0) + f(1))
\end{aligned} \tag{10}
$$

Note that on the voxel positions these methods are identical.

5 Frequency domain analysis

To compare some of these practical gradient estimators to the ideal case we compare the frequency transforms of these estimators to the frequency transform of the ideal case.

To analyze discrete signals in the frequency domain, the discrete Fourier transform (DFT) can be applied. For samples of a periodic function with period NT, the DFT transforms a finite sequence of samples $x(n)$ of length N to the frequency domain by

$$
X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{2\pi i k n}{N}} \qquad \left(-\frac{N}{2} \leq k \leq \frac{N}{2}\right) \tag{11}
$$

The ideal gradient filter is a constant slope function between $-\pi$ and π, as shown earlier. First we will look at truncating the ideal gradient filter given in equation 3. Figure 2 the effects in the frequency domain can be seen of using such a filter truncated to a width of 4,6 and 10 voxels.

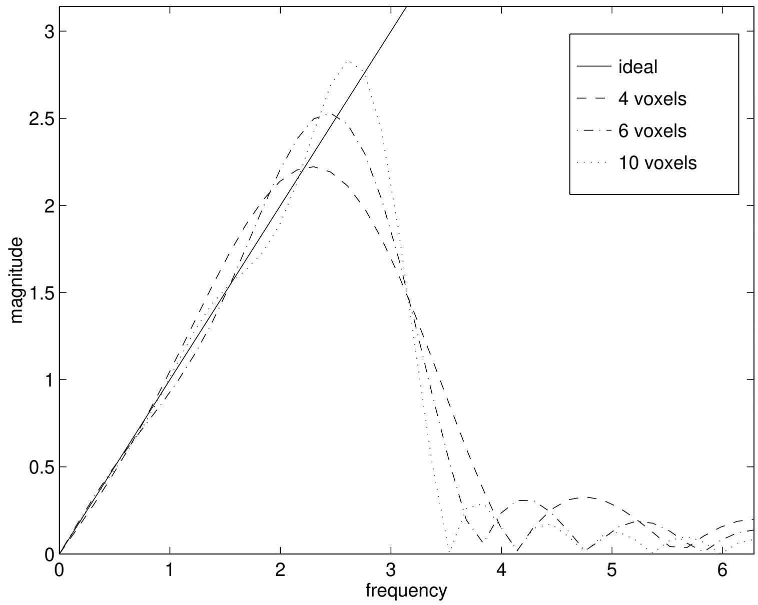

Figure 2: *Frequency response of the truncated ideal gradient filter.*

The frequency response of the central difference method can be calculated analytically. The DFT of the three element sequence

$$h_c(n) = [\frac{1}{2}, 0, -\frac{1}{2}] \qquad n = [-1, 0, 1] \tag{12}$$

is given by

$$H_c(k) = \frac{1}{2}\left(e^{-2\pi ik/N} - e^{2\pi ik/N}\right) = -i\sin 2\pi\frac{k}{N} \qquad -\frac{N}{2} \leq k \leq \frac{N}{2} \tag{13}$$

The frequency response of the intermediate difference method can also be calculated analytically. The DFT of the two element sequence

$$h_i(n) = [1, -1] \tag{14}$$

and shifted by half a voxel to match the individual grids results in

$$H_i(k) = e^{-2\pi ik\frac{1}{2}/N} - e^{2\pi ik\frac{1}{2}/N} = -2i\sin\pi\frac{k}{N} \tag{15}$$

There are two important differences with the DFT of the central difference method. The amplitude is twice as high and the period is twice as low. This makes the intermediate difference a better estimator in the passband, but the aliasing energy is much more than that of the central difference estimator. See Figure 3.

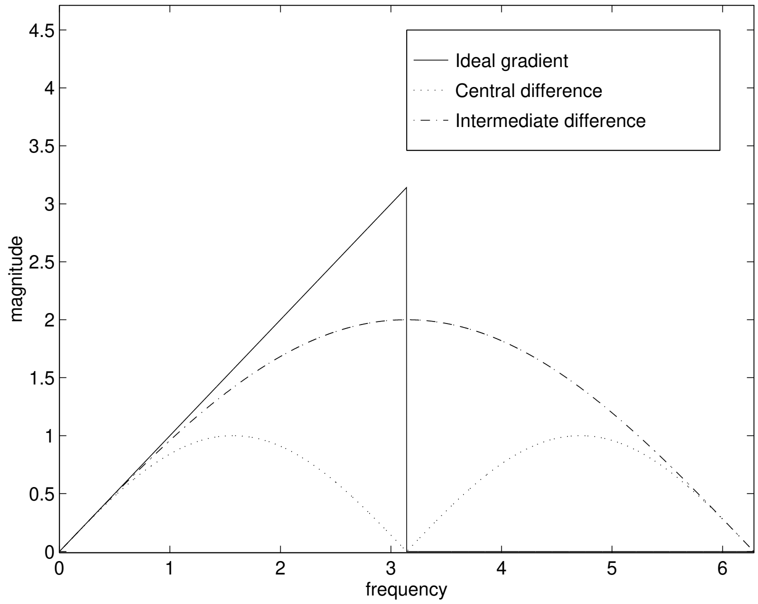

Figure 3: *Frequency response of the central difference and intermediate difference estimator.*

5.1 combining gradient estimation and interpolation

Frequencies above the cutoff frequency π in Figure 2 and Figure 3 are aliased back since the interpolation filter, used to calculate sample values on non voxel positions, is not ideal. If the interpolation filter was perfect, it would not matter what the energy content of the gradient estimator is for frequencies above π since these would all be filtered out by the interpolation.

Figure 4 shows the effect of estimating the gradient on a sample position using central differences to calculate the gradient on voxel positions and linearly interpolating the gradient to the sample position. This is the conventional method of calculating the gradient. In the frequency domain this can be viewed as first filtering with the frequency transform of the central difference operator, and then filtering with the frequency transform of the linear interpolation operator. This means that we can take the product of these transforms to form one effective filter which describes the effects of the standard method. We will call this resulting filter the effective central difference operator. These three filters are plotted in Figure 4. Compared to Figure 3 the effective central difference operator falls off even quicker at frequencies below π but has less aliasing energy in the stop band at frequencies above π.

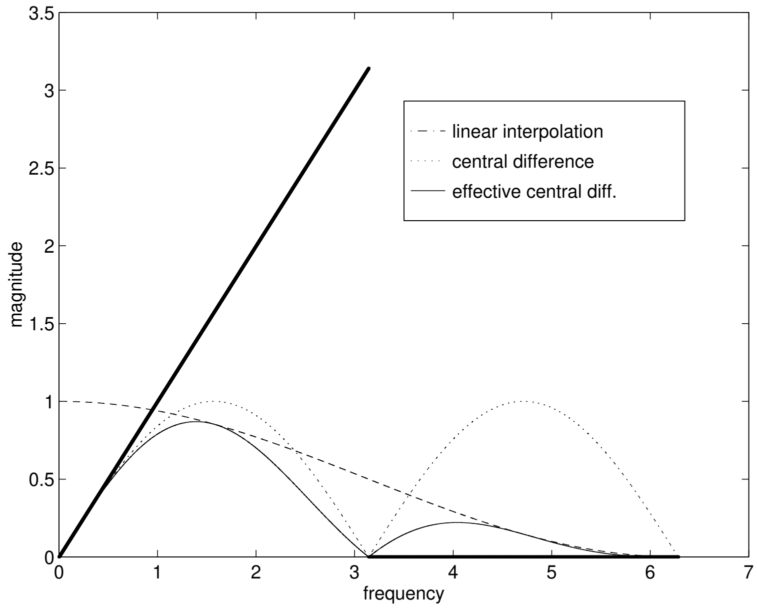

Figure 4: *Frequency response of central difference and linear interpolation and their product.*

6 Using the derivative of the Interpolation function

As described before, the gradient is the derivative of the continuous function $f(x, y, z)$. Since we apply an interpolation filter to the voxels to estimate this function, using the derivative of this interpolation filter will estimate the gradient on sample positions directly. This is opposed to the conventional method which calculates the gradient on a voxel point and then interpolates to the actual sample point.

6.1 Interpolation and Gradient Estimation with the Cubic Spline

In [9] [6] [10] and [1] it is shown that the cubic spline function performs very well for interpolation. In this section we will discuss the cubic spline filter and its application as a gradient estimator.

The general cubic spline is given by

$$\begin{array}{rcl} r(x) = & (a+2)\,|x|^3 - (a+3)\,|x|^2 + 1 & 0 \le |x| \le 1 \\ r(x) = & a\,|x|^3 - 5a\,|x|^2 + 8a\,|x| - 4a & 1 \le |x| \le 2 \\ r(x) = & 0 & |x| > 2 \end{array} \tag{16}$$

This can be rewritten as

$$r(x) = r_0(x) + ar_1(x) \tag{17}$$

Where

$$\begin{array}{rcl} r_0(x) = & 2|x|^3 - 3|x|^2 + 1 & 0 \leq |x| \leq 1 \\ r_0(x) = & 0 & otherwise \end{array} \tag{18}$$

$$\begin{array}{rcl} r_1(x) = & |x|^3 - |x|^2 & 0 \leq |x| \leq 1 \\ r_1(x) = & |x|^3 - 5|x|^2 + 8|x| - 4 & 1 \leq |x| \leq 2 \end{array} \tag{19}$$

This leaves the parameter a free to choose. a has physical significance. It is the slope of $r(x)$ at $|x| = 1$. In [6] is shown that for $a = -0.5$ the reconstructed function $g(x)$ matches the Taylor expansion of $f(x)$ exactly up till the third order term. If $a = -1$ then the slope at $|x| = 1$ is the same as the slope of the $sinc(x)$ at $|x| = 1$. Finally, $a = -0.75$ assures that the second derivative at $x = 1$ is continuous, which may be important if the derivative of the cubic spline is used as the gradient estimator.

The derivative of Equation 17 is

$$r'(x) = r'_0(x) + ar'_1(x) \tag{20}$$

$$\begin{array}{rcl} r'_0(x) = & 6|x|^2 - 6|x| & 0 \leq |x| \leq 1 \\ r'_0(x) = & 0 & otherwise \end{array} \tag{21}$$

$$\begin{array}{rcl} r'_1(x) = & 3|x|^2 - 2|x| & 0 \leq |x| \leq 1 \\ r'_1(x) = & 3|x|^2 - 10|x| + 8 & 1 \leq |x| \leq 2 \end{array} \tag{22}$$

The performance of this gradient filter is compared to other methods in the next section. In order to do so the Fourier Transform of Equation 20 is needed. It can be shown that the Fourier transform of Equation 17 is

$$R(\omega) = R_0(\omega) + aR_1(\omega) \tag{23}$$

Where the capital R stands for the Fourier Transform and

$$\begin{aligned} R_0(\omega) = & \quad \tfrac{12}{\omega^2}[sinc^2(\omega/2) - sinc(\omega)] \\ R_1(\omega) = & \tfrac{8}{\omega^2}[3sinc^2(\omega) - 2sinc(\omega) - sinc(2\omega)] \end{aligned} \tag{24}$$

The derivation of Equation 24 can be found in the appendix. Note that at the Nyquist or foldover frequency $\omega = \pi$ the magnitude of $R(\omega)$ is independent of a and is equal to $48/\pi^4$. The Fourier Transform of Equation 20 is $j\omega$ times Equation 23.

In Figure 5 a plot of $r(x)$ for different values of a is shown. The same Figure shows $r'(x)$ also for different values of a, while Figure 6 show the respective Fourier Transforms.

The bottom section of Figure 6 shows the frequency responses as a function of a. By varying this parameter the response to higher frequencies can be adjusted as desired and thus can be used as an adjustable gradient filter. All energy above the Nyquist frequency π is aliased back. This is due to imperfect interpolation with the cubic spline. See the top of Figure 6 for the frequency response of the cubic spline.

6.2 Evaluation of the cubic spline based gradient filter

While most of the gradient estimators show a good approximation of the ideal filter at lower frequencies, they show a rapid fall off at higher frequencies. See Figure 7. This figure shows several effective gradient estimators. Thus the effect of linear interpolation to the sample point is already included in the plots for the intermediate difference, central difference and adjustable filter. Since fine details, like bone fractures in medical images, contain a lot of high frequencies, some errors will occur. Although this is a disadvantage, fall off at higher frequencies has also some desirable properties. Volume data often contains a lot of noise and aliasing energy, especially for PET and SPECT data. This energy may be concentrated along the high frequencies. The gradient filter has its greatest sensitivity along these high frequencies. Attenuation of these high frequencies reduced artifacts caused by noise and aliasing. As always, a trade off has to be made. If it is possible to adjust the frequency response the user can control this trade off themself. In [5] an adjustable gradient filter is discussed. This filter is based on truncating the $cos(x)/x$ and windowing that truncated filter with a Kaiser window. By adjusting the Kaiser window parameter α the frequency response can be varied. Figure 7 shows a plot of this filter with $\alpha = 4$ as the Kaiser window parameter.

Note that the cubic spline (and its derivative) only needs 4 voxels to evaluate in the 1-D case, compared to 6 for the adjustable gradient filter proposed in [5]. Furthermore this adjustable

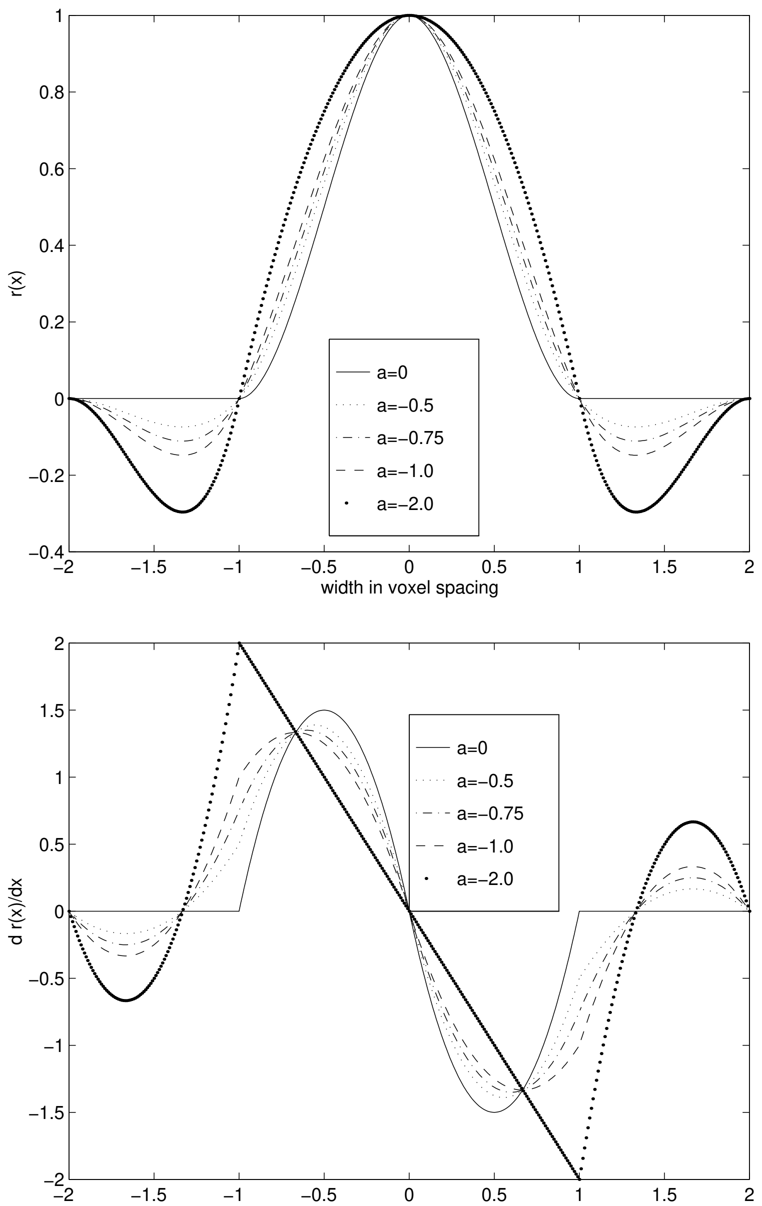

Figure 5: *The cubic spline (top) and its derivative (bottom) for different values of a.*

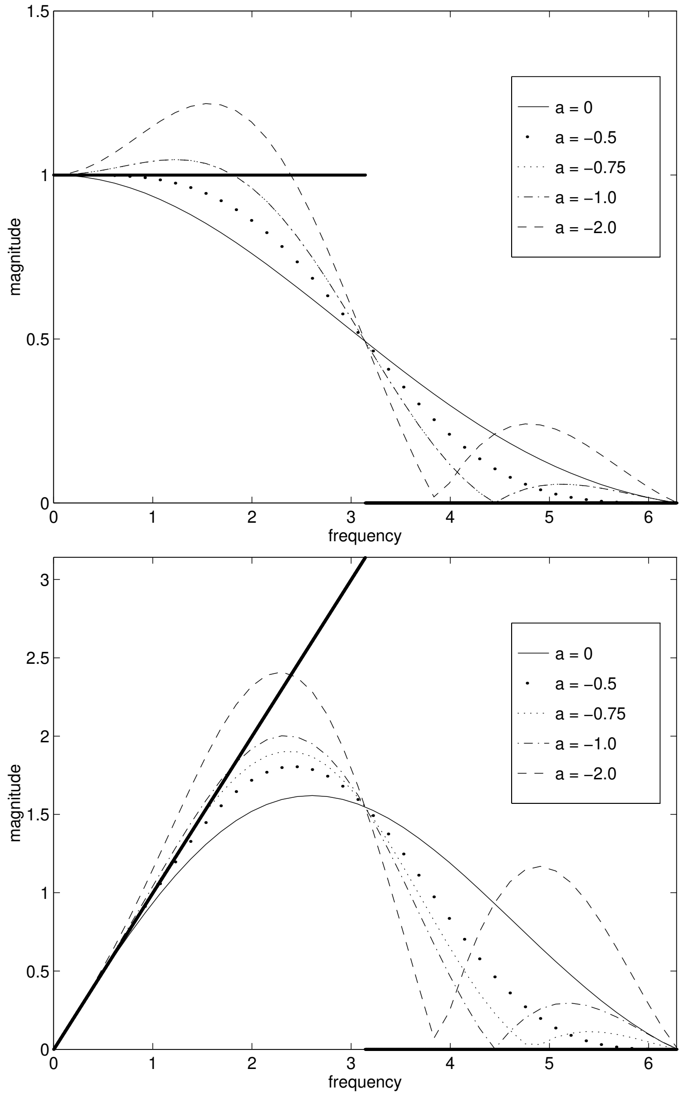

Figure 6: *The Fourier Transform of the cubic spline (top) and the Fourier Transform of the derivative (bottom) for different values of a. The thick lines denote the ideal cases*

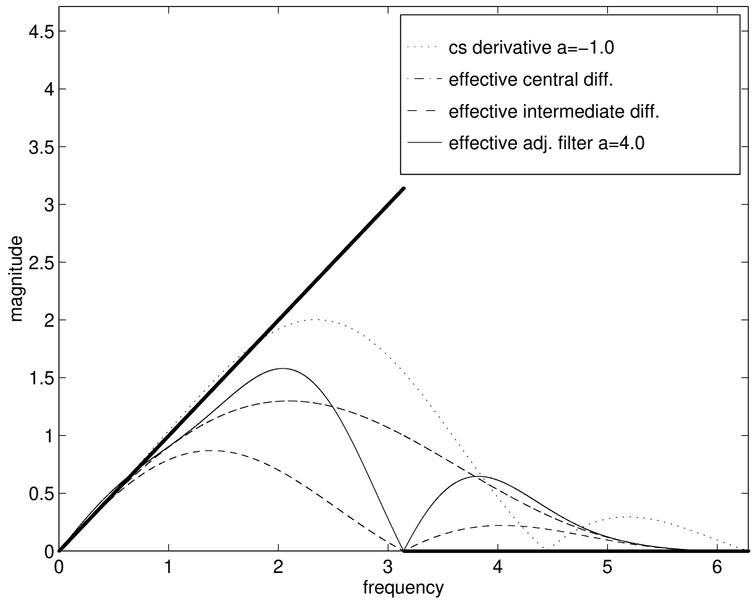

Figure 7: *Frequency responses of different methods to estimate the local gradient. The thick line is the ideal case.*

gradient filter calculates gradients on voxel positions, not on sample positions. In order to get the gradient on sample positions, some kind of interpolation has to be done between the gradients on voxels positions. The cubic spline with the a parameter set to $a = -1.0$ falls of at a higher frequency than the adjustable gradient filter but has about the same energy at frequencies higher than π. See Figure 7.

In Figure 8 a comparison is made between the cubic spline derivative filter for several values of a and the truncated version of the ideal gradient estimator on sample positions. All the filters have an extend of 4 in the spatial domain. Three cubic spline based filters are plotted, with a value of the a parameter of respectively -0.75, -1.5 and -2.0.

Figure 8 shows that the cubic spline derivate filter with $a = -2.0$ approximates a constant slope up to a slightly higher frequency than the truncated ideal filter, but it has much more energy above the foldover frequency π. The cubic spline derivate filter with $a = -1.5$ is nearly identical to the truncated ideal filter up to π. Above that it has somewhat more energy.

Knowing this, the truncated ideal gradient filter performs best, but it is not adaptive. If small features in the original data should be detected with more sensitivity, the cubic spline based method with $a = -2.0$ gives a better high frequency behavior. Of course the aliasing energy in this case is much higher than using the truncated ideal filter.

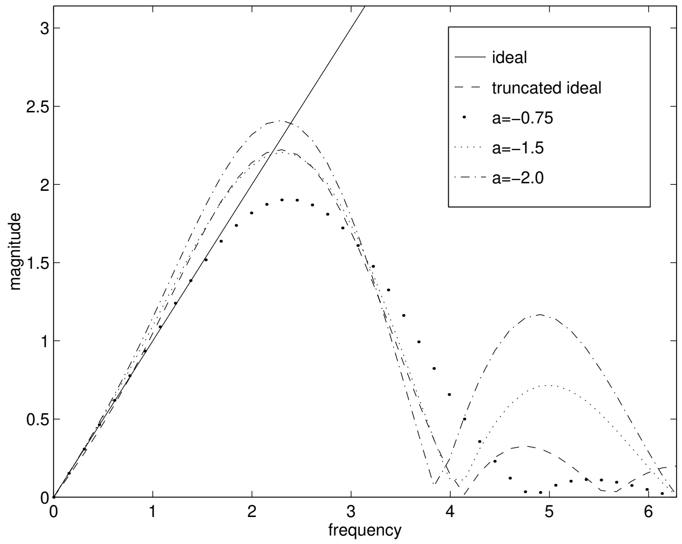

Figure 8: *Frequency responses of the ideal estimator truncated to 4 voxels and of the derivative of the cubic spline for different values of a.*

7 Implementing the cubic spline based gradient filter

This section will address two different ways of implementing the cubic spline based gradient filter in volume rendering. First the two dimensional case will be discussed. After that it will be extended to three dimensions.

In Figure 9 the two dimensional situation is shown, with P_i the sample position. $P_{x,y}$ are the known pixel (3D: voxel) values.

Using an interpolation kernel $h(x, y)$, P_i can be estimated as follows

$$
\begin{aligned}
&P_i(x,y) = \\
&\quad P_{-2,-2}h(x-2,y-2) + P_{-1,-2}h(x-1,y-2) + P_{0,-2}h(x,y-2) + P_{1,-2}h(x+1,y-2) + \\
&\quad P_{-2,-1}h(x-2,y-1) + P_{-1,-1}h(x-1,y-1) + P_{0,-1}h(x,y-1) + P_{1,-1}h(x+1,y-1) + \\
&\quad P_{-2,0}h(x-2,y) + P_{-1,0}h(x-1,y) + P_{0,0}h(x,y) + P_{1,0}h(x+1,y) + \\
&\quad P_{-2,1}h(x-2,y+1) + P_{-1,1}h(x-1,y+1) + P_{0,1}h(x,y+1) + P_{1,1}h(x+1,y+1) \qquad (25)
\end{aligned}
$$

This means that in order to interpolate P_i the pixel values are multiplied with a 2-D kernel of which the weights are determined by $h(x, y)$. One method to achieve this is to sample $h(x, y)$ and storing these values in a lookup table which reduces the interpolation to 16 multiplications and 15 additions. We call this non-seperable interpolation.

For seperable interpolation kernels $h(x, y)$ can be calculated as a product of two 1-dimensional

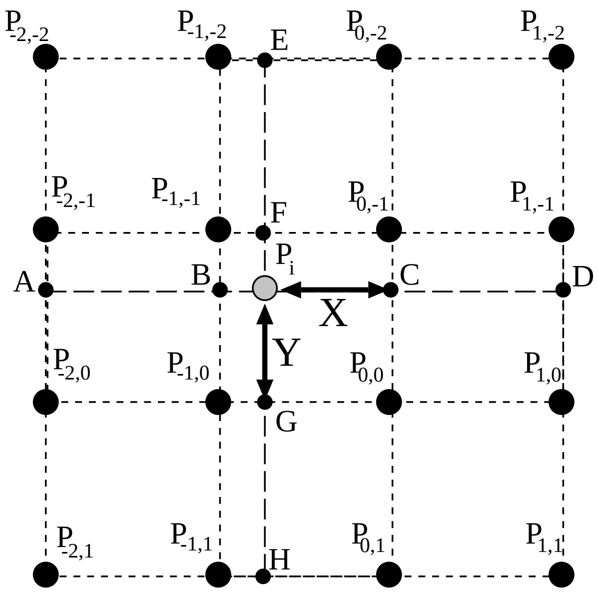

Figure 9: *Two dimensional interpolation situation. P_i is the sample position.*

interpolation kernels.

$$h(x, y) = h(x)h(y) \tag{26}$$

Now Equation 25 can be rewritten as

$$\begin{aligned} P_i(x, y) = & \\ & h(x-2)[P_{-2,-2}h(y-2) + P_{-2,-1}h(y-1) + P_{-2,0}h(y) + P_{-2,1}h(y+1)] + \\ & h(x-1)[P_{-1,-2}h(y-2) + P_{-1,-1}h(y-1) + P_{-1,0}h(y) + P_{-1,1}h(y+1)] + \\ & h(x)[P_{0,-2}h(y-2) + P_{0,-1}h(y-1) + P_{0,0}h(y) + P_{0,1}h(y+1)] + \\ & h(x+1)[P_{1,-2}h(y-2) + P_{1,-1}h(y-1) + P_{1,0}h(y) + P_{1,1}h(y+1)] \\ = \; & h(x-2)A + h'(x-1)B + h'(x)C + h'(x+1)D \end{aligned} \tag{27}$$

Where $A..D$ are interpolated values in the X-direction. Figure 9 shows this seperable interpolation method.

The computational expense of calculating the gradient on the sample position (x, y) depends on the way interpolation is done. In the non-seperable case it is not possible to use any results of the interpolation to speed up the gradient calculation. If a seperable filter is used, this is possible.

The gradient at the sample position (x, y) is the partial derivative in the x and y direction of $P_i(x, y)$

$$G_i(x,y) = [G_{ix}, G_{iy}] = [\frac{\delta}{\delta x} P_i(x,y), \frac{\delta}{\delta y} P_i(x,y)] \tag{28}$$

This leads to the following gradient in the x-direction

$$\begin{aligned}
G_{ix} = & \\
& h'(x-2)[P_{-2,-2}h(y-2) + P_{-2,-1}h(y-1) + P_{-2,0}h(y) + P_{-2,1}h(y+1)] + \\
& h'(x-1)[P_{-1,-2}h(y-2) + P_{-1,-1}h(y-1) + P_{-1,0}h(y) + P_{-1,1}h(y+1)] + \\
& h'(x)[P_{0,-2}h(y-2) + P_{0,-1}h(y-1) + P_{0,0}h(y) + P_{0,1}h(y+1)] + \\
& h'(x+1)[P_{1,-2}h(y-2) + P_{1,-1}h(y-1) + P_{1,0}h(y) + P_{1,1}h(y+1)] \\
= & \; h'(x-2)A + h'(x-1)B + h'(x)C + h'(x+1)D
\end{aligned} \tag{29}$$

Thus the x-component of the gradient can be calculated using the interpolated values A, B, C and D. If interpolation is done using a seperable filter these values already were computed. Thus calculating the x-component of the gradient only requires 4 multiplications and 3 additions. If on the other hand interpolation was done using a non-seperable filter, no savings would be achieved.

G_{iy} can be derived in the same way

$$G_{iy} = h'(y-2)E + h'(y-1)F + h'(y)G + h'(y+1)H \tag{30}$$

Unfortunately the values E, F, G and H are not available and have to be computed too.

The above can be easily extended to the 3-D case

$$\begin{aligned}
P_i(x,y,z) &= \sum_{i=-2}^{1} \sum_{j=-2}^{1} \sum_{k=-2}^{1} P_{i,j,k} h(x+i, y+j, z+l) \\
&= \sum_{i=-2}^{1} \sum_{j=-2}^{1} \sum_{k=-2}^{1} P_{i,j,k} h(x+i)h(y+j)h(z+k)
\end{aligned} \tag{31}$$

The local gradient at the sample position is

$$G_i(x,y,z) = [G_{ix}, G_{iy}, G_{iz}] = [\frac{\delta}{\delta x} P_i(x,y,z), \frac{\delta}{\delta y} P_i(x,y,z), \frac{\delta}{\delta z} P_i(x,y,z)] \tag{32}$$

With

$$
\begin{aligned}
G_{ix} = \quad & h'(x-2)\left(\sum_{j=-2}^{1}\sum_{k=-2}^{1} P_{-2,j,k}h(y+j)h(z+1)\right) \\
+ \quad & h'(x-1)\left(\sum_{j=-2}^{1}\sum_{k=-2}^{1} P_{-1,j,k}h(y+j)h(z+1)\right) \\
+ \quad & h'(x)\left(\sum_{j=-2}^{1}\sum_{k=-2}^{1} P_{0,j,k}h(y+j)h(z+1)\right) \\
+ \quad & h'(x+1)\left(\sum_{j=-2}^{1}\sum_{k=-2}^{1} P_{1,j,k}h(y+j)h(z+1)\right) \qquad (33)
\end{aligned}
$$

The sum-terms between the large brackets in this equation are interpolated values in the yz-plane. Thus the x-component of the 3-D gradient can again be calculated with very little extra computation, if interpolation is done taking advantage of the seperability of the interpolation filter.

For the gradient vector in the y- and z-direction the same kind of formulas can be derived.

7.1 Computational expense of the new gradient filters

Interpolation in 3-D with a filter of extend 4 using a non-seperable filter requires 64 multiplications and 63 additions. This assumes that the interpolation filter is sampled and that all possible values of $h(x, y, z)$ are stored in a lookup table. This can possibly be a very large lookuptable. Gradient estimation requires three times as much computation and a lookuptable to store the sampled derivative of the interpolation filter $h'(x, y, z)$ in. Thus interpolation and gradient estimation in total amounts to $4 \times 64 = 256$ multiplications and $4 \times 63 = 252$ additions.

A 3-D interpolation using a seperable filter with an extent of 4 points in one dimension requires $4 \times 5 + 1 = 21$ 1D interpolations[1], or convolutions with the interpolation filter $h(x)$. In that case the x-component of the gradient can be calculated with one 1-D convolution with the gradient filter. For the y and z components however, a totally new interpolation scheme must be applied to obtain the 4 interpolated points, necessary to calculate the y and z gradient components. That means that a total of $21 + 21 + 1 = 43$ convolutions[2]

[1] A 4×4 2-D interpolation requires 5 1-D interpolations. See Figure 9.

[2] One convolution in the x direction because values calculated in the interpolation stage can be reused to calculate the x component of the gradient. Twice 21 convolutions because for the other two directions this is not the case.

are needed to obtain the complete gradient vector. Thus the calculation of the gradient vector is about twice as expensive as the interpolation itself. A 1-D convolution requires 4 multiplications and 3 additions. Thus interpolation and gradient estimation together amounts to $4 \times (21 + 43) = 256$ multiplications and $3 \times (21 + 43) = 192$ additions. This assumes however that the interpolation function $h(x)$ and its derivative $h'(x)$ are sampled and the values stored in a lookup table. Note that this lookup table will be much smaller than the one needed if a non-seperable filter is used.

It is slightly cheaper to use a seperable filter to do interpolation and gradient estimation. Note that no extra memory fetches are required to calculate the gradient. The gradient vector calculation uses the same memory addresses as the interpolation unit.

8 Results

Figure 10 and Figure 11 present two renderings using various gradient estimators. Figure 10(a) shows a volume rendering of a dataset from the University of North Carolina Volume Rendering Test Dataset Volume II. The dataset is a $256 \times 256 \times 109$ voxel magnetic resonance image of a head with the brain surface exposed. Figure 10(b),(c) and (d) show closeups of the center of Figure 10(a) but rendered using different gradient estimators. In Figure 11 three rendered images are shown of a solid cone with grooves in it.

Figure 10(b) and Figure 11(a) show the standard gradient and opacity method in which the opacity and color is calculated on grid points and interpolated to estimate the color and opacity on sample positions. Gradient estimation was done using the central difference method. Figure 10(c) and Figure 11(b) are rendered using the intermediate difference method to estimate the gradient. Linear interpolation is used to estimate the gradient on sample positions. Figure 10(d) and Figure 11(c) images are rendered using the gradient method we propose, the derivative of the cubic spline interpolation function.

As can be seen in both figures there is a substantial difference between the image quality with precalculated color and opacity and direct gradient estimation. The differences between the intermediate difference method and the cubic spline based gradient method are smaller in Figure 11. For detecting fine features the cubic spline method performs slightly better. The differences are significant in Figure 10. Figure 10(d) shows the most details because high frequencies are better preserved. These figures show that the choice of a gradient estimator depends on the dataset and the application.

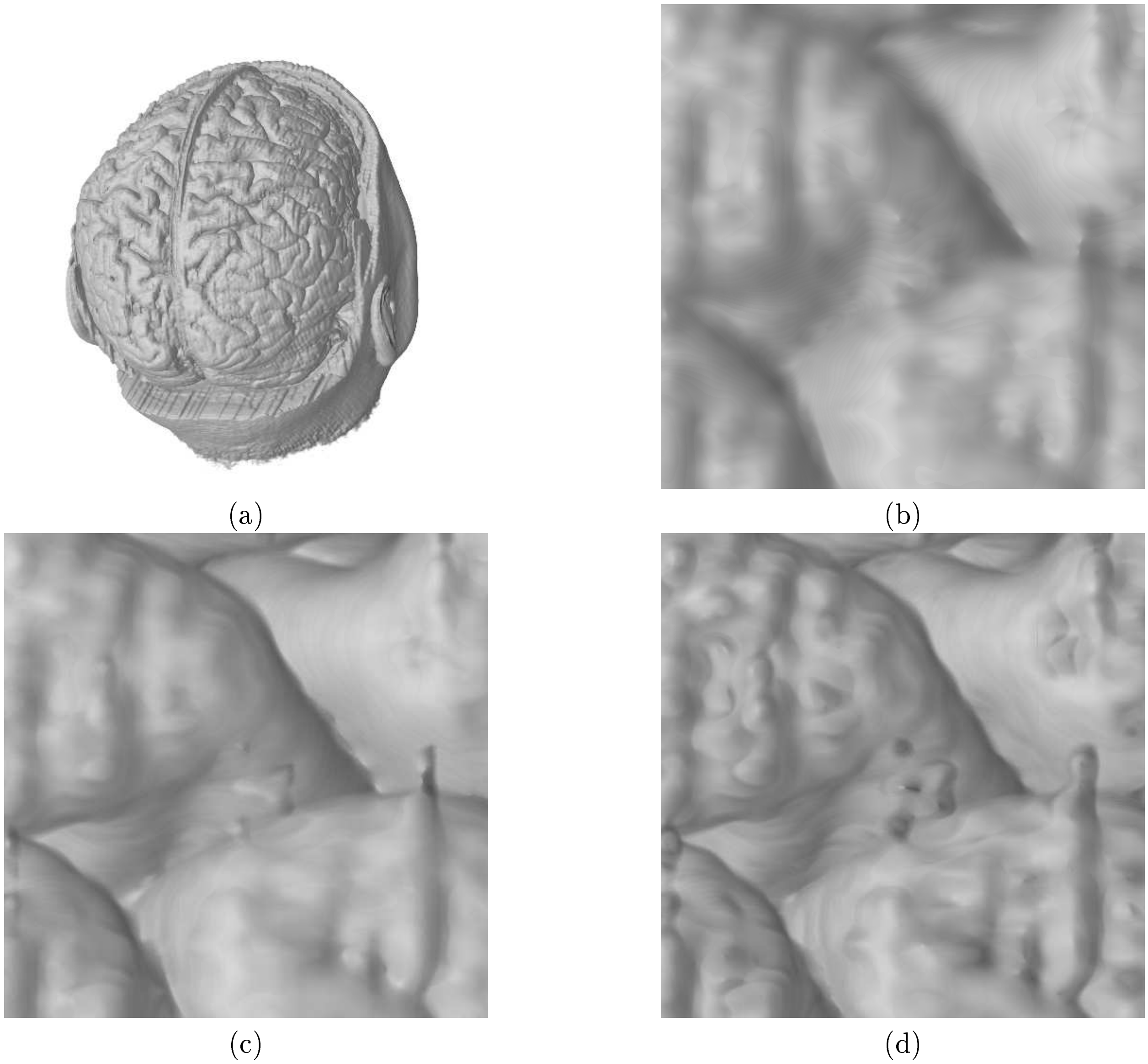

Figure 10: *Difference between gradient filters. (a) Original dataset. (b) Zoomed in on the center using precalculated color and opacity on grid positions (c) Zoomed in on the center using the intermediate difference method (d) Zoomed in on the center using the cubic spline based gradient method with a=-0.5.*

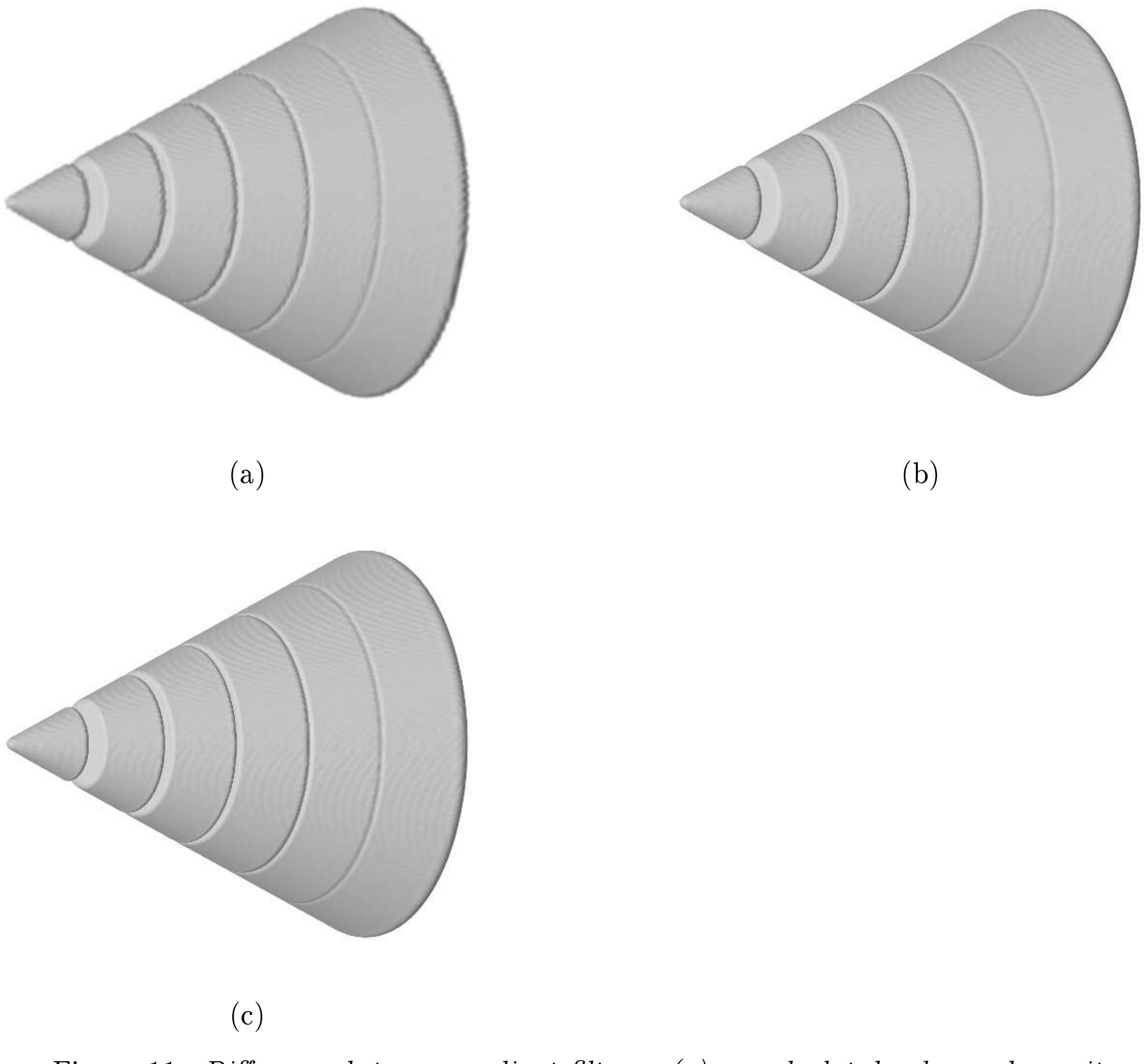

Figure 11: *Difference between gradient filters. (a) precalculated color and opacity on grid positions (b) Intermediate difference method (c) Cubic spline based gradient method with a=-0.5.*

9 Discussion

In this paper a few methods are presented to estimate the gradient in a three dimensional dataset. The most commonly used gradient filter, the central difference, is fixed and can not be tuned for optimal balance between fine details on one hand and aliasing and noise rejection on the other hand. The adjustable filter techniques of Goss [5] and the derivative cubic spline filter do have this feature. Both filters also have a better high frequency behavior and less energy in the frequency range above π compared to the central difference and the intermediate difference methods.

It would be possible to make the filter choice adaptive. If several lookup tables are used, one for each different gradient filter, choosing between filters means switching between lookup tables or changing the contents of a lookup table. This leads to a gradient estimator based on local information around the sample point being processed at that moment.

10 Acknowledgments

We like to thank all the people who helped in this research. We especially like to thank Irwin Sobel and Ron Schafer for their comments and suggestions.

The investigations were partly supported by the foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organization for Scientific Research (NWO) and the Visual Computing Department of Hewlett Packard Laboratories.

APPENDIX Fourier Transform of the Cubic Spline

This appendix derives the Fourier Transform of the cubic spline function given in Equation 17 analytically.

For convenience the cubic spline equation is repeated here

$$r(x) = r_0(x) + ar_1(x) \tag{34}$$

Where

$$\begin{aligned} r_0(x) = &\quad 2|x|^3 - 3|x|^2 + 1 \quad 0 \le |x| \le 1 \\ r_0(x) = &\quad 0 \quad otherwise \end{aligned} \tag{35}$$

$$\begin{aligned} r_1(x) = &\quad |x|^3 - |x|^2 \quad 0 \le |x| \le 1 \\ r_1(x) = &\quad |x|^3 - 5|x|^2 + 8|x| - 4 \quad 1 \le |x| \le 2 \end{aligned} \tag{36}$$

The Fourier transform of $r(x)$ can be calculated using $r_0(x)$ and $r_1(x)$ and by using the linear property of the Fourier transform

$$\alpha f(x) + \beta g(x) \Longleftrightarrow \alpha F(\omega) + \beta G(\omega) \tag{37}$$

Then the Fourier transform of $R(\omega)$ is

$$R(\omega) = R_0(\omega) + aR_1(\omega) \tag{38}$$

First the Fourier Transform of $r_0(x)$ will be derived. It is easiest to use the Laplace Transform as a means to derive the Fourier Transform. Let $p(x)$ be

$$\begin{aligned} p(x) = &\quad 1 \quad 0 \le x \le 1 \\ p(x) = &\quad 0 \quad otherwise \end{aligned} \tag{39}$$

and

$$\begin{aligned} h_0(x) = &\quad 2x^3 - 3x^2 + 1 \quad 0 \le x \le 1 \\ h_0(x) = &\quad 0 \quad otherwise \end{aligned} \tag{40}$$

Now define $g(x)$ as

$$g(x) = p(x)h(x) = p(x)\{2x^3 - 3x^2 + 1\} \tag{41}$$

then

$$r_0(x) = g(x) + g(-x) \tag{42}$$

and

$$R_0(\omega) = G(\omega) + G(-\omega) \tag{43}$$

The Laplace Transform of Equation 41 can be computed by using the derivative theorem of the Laplace Transform. If

$$P(s) = \int_{-\infty}^{\infty} p(x)e^{-sx}dx \tag{44}$$

Then

$$\frac{dP(s)}{ds} = \int_{-\infty}^{\infty} -xp(x)e^{-sx}dx \tag{45}$$

Or in words, $P'(s)$ and $-xp(x)$ are a Laplace Transform pair. The same reasoning holds for the second and higher order derivatives.

$$\begin{aligned} -xp(x) &\iff P'(s) \\ x^2p(x) &\iff P''(s) \\ -x^3p(x) &\iff P'''(s) \end{aligned} \tag{46}$$

Where $\iff$ denotes a Laplace Transform pair. Thus the Laplace Transform of Equation 41 has the following form

$$G(s) = -2P'''(s) - 3P''(s) + P(s) \tag{47}$$

$P(s)$ is the Laplace Transform of $p(x)$ and is

$$P(s) = \int_0^1 e^{-sx}dx = \frac{1 - e^{-s}}{s} \tag{48}$$

Now

$$
\begin{aligned}
P'(s) &= \frac{se^{-s} - 1 + e^{-s}}{s^2} \\
P''(s) &= \frac{-s^2e^{-s} - 2se^{-s} + 2 - 2e^{-s}}{s^3} \\
P'''(s) &= \frac{s^3e^{-s} + 3s^2e^{-s} + 6se^{-s} - 6 + 6e^{-s}}{s^4}
\end{aligned} \tag{49}
$$

Thus

$$
\begin{aligned}
G(s) &= -2P'''(s) - 3P''(s) + P(s) \\
&= \frac{1}{s^3}(-6 - 6e^{-s}) + \frac{1}{s^4}(12 - 12e^{-s})
\end{aligned} \tag{50}
$$

Then

$$
\begin{aligned}
R_0(\omega) &= G(\omega) + G(-\omega) \\
&= \frac{6}{j^3\omega^3}(e^{j\omega} - e^{-j\omega}) - \frac{12}{j^4\omega^4}(e^{j\omega} + e^{-j\omega} - 2) \\
&= \frac{12}{\omega^2}\left(\mathrm{sinc}^2\left(\frac{\omega}{2}\right) - \mathrm{sinc}(\omega)\right)
\end{aligned} \tag{51}
$$

Next the Fourier Transform of $r_1(x)$ will be derived.

$$
\begin{array}{rcl}
r_1(x) = & |x|^3 - |x|^2 & 0 \leq |x| \leq 1 \\
r_1(x) = & |x|^3 - 5|x|^2 + 8|x| - 4 & 1 \leq |x| \leq 2
\end{array} \tag{52}
$$

Again the function $p(x)$ is used and also the function $q(x)$

$$
\begin{array}{rcl}
q(x) = & 1 & 1 \leq x \leq 2 \\
q(x) = & 0 & otherwise
\end{array} \tag{53}
$$

$g(x)$ is now defined as

$$
g(x) = p(x)(x^3 - x^2) + q(x)(x^3 - 5x^2 + 8x - 4) \tag{54}
$$

Then

$$r_1(x) = g(x) + g(-x) \tag{55}$$

and

$$R_1(\omega) = G(\omega) + G(-\omega) \tag{56}$$

Using the same strategy, $G(\omega)$ can be calculated as follows

$$G(s) = -P'''(s) - P''(s) - Q'''(s) - 5Q''(s) - 8Q'(s) - 4Q(s) \tag{57}$$

$Q(s)$ can be calculated using the Laplace transform

$$Q(s) = \int_{-\infty}^{\infty} q(x)e^{-sx}dx = \int_{1}^{2} e^{-sx}dx = \frac{e^{-s} - e^{-2s}}{s} = e^{-s}P(s) \tag{58}$$

This results in the following derivatives of $Q(s)$

$$\begin{aligned} Q'(s) &= e^{-s}P'(s) - e^{-s}P(s) \\ Q''(s) &= e^{-s}P''(s) - 2e^{-s}P'(s) + e^{-s}P(s) \\ Q'''(s) &= e^{-s}P'''(s) - 3e^{-s}P''(s) + 3e^{-s}P'(s) - e^{-s}P(s) \end{aligned} \tag{59}$$

Substitution in Equation 57 yields

$$\begin{aligned} -G(s) &= P'''(s)[1 + e^{-s}] + P''(s)[1 + 2e^{-s}] + P'(s)[e^{-s}] \\ &= \frac{1}{s^3}\left(2 + 8e^{-s} + 2e^{-2s}\right) + \frac{1}{s^4}\left(6e^{-2s} - 6\right) \end{aligned} \tag{60}$$

Now $R_1(\omega)$ is

$$\begin{aligned} R_1(\omega) &= G(\omega) + G(-\omega) \\ &= \frac{8}{j^3\omega^3}\left(e^{j\omega} - e^{-j\omega}\right) + \frac{2}{j^3\omega^3}\left(e^{2j\omega} - e^{-2j\omega}\right) - \frac{6}{j^4\omega^4}\left(e^{2j\omega} + e^{-2j\omega} - 2\right) \\ &= \frac{8}{\omega^2}\left(3\mathrm{sinc}^2(\omega) - 2\mathrm{sinc}(\omega) - \mathrm{sinc}(2\omega)\right) \end{aligned} \tag{61}$$

Finally the Fourier transform of the cubic spline interpolation function is given by

$$\begin{aligned}
R(\omega) &= \frac{12}{\omega^2}\left(\mathrm{sinc}^2\left(\frac{\omega}{2}\right) - \mathrm{sinc}(\omega)\right) \\
&\quad + a\frac{8}{\omega^2}\left(3\mathrm{sinc}^2(\omega) - 2\mathrm{sinc}(\omega) - \mathrm{sinc}(2\omega)\right) \qquad (62)
\end{aligned}$$

Bibliography

[1] M.J. Bentum, M.A. Boer, A.G.J. Nijmeijer, M.M. Samsom, and C.H. Slump. Resampling of Images in Real-Time. In *Proceedings of the IEEE ProRISC workshop on Circuit, Systems and Signal Processing*, volume March, pages 21–26, 1994.

[2] M. Bosma, J. Smit, and J. Terwisscha van Scheltinga. Super Resolution Volume Rendering Hardware. In *Proceedings of the Tenth Workshop on Graphics Hardware*, volume EG95 HW, page . EuroGraphics Technical Report Series, 1995.

[3] Ronald N. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, 1986.

[4] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. *Computer Graphics*, 22(4):65–74, August 1988.

[5] M.E. Goss. An Adjustable Gradient Filter for Volume Visualization Image Enhancement. In *Proceedings Graphics Interface '94*, volume , pages 67–74. Canadian Inf. Process. Soc, Toronto, Ont., Canada, 1994.

[6] R.G. Keys. Cubic Convolution Interpolation for Digital Image Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(6):1153–1160, December 1981.

[7] M.S. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, pages 29–37, May 1988.

[8] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[9] S.K. Park and R.A. Schowengerdt. Image Reconstruction by Parametric Cubic Convolution. *Computer Vision, Graphics, and Image Processing*, 23:258–272, 1983.

[10] J.A. Parker, R.V. Kenyon, and D.E. Troxel. Comparison of Interpolating Methods for Image Resampling. *IEEE Transactions on Medical Imaging*, 2(1):31–39, March 1983.

[11] B.T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[12] A. Pommert, U. Tiede, G. Wiebecke, and K.H. Hohne. Surface Shading in Tomographic Volume Visualization. In *Proceedings of the First Conference on Visualization in Biomedical Computing*, volume 1, pages 19–26. IEEE Comput. Soc. Press, 1990.

[13] L.R. Rabiner and R.W. Schafer. On The Behavior Of Minimax Relative Error FIR Digital Differentiators. *The Bell System Technical Journal*, 53(2):333–361, Februari 1974.

[14] P. Sabella. A Rendering Algorithm for Visualizing 3D Scalar Fields. *Computer Graphics*, 22(4):51–58, August 1988.

[15] C.E. Shannon. Communication in the Process of Noise. *Proceedings of the IRE*, 37( ):10–21, January 1949.

[16] C. Upson and M. Keeler. V-Buffer: Vissible Volume Rendering. *Computer Graphics*, 22(4):59–64, August 1988.

[17] D. Vandermeulen, D. Delaere, P. Suetens, H. Bosmans, and G. Marchal. Local filtering and Global optimisation Methods for 3D Magnetic Resonance Angiography (MRA) Image enhancement. In *SPIE Visualization in Biomedical Computing*, volume 1808, pages 274–288, October 1992.

[18] L. Westover. Footprints Evaluation for Volume Rendering. *Computer Graphics*, 24(4):367–376, August 1990.

[19] Lee Westover. Interactive volume rendering. In *Chapell Hill Workshop on Volume Visualization*, pages 9–16, May 1989.

[20] R. Yagel, D. Cohen, and A. Kaufman. Normal Estimation In 3d Discrete Space. *The Visual Computer*, 8(5-6):278–291, June 1992.

[21] S.W. Zucker and R.A. Hummel. A Three-Dimensional Edge Operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(3):324–331, May 1981.