

# Frequency analysis of gradient estimators in volume rendering

Mark J. Bantum, Barthold B.A. Lichtenbelt and Thomas Malzbender

## Abstract

Gradient information is used in volume rendering to classify and color samples along a ray. In this paper we present an analysis of the theoretically ideal gradient estimator and compare it to some commonly used gradient estimators. A new method is presented to calculate the gradient at arbitrary sample positions, 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. Comparisons with several other methods are demonstrated. Computational efficiency can be realized since parts of the interpolation computation can be leveraged in the gradient estimation.

## Keywords

Volume rendering, volume visualization, gradient filters

## I. INTRODUCTION

The computation of dataset gradients is an essential operation in many visualization techniques. Visualizing a given three dimensional dataset can be done by surface rendering algorithms, such as the Marching Cubes Algorithm [24] [9] or by direct volume rendering algorithms, eg. raycasting [8] or splatting [21]. For direct volume rendering methods the voxel intensity, gradient direction and magnitude are often used to shade and classify the dataset. For surface rendering techniques the gradient is used as an estimation of the surface normal which is used for shading.

Gradient operators are also often used during the classification of data as well. The classification procedure provides an optical density value for each voxel in the dataset, called opacity. Opacities are typically calculated using either voxel intensities or a combination of voxel intensities and gradient information.

Once the dataset is classified and opacity and color is calculated, it is rendered. Color and opacity values are composited to achieve the final two dimensional projection. Several rendering techniques are known. An important distinction between the techniques is the order in which the voxels are processed to create an image, image-order vs object-order algorithms. Examples of image-order algorithms are raycasting [8] and Sabella's method [17]. Examples of object-order algorithms are the splatting algorithm [22], V-buffer algorithm [20] and the Slice Shearing algorithm [5]. An additional class of volume rendering algorithms are those employing frequency domain techniques to achieve fast rendering speeds [10] [19].

It is possible to classify and shade as a preprocessing step, yielding two new voxel datasets, the opacity and color datasets. During the rendering stage the color and opacity at sample points (generally not at 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 at the sample positions. As we will see this has consequences for the complexity of the algorithm.

This paper analyses some commonly used gradient operators, and introduces a new method based on taking the derivative of the interpolation function itself. Specifically, in section two ideal interpolation will be briefly discussed. Section three discusses the notion of ideal 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. In section six gradient estimation and interpolation are combined. Section seven 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 eight evaluates the proposed cubic spline based gradient filter. Section nine analyzes the performance of several gradient filters at different offsets between voxels. Section ten discusses the implementation of the cubic spline based filter in volume rendering. Section eleven discusses the computational expense of this filter. Section twelve gives some results and section thirteen discusses the proposed methods.

## II. IDEAL INTERPOLATION

The process of interpolation is a fundamental operation in digital signal processing and computer graphics. Its purpose 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 sample positions along rays, since it is unlikely these points will be positioned on gridpoints.

According to the sampling theorem [18] 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

$$r(x)_{ideal} = \text{sinc}(x) = \frac{\sin x}{x} \quad (1)$$

Since the magnitude of the *sinc* function is a pulse in the frequency domain, it is able to remove all replicates of the frequency spectrum introduced by sampling the original continuous function.

In computer graphics one has a set of discrete points, an image or volume. This set of points is often obtained by sampling the continuous function  $f(x, y, z)$  in the three dimensional case. Often the discrete dataset does not exactly 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 the set of discrete points of course still can be manipulated and interpolated. The *sinc* is still the ideal reconstruction filter, but it will reconstruct a continuous function  $f'(x, y, z)$  that is (slightly) different from the one originally sampled.

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],[7],[13],[14].

Throughout this paper we assume interpolation filters that will exactly reproduce the original voxel values if resampling is done on the voxels themselves. In [12] the more general family of cubic filters defined by piecewise cubic polynomials is discussed. Although these filters are useful for function approximation, in general the continuous function that they reconstruct do not pass through the given data points. We restrict our analysis to interpolating filters.

### III. IDEAL GRADIENT ESTIMATION

The gradient of a 3-D intensity function is the partial derivative of that intensity function with respect to all three directions. Given a function  $f(x, y, z)$  the gradient is:

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

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 is calculated using these voxels.

Gradient estimation can be analyzed in a manner similar to 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 interpolates  $f(x, y, z)$  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* function will reconstruct that function, the gradient can be reconstructed exactly by using the derivative of the *sinc* function as a reconstruction kernel.

In one dimension the ideal gradient reconstruction filter is:

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

This result is consistent with the results found in [16].

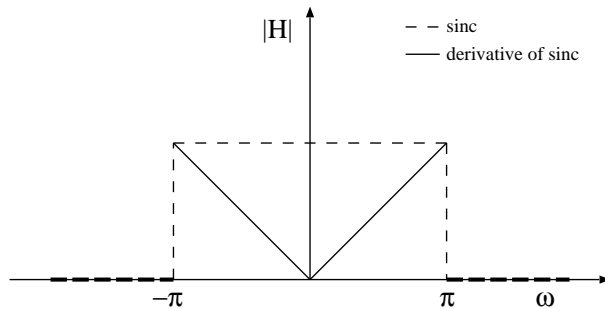


Fig. 1. Frequency response of the ideal gradient estimator and of the sinc function.

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 pulse in the interval  $-\pi \leq \omega \leq \pi$ . 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 spectral magnitude with constant slope in the frequency domain for the ideal

gradient filter in the interval  $0 \leq \omega \leq \pi$ . See Fig. 1. Not shown in Fig. 1 is the constant phase shift introduced by multiplying by the derivative operator  $j\omega$ .

#### IV. 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:

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

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 [6] and [15]. The Zucker-Hummel 3-D edge operator [26] 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 [15]. It selects between 3-6 adjacent 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 uses the difference between the voxel itself and the neighbor with the lower (higher) gray value; otherwise it uses the difference between both neighbors. The same method is used for the  $y$  and  $z$  components of the gradient vector. This method can be generalized to all 26 neighbors and is called adaptive gray-level gradient estimation.

Another adaptive method is proposed in [25]. 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:

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

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

Where  $g(x)$  denotes the gradient at position  $x$ . Using the intermediate difference method, gradients are calculated in between voxel positions:

$$g(-0.5) = f(0) - f(-1) \quad (7)$$

$$g(0.5) = f(1) - f(0) \quad (8)$$

If the gradient is required on position  $x$  with  $x$  between 0 and 0.5, the approximation with the central difference method equals:

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

Here we used linear interpolation to get the gradient on position  $x$ . The intermediate difference method results in:

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

Note that on the voxel positions these methods yield identical results!

Goss [6] proposes a gradient operator of which the frequency response can be varied, thus allowing the user control over the amount of smoothing introduced in the final image. This filter is discussed in section VIII.

## V. 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:

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

Since gradient operators are typically non-causal, we follow Bracewell [3] in using a version of the DFT centered at zero.

The ideal gradient filter magnitude has constant slope between  $-\pi$  and  $\pi$ , as shown earlier. First we will look at truncating the ideal gradient filter given in equation 3. In Fig. 2 the effects in the frequency domain can be seen of using such a filter truncated to a width of 4, 6 and 10. These plots have been numerically generated using the DFT.

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

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

is given by:

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

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

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

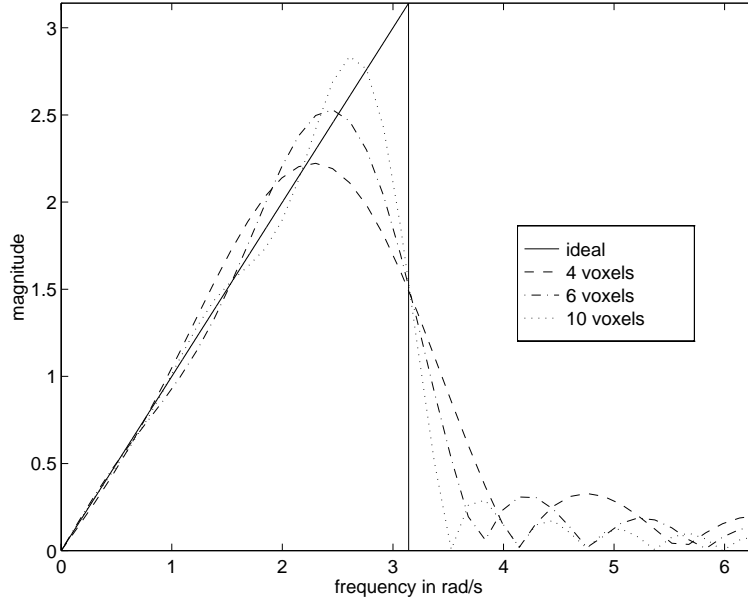


Fig. 2. *Frequency response of the truncated ideal gradient filter.*

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

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

There are two important differences between the DFT of the central difference method and the intermediate difference method. The amplitude of the DFT of the intermediate difference method is twice as high and the period is twice as low. This makes the intermediate difference a better gradient filter in the passband, but worse outside. See Fig. 3.

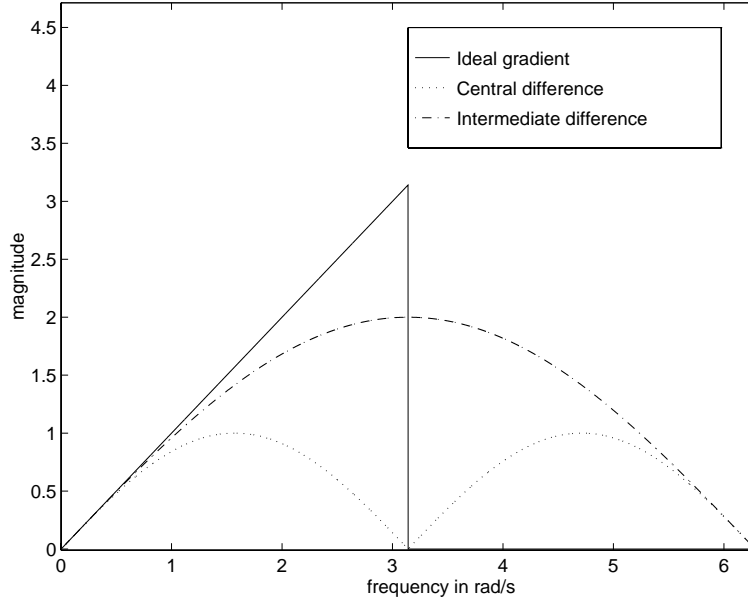


Fig. 3. *Frequency response of the central difference and intermediate difference estimator.*

## VI. COMBINING GRADIENT ESTIMATION AND INTERPOLATION

Frequencies above the cutoff frequency  $\pi$  in Fig. 2 and Fig. 3 are aliased back since the interpolation filter, used to calculate the gradient 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  $\pi$  since these would be filtered out by the interpolation.

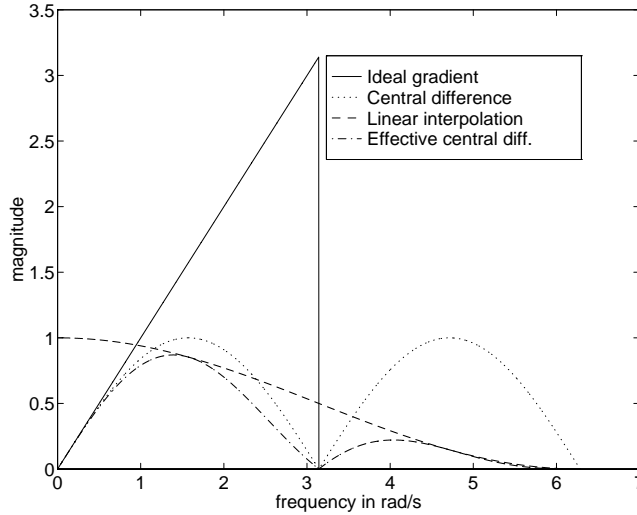


Fig. 4. Frequency response of central difference and linear interpolation and their product.

Fig. 4 shows the effect of estimating the gradient on a sample position by 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 Fig. 4. Compared to Fig. 3 the effective central difference operator falls off even quicker at frequencies below  $\pi$  but has less aliasing energy in the stop band at frequencies above  $\pi$ .

## VII. USING THE DERIVATIVE OF THE INTERPOLATION FUNCTION

As described before, the gradient is the first derivative of the continuous function  $f(x, y, z)$ .  $f(x, y, z)$  itself is gotten by convolving our sampled dataset with some interpolation function. One method of computing the gradient of the continuous function is to apply the gradient operator to the interpolation function itself and then convolve the dataset with the resultant continuous gradient operator. An example of such a procedure from the image processing literature is the use of the Laplacian of Gaussian (L.O.G.) edge operator which results from taking a second derivative of a gaussian convolution kernel [11]. This approach is valid due to linearity of both the interpolation and differentiation operator. This is contrasted to the more conventional volume rendering approach which evaluates gradients only at voxel positions, and may then interpolate these down to the sample points.

In [13] [7] [14] and [1] it is shown that the cubic spline<sup>1</sup> interpolation function defined in Equation 16 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:

$$r(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| \leq 2 \\ 0 & |x| > 2 \end{cases} \quad (16)$$

<sup>1</sup>In [13] and [7] the authors use the term cubic convolution. In [14] and [1] the authors use the term cubic spline. They all however are referring to the same function as defined in Equation 16. We adopted the latter naming convention.

This can be rewritten as:

$$r(x) = r_0(x) + ar_1(x) \quad (17)$$

Where:

$$r_0(x) = \begin{cases} 2|x|^3 - 3|x|^2 + 1 & 0 \leq |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

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

This leaves the parameter  $a$  free to be chosen. The parameter  $a$  is the slope of  $r(x)$  at  $x = 1$ . In [7] it 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. This case is also known as the Catmull-Rom spline [4] [12] and is often used for interpolation. If  $a = -1$  then the slope at  $x = 1$  is the same as the slope of the  $\text{sinc}(x)$  at  $x = 1$ .  $a = -0.75$  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. Finally, if  $a = 0$ , the spline function reduces to a 2-point interpolation function.

The derivative of Equation 17 is:

$$r'(x) = r'_0(x) + ar'_1(x) \quad (20)$$

with:

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

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

The performance of this gradient filter will be 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:

$$R(j\omega) = R_0(j\omega) + aR_1(j\omega) \quad (23)$$

Where the capital  $R$  stands for the Fourier transform of the cubic spline interpolation function. Now:

$$R_0(j\omega) = \frac{12}{\omega^2} [\text{sinc}^2(\omega/2) - \text{sinc}(\omega)] \quad (24)$$

$$R_1(j\omega) = \frac{8}{\omega^2} [3\text{sinc}^2(\omega) - 2\text{sinc}(\omega) - \text{sinc}(2\omega)] \quad (25)$$

The derivation of  $R_0(j\omega)$  and  $R_1(j\omega)$  can be found in the appendix. Note that at the Nyquist or fold over frequency  $\omega = \pi$  the magnitude of  $R(j\omega)$  is independent of  $a$  and is equal to  $48/\pi^4$ . The Fourier transform of the derivative of the cubic spline (Equation 20) is  $j\omega$  times Equation 23.

In Fig. 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 Fig. 6 shows the respective Fourier transforms.

Fig. 6 shows the frequency responses as a function of  $a$ . By varying this parameter the response to higher frequencies can be adjusted as desired. In Fig. 6 bottom all energy above the Nyquist frequency  $\pi$  is aliased back. This is due to imperfect interpolation with the cubic spline. See the top of Fig. 6 for the frequency response of the cubic spline.

## VIII. EVALUATION OF GRADIENT FILTERS

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 Fig. 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 [6]. Since fine details, like bone fractures in

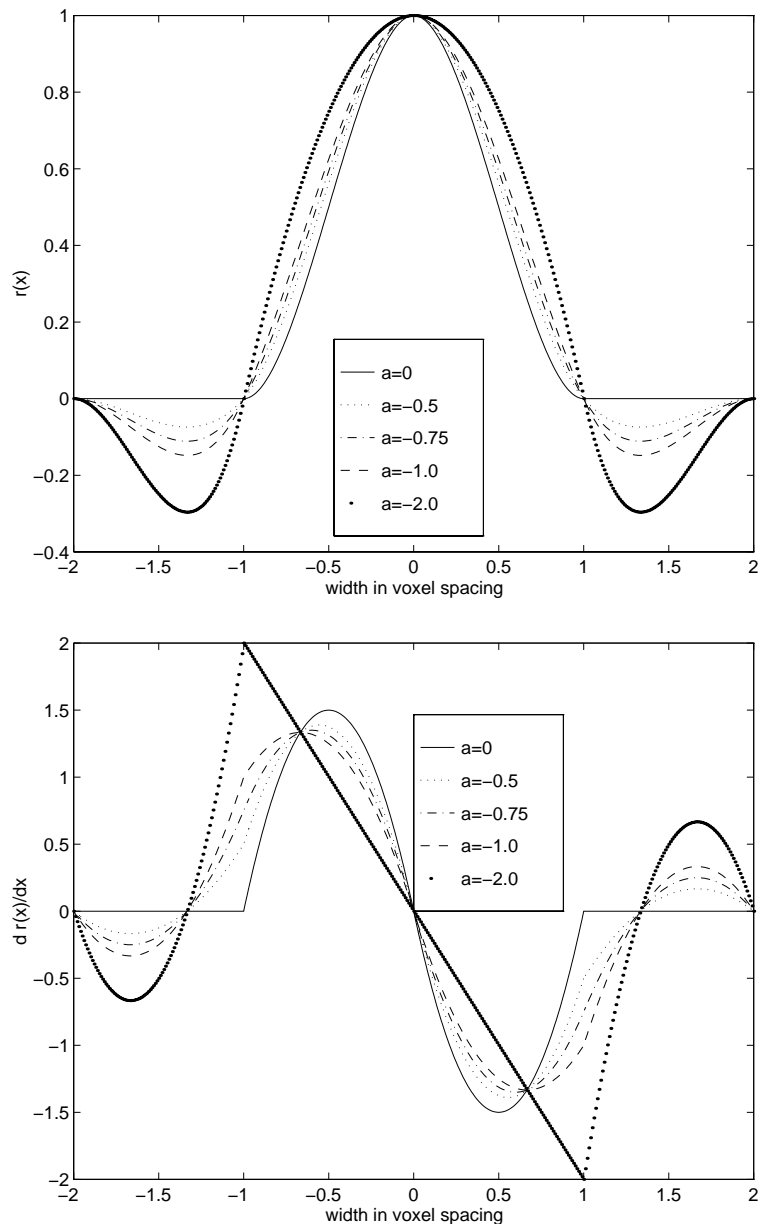


Fig. 5. The cubic spline (top) and its derivative (bottom) for different values of  $a$ .

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 unwanted noise and energy in the stop band, 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 reduces 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. In [6] an adjustable gradient filter is discussed. This filter is based on truncating  $\cos(x)/x$  and windowing that truncated filter with a Kaiser window. The Kaiser window of  $N$  samples is defined as:

$$w_{\alpha}(n) = \frac{I_0(\beta)}{I_0(\alpha)} \quad -\frac{N}{2} \leq n \leq \frac{N}{2} \quad (26)$$

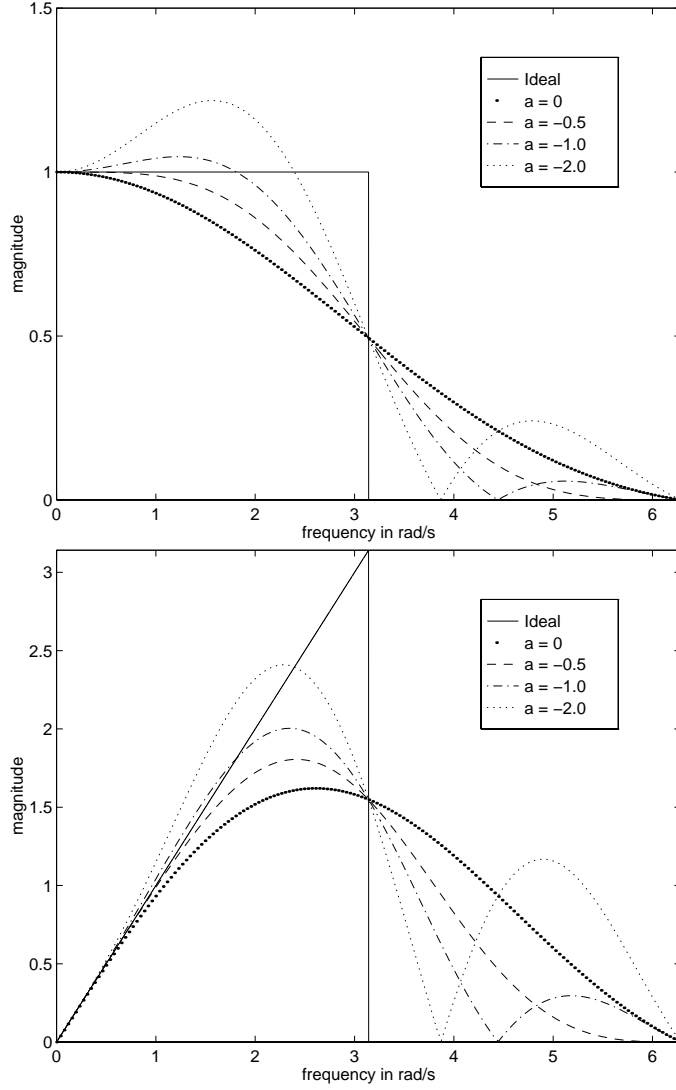


Fig. 6. *The Fourier Transform of the cubic spline (top) and the Fourier Transform of the derivative (bottom) for different values of  $a$ . The thick lines denote the ideal cases.*

where  $I_0(x)$  is the order 0 modified Bessel function [23] of  $x$ , and  $\beta$  is calculated from  $\alpha$  as

$$\beta = \alpha \sqrt{1 - \left(\frac{2n}{N-1}\right)^2} \quad (27)$$

By adjusting the Kaiser window parameter  $\alpha$  the frequency response can be varied. Fig. 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 samples to evaluate in the 1-D case, compared to 6 for the adjustable gradient filter proposed in [6]. Furthermore, this adjustable gradient filter calculates gradients on voxel positions, not on sample positions. In order to get the gradient on sample positions, some sort 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 off at a higher frequency than the effective adjustable gradient filter but has more energy at frequencies higher than  $\pi$ . See Fig. 7.

In Fig. 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 extent of 4 in the spatial domain. Three cubic spline based filters are plotted, with values of the  $a$  parameter of -0.75, -1.5 and -2.0.

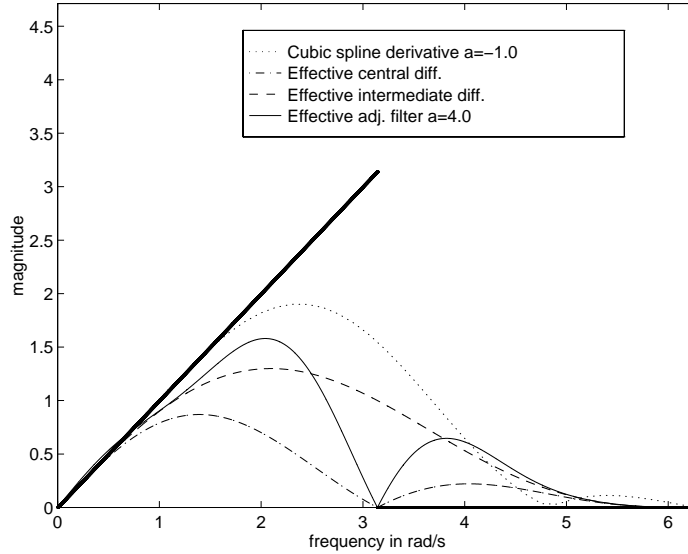


Fig. 7. Frequency responses of different methods to estimate the local gradient. The thick line is the ideal case.

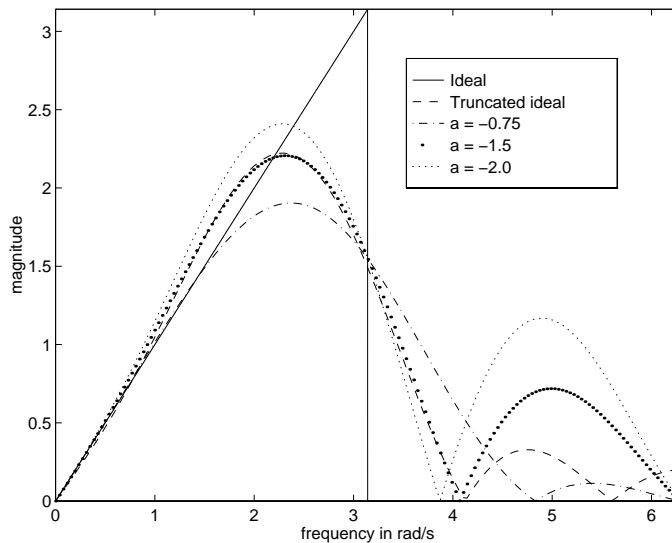


Fig. 8. Frequency responses of the ideal estimator truncated to 4 voxels and of the derivative of the cubic spline for different values of  $a$ .

Fig. 8 shows that the cubic spline derivative 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 fold over frequency  $\pi$ . The cubic spline derivative filter with  $a = -1.5$  is nearly identical to the truncated ideal filter up to  $\pi$ . 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.

## IX. SAMPLED GRADIENT FILTERS

When the gradient filters are used in practice, sampled gradient functions must be considered with different offset values. By offsets we mean the distance of the current sample point to the nearest voxel in each axis.

The resampling will alias any data which is passed above the Nyquist frequency into the pass band. This results in considerably different performance compared with the unsampled gradient filters discussed in the previous sections. In [14] a similar analysis is done for interpolation filters.

For the resampling of the intermediate difference and central difference gradient filters, the interpolation step is taken into account (the so called *effective* central difference and *effective* intermediate difference gradient filters). This is not necessary for the cubic spline based gradient filter, since this filter calculates the gradient on the sample position directly. In Fig. 9 and 10 the sampled gradient filters are shown for offsets of 0, 0.2, 0.4, and 0.5.

Several observations can be made from Fig. 9 and 10. The effective central difference method performs best at zero offset. If the offset increases, the performance at higher frequencies deteriorates. The effective intermediate difference method performs much better, especially at higher frequencies for increasing offsets. Note that at zero offset the frequency response of the effective intermediate difference method is equal to the frequency response of the effective central difference method. The effective central difference gradient filter has its best response at zero offset, while the effective intermediate difference gradient filter has its best response at an offset of 0.5. This is due to the fact that the original gradients are calculated in between integer positions (at an offset of 0.5) and then linearly interpolated to other offset positions. The worst performance for the intermediate difference filter is obtained at zero offset.

Although the effective intermediate difference method performs much better than the effective central difference method, the cubic spline based gradient filter performs even better, as is seen in Fig. 10. At zero offset, this gradient filter response is equal to the other two filters. This is because the filter coefficients of the cubic spline based gradient filter at zero offset are proportional to the other filters (positions 0, -2 and 2 are zero and positions -1 and 1 are non-zero, as can be seen in Fig. 5(bottom)). If the offset increases, the performance of the gradient filter also increases. Above an offset of 0.3, the response of the cubic spline based gradient filter closely approximates that of the ideal gradient filter.

## X. 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 followed by an extension into three dimensions.

In Fig. 11 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  $r(x, y)$ ,  $P_i$  can be estimated as follows:

$$\begin{aligned}
 P_i(x, y) = & \\
 & P_{-2,-2}r(x-2, y-2) + P_{-1,-2}r(x-1, y-2) + P_{0,-2}r(x, y-2) + P_{1,-2}r(x+1, y-2) + \\
 & P_{-2,-1}r(x-2, y-1) + P_{-1,-1}r(x-1, y-1) + P_{0,-1}r(x, y-1) + P_{1,-1}r(x+1, y-1) + \\
 & P_{-2,0}r(x-2, y) + P_{-1,0}r(x-1, y) + P_{0,0}r(x, y) + P_{1,0}r(x+1, y) + \\
 & P_{-2,1}r(x-2, y+1) + P_{-1,1}r(x-1, y+1) + P_{0,1}r(x, y+1) + P_{1,1}r(x+1, y+1)
 \end{aligned} \tag{28}$$

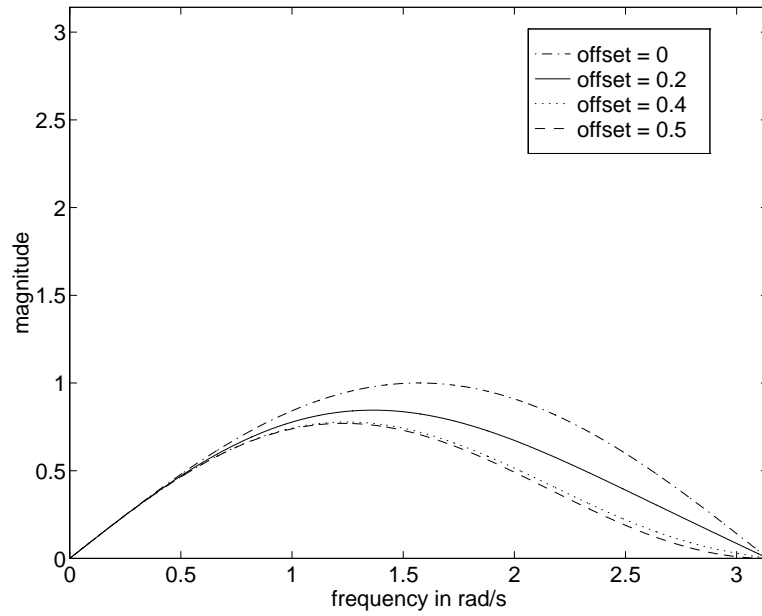
This means that in order to interpolate  $P_i$  the pixel values are multiplied with a 2-D kernel of which the weights are determined by  $r(x, y)$ . One method to achieve this is to sample  $r(x, y)$  and store these values in a lookup table which reduces the interpolation to 16 multiplications and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

For separable interpolation kernels (which has been assumed throughout this paper),  $r(x, y)$  can be calculated as a product of two 1-dimensional interpolation kernels.

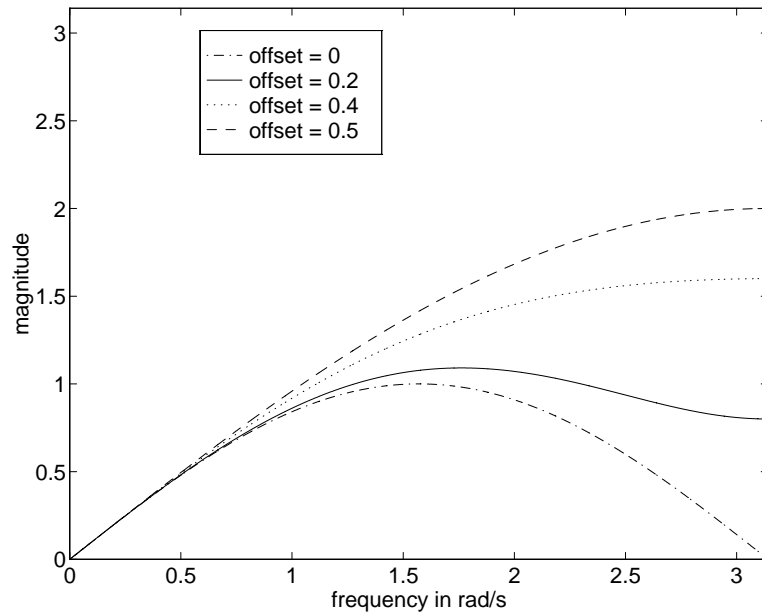
$$r(x, y) = r(x)r(y) \tag{29}$$

Now Equation 28 can be rewritten as:

$$\begin{aligned}
 P_i(x, y) = & \\
 & r(x-2)[P_{-2,-2}r(y-2) + P_{-2,-1}r(y-1) + P_{-2,0}r(y) + P_{-2,1}r(y+1)] + \\
 & r(x-1)[P_{-1,-2}r(y-2) + P_{-1,-1}r(y-1) + P_{-1,0}r(y) + P_{-1,1}r(y+1)] + \\
 & r(x)[P_{0,-2}r(y-2) + P_{0,-1}r(y-1) + P_{0,0}r(y) + P_{0,1}r(y+1)] +
 \end{aligned}$$



(a)



(b)

Fig. 9. Amplitude spectra of the sampled gradient functions, (a) is the effective central difference gradient method and (b) the effective intermediate difference gradient method.

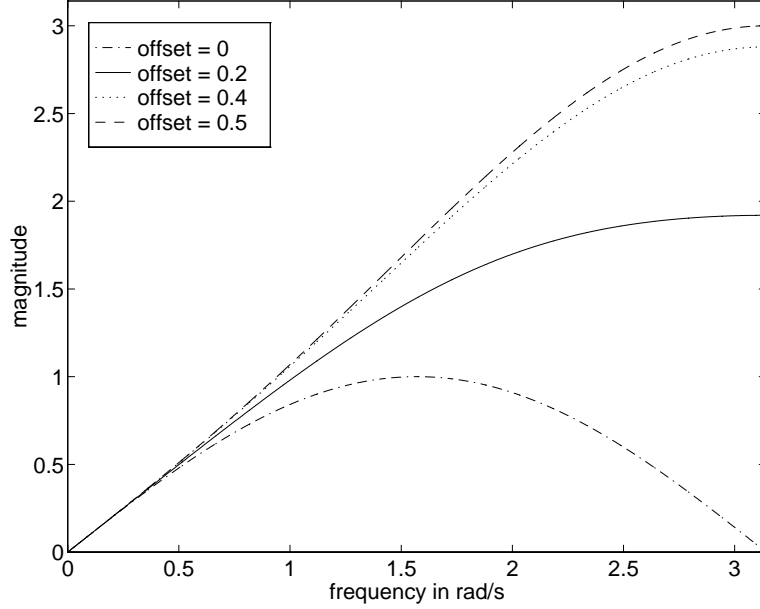


Fig. 10. Amplitude spectrum of the sampled cubic spline based gradient filter ( $a = -0.5$ ).

$$\begin{aligned}
 & r(x+1)[P_{1,-2}r(y-2) + P_{1,-1}r(y-1) + P_{1,0}r(y) + P_{1,1}r(y+1)] \\
 = & r(x-2)A + r(x-1)B + r(x)C + r(x+1)D
 \end{aligned} \tag{30}$$

Where  $A..D$  are interpolated values in the  $Y$ -direction. 20 multiplications and 15 additions are needed to compute  $P_i(x, y)$ . Fig. 11 shows this separable interpolation method.

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

$$G_i(x, y) = [G_{ix}, G_{iy}] = \left[ \frac{\partial}{\partial x} P_i(x, y), \frac{\partial}{\partial y} P_i(x, y) \right] \tag{31}$$

This leads to the following gradient in the  $x$ -direction:

$$\begin{aligned}
 G_{ix} = & \\
 & r'(x-2)[P_{-2,-2}r(y-2) + P_{-2,-1}r(y-1) + P_{-2,0}r(y) + P_{-2,1}r(y+1)] + \\
 & r'(x-1)[P_{-1,-2}r(y-2) + P_{-1,-1}r(y-1) + P_{-1,0}r(y) + P_{-1,1}r(y+1)] + \\
 & r'(x)[P_{0,-2}r(y-2) + P_{0,-1}r(y-1) + P_{0,0}r(y) + P_{0,1}r(y+1)] + \\
 & r'(x+1)[P_{1,-2}r(y-2) + P_{1,-1}r(y-1) + P_{1,0}r(y) + P_{1,1}r(y+1)] \\
 = & r'(x-2)A + r'(x-1)B + r'(x)C + r'(x+1)D
 \end{aligned} \tag{32}$$

Where  $r'(x)$  is the derivative of  $r(x)$ . 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 separable 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 is done using a non-separable filter, no savings can be achieved.

$G_{iy}$  can be derived in the same way:

$$G_{iy} = r'(y-2)E + r'(y-1)F + r'(y)G + r'(y+1)H \tag{33}$$

Where  $r'(y)$  is the derivative of  $r(y)$ . Unfortunately the values  $E, F, G$  and  $H$  are not available and have to be computed too.

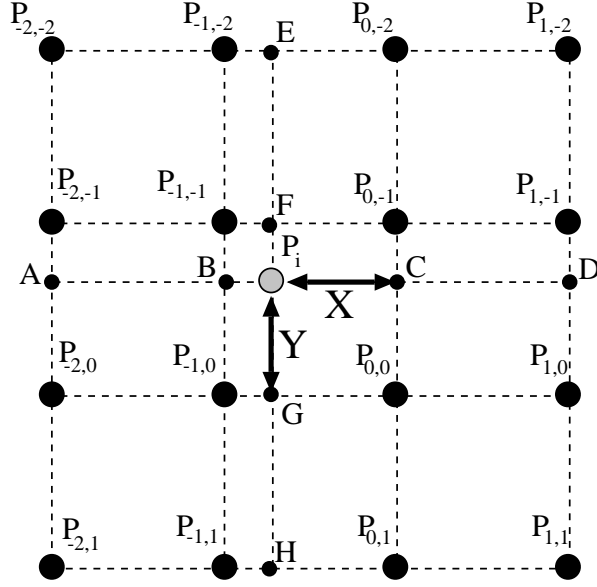


Fig. 11. Two dimensional interpolation situation.  $P_i$  is the sample position.

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

$$\begin{aligned}
 P_i(x, y, z) &= \sum_{j=-2}^1 \sum_{k=-2}^1 \sum_{l=-2}^1 P_{j,k,l} r(x+j, y+k, z+l) \\
 &= \sum_{j=-2}^1 \sum_{k=-2}^1 \sum_{l=-2}^1 P_{j,k,l} r(x+j) r(y+k) r(z+l)
 \end{aligned} \tag{34}$$

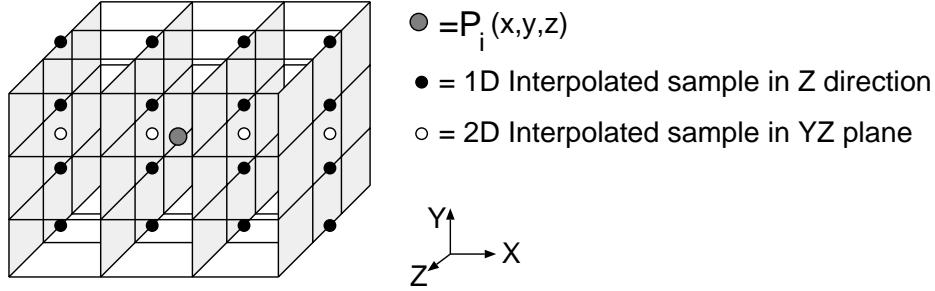


Fig. 12. Three dimensional interpolation situation.

Fig. 12 shows how this 3-D interpolation is split into consecutive 1-D interpolations. Shown is interpolation in the z-direction, then in the y-direction and finally in the x direction. The order in which this is done can be chosen arbitrarily.

The local gradient at the sample position is:

$$G_i(x, y, z) = [G_{ix}, G_{iy}, G_{iz}] = \left[ \frac{\partial}{\partial x} P_i(x, y, z), \frac{\partial}{\partial y} P_i(x, y, z), \frac{\partial}{\partial z} P_i(x, y, z) \right] \tag{35}$$

If we now rewrite Equation 34:

$$P_i(x, y, z) = \sum_{j=-2}^1 r(x+j) \sum_{k=-2}^1 \sum_{l=-2}^1 P_{j,k,l} r(y+k) r(z+l) \tag{36}$$

Then the x-component of the local gradient is:

$$\frac{\partial}{\partial x} P_i(x, y, z) = \sum_{j=-2}^1 r'(x+j) \sum_{k=-2}^1 \sum_{l=-2}^1 P_{j,k,l} r(y+k) r(z+l) \quad (37)$$

The summations over  $k$  and  $l$  are already computed in Equation 36. Thus the calculation of the x-component of the gradient takes only one 1-D interpolation with  $r'(x)$  in the x-direction.

For the y-component of the local gradient a similar scheme can be applied.

$$\frac{\partial}{\partial y} P_i(x, y, z) = \sum_{k=-2}^1 r'(y+k) \sum_{j=-2}^1 r(x+j) \sum_{l=-2}^1 P_{j,k,l} r(z+l) \quad (38)$$

The summation over  $l$  is already computed in Equation 36 for every  $kj$  pair. This means that only 4 1-D interpolations in the x-direction with  $r(x)$  have to be done, over the solid black circles in Fig. 12, and one final 1-D interpolation in the y-direction with  $r'(y)$  to calculate the y-component of the gradient.

For the z-component of the local gradient no savings can be achieved. It has to be computed completely from scratch, either using 21 interpolations or a 3-D table of  $r(x, y, z)$  and a brute force calculation or a hybrid method with a 2-D table of  $r(x, y)$  or a 1-D table of  $r(x)$ .

## XI. COMPUTATIONAL EXPENSE OF THE NEW GRADIENT FILTERS

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

A 3-D interpolation using a separable filter with an extent of 4 points in one dimension requires  $4 \times 5 + 1 = 21$  1-D interpolations<sup>2</sup>, or convolutions with the interpolation filter  $r(x)$ . In that case the x-component of the gradient can be calculated with one 1-D convolution with the gradient filter  $r'(x)$ . The y-component can be computed with 5 more 1-D convolutions. For the z components however, a totally new interpolation scheme must be applied to obtain the 4 interpolated points, necessary to calculate the z gradient components. That means that a total of  $1 + 5 + 21 = 27$  convolutions<sup>3</sup> are needed to obtain the complete gradient vector. Thus the calculation of the gradient vector is slightly more expensive than the interpolation itself, which cost 21 convolutions. A 1-D convolution requires 4 multiplications and 3 additions. Thus interpolation and gradient estimation together amount to  $4 \times (21 + 27) = 192$  multiplications and  $3 \times (21 + 27) = 144$  additions. This assumes however that the interpolation function  $r(x)$  and its derivative  $r'(x)$  are sampled and the values stored in a lookup table. Note that this lookup table will be much smaller than the one needed for a non-separable filter.

It is cheaper to use a separable 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.

## XII. RESULTS

Fig. 14 and 15 present renderings using various gradient estimators. The original datasets are shown in Fig. 13(a) and 13(b). They are rendered using the cubic spline based gradient filter with  $a = -0.5$ .

Fig. 13(a) shows a volume rendering of a dataset from the University of North Carolina Volume Rendering Test Dataset Volume II. The dataset is a  $256 \times 256 \times 109$  magnetic resonance image of a head with the brain surface exposed.

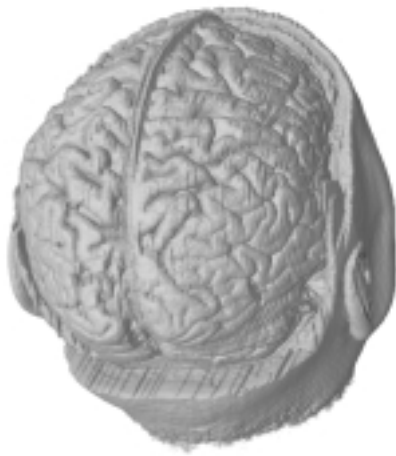
<sup>2</sup>A  $4 \times 4$  2-D interpolation requires 5 1-D interpolations. See Fig. 11 and 12.

<sup>3</sup>One convolution in the x direction and 5 convolutions in the y direction because values calculated in the interpolation stage can be reused to calculate the x and y component of the gradient. 21 convolutions because this is not the case for the z direction.

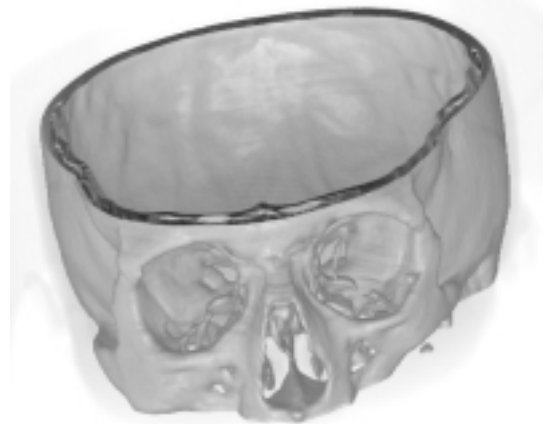
Fig. 13(b) shows a volume rendering of a CT dataset from the Mayo Clinic. The size of the dataset is  $256 \times 256 \times 46$ .

Fig. 14(a),(b),(c) and (d) show closeups of the dataset shown in Fig. 13(a) but rendered using different gradient estimators. Fig. 15(a),(b),(c) and (d) show closeups of the area above the nose of Fig. 13(b) also rendered using various different gradient estimators.

Fig. 14(a) and 15(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. Fig. 14(b) and 15(b) were rendered using the intermediate difference method to estimate the gradient. Linear interpolation is used to estimate the gradient on the sample positions, used for the opacity calculation. Fig. 14(c) and 15(c) were rendered using the gradient method we propose, the derivative of the cubic spline interpolation function with the parameter  $a = -0.5$ . Fig. 14(d) and 15(d) were rendered with the same gradient method, but with  $a = -2.0$ .



(a)



(b)

Fig. 13. See text also. (a) Rendering of the whole MR brain dataset. (b) Rendering of the whole CT dataset.

As can be seen in the 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. However, for detecting fine features the cubic spline method performs better. Fig. 14(d) shows the most details because high frequencies are better preserved.

### XIII. DISCUSSION

In this paper several existing gradient estimation methods were analysed. The most commonly used gradient filter, the central differences method, is a fixed operator and can not be tuned for optimal balance between fine details on the one hand and aliasing and noise rejection on the other. The adjustable filter techniques of Goss [6] does have this feature.

We presented a new class of volumetric gradient operators that are adjustable as well. These operators take the gradient of the interpolation function itself, and we demonstrated a family of examples based on the cubic-spline as an interpolation function. Performance of these cubic spline derivative filters in the frequency domain was analysed. We demonstrated the computational advantages of such seperable filters and showed that calculations used in interpolating to a sample point can be reused in the gradient computation, reducing its complexity. We also provided an analysis of frequency preformance of various gradient operators as a function of subvoxel offset. In general the offset value having optimum performance varies among the various filters.

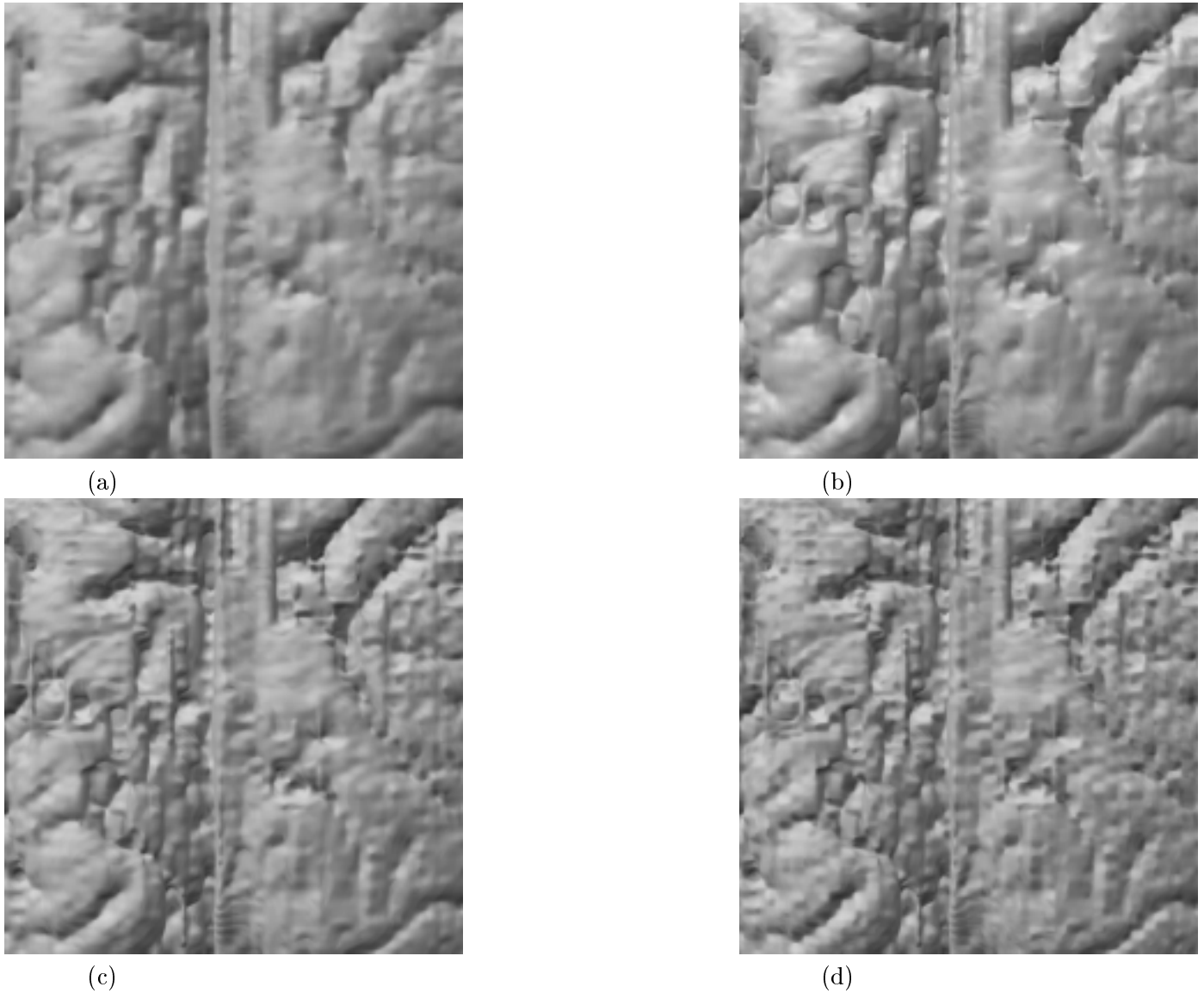


Fig. 14. Difference between gradient filters rendering the MR brain dataset. (a) Zoomed in using precalculated color and opacity on grid positions. (b) Zoomed in using the intermediate difference method. (c) Zoomed in using the cubic spline based gradient method with  $a=-0.5$ . (d) Zoomed in using the cubic spline based gradient method with  $a=-2.0$ .

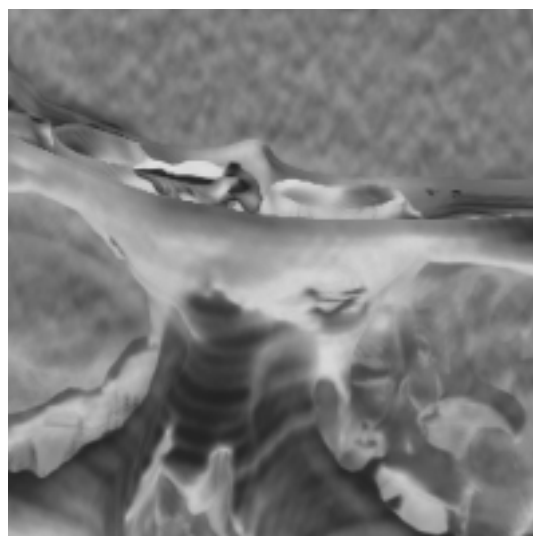
#### XIV. ACKNOWLEDGMENTS

We like to thank all the people who helped in this research. We especially like to thank Marco Bosma and Jeroen Terwischa van Scheltinga for the Vivid software to generate the images in this paper. We are very thankful for the useful comments and suggestions Irwin Sobel made. We would like to thank Ron Schafer for the derivation presented in the appendix. Lastly we are very grateful for the useful comments of the anonymous reviewers.

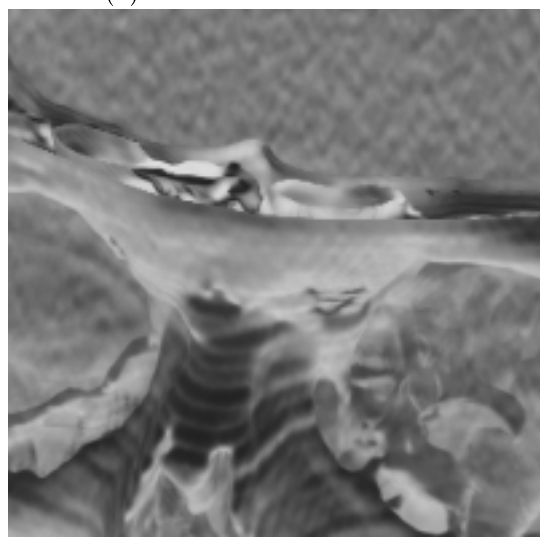
The investigations were partly supported by the foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organization for Scientific Research (NWO) and the Visual Computing Department of Hewlett Packard Laboratories.



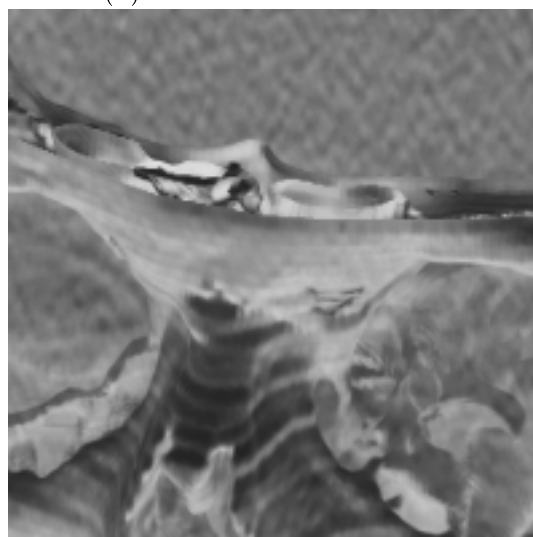
(a)



(b)



(c)



(d)

Fig. 15. Difference between gradient filters rendering the CT head dataset. (a) Zoomed in using precalculated color and opacity on grid positions. (b) Zoomed in using the intermediate difference method. (c) Zoomed in using the cubic spline based gradient method with  $a=-0.5$ . (d) Zoomed in using the cubic spline based gradient method with  $a=-2.0$ .

## BIOGRAPHY

Tom Malzbender heads the Graphics Algorithms and Architecture group at Hewlett-Packard Laboratories in Palo Alto, California. He developed the capacitive sensing technology that forms the basis of HP's line of graphics tablets. His research interests include the role of signal processing in volume rendering, 3D ultrasound, statistical texture modelling, neural and genetic optimization techniques. He holds a BS in Electrical Engineering from Cornell University.

Mark J. Bentum was born in Smilde, The Netherlands in 1967. In 1988 he received his BS in Electrical Engineering from the Polytechnical High School of Groningen, The Netherlands. In 1991 he received his MS, with honors, in Electrical Engineering from the University of Twente, Enschede, The Netherlands. In 1995 he obtained his PhD in Electrical Engineering, also from the University of Twente. His thesis was about interactive visualization of volume data. Currently he is working for the Netherlands Foundation for Research in Astronomy, Westerbork, The Netherlands. His main research interests are computer graphics, signal and image processing.

Barthold Lichtenbelt currently works in the Graphics Software Lab at Hewlett-Packard Company in Fort Collins, Colorado. He received his MS in Electrical Engineering in 1994 from the University of Twente, Enschede, The Netherlands. Most of the work presented in this paper was done while he was working at Hewlett-Packard Laboratories. His research interests include hardware designs for 3-D graphics systems and signal processing applied to volume rendering and image processing.

APPENDIX FOURIER TRANSFORM OF THE CUBIC SPLINE INTERPOLATION FUNCTION

In this appendix the Fourier transforms of the cubic spline interpolation function will be derived analytically. If  $f(x)$  is an arbitrary function in the spatial domain, the Fourier transform  $F(j\omega)$  is given by:

$$F(j\omega) = \int_{-\infty}^{\infty} f(x)e^{-j\omega x} dx \quad (39)$$

The inverse Fourier transform is given by:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega)e^{j\omega x} d\omega \quad (40)$$

The cubic spline interpolation function is given by:

$$r(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

For convenience the cubic spline equation can be written as:

$$r(x) = r_0(x) + ar_1(x) \quad (42)$$

Where:

$$r_0(x) = \begin{cases} 2|x|^3 - 3|x|^2 + 1 & 0 \leq |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (43)$$

$$r_1(x) = \begin{cases} |x|^3 - |x|^2 & 0 \leq |x| \leq 1 \\ |x|^3 - 5|x|^2 + 8|x| - 4 & 1 \leq |x| \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (44)$$

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

$$\alpha f(x) + \beta g(x) \iff \alpha F(j\omega) + \beta G(j\omega) \quad (45)$$

Then the Fourier transform of  $r(x) = R(j\omega)$  equals:

$$R(j\omega) = R_0(j\omega) + aR_1(j\omega) \quad (46)$$

$R_0(j\omega)$  and  $R_1(j\omega)$  are the Fourier transforms of  $r_0(x)$  and  $r_1(x)$  respectively.

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 <sup>4</sup>.

Let  $p(x)$  be:

$$p(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (47)$$

Now define  $g(x)$  as:

$$g(x) = p(x)\{2x^3 - 3x^2 + 1\} \quad (48)$$

then:

$$r_0(x) = g(x) + g(-x) \quad (49)$$

and:

$$R_0(j\omega) = G(j\omega) + G(-j\omega) \quad (50)$$

<sup>4</sup>The Fourier transform is a special case of the Laplace transform. Let  $s = \alpha + j\omega$  for the Laplace transform. Setting  $\alpha = 0$  will then result in the Fourier transform.

$G(j\omega)$  is the Fourier transform of  $g(x)$ . The Laplace transform of Equation 48 can be computed using the derivative theorem of the Laplace transform. If:

$$P(s) = \int_{-\infty}^{\infty} p(x)e^{-sx} dx \quad (51)$$

Then:

$$\frac{dP(s)}{ds} = \int_{-\infty}^{\infty} -xp(x)e^{-sx} dx \quad (52)$$

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

$$\begin{aligned} -xp(x) &\iff P'(s) \\ x^2p(x) &\iff P''(s) \\ -x^3p(x) &\iff P'''(s) \end{aligned} \quad (53)$$

Where  $\iff$  denotes a Laplace transform pair. Thus the Laplace transform of Equation 48 has the following form:

$$G(s) = -2P'''(s) - 3P''(s) + P(s) \quad (54)$$

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

$$P(s) = \int_0^1 e^{-sx} dx = \frac{1 - e^{-s}}{s} \quad (55)$$

Now:

$$\begin{aligned} P'(s) &= \frac{se^{-s} - 1 + e^{-s}}{s^2} \\ P''(s) &= \frac{-s^2e^{-s} - 2se^{-s} + 2 - 2e^{-s}}{s^3} \\ P'''(s) &= \frac{s^3e^{-s} + 3s^2e^{-s} + 6se^{-s} - 6 + 6e^{-s}}{s^4} \end{aligned} \quad (56)$$

So:

$$\begin{aligned} G(s) &= -2P'''(s) - 3P''(s) + P(s) \\ &= \frac{1}{s^3}(-6 - 6e^{-s}) + \frac{1}{s^4}(12 - 12e^{-s}) + \frac{1}{s} \end{aligned} \quad (57)$$

By substituting  $s = j\omega$  we go back to the Fourier transform.

$$\begin{aligned} R_0(j\omega) &= G(j\omega) + G(-j\omega) \\ &= \frac{6}{j^3\omega^3}(e^{j\omega} - e^{-j\omega}) - \frac{12}{j^4\omega^4}(e^{j\omega} + e^{-j\omega} - 2) \\ &= \frac{12}{\omega^2} \left( \text{sinc}^2\left(\frac{\omega}{2}\right) - \text{sinc}(\omega) \right) \end{aligned} \quad (58)$$

Now the Fourier transform of  $r_1(x)$  will be derived.

$$r_1(x) = \begin{cases} |x|^3 - |x|^2 & 0 \leq |x| \leq 1 \\ |x|^3 - 5|x|^2 + 8|x| - 4 & 1 \leq |x| \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (59)$$

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

$$q(x) = \begin{cases} 1 & 1 \leq x \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (60)$$

$g(x)$  is now defined as:

$$g(x) = p(x)\{x^3 - x^2\} + q(x)\{x^3 - 5x^2 + 8x - 4\} \quad (61)$$

Then:

$$r_1(x) = g(x) + g(-x) \quad (62)$$

and:

$$R_1(j\omega) = G(j\omega) + G(-j\omega) \quad (63)$$

Using the same strategy,  $G(j\omega)$  can be calculated. Again the Laplace domain is used for convenience.

$$G(s) = -P'''(s) - P''(s) - Q'''(s) - 5Q''(s) - 8Q'(s) - 4Q(s) \quad (64)$$

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

$$Q(s) = \int_{-\infty}^{\infty} q(x)e^{-sx}dx = \int_1^2 e^{-sx}dx = \frac{e^{-s} - e^{-2s}}{s} = e^{-s}P(s) \quad (65)$$

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

$$\begin{aligned} Q'(s) &= e^{-s}P'(s) - e^{-s}P(s) \\ Q''(s) &= e^{-s}P''(s) - 2e^{-s}P'(s) + e^{-s}P(s) \\ Q'''(s) &= e^{-s}P'''(s) - 3e^{-s}P''(s) + 3e^{-s}P'(s) - e^{-s}P(s) \end{aligned} \quad (66)$$

Substitution yields:

$$\begin{aligned} G(s) &= -P'''(s)[1 + e^{-s}] - P''(s)[1 + 2e^{-s}] - P'(s)[e^{-s}] \\ &= -\frac{1}{s^3} \left( 2 + 8e^{-s} + 2e^{-2s} \right) - \frac{1}{s^4} \left( 6e^{-2s} - 6 \right) \end{aligned} \quad (67)$$

By substituting  $s = j\omega$   $R_1(j\omega)$  equals:

$$\begin{aligned} R_1(j\omega) &= G(j\omega) + G(-j\omega) \\ &= \frac{8}{j^3\omega^3} \left( e^{j\omega} - e^{-j\omega} \right) + \frac{2}{j^3\omega^3} \left( e^{2j\omega} - e^{-2j\omega} \right) - \frac{6}{j^4\omega^4} \left( e^{2j\omega} + e^{-2j\omega} - 2 \right) \\ &= \frac{8}{\omega^2} \left( 3\text{sinc}^2(\omega) - 2\text{sinc}(\omega) - \text{sinc}(2\omega) \right) \end{aligned} \quad (68)$$

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

$$\begin{aligned} R(j\omega) &= \frac{12}{\omega^2} \left( \text{sinc}^2 \left( \frac{\omega}{2} \right) - \text{sinc}(\omega) \right) \\ &\quad + a \frac{8}{\omega^2} \left( 3\text{sinc}^2(\omega) - 2\text{sinc}(\omega) - \text{sinc}(2\omega) \right) \end{aligned} \quad (69)$$

Note that at the Nyquist fold over frequency  $\omega = \pi$  the magnitude of  $R(j\omega)$  is independent of the parameter  $a$  and is equal to  $48/\pi^4$ .

## REFERENCES

- [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*, 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. EuroGraphics Technical Report Series, 1995.
- [3] Ronald N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, 1986.
- [4] E. Catmull and R. Rom. *Computer Aided Geometric Design*. Academic Press, 1974.
- [5] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. *Computer Graphics*, 22(4):65–74, August 1988.
- [6] M.E. Goss. An Adjustable Gradient Filter for Volume Visualization Image Enhancement. In *Proceedings Graphics Interface '94*, pages 67–74. Canadian Inf. Process. Soc, Toronto, Ont., Canada, 1994.
- [7] 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.
- [8] M.S. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, pages 29–37, May 1988.
- [9] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, July 1987.
- [10] T. Malzbender. Fourier Volume Rendering. *ACM Transaction on Graphics*, 12(3):233–250, July 1993.
- [11] David Marr. Theory of edge detection. In *Proceedings of the Royal Society, London*, volume B207, pages 187–217, 1980.
- [12] Don P. Mitchell and Arun N. Netravali. Reconstruction Filters in Computer Graphics. *Computer Graphics*, 22(4):221–228, August 1988.
- [13] S.K. Park and R.A. Schowengerdt. Image Reconstruction by Parametric Cubic Convolution. *Computer Vision, Graphics, and Image Processing*, 23:258–272, 1983.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] P. Sabella. A Rendering Algorithm for Visualizing 3D Scalar Fields. *Computer Graphics*, 22(4):51–58, August 1988.
- [18] C.E. Shannon. Communication in the Process of Noise. *Proceedings of the IRE*, 37:10–21, January 1949.
- [19] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. *Computer Graphics*, pages 271–278, August 1993.
- [20] C. Upson and M. Keeler. V-Buffer: Visible Volume Rendering. *Computer Graphics*, 22(4):59–64, August 1988.
- [21] L. Westover. Interactive volume rendering. In *Chapell Hill Workshop on Volume Visualization*, pages 9–16, May 1989.
- [22] L. Westover. Footprints Evaluation for Volume Rendering. *Computer Graphics*, 24(4):367–376, August 1990.
- [23] Saul Teukolsky William Press, Brian Flannery and William Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge UP, Cambridge England, 1988.
- [24] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.
- [25] R. Yagel, D. Cohen, and A. Kaufman. Normal Estimation In 3d Discrete Space. *The Visual Computer*, 8(5-6):278–291, June 1992.
- [26] 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.