

A Comparison Of Normal Estimation Schemes

Torsten Möller^{1,2}, Raghu Machiraju^{3,4}, Klaus Mueller¹, Roni Yagel^{1,2}

¹Department of Computer and Information Science

²The Advanced Computing Center for
the Arts and Design

The Ohio State University, Columbus, Ohio

³NSF Engineering Research Center for Computational
Field Simulation

⁴Department of Computer Science
Mississippi State University, Mississippi

Abstract

The task of reconstructing the derivative of a discrete function is essential for its shading and rendering as well as being widely used in image processing and analysis. We survey the possible methods for normal estimation in volume rendering and divide them into two classes based on the delivered numerical accuracy. The three members of the first class determine the normal in two steps by employing both interpolation and derivative filters. Among these is a new method which has never been realized. The members of the first class are all equally accurate. The second class has only one member and employs a continuous derivative filter obtained through the analytic derivation of an interpolation filter. We use the new method to analytically compare the accuracy of the first class with that of the second. As a result of our analysis we show that even inexpensive schemes can in fact be more accurate than high order methods. We describe the theoretical computational cost of applying the schemes in a volume rendering application and provide guidelines for helping one choose a scheme for estimating derivatives. In particular we find that the new method can be very inexpensive and can compete with the normal estimations which pre-shade and pre-classify the volume [8].

Keywords: interpolation filters, derivative filters, filter design, normal estimation, Taylor series expansion, efficient volume rendering

1 INTRODUCTION

Reconstruction of a continuous function and its derivatives from a set of samples is one of the fundamental operations in visualization algorithms. In volume rendering, for instance, we must be able to interpolate the function at arbitrary locations to obtain the volume densities. The gradient (the first derivative of the function) is employed in both volume classification and shading [3][8]. If the gradient estimation is done carelessly, shading and classification will yield wrong colors and opacities. Since the derivative of a function indicates the velocity of change of the function values, the presence of noise especially will lead to incorrect images [4].

There have been various studies and comparisons of accurate interpolation filters, a summary of which is given in [10][12]. However, as is also shown in [12], the derivative approximation has a larger impact on the quality of the image and therefore deserves a thorough analysis, which is the goal of this paper.

The ideal derivative filter is the *Cosc* filter, which is the derivative of the ideal interpolation filter (*Sinc*) [1][4]. For a practical use of the *Sinc* filter, windowing is suggested [7]. Goss [6] extends the idea of windowing from interpolation filters to derivative filters. He uses a Kaiser window to mitigate the adverse effects of the truncated ideal derivative filter. Bentum et al. [1] use the Cardinal

cubic splines to develop derivative filters. A good survey of existing digital derivative filters can be found in the paper by Dutta Roy and Kumar [4].

While all of the previous work focuses on the design of derivative filters, no work is known to us, that tries to conduct a comparative study of gradient filters. Especially, in the case of volume rendering, most algorithms are driven by efficiency considerations and may decompose the gradient estimation in one or two steps. One step is typically the interpolation of the normals or of the data values with a continuous interpolation filter. The other step is the application of a digital derivative filter (e.g. central differences) in order to compute the normal at the sampling location. However, there have been schemes proposed, that estimate the normal at an arbitrary point in the volume in one step [1]. The goal of this paper is to enumerate and classify the different schemes of gradient estimation and to analyze them in terms of accuracy and efficiency.

In this paper, we denote by $f(t)$ a continuous function (the signal) which is sampled into the discrete function $f[k] = f(kT)$, where T is the sampling distance and k is an integer. In computer imaging, $f(t)$ is not available; we only have $f[k]$. We denote by $h(t)$ the continuous function kernel used for interpolation and by $d[k]$ the digital (i.e. only defined for integer arguments) derivative filter.

We employ a Taylor series expansion of the convolution sum for our numerical analysis, as introduced in [12]. Our Taylor series expansion provides both qualitative and quantitative means of analyzing filters. In Section 3, this analysis is expanded to the convolution of two filters. The methods of [12] are briefly summarized.

1.1 Taylor Expansion of the Convolution Sum

To reconstruct a continuous function $f(t)$ or its derivative $f'(t)$ from a set of sample points $f[k]$, we convolve $f[k]$ with a filter kernel, i.e. we compute a weighted average of these samples. By convolving the sampled signal $f[k]$ with a continuous interpolation filter h , we reconstruct an approximation of the original function $f(t)$. Similarly, if we convolve the samples with a continuous derivative filter d , we reconstruct an approximation of the derivative of the original function. We denote the result of this operation by $f_r^v(t)$, where w is the filter used. Formally, this can be written as:

$$f_r^v(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot w\left(\frac{t}{T} - k\right) \quad (1)$$

Now we can expand $f[k] = f(kT)$ into a Taylor series about t . The Taylor series expansion at that point would be:

$$f[k] = \sum_{n=0}^N \frac{f^{(n)}(t)}{n!} (kT - t)^n + \frac{f^{(N+1)}(\xi_k)}{(N+1)!} (kT - t)^{(N+1)}$$

where $f^{(n)}(t)$ is the n -th derivative of f and $\xi_k \in [t, kT]$

Substituting the Taylor series expansion into the convolution sum of Equation 1, leads to an alternative representation for the reconstructed value at a point t :

^{1,2} 395 Dreese Lab, Columbus, OH 43210,
{moeller, mueller, yagel}@cis.ohio-state.edu

^{3,4} NSF Engineering Research Center for Computational Field Simulation,
P.O. Box 9627, Mississippi State, MS 39762,
raghu@erc.msstate.edu

$$f_r^w(t) = \sum_{n=0}^N a_n^w(\tau) f^{(n)}(t) + r_{N,i}^w(\tau)$$

$$a_n^w(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n w(\tau-k)$$
(2)

$$r_{N,i}^w(\tau) \leq \left(\xi \in [(i-M)T, (i+M)T] \right) \left(f^{(N+1)}(\xi) \right) \left| a_{N+1}^w(\tau) \right|$$

or

$$r_{N,i}^w(\tau) \approx a_{N+1}^w(\tau) f^{(N+1)}(t)$$

where τ is chosen such that $t = (i + \tau)T$, with $0 \leq \tau < 1$, and i is an integer. It is noteworthy that the derived Taylor coefficients a and the remainder term r only depend on the offset to the nearest sampling point, i.e., they are periodic in the sampling distance T . For further details, please refer to [12].

The characterization of the filtering process in Equation 2 imposes 4 different criteria for a good normal estimation scheme. First of all, we require a_0^w to be zero. Secondly we have to normalize by a_1^w in order to reconstruct the actual derivative as opposed to a multiple of it. Further by determining the largest N , such that a_N^w is zero, we can determine the asymptotic error behavior of a filter for a decreasing sampling distance T . Finally, the remainder term r gives us an indication of the absolute error of that filter.

This expansion of the convolution sum assumes that at least the first N derivatives of the function f exist, where N depends on our error analysis. This condition is generally met in practice since image and volume acquisition devices such as scanners and cameras inherently perform a low-pass filtering operation that bandlimits the functions [2]. Numerical simulations of physical phenomena, as performed in the field of computational fluid dynamics, usually generate bandlimited images as well since typically robust numerical solutions can be obtained only if the algorithm incorporates a smoothing step [15]. Finally, all rendering and scan-conversion algorithms, in order to provide antialiased images, usually also employ a filtering step that bandlimits the image. Bandlimited functions do not contain frequencies higher than a certain limiting frequency in their spectra. One can conclude, that bandlimited functions are analytic functions and all N derivatives exist.

The remainder of the paper is organized as follows. In Section 2, we summarize the different schemes for normal estimation. In Section 3, we modify the Taylor series expansion of the convolution operation for the specific use of cascading two filters, and compare the schemes of Section 2 numerically. In Section 4, we examine possible implementations of the normal estimation schemes and compare their efficiency. Experimental results are also presented in Section 5. Finally, in Section 6, we summarize the results of this paper and discuss some open questions.

2 GRADIENT RECONSTRUCTION FROM THE VOLUME SAMPLES

We will use the symbol F to represent the discrete function $ff[k]$. Further, we let D and H denote the derivative and interpolation operators, respectively. In the process of volume rendering there are two additional operators applied to the data. The first is the *transfer function*, which maps the raw data values into material attributes such as color, opacity, and reflectivity. We denote this operator, also called *classification function*, by C . The second operator applied to the data is *shading*, which illuminates the data. The shading operator, which we denote by S , takes as input material attributes, light attributes, and the surface normal, and produces a

displayable value (e.g., $RGB\alpha$).

Since S needs the output of C , shading will always be performed after classification. Since S needs the function's derivative, it will always be after D . We now present four different ways of computing the function derivatives. Except for the first approach, $(FD)H$, in all others the operators CS will be performed *after* the interpolated derivative has been computed.

2.1 Method $(FD)H$ - Derivative First

One way of computing the derivative at a point t of a discrete function $f[k]$ is to first compute the normal at the grid points kT and then interpolate these normals, producing the derivative at the desired location t . This is the method most commonly used in volume graphics [6][9]. The first step, the computation of the derivative at the grid points, can be expressed in the following convolution:

$$f^d[k] = \sum_{l=-\infty}^{\infty} f[l] \cdot d[k-l]$$

Now the derivative at an arbitrary point can be interpolated as:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} f^d[k] \cdot h\left(\frac{t}{T} - k\right)$$

$$= \sum_{k=-\infty}^{\infty} \left(\sum_{l=-\infty}^{\infty} f[l] \cdot d[k-l] \right) \cdot h\left(\frac{t}{T} - k\right)$$

Square brackets are used to emphasize the discrete nature of the operator. Since a convolution in spatial domain is the same as a multiplication in frequency domain, we conclude the following frequency characterization of the above operation:

$$F_r^{dh}\left(\frac{\omega}{T}\right) = (F_D(\omega)D_D(\omega))H(\omega) \quad (3)$$

Here $D_D(\omega)$ denotes the Fourier transform of the discrete derivative filter and $F_D(\omega)$ denotes the Fourier transform of the sampled function $f[k]$. The Fourier transform of a discrete function contains replicated frequency spectra at $k2\pi$ (where k is an arbitrary integer). Therefore $D_D(\omega)$ and $F_D(\omega)$ are periodic functions with period 2π . Following the Fourier transform in Equation 3, we will refer to this method as $(FD)H$.

Unlike all other methods described in this paper, some algorithms ([8][3][16]) perform interpolation after classification and shading. Normal values are computed at the grid points and classification is also applied to the original data values. Then, these data points are shaded. The final $RGB\alpha$ volume is then interpolated at the appropriate sampling points. Using our notation, this method can be summarized by $((FD)C)S)H$. This is indeed an efficient method, since CS does not have to be computed for every sample point (which is the case for all other methods described in this paper where interpolation is done *before* CS) but rather it is computed only for the original data points. However, this method will produce correct results only if both C and S are linear operators. The result of employing a non-linear transfer function or illumination model may, for example, cause the appearance of errors or pseudo-features that are non-existent in the original data. In the case of S , one must therefore allow illumination models consisting of only ambient lighting. In the case of C , the linearity restriction may not be acceptable for many applications. For example, if we want to find the opacity in-between two data values a and b (using linear interpolation), we would find $(C(a)+C(b))/2$ by performing classification first. However we would find $C((a+b)/2)$ by performing interpolation first. Obviously, if C is a non-linear operator, the two results will be different. We therefore concentrate our analysis and discussion in the more general and accurate methods that

perform *CS* only *after* gradient estimation and interpolation.

2.2 Method (FH)D - Interpolation First

In this approach, we first reconstruct the continuous function $f(t)$ from the sampled values f_k and then apply the discrete derivative filter d [10][14]. Since the derivative filter is discrete, we only need to evaluate the convolution sum of the interpolated function at discrete points. The interpolated function $f_r^h(t)$ can be expressed as a convolution of the samples f_k using the interpolation filter h :

$$f_r^h(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot h\left(\frac{t}{T} - k\right)$$

The reconstructed derivative can be computed by:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} d[k] \cdot f_r^h(t - kT)$$

Using similar arguments as above, we find the Fourier Transform to be:

$$F_r\left(\frac{\omega}{T}\right) = (F_D(\omega)H(\omega))D_D(\omega) \quad (4)$$

Using our previous notation scheme, we refer to this method as method (FH)D.

2.3 Method F(DH) - Continuous Derivative

Looking at all possible combinations of applying the interpolation filter and the derivative filter to the discrete signal, we are led to a theoretical result. Namely, that we can first convolve the digital derivative filter with the continuous interpolation filter. The result will be a new continuous derivative filter which we can apply to the data samples, enabling us to reconstruct the derivative at any arbitrary point t directly. This can be written as:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot dh\left(\frac{t}{T} - k\right)$$

where the continuous derivative filter $dh(t)$ is obtained as the convolution of the digital filter $d[k]$ with the interpolation filter h :

$$dh(t) = \sum_{k=-\infty}^{\infty} d[k] \cdot h\left(\frac{t}{T} - k\right)$$

We can show that the frequency representation of this process is:

$$F_r^{dh}\left(\frac{\omega}{T}\right) = F_D(\omega)(D_D(\omega)H(\omega)) \quad (5)$$

therefore referring to this method as $F(DH)$. The benefit of this scheme is more conceptual at this moment. In Section 3 we show how it can be used for a convenient analysis of the normal estimation process. Further we will show in Section 4, that this method can also be the most efficient to use for volume rendering algorithms.

2.4 Method FH' - Analytic Derivative

A fourth method to compute the gradient of a discrete function is to convolve the samples $f[k]$ with the analytical derivative of the interpolation filter h :

$$f_r^{h'}(t) = \sum_{k=-\infty}^{\infty} \frac{f_k}{T} \cdot h'\left(\frac{t}{T} - k\right)$$

In this case, h' represents a continuous derivative filter, allowing us to reconstruct the continuous derivative $f'(t)$ directly from the samples $f[k]$ in just one convolution. This is very similar to the previous method $F(DH)$. It differs only in the way we construct the

derivative filter: In method $F(DH)$ we compute a convolution sum for the continuous derivative filter, while in this method we compute the continuous derivative filter analytically. Bentum et al. [1] apply this idea to cardinal splines, and Marschner and Lobb [11] use this for the BC-splines. The Fourier transform of the derivative of a function is simply the scaled Fourier transform of that function multiplied by $i\omega$ ($i^2 = -1$)[2]. Therefore, we find that the Fourier transform of $f_r^{h'}(t)$ is:

$$F_r^{h'}\left(\frac{\omega}{T}\right) = F_D(\omega)\left(\frac{i\omega}{T}H(\omega)\right)$$

and we refer to this method as FH' .

3 NUMERICAL ACCURACY

Comparing Equations 3, 4, and 5 we easily find that these three methods are numerically equivalent and thus produce the exact same reconstructed derivative of f . Therefore, we will concentrate on comparing the methods $(FD)H$, $(FH)D$, $F(DH)$ with the method FH' . In order to compare the numerical accuracy of the methods, we use the tools developed in [12] and summarized in Section 1.1.

For method FH' , w in Equation 1 is simply the derivative of the interpolation filter h . For other methods, we choose the derivative filter described in Section 2.3. To clarify the notation, we will replace w by dh . To better compute the coefficients $a_n^{dh}(\tau)$ of Equation 2 for the derivative filter dh , we will substitute the convolution sum of the derivative and interpolation filters into the expression for $a_n^{dh}(\tau)$ in Equation 2:

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k - \tau)^n \left(\sum_{l=-\infty}^{\infty} d[l] \cdot h(\tau - k - l) \right)$$

which simplifies to:

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \cdot \left(\sum_{k=-\infty}^{\infty} (k - \tau)^n h(\tau - k - l) \right)$$

Substituting m for $k+l$ in the inner sum, we get:

$$\begin{aligned} a_n^{dh}(\tau) &= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \cdot \left(\sum_{m=-\infty}^{\infty} (m - \tau - l)^n h(\tau - m) \right) \\ &= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \left(\sum_{m=-\infty}^{\infty} \left(\sum_{i=0}^n \binom{n}{i} (m - \tau)^i (-l)^{n-i} \right) h(\tau - m) \right) \end{aligned}$$

which resolves to

$$\begin{aligned} a_n^{dh}(\tau) &= \frac{T^n}{n!} \sum_{i=0}^n \binom{n}{i} \left(\sum_{l=-\infty}^{\infty} (-l)^{n-i} d[l] \right) \left(\sum_{m=-\infty}^{\infty} (m - \tau)^i h(\tau - m) \right) \\ &= \sum_{i=0}^n a_{n-i}^d(0) a_i^h(\tau) \end{aligned}$$

This means that the error coefficient of a convolution filter is simply the convolution of the error coefficients of both filters. In Table 1, we have computed the coefficients for some commonly used filter combinations. The first column shows the error coefficients for the probably most common used filter combination of linear interpolation and central differences, abbreviated by DL. Another common choice is the combination of a cubic interpolation filter (we have chosen the class of cubic cardinal splines) with central differences. We let DC denote this filter class. For the class of analytic derivative filters we have chosen the derivative of the cubic interpolation filter, as introduced in [1]. We use C to represent this filter class.

	central difference + linear interpolation (DL)	central difference + cubic interpolation (DC)	cubic derivative (C)
a_0^d	0	0	0
a_1^d	T	T	$T(1 + (2\alpha + 1)(-6\tau^2 + 6\tau - 1))$
a_2^d	0	$-T^2(2\alpha + 1)\tau(1 - \tau)(1 - 2\tau)$	$3T^2(2\alpha + 1)\tau(1 - \tau)(1 - 2\tau)$
a_3^d (normalized)	$\frac{T^2}{6}(1 + 3\tau - 3\tau^2)$	$\alpha = -0.5 : \frac{T^2}{6}$ $\tau = 0 : \frac{T^2}{6}$ $\tau = 0.5 : \frac{T^2}{6}\left(\frac{6}{4}\alpha + \frac{14}{8}\right)$	$\alpha = -0.5 : \frac{T^2}{6}(6\tau^2 - 6\tau + 1)$ $\tau = 0 : \frac{T^2}{6}$ $\tau = 0.5 : \frac{T^2}{6}\left(\frac{14\alpha + 3}{8\alpha + 12}\right)$

Table 1. Coefficients for some commonly used filter combinations

In the case that $\alpha = -0.5$, a_2^d is zero for all three methods and we must compare a_3^d . One can easily prove that this coefficient for DL is always greater than $T^2/6$ (the coefficient for DC), which in turn is greater than the coefficient for C. This implies that, the worst behavior is observed for DL, and C is more accurate than DC.

Therefore we conclude, that the optimal filter to use is C for $\alpha = -0.5$. However, one might be interested to use different α in different situations. For instance Park and Schowengerdt [13] conclude from a frequency study of the cardinal cubic splines, that some α (different than -0.5) might yield better images. They find that α depends strongly on the underlying function to be reconstructed. Therefore it is of interest to analyze the spatial error for different α as well.

In the case that $\alpha \neq -0.5$, the coefficient for a_2^d is zero only for the method DL. In order to compare the error coefficient among the methods DC and C, we compare a_2^d for both filters. As we have pointed out in [12], these coefficients need to be normalized. Fig. 1 shows a plot of a_2^d after its normalization. Note that T simply scales both plots equally. Therefore, it can be set to one. In Fig. 1, one can clearly see that the error coefficient for DC is smaller than the error coefficient for C. Therefore, we conclude that DC is superior to C when $\alpha \neq -0.5$. This is a rather unexpected result, since one would naturally expect the analytic derivative of a filter to be more exact and therefore to perform better. As we have just seen, this is not necessarily the case.

For the special cases that $\tau = 0$ and $\tau = 0.5$ (where $a_3^d = 0$) we found by comparing a_3^d , that C is more accurate than DC for $\alpha \in [-3, -0.6]$. Another value to consider is the second derivative of the underlying function. When it goes to zero, we also have to use

the error coefficient a_3^d for an error comparison.

We are left to compare the error behavior of the most common method DL with the other two methods. Again, for the special cases, where the second coefficient or the second derivative of the function go to zero, we must compare a_3^d in order to find the most accurate filter. For the other cases however, we can follow the following analysis. If we have influence on the original sampling distance T for our applications, we can always find a T , such that the combination of central difference and linear interpolation is superior to the other two methods. In other words, DL is asymptotically better than DC and C. However in most practical applications we are given a data set with fixed sampling distance T . In these cases we need to weight the actual error of the filters and conclude from this comparison which filter is more accurate. If we are comparing DL and DC, we want to find out for which α DC performs better than DL. Mathematically:

$$\|\varepsilon_{DL}\| \geq \|\varepsilon_{DC}\|$$

Using the second error approximation of Equation 2, we find the following criteria:

$$\|a_3^{DL}(\tau)f^{(3)}(t)\| \geq \|a_2^{DC}(\tau)f^{(2)}(t)\|$$

We can conclude that the choice of α very much depends on the resampling offset τ and the actual data. After some algebraic manipulations, we can conclude:

$$T\frac{5}{4}\left\|\frac{f^{(3)}(t)}{f^{(2)}(t)}\right\| \geq \alpha + 0.5 \geq -T\frac{5}{4}\left\|\frac{f^{(3)}(t)}{f^{(2)}(t)}\right\| \quad (6)$$

For α in this range the method DC is more accurate than DL. As expected, the choice of the most accurate filter strongly depends on the underlying data.

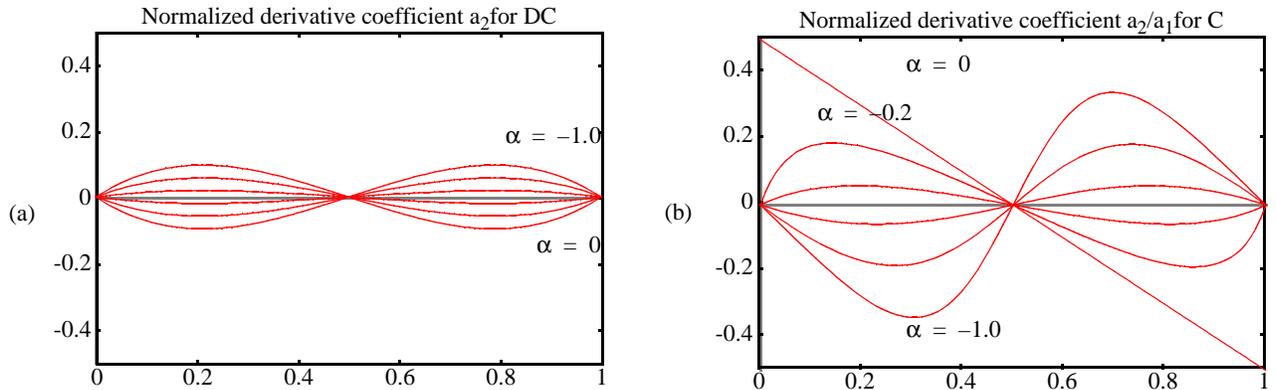


FIGURE 1. In both plots we set T to 1.0. α takes the values 0, -0.2, -0.4, -0.6, -0.8, -1. (a) The coefficient a_2^d of the central difference and cubic interpolation filter for varying α . (b) The normalized coefficient a_2^d/a_1^d of the cubic derivative filter for varying α .

For a similar comparison of the methods *DL* and *C* we find:

$$\frac{1}{\frac{14}{T5} \left\| \frac{f^{(2)}(t)}{f^{(3)}(t)} \right\| + 2} \geq \alpha + 0.5 \geq \frac{1}{\frac{14}{T5} \left\| \frac{f^{(2)}(t)}{f^{(3)}(t)} \right\| + 2} \quad (7)$$

An important observation that we draw from Equation 6 and Equation 7 is the dependency of the comparative accuracy on the sampling distance. The higher the sampling rate the smaller the range in which *C* or *DC* performs better than *DL*. This means that for densely sampled data sets a combination of linear interpolation and central difference is not only efficient, but also recovers the derivative accurately. That can also be explained in the frequency domain. The higher the sampling rate, the further apart the frequency spectra are placed. In other words, the signal's aliases are more separated. Thus, the deficiencies of the central difference operator at higher frequencies do not impose a problem since no signal aliases exist in this frequency range. This is an important and new result, since it tells us, that for some data sets *DL* is just as accurate, as the other two (more expensive) methods *DC* and *C*.

4 EFFICIENCY CONSIDERATIONS

In this section, we compare the four methods $(FH)D$, $(FD)H$, $F(HD)$, and FH' from an efficiency perspective. While the first three methods are equivalent from an accuracy standpoint, they are not so from an efficiency point of view. This section also contrasts the overall computational effort of these four shading-deferring methods with the demands of the popular, but less accurate, pre-shading scheme [8]. We denote this approach as $((FD)CS)H$, where *C* and *S* stand for classification and shading that occur after gradient computation but before color interpolation. Our comparisons will be valid for the 3D case only (a typical application will be volume rendering algorithms). If we compare normal estimation schemes in other dimensions, our analysis will be similar.

In the following discussion, we distinguish between implementations that compute all results on-the-fly, and implementations that utilize some form of caching to reuse previously computed results. The latter approaches obviously requires an extra amount of memory and cache management overhead. We now introduce some terminology:

E_λ : The computational effort to apply the operator λ where λ can be *H*, *D*, *DH*, *H'*, or *CS*.

$\|\lambda\|$: Number of filter weights used for applying the operator λ .

n: Number of data elements (voxels).

m: Number of samples.

For digital filters $\|\lambda\|$ is obviously the length of the filter, but for continuous filters (e.g. cardinal splines) $\|\lambda\|$ is usually the filter support, i.e. the number of sample values, that are influenced by the filter. Since the filter operation is the weighted sum of $\|\lambda\|$ elements, we usually have $E_\lambda = 2\|\lambda\| - 1$ for a straight forward implementation of $\|\lambda\|$ multiplications and $\|\lambda\| - 1$ additions. However, for some special filters, there will be a more efficient implementation. For instance, the central difference filter (in one dimension), can be implemented in 2 operations (one subtraction and one division by 2) as opposed to 3 operations (two multiplications by 0.5 and one subtraction). Therefore, we find it important to separate between E_λ and $\|\lambda\|$.

In the following discussion we will discuss the cost of reconstructing the function and its derivative at all the sample points. We will also comment on the cost of applying the classification and shading operators.

4.1 $(FD)H$ - Derivative First

In this method, we first compute the gradient at all grid voxels

within the extent of the interpolation filter *h*, and then interpolate these gradients using *H*. An on-the-fly approach would have to compute $\|H\|$ gradients for a total cost of $\|H\|E_D$, followed by three interpolations to compute the three gradient components and one interpolation to compute the data value itself. The total cost is thus:

$$m(\|H\| \cdot E_D + 3E_H + E_H)$$

By storing computed gradients in a gradient volume, one could reduce the cost to:

$$n \cdot E_D + m(3E_H + E_H)$$

The process of classification and shading will require additional $m \cdot E_{CS}$ cost and the total cost will then be:

$$n \cdot E_D + m(4E_H + E_{CS})$$

However, in the $((FD)CS)H$ method, classification and shading are applied to the data values, and the interpolation filter is applied to the resulting $RGB\alpha$ values. Therefore, the total cost for this method, assuming caching, is:

$$n \cdot (E_D + E_{CS}) + m(4E_H)$$

Since in most cases, to assure proper sampling, $n \leq m$, the computational advantage of this method is clear. Moreover, when classification and illumination does not change for multiple rendering, the cost of the first component in the last two equations is amortized and can therefore be ignored. If we ignore the shading component then the cost of reconstructing the function and its derivative assuming caching is given by:

$$n \cdot E_D + m(4E_H)$$

4.2 $(FH)D$ - Interpolation First

The $(FH)D$ method computes the derivative at a ray sample location from a set of additionally interpolated samples in the neighborhood of the sample location. In parallel (orthographic) rendering of volumes the data is resampled into a new grid. If this grid is cached somewhere, one can perform the derivative calculations using the data values at that grid.

Without caching, in order to compute the derivative at a sample location, $(FH)D$ interpolates $\|D\|$ additional samples, each at a cost of E_H , and uses them to obtain three axis derivatives at the cost E_D . Another interpolation at the sample location, each at a cost of E_H , yields the function value. The total cost for reconstructing the function and its derivative is:

$$m(\|D\| \cdot E_H + E_D + E_H)$$

Later, these samples are classified and shaded, with an additional cost (for the whole volume) of $m \cdot E_{CS}$. However, if caching is employed, only one interpolation is needed per sample, and the *D* operator uses only existing samples. Therefore the total cost for reconstructing the function and its derivative:

$$m(E_D + E_H)$$

4.3 $F(DH)$ - Continuous Derivative

Here the derivative filter is pre-convolved with the interpolation filter which increases its size. The gradients are then computed by convolving the volume by this combined *DH*-filter. The total cost for computing the function and its derivative is then given by:

$$m(E_{DH} + E_H)$$

This is the most direct method of the three methods presented so far and there is no caching mechanism available to gain some speedup.

4.4 Method FH' - Analytic Derivative

This method is not equivalent to the previous three in terms of accuracy, as other sections of this paper demonstrate. FH' uses a special gradient filter derived from the interpolation filter to estimate the gradients. Since this derivative filter has the same size as H the corresponding cost for computing the function and its derivative is:

$$m(E_H + E_{H'}) .$$

4.5 Summary and Numerical Examples

We are now ready to compare the theoretical cost functions presented in the previous subsections and provide some numerical examples to highlight the differences. Table 2 lists all costs derived above and gives two numerical examples: In case 1, H and H' are cubic filters ($\|H\| = 4^3$), D is a central difference filter ($\|D\| = 6$), and in case 2, H is a trilinear filter ($\|H\| = 2^3$) and D is again the central difference filter. For the following discussion we count the number of floating point operations associated with each operator, but we do not distinguish between additions, multiplications or divisions. In this case, the cost of E_H for H being cubic is $2 \cdot 4^3 - 1 = 127$, of E_H for H being trilinear is $2 \cdot 2^3 - 1 = 15$, of E_D for D being central difference the cost is 6. Since the derivative filters are directional filters, and E_{DH} denotes the cost of computing all three derivative components, we find that E_{DH} is three times the cost of one derivative component operation. In order to find the directional derivative, we convolve the interpolation filter of size $s \cdot s \cdot s = s^3$ with a 1D derivative filter of length k (in our case - central differences - $k = 2$). That results in a filter of size $(s + k - 1) \cdot s \cdot s$. Therefore we find the cost of E_{DH} for H being cubic is 477 and for H being trilinear is 69.

As expected the analytical derivative method (FH') is the most efficient one. However, as we showed in Section 3, it is not necessarily the most accurate. Among the other three schemes (which are numerically identical), we find our new method ($F(DH)$) most efficient if there is no caching. However, if caching is available, (FH) D is certainly the most efficient way to compute the normal and the data value at this point. Therefore, we conclude that in terms of efficiency and in terms of accuracy, there is no need for the most commonly used method (FD) H (in the case of deferred shading). As was pointed out already in Section 4.1, if we do shading at the grid locations, we might find a more efficient algorithm, yet trading speed for accuracy.

5 EXPERIMENTS

The images were rendered employing a simple raycaster to find isosurfaces. The volumes were resampled at an interval of 0.05 voxel lengths. At each sampling point, the raycaster first applied the interpolation kernel to reconstruct the function at that point. If the reconstructed value was above a pre-set isovalue, the derivative filter was used to compute the 3D gradient. Shading was then performed using the traditional Phong lighting model [5] with diffuse and specular reflections. The obtained color and opacity were composited with the previous ray values, and the ray was terminated after the opacity reached a value close to one. Since for all our filters both the interpolation and the derivative kernel were separable, the filter operations could be efficiently performed using a scheme similar to the one given by Bantum et al [1].

For our experiments we used an analytic data set, derived from the same function as the one used by Marschner and Lobb [11]. Specifically, we used:

$$f(x, y, z) = \frac{1}{2} - \frac{2}{5} \sin\left(\frac{\pi}{2} z\right) + \frac{1}{10} \cos\left(12\pi \cos\left(\frac{\pi}{2} \sqrt{x^2 + y^2}\right)\right)$$

Since we study different derivative filters, we have fixed the

interpolation filter to be the Catmull-Rom interpolation filter - a cubic filter with small error as was also shown in [12]. From Equations 6 and 7 we learn that the range of α where C and/or DC performs better than DL is dependent on the data set. To address this issue, we have computed the ratio $\|f^{(3)}(t)/f^{(2)}(t)\|$ analytically for the data points for the three axis directions x , y , and z , where we reconstruct and collected them in a histogram, plotted in Fig. 2. In order to guarantee that all data points are reconstructed more accurately using DC (or C) than DL, we would have to choose the minimal ratio. This ratio is zero and therefore we can conclude that only for $\alpha = -0.5$ we can guarantee, that the derivative reconstruction at any single point will be better for the methods DC and C as opposed to DL. In order to get practical results, we could choose a higher ratio of $\|f^{(3)}(t)/f^{(2)}(t)\|$, giving up on the accuracy assurance for some reconstructed values. If we for instance choose the ratio 7, we still guarantee all z directional derivatives to be estimated more accurately. Approximately 8% of the directional derivatives in y will be more accurate by DL, and only 3.8% of the directional derivatives in x will be better by DL.

When we plug in the ratio of 7 into Equations 6 and 7, we find the theoretical result that for $\alpha \in [-0.78, -0.22]$, DC performs better than DL and for $\alpha \in [-0.65, -0.34]$, C performs better than DL. These theoretical ranges have steered our experiments and in Fig. 3 (see color plates) we have rendered the Marschner-Lobb data set for several different α . For a better (analytical) understanding of these rendered images, we have also drawn the angular error images in Fig. 4. For each reconstructed normal we computed the actual normal and recorded their angular difference. The grey value of 255 was displayed for an angular error of 5 degrees.

For the first row of images we have used $\alpha = -0.5$. Following our analysis in Section 3, we expect that $\epsilon(C) < \epsilon(DC) < \epsilon(DL)$, where $\epsilon(A)$ denotes the error measure of image A. The first row of Fig. 3 shows the different images for $\alpha = -0.5$. Although the differences are small, one can find DC to be better, than DL. Although the image for DC is overall smoother, it's error image in Fig. 4 reveals a much higher error than for C.

The images for $\alpha = -1.0$ show the opposite behavior. From our analysis we deduce the following error behavior: $\epsilon(DL) < \epsilon(DC) < \epsilon(C)$. From Fig. 3 we conclude, that C clearly is the worst image. Also a visual comparison of DC and DL leads to the conclusion, that DL is better than DC. The error images in Fig. 4 support this analysis.

The rows for $\alpha = -0.6$ and $\alpha = -0.7$ show rather a transitional phase. Since the change of the filter weights happens continuously,

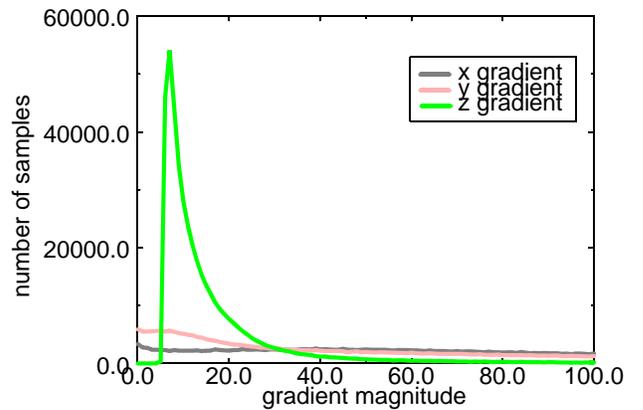


FIGURE 2. The ratio of $\|f^{(3)}(t)/f^{(2)}(t)\|$ for the directional derivatives in x , y , and z direction respectively for the Marschner Lobb data set.

	Theoretical Cost		Cubic Interp. and Central Diff.		Trilinear Interp. and Central Diff.	
	No Caching	Caching	No Cache	Cache	No Cache	Cache
$(FD)H$	$m(\ H\ \cdot E_D + 3E_H + E_H)$	$n \cdot E_D + m(4E_H)$	1084m	9n+508m	132m	9n+60m
$((FD)CS)H$	---	$n \cdot E_D + m(4E_H)$	---	9n+508m	---	9n+60m
$(FH)D$	$m(\ D\ \cdot E_H + E_D + E_H)$	$m(E_D + E_H)$	898m	136m	114m	24m
$F(DH)$	$m(E_{HD} + E_H)$	---	604m	---	84m	---
FH'	$m(2E_H)$	---	254m	---	30m	---

Table 2. Comparison of efficiency of the normal estimation schemes

we cannot necessarily expect a sudden sharp change in the image quality. The differences in the image quality can be better studied using the error images in Fig. 4. We can conclude, that for $\alpha = -0.6$ our results follow our theoretical analysis: $\epsilon(\text{DC}) < \epsilon(\text{C}) < \epsilon(\text{DL})$. However, for $\alpha = -0.7$ it is debatable, which method is preferable in terms of image quality. Analytically we show $\epsilon(\text{DC}) < \epsilon(\text{DL}) < \epsilon(\text{C})$. It is clear, that the image for C is the least appealing to the viewer.

6 CONCLUSIONS AND FUTURE GOALS

We have classified the different techniques of normal estimation into four groups, and we have developed a new scheme $F(DH)$. We showed that the schemes $(FD)H$, $(FH)D$ and $F(DH)$ are numerically equivalent, and then extended the idea of classifying filters using Taylor series expansion to the convolution of two filters. We found that computing the analytic derivative of a filter kernel (method FH') is not always more accurate than using a combination of that filter with the central difference kernel (any of the methods FDH). Therefore, a careful analysis of existing filters and filter combinations is suggested.

The new scheme $F(DH)$ opens up new ways to design continuous derivative filters. Furthermore, this method of normal estimation is also the second most cost-efficient one, if no caching is performed (with FH' being the most cost effective one). However, if caching is enabled, then the method $(FH)D$ is clearly preferable over any other method in terms of efficiency. In fact, what is believed as one of the most commonly used methods, $(FD)H$, is one of the slowest normal estimation method. The only advantage one could gain is the pre-calculation of the shading operation at the grid voxels, as Levoy [8] has proposed it. However, as was pointed out in Section 2, this method is certainly not preferable if accurately rendered images are required.

One of our immediate goals is to compare various combinations of known derivative and interpolation filters in order to find new derivative filters. We also would like to extend the error analysis to frequency space so that we can examine any aliasing and smoothing errors. Finally, it would contribute to the accuracy of our analysis to include a noise model. We also believe that it is very important to further investigate the shading and classification steps in terms of numerical accuracy.

Acknowledgments

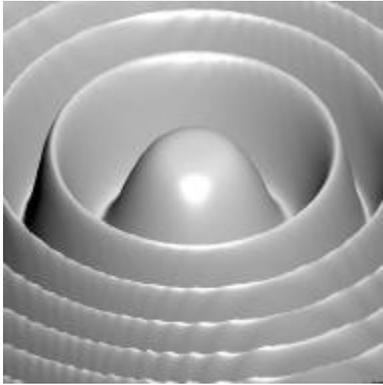
We thank Prof. Wayne Carlson and the Advanced Computing Center for the Arts and Design for the use of their computing facilities and Prof. Robert Moorhead of the NSF Engineering Research Center, Mississippi State University as well as Tom Malzbender of Hewlett-Packard Labs for providing encouragement and support. This project was partially supported by the Department of Defense USAMRDC 94228001, by the Advanced Research Projects Agency Contract DABT63-C-0056 and the NSF Research Center

for Computational Field Simulations.

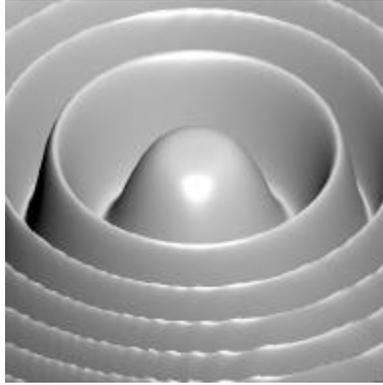
References

- [1] Benthum M.J., Lichtenbelt B.B.A., Malzbender T., "Frequency Analysis of Gradient Estimators in Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242-254, September 1996.
- [2] Bracewell R.N., *Two Dimensional Imaging*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1995.
- [3] Drebin R.A., L. Carpenter L., Hanrahan P., "Volume Rendering", *Computer Graphics*, 22(4):51-58, August 1988.
- [4] Dutta Roy S.C., Kumar B., "Digital Differentiators", in *Handbook of Statistics*, N. K. Bise and C. R. Rao eds., vol. 10: 159-205, 1993.
- [5] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., *Computer Graphics, Principles and Practice*, Addison-Wesley, Reading, Massachusetts, 1990.
- [6] Goss M.E., "An Adjustable Gradient Filter for Volume Visualization Image Enhancement", *Proceedings of Graphics Interface '94*, pp. 67-74, Toronto, Ontario, 1994.
- [7] Hamming R.W., *Digital Filters*, Prentice Hall Inc., Second Edition, Englewoods Cliffs, NJ, 1983.
- [8] Levoy M., "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, 8(5):29-37, May 1988.
- [9] Lorensen W.E., Cline H., "Marching Cubes: a High Resolution 3D Surface Reconstruction Algorithm", *Computer Graphics*, 21(4):163-169, July 1987.
- [10] Machiraju R.K., Yagel R., "Reconstruction Error Characterization and Control: A Sampling Theory Approach", *IEEE Transactions on Visualization and Computer Graphics*, ITVCG 2(4):364-376, December 1996.
- [11] Marschner S.R. and Lobb R.J., "An Evaluation of Reconstruction Filters for Volume Rendering", *Proceedings of Visualization '94*, IEEE CS Press, pp. 100-107, October 1994.
- [12] Möller T., Machiraju R.K., Mueller K., Yagel R., "Evaluation and Design of Filters Using a Taylor Series Expansion", *IEEE Transactions on Visualization and Computer Graphics*, ITVCG 3(2): 184-199, June 1997.
- [13] Park S.K., Schowengerdt R.A., "Image Reconstruction by Parametric Cubic Convolution", *Computer Vision, Graphics, and Image Processing*, vol. 23:258-272, 1983
- [14] Pfister H., Kaufman A.E., "Cube-4 - A Scalable Architecture for Real-Time Volume Rendering", *Proceedings of the 1996 Symposium on Volume Visualization*, pp. 47-54, October 1996.
- [15] Warsi Z.U.A., *Fluid Dynamics: Theoretical and Computational Approaches*, CRC Press, Boca Raton, FL, 1992.
- [16] Westover L., "Footprint Evaluation for Volume Rendering", *Computer Graphics (SIGGRAPH '90 Proceedings)*, vol. 24:367-376, August 1990.

central difference +
linear interpolation (DL)



central difference +
cubic interpolation (DC)



cubic derivative (C)

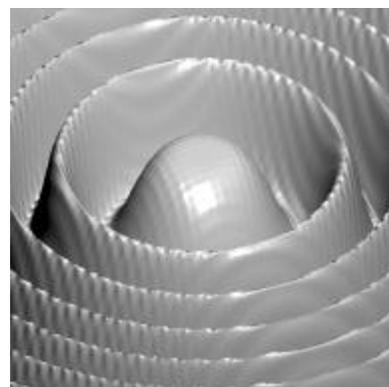
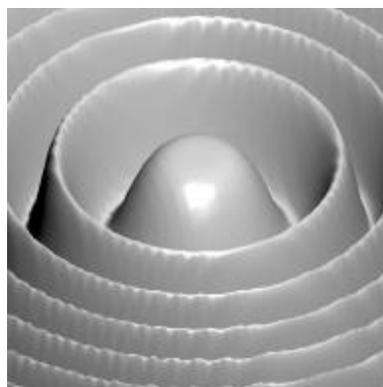
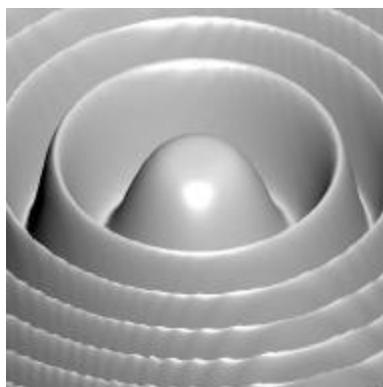
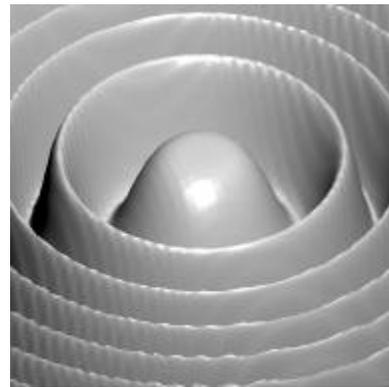
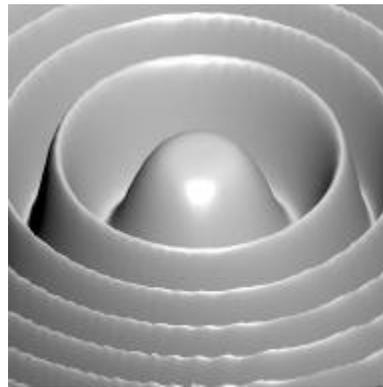
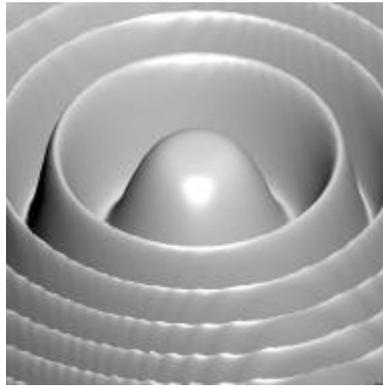
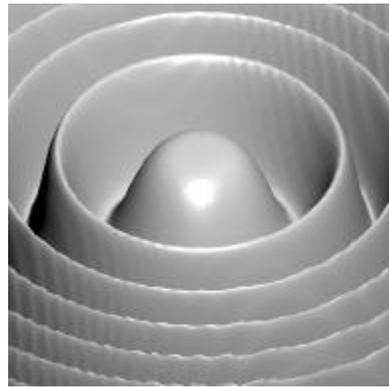
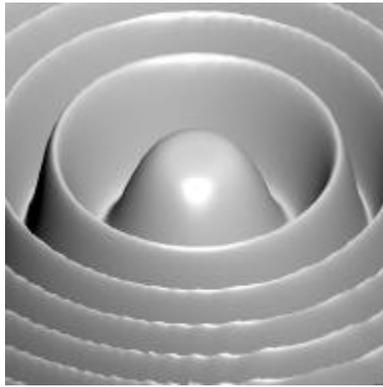
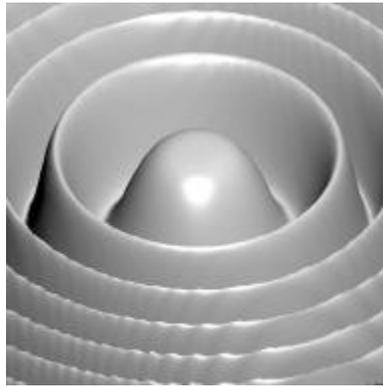
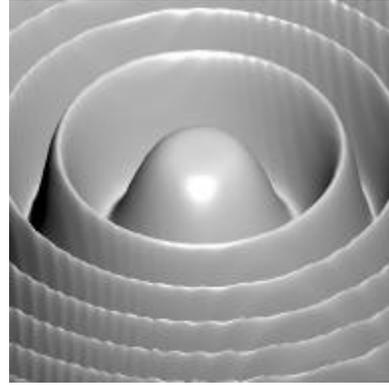


FIGURE 3. Marschner Lobb data set.

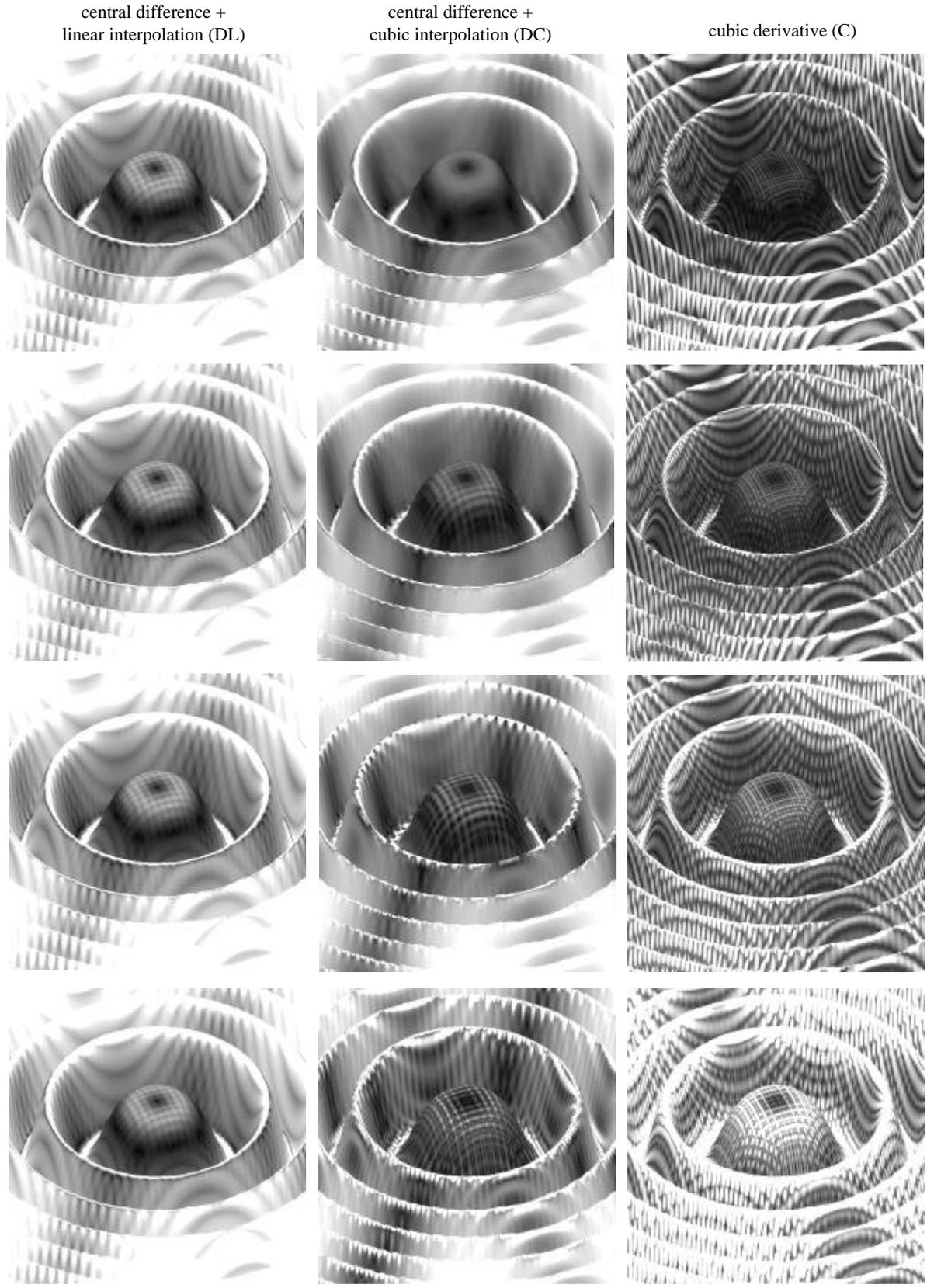


FIGURE 4. Error images of the Marschner Lobb images in Fig. 3 (see color plates)