Evaluation and Design of Filters Using a Taylor Series Expansion

Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel

Abstract—We describe a new method for analyzing, classifying, and evaluating filters that can be applied to interpolation filters as well as to arbitrary derivative filters of any order. Our analysis is based on the Taylor series expansion of the convolution sum. Our analysis shows the need and derives the method for the normalization of derivative filter weights. Under certain minimal restrictions of the underlying function, we are able to compute tight absolute error bounds of the reconstruction process. We demonstrate the utilization of our methods to the analysis of the class of cubic BC-spline filters. As our technique is not restricted to interpolation filters, we are able to show that the Catmull-Rom spline filter and its derivative are the most accurate reconstruction and derivative filters, respectively, among the class of BC-spline filters. We also present a new derivative filter which features better spatial accuracy than any derivative BC-spline filter, and is optimal within our framework. We conclude by demonstrating the use of these optimal filters for accurate interpolation and gradient estimation in volume rendering.

Index Terms—Interpolation filters, derivative filters, filter design, normal estimation, gradient estimation, normalization of filters, Taylor series expansion, volume rendering, cubic filters.

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 evaluate the rendering integral. The gradient (the first derivative of the function) is employed in volume classification and shading [5], [10].

1.1 Assumptions

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. In order to employ Taylor series expansion, the foundation of our method, we require that the first N derivatives of the function exist, where N depends on our error analysis. This condition is usually met in practice, since image and volume acquisition devices, such as scanners and cameras, inherently perform a lowpass filtering operation that bandlimits the functions [2]. Numerical simulations of physical phenomena, as performed in 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. Finally, all rendering and scan-conversion algorithms, in order to provide antialiased images, generally employ a filtering step that bandlimits the image.

1.2 Motivation

Before resampling we must reconstruct from f_k the continuous function, which we denote by $f_r^h(t)$. Here, h denotes the low-pass interpolation filter. Some applications (e.g., volume rendering) perform a gradient-based shading operation. Thus, we need to reconstruct the derivative of $f(t)$ from the known samples f_k. In the following, the derivative of the continuous function $f(t)$ is denoted by $f'(t)$, and the reconstructed derivative is denoted by $f_r^d(t)$. Here, d stands for the high-pass derivative filter. In most applications, efficiency considerations steer the filter selection, ignoring the adverse effect the chosen filters h and d may have on image quality. The trilinear and central difference filters are often used for the reconstruction of the underlying function and its derivative, respectively, because they do not require much computational effort. However, the use of the trilinear filter leads to blurring and aliasing in the final image [13], while the application of the central difference filter results in the loss of fine detail [1].

With recent advances in hardware (e.g., the emergence of fast CPUs, DSP boards, texture mapping hardware), we can now afford to use computationally more expensive filters with better image quality characteristics than trilinear and central-difference filters. To enable the evaluation of the performance of these more expensive filters, there exists a need for both subjective and objective metrics. Subjective measures usually involve real human subjects who judge the suitability for particular applications. Objective metrics are computational in nature, and may or may not include the human observer. If the human visual system is included, it is usually in the form of a numerical model. For

- *T. Möller and R. Yagel are with the Department of Computer and Information Science and The Advanced Computing Center for the Arts and Design, The Ohio State University, 395 Dreese Lab, 2015 Neal Avenue, Columbus, Ohio 43210-1277. E-mail: {moeller, yagel}@cis.ohio-state.edu.*
- *R. Machiraju is with the U.S. National Science Foundation Engineering Research Center for Computational Field Simulation, Department of Computer Science, Mississippi State University, P.O. Box 9627, Mississippi State, Mississippi 39762-9627. E-mail: raghu@erc.msstate.edu.*
- *K. Mueller is with the Department of Computer and Information Science, The Ohio State University, 395 Dreese Lab, 2015 Neal Avenue, Columbus, Ohio 43210-1277. E-mail: mueller@cis.ohio-state.edu.*

For information on obtaining reprints of this article, please send e-mail to: transvcg@computer.org, and reference IEEECS Log Number 104720.0.

example, the frequency response of the human visual system can be represented by the *Contrast Sensitivity Function* [12]. Thus, the resulting metric will measure the response of the visual system to frequencies. Objective metrics are used to measure the numerical accuracy of the filters, and can be further subdivided into quantitative and qualitative metrics. Qualitative methods allow us to make general judgments about the numerical accuracy of filters, whereas quantitative methods allow us to evaluate the absolute error. The latter has the potential to lead to improved local reconstruction methods as was shown by Machiraju and Yagel [11]. Quantitative methods are useful, since they provide an error metric to compare and contrast filters. They also provide a means to select optimal filters. On the other hand, qualitative methods allow the classification of the filters into categories, and are based on the asymptotic behavior of the error in terms of the original sampling distance.

For our evaluation methods, we require that the function is included in the evaluation. In general, metrics should be independent of the function. However, for certain situations, it is useful that the function be included, especially if the filter is to be optimized for a particular function. Also, the evaluation should be conducted in the spatial domain instead of the frequency domain. Frequency domain methods, when used for evaluating function reconstruction, can only measure global errors, as it is assumed that the convolution operation in the spatial domain is invariant (e.g., the function interpolation occurs at the same inter-sample distance everywhere using the same filter kernel).

The outline of this paper is as follows. Section 2 summarizes previous research that has been done in this field. In Section 3, we introduce the Taylor series expansion of the convolution sum. Due to their importance, we focus on interpolation and first derivative filters. In Section 4, we illustrate an application of our method to the group of cubic BC-spline interpolation and first derivative filters. In Section 5, we present some experimental results, and in Section 6, we suggest steps for furthering this research. Finally, in Section 7, we summarize our findings.

2 Related Work

Researchers have generally studied and evaluated filters in the frequency domain. One of the earliest comparative studies of interpolation filters for image resampling was done by Parker et al. [17]. Here, a comparison of nearest neighbor, linear, cubic B-splines, and two members of the class of Cardinal cubic splines was conducted via a discussion of their respective frequency spectra.

A thorough study of Cardinal cubic splines in the frequency domain was performed by Park and Schowengerdt [16]. They found that the optimal interpolation filter of this class highly depends on the signal to which it will be applied. For most signals, the parameter α was found to be around –0.6 and –0.5, where the latter corresponds to the Catmull-Rom spline. Keys [9] showed that, within the class of Cardinal cubic splines, the Catmull-Rom spline is optimal, in the sense that it interpolates the original function with the smallest asymptotic spatial error. By using a Taylor series expansion of the convolution sum, he found that the Catmull-Rom interpolation filter has an error proportional to the cube of the sampling distance.

Mitchell and Netravali [14] introduce a more general class of cubic splines, which we refer to as BC cubic splines, or, for short, BC-splines. Cardinal cubic splines are a subclass of the BC-splines. Mitchell and Netravali conducted a study involving more than 500 sample images, classifying the parameter space into different regions of dominating reconstruction artifacts, such as blurring, ringing, and anisotropy. They found, by using a Taylor series expansion, that filters for which $B + 2C = 1$ are the most accurate numerically within the class of BC-splines, and have an error proportional to the square of the sampling distance. Neither Keys nor Mitchell and Netravali evaluate the absolute error of their filters.

A comparative study by Marschner and Lobb [13] proposes the use of different error metrics for various reconstruction artifacts. These error metrics are based in the frequency domain, and measure the smoothing, postaliasing, and overshoot of an interpolation filter. In this study, the windowed *Sinc* filter was found to behave the best. However, their metrics do not depend on the actual function to be reconstructed or intersample distance. This is unfortunate in light of our earlier comments on filter optimization, and was found to be crucial by Park and Schowengerdt in their frequency domain analysis.

All the aforementioned approaches neglect to take into account the effect of derivative filters on the quality of the rendered image. In studies that aim at comparing a given set of filters, the effects of interpolation and derivative filters must be clearly separated. Otherwise, one may mask the effect of the interpolation filter by that of the derivative filter, or vice-versa. For instance, Marschner-Lobb [13], in their analysis of BC-spline interpolation filters, varied both the interpolation and derivative filters at the same time. We will show that many of the effects attributed to the interpolation filter are really due to the corresponding derivative filter. A good survey of existing derivative filters can be found in the paper by Dutta Roy and Kumar [6], in which the design of maximal linear filters in the frequency domain is described. The filter design outlined there can be easily adapted to various frequency ranges, which is an important property for practical applications.

Goss [8] 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. In [1], the ideal derivative filter is shown to be the derivative of the *Sinc* filter, which we denote as the *Cosc* filter. Bentum et al. [1] use the Cardinal cubic splines as a basis to develop derivative filters. Although the authors illustrate the effect of various parameters on these filters via a number of frequency plots, they do not analytically compare the different filters.

While most of the existing research concentrates on frequency analysis, we believe that spatial domain analysis is just as important. If the local error can be kept small, then the effect of image artifacts also diminishes. We find that the results of Keys' [9] spatial analysis are nearly identical to the results of the frequency analysis done by Park and Schowengerdt [16].

In this work, we develop tools for the spatial analysis of both interpolation and derivative filters of arbitrary order. We show the importance of normalizing the filter coefficients, and how this step is performed. Our method also yields an absolute error bound of the reconstruction process. Specifically, for the class of BC-splines, the application of the new method allows us to derive both known and new results (see Section 4).

3 Taylor Series Expansion of the Convolution Sum

To reconstruct a continuous function *f*(*t*) or its *n*th derivative $f^{(n)}(t)$ from a set of sample points f_k, we compute a weighted average of these samples. The sample points are multiplied with the proper filter weights when convolving them with a continuous filter *h*. By convolving the sampled signal f_k with *h*, we can reconstruct an approximation of the original function *f*(*t*). Similarly, if we convolve the samples with a continuous derivative filter *d*, we can reconstruct an approximation of the *n*th derivative of the original function. We denote the result of this operation by $f_r^w(t)$, where *w* denotes the type of filter used. Formally, this can be written as:

$$f_r^w(t) = \sum_{k=-\infty}^{\infty} f_k \cdot w\left(\frac{t}{T} - k\right) \tag{1}$$

The values f_k represent the samples of the original function *f*(*t*) at the locations *kT*, where *T* is the sampling distance. Assuming that the first (*N* + 1) derivatives of *f*(*t*) exist, we can expand $f_k = f(kT)$ as a Taylor series in *f* about *t*. Therefore, we write:

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

where $(\xi_k) \in [t, kT]$. Substituting this Taylor series expansion into the convolution sum of (1) and reordering the terms according to the derivatives of *f*(*t*), we can rewrite (1) as:

$$f_r^w(t) = \sum_{n=0}^{N} a_{n,T}^w(t) f^{(n)}(t) + r_{N,T}^w(t) \tag{2}$$

where the coefficients $a_{n,T}^w(t)$ and the remainder term $r_{N,T}^w(t)$ are, respectively:

$$a_{n,T}^w(t) = \frac{1}{n!} \sum_{k=-\infty}^{\infty} (kT - t)^n w\left(\frac{t}{T} - k\right)$$

$$r_{N,T}^w(t) = \frac{1}{(N+1)!} \sum_{k=-\infty}^{\infty} f^{(N+1)}(\xi_k)(kT - t)^{(N+1)} w\left(\frac{t}{T} - k\right)$$

Let us first have a look at the coefficients $a_{n,T}^w$. In practice, a filter *w* cannot be infinitely long, but must have a finite extent *M*. Therefore, *w* is defined to be zero outside this interval [−*M*, *M*]. Furthermore, we observe that $a_{n,T}^w$ is a periodic function with period *T*. That can easily be verified:

$$\begin{aligned} a_{n,T}^w(t+T) &= \frac{1}{n!} \sum_{k=-\infty}^{\infty} \left(kT - (t+T)\right)^n w\left(\frac{t+T}{T} - k\right) \\ &= \frac{1}{n!} \sum_{k=-\infty}^{\infty} \left((k-1)T - t\right)^n w\left(\frac{t}{T} - (k-1)\right) \\ &= a_{n,T}^w(t) \end{aligned}$$

Therefore, $a_{n,T}^w$ can be more efficiently studied, if we express *t* in terms of a normalized offset *τ*:

$$t = (i + \tau)T, \text{ where } 0 \le \tau < 1, \text{ and } i = \left\lfloor \frac{t}{T} \right\rfloor, \text{ or}$$

$$\tau = \frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor$$

Hence, the coefficients $a_{n,T}^w$ can be expressed in terms of this offset:

$$a_{n,i,T}^w(\tau) = a_n^w\left((i+\tau)T\right) = \frac{1}{n!} \sum_{k=i-M}^{i+M} \left((k-i)T - \tau T\right)^n w\left(\tau - (k-i)\right)$$

which can be simplified to:

$$a_{n,i,T}^w(\tau) = \frac{T^n}{n!} \sum_{k=-M}^{M} (k - \tau)^n w(\tau - k) \tag{3}$$

Since $a_{n,i,T}^w$ does not depend on *i*, and *T* is set by the data acquisition step (and thus cannot be changed during the reconstruction process), we will drop the appropriate subscripts in the expression for a_n^w. Here, we note that this formulation provides a convenient way to study the performance of the filter *w* according to the inter-sample distance *τ*.

If we repeat the same analysis for the remainder term r_n^w, we find:

$$r_{N,i}^w(\tau) = \frac{T^{(N+1)}}{(N+1)!} \sum_{k=-M}^{M} f^{(N+1)}(\xi_{k,i})(k - \tau)^{(N+1)} w(\tau - k) \tag{4}$$

where $\xi_{k,i} \in [t, (k+i)T]$. Finally, the convolution sum in (1) can be expressed as:

$$f_r^w(t) = \sum_{n=0}^{N} a_n^w(\tau) f^{(n)}(t) + r_{N,i}^w(\tau) \tag{5}$$

Equation (5) resulted from applying a sequence of algebraic manipulations that transformed the convolution sum of (1) into a form that we can better study. Our next goal is to interpret this equation, and to derive criteria to help us evaluate our filter *w*. In order to do so, let us rewrite (5) in a more convenient form:

$$\begin{aligned} f_r^w(t) = a_0^w(\tau)f(t) + a_1^w(\tau)f'(t) + a_2^w(\tau)f''(t) + \ldots + \\ a_N^w(\tau)f^{(N)}(t) + r_{N,i}^w(\tau) \end{aligned} \tag{6}$$

From this expansion, we see that, in the ideal case, all coefficients a_k^w are zero, except for the one belonging to the derivative that we want to reconstruct. Furthermore, the remainder term should also evaluate to zero in the

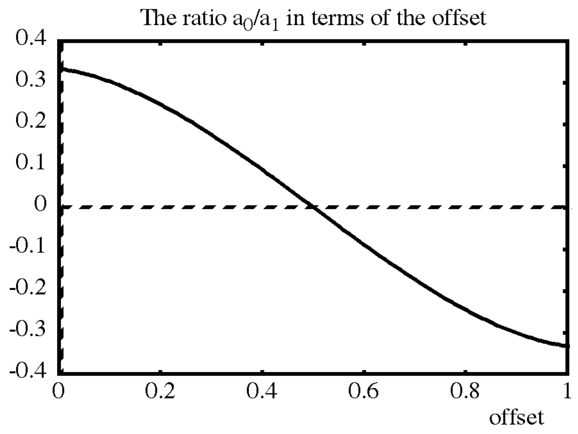


Fig. 1. The normalized coefficient a_0^w of the truncated *Cosc* filter in terms of the offset to the next sampled value. It is only zero for $\tau = 0.5$.

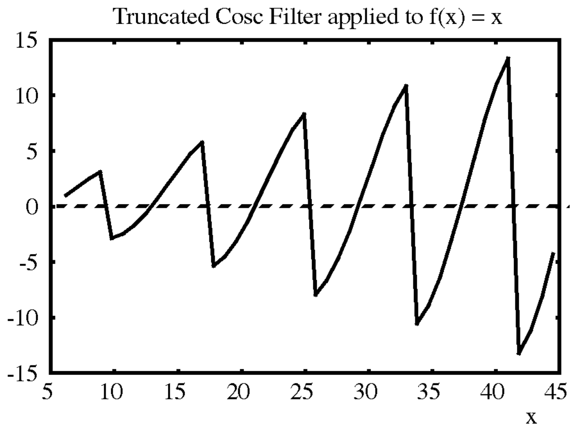


Fig. 2. Derivative reconstruction of the function $f(x) = x$ using the truncated *Cosc* filter at $M = 3$. Since the coefficient a_0^w is not zero, we end up with artifacts that make the filter useless. (The expected result is the constant function $f(x) = 1$.)

ideal case. Thus, for interpolation filters, we need a_0^w to evaluate to one and all other coefficients to evaluate to zero. On the other hand, for first derivative filters, we need a_1^w to be one with the other coefficients being zero. As a result of this observation, we use (6) to derive four filter evaluation criteria dealing with analysis, normalization, classification, and error estimation. We describe these criteria in the following sections.

Before we end this section, we would like to compare our methods to previous methods in filter design. This method outlined here has been used to specify and design filters, especially those used in wavelet-based multiresolution analyses [3]. In [3], a filter w is designed to have the following n-moment property, namely:

$$\int x^p w(x)dx = 0 \qquad 0 \le p \le n$$

Thus, the filter w can represent functions which are n-polynomials accurately. The design of filters as espoused here is different from classical techniques. Similarly, in [18], polynomial smoothing filters are designed which match the function in a least-square sense. On close inspection of (3), one can conclude that it represents a discrete evaluation of function moments. Also, the evaluation depends on the location within each cell. The novelty of the approach lies not in designing filters, but to actually use the technique to evaluate the accuracy of function and (any order) derivative reconstruction. Rather, filter design occurs as a by-product of the analysis.

3.1 Analysis

According to (6), a filter's characteristics can be analyzed by its Taylor series coefficients. For example, an interpolation filter will have a nonzero value for a_0, while a first derivative filter will require a_0 to be zero, and a_1 to be nonzero. We can determine the type of the filter we are dealing with by finding the *first* Taylor coefficient a_n^w that is nonzero. More formally speaking, we need to find the largest n, such that $a_i^w = 0, \forall(i < n)$. The importance of this test can be demonstrated by an example where the filter, at first glance, seems to be a first derivative filter. An analysis using (6), however, shows that this filter is far from perfect.

The reconstruction of the first derivative requires a_0^w to be zero. In fact, if this condition is violated, the reconstruction result may not be very useful. For our example, let us examine the *Cosc* filter, the ideal derivative filter. It has been shown that the ideal derivative filter is simply the derivative of the ideal interpolation filter *Sinc* [1]. Therefore, we have

$$\mathrm{Cosc}(t) = \mathrm{Sinc}'(t) = \begin{cases} 0 & t = 0 \\ \dfrac{1}{t}\left(\cos(\pi t) - \dfrac{\sin(\pi t)}{\pi t}\right) & t \neq 0 \end{cases}$$

The *Cosc* filter (like the *Sinc* filter) is an infinite filter (IIR), and thus not very practical for graphics and rendering. One possible solution could be to use a truncated *Cosc* filter, which is equal to the *Cosc* filter for all $|t| \le M$ and zero outside of this interval. However, we cannot simply use a truncated *Cosc* filter, since a_0^w will not be zero. To demonstrate this, let us set M to three, which results in six filter weights for the reconstruction. In Fig. 1, we plot the normalized coefficient $a_0^w(\tau)/a_1^w(\tau)$ of the truncated *Cosc* filter (we introduce the concept of normalization in the next section). It can be seen that this function varies between –0.4 and 0.4 instead of being zero as required from a derivative filter. Notice the behavior of this specific truncated *Cosc* filter when applied to a linear function as shown in Fig. 2. We would expect a function close to constant one (the derivative of the linear function $f(x) = x$), but, instead, we see a linearly increasing error as $f(t)$ increases. In order to get a correct result, we need to subtract $\left(a_0^w(\tau)/a_1^w(\tau)\right)f(t)$ from the reconstructed value. This means that we actually need to reconstruct the original function $f(t)$ in order to compute its derivative. But this would require another convolution with an interpolation filter, which would clearly be inefficient. A proposed way around this problem is to window the truncated *Cosc* filter [8].

We conclude that a careful analysis of a filter is necessary before its use.

3.2 Normalization

Let us assume we have an nth derivative filter that fulfills the above criteria: a_n^w is nonzero and $\forall(i < n)\, a_i^w = 0$. Since we want to reconstruct the exact value, we need to require a_n^w to be one. This can be achieved by normalizing the filter weights by a_n^w.

In the case of an interpolation filter h, we need to require that a_0^h be exactly one. Using (3), we find:

$$a_0^w(\tau) = \sum_{k=-M}^{M} w(\tau - k),$$

Thus, the sum of the filter weights needs to evaluate to one. This requirement is well known, and is also recommended by Mitchell and Netravali [14] for the design of filters that fall into the class of BC-splines. To fulfill this requirement, we simply normalize the filter weights by dividing them by a_0^n.

As with interpolation filters, we also need to normalize the first derivative filter d. Since we are reconstructing the derivative of the function instead of the function itself, we now set the coefficient for $f'(t)$ to one. Normalization is performed by dividing the filter weights by a_1^d. Using (3) again, the normalization factor is

$$a_1^w(\tau) = T \sum_{k=-M}^{M} (k - \tau) w(\tau - k)$$

The normalization step for derivative filters is a lesser known fact, but, nevertheless, is very important for accurate normal estimation. Without this normalization step, we cannot rely on the accuracy of the reconstructed derivative.

3.3 Classification

One of the major objectives of the Taylor series analysis of a numerical algorithm is to characterize its asymptotic error behavior. We can do this by comparing the a_n^w and r_N^w of various filters w. The principal idea is to choose the largest N such that all coefficients $a_m^w\,(m \le N)$ evaluate to zero, with the only exception being a_0^w for interpolation filters and a_n^w for nth order derivative filters, which should evaluate to one. Choosing the value of N in this way, the reconstruction error is simply the (normalized) remainder term r_N^w. Equation (5) can therefore be rewritten as:

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

or

$$\left| f^{(n)}(t) - f_r^w(t) \right| = \varepsilon = \left| \frac{r_{N,i}^w(\tau)}{a_n^w(\tau)} \right| \tag{7}$$

Having made these observations, we are now ready to outline the classification property of our filter design method. We observe that the coefficients depend solely on the underlying filter w. This provides us with an intuitive scheme to compare and classify different filters: All filters that are characterized by the same asymptotic error behavior $O\left(T^k\right)$ are grouped into one class and are called kth degree error filters (k-EF) to comply with standard nomenclature in numerical mathematics. The reconstruction error r_N^w is of the order $O\left(T^{N+1}\right)$, and the normalization factor a_n^w is of the order $O\left(T^n\right)$. Therefore, the asymptotic behavior of the error ε of the reconstruction process is $O\left(T^{N+1-n}\right)$. That means that for most applications, $\left|r_{N_1}^w\right|$ will be smaller than $\left|r_{N_2}^w\right|$, if and only if $N_1 > N_2$. Therefore, in general, we prefer filters in a class with largest N. We can further distinguish among filters in the same class using their absolute errors $\left|r_N^w\right|$.

This classification is very important and should be considered when selecting a filter for a given application. An N-EF filter will reconstruct a polynomial of (N – 1)th or lower degree (the original function as well as its derivatives) without any errors. Since, in most practical applications, the underlying data can be sufficiently modeled with low degree polynomials, a 3EF or 4EF should be sufficient.

The presented classification scheme is important for determining the sufficient and necessary resolution of voxelization (discretization) algorithms, since the reconstruction error is tightly bound to the sampling distance. Note that the placement of a filter in a k-EF group depends on the asymptotic behavior of the filter error. One can find examples where, for some T, a specific filter in the 2-EF group will perform better than a specific filter of the 3-EF group. For instance, if we compare a 2-EF filter with error $T^2\tau$ and a 3-EF filter with error $T^3\tau$, then it is clear that, for $T > 1$, the 2-EF filter outperforms the 3-EF filter. By the same token, there are real-life situations where one is forced to accept a certain sampling distance T, e.g., when T is predetermined by the maximal resolution of an MRI scanner. In such cases, one would consider the error term quantitatively only.

3.4 Absolute Error Approximation

The error as computed in (4) is more of theoretical interest, because we do not know the $\xi_{k,i}$. In order to find an approximation to the actual error, one could develop one more term of the Taylor series expansion of (6). That would allow us to write it as:

$$\begin{aligned} f_r^w(t) &= f^{(n)}(t) + \left(\frac{a_{N+1}^w(\tau)}{a_n^w(\tau)} f^{(N+1)}(t) + \frac{r_{N+1,i}^w(\tau)}{a_n^w(\tau)} \right) \\ &= f^{(n)}(t) + \frac{a_{N+1}^w(\tau)}{a_n^w(\tau)} f^{(N+1)}(t) + O\left(T^{N+2-n}\right). \end{aligned} \tag{8}$$

This equation reconstructs the nth derivative with the normalization step already performed. For small T, the dominant error term is clearly $f^{(N+1)}(t) a_{N+1}^w(\tau) \big/ a_n^w(\tau)$.

An alternative way to approximate the error would be to evaluate the remainder term. We suggest to use the following approximation of $r_{N,i}^w$. In practical applications, the length M of the used filter w is usually small, since the

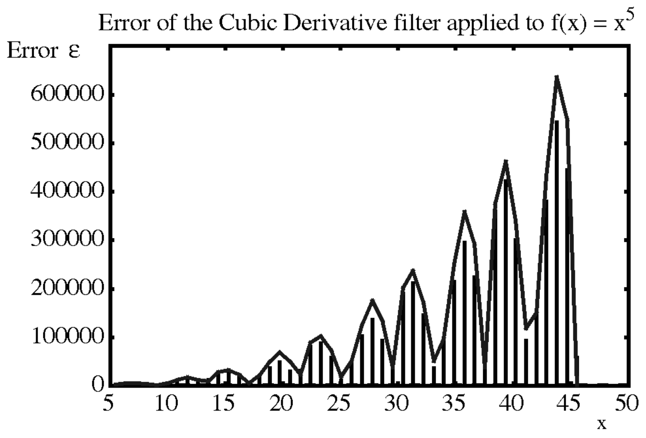


Fig. 3. The absolute error of the cubic derivative filter with the parameters $B = 0$ and $C = 1$ applied to a polynomial of fifth degree. We have also *computed* the error bound according to (9) using the result from Table 3 in Section 4.2. We can see that our error bound matches the actual error well. The actual error is shown as bars and the computed error-bound is drawn as a connected curve.

use of a larger filter is expensive. Therefore, the interval $[(i - M)T, (i + M)T]$, in which all the $\xi_{k,i}$ are to be found, is relatively short. In addition, practical data sets, like MRI or CT, are usually bandlimited with only little contributions coming from high frequencies. Due to this lack of a significant amount of high frequencies, which would imply a fast changing function within a short interval, it is reasonable to assume that high derivatives, in particular $f^{(N+1)}(t)$, will not change much on a short interval of length $2M$. We conclude, that:

$$\left|r^w_{N,i}(\tau)\right| \le \left|\max_{\xi\in[(i-M)T,(i+M)T]}\left(f^{(N+1)}(\xi)\right)\right| \left|\frac{T^{(N+1)}}{(N+1)!}\sum_{k=-M}^{M}(k-\tau)^{(N+1)}w(\tau-k)\right|$$

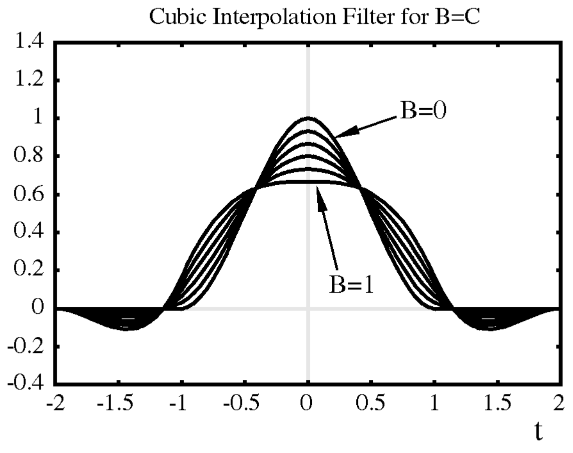
or,

$$\left|r^w_{N,i}(\tau)\right| \le \left|\max_{\xi\in[(i-M)T,(i+M)T]}\left(f^{(N+1)}(\xi)\right)\right|\left|a^w_{N+1}w(\tau)\right|.$$

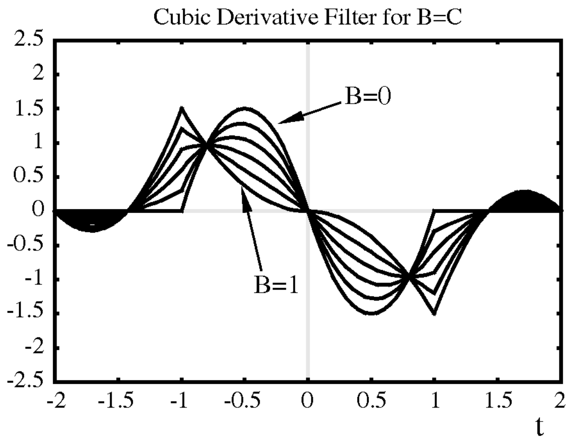
After evaluating the first three criteria, we find with (7) a bound of the absolute error:

$$\left|\frac{r^w_{N,i}(\tau)}{a^w_n(\tau)}\right| \le \left|\max_{\xi\in[(i-M)T,(i+M)T]}\left(f^{(N+1)}(\xi)\right)\right|\left|\frac{a^w_{N+1}(\tau)}{a^w_n(\tau)}\right| \qquad (9)$$

If we can approximate the $(N + 1)$st derivative of the underlying function, then we can approximate the actual error. Even if this is not possible, we can at least compute a^w_{N+1}/a^w_n to get an idea about the scale of the error. How well the error bound of (9) approximates the actual error can be seen in Fig. 3, where we used the derivative of the cubic interpolation filter (using $B = 0$ and $C = 1$) to compute the derivative of a quintic polynomial. It is not unreasonable to assume that real-life data sets can be modeled by cubic polynomials, at least within a piecewise local neighborhood. Therefore, a quintic polynomial, which is a supergroup of the cubic polynomials and has even larger variations and faster changing derivatives, should well suffice to demonstrate the correctness of our error bound computation.



(a)



(b)

Fig. 4. Representatives of the class of (a) cubic BC-spline interpolation filters and (b) cubic derivative filters. For our example, we chose to have $B = C$ and varied B from 0 to 1 in steps of 0.2.

4 Optimal Cubic Filter

Now, let us apply the methods we developed in Section 3 to a group of cubic filters. This group of cubic filters is described as (for some examples see Fig. 4a):

$$h(t) = \begin{cases} \left(2 - \frac{3}{2}B - C\right)|t|^3(-3 + 2B + C)|t|^2 + \left(1 - \frac{B}{3}\right) & 0 \le |t| < 1 \\ \left(-\frac{1}{6}B - C\right)|t|^3 + (B + 5C)|t|^2 + (-2B - 8C)|t| + \left(\frac{4}{3}B + 4C\right) & 1 \le |t| < 2 \\ 0 & 2 \le |t| \end{cases}$$

and are the aforementioned BC-splines. This class of filters has been derived by Mitchell and Netravali [14]. They show that, for $2C + B = 1$, this filter is a 2EF. Keys [9] investigated the Cardinal splines for numerical accuracy. The Cardinal splines are a subgroup of the BC-splines and are obtained by setting $B = 0$ and $C = -\alpha$.

Utilizing a Taylor series expansion, Keys found a 3EF for $\alpha = -0.5$. In Section 4.1 we will replicate these results using our method, which demonstrates its validity and power. Bentum et al. [1] derived a continuous derivative filter

TABLE 1
COEFFICIENTS FOR THE BC-SPLINE INTERPOLATION FILTER

a_0^h	1
a_1^h	$T(2C+B-1)\tau(2\tau-1)(\tau-1)$
a_2^h	$\frac{T^2}{2}\left(\frac{1}{3}B-4(2C+B-1)\tau^2(\tau-1)^2\right)$
a_3^h	$\frac{T^3}{6}\tau(2\tau-1)(\tau-1)(2C+3(2C+B-1)\tau(\tau-1))$
a_4^h	$\frac{T^4}{24}\left(\frac{1}{3}B-\tau^2(\tau-1)^2\left(1+16C+4(2C+B-1)\left(2\tau^2-2\tau+1\right)\right)\right)$

based on the Cardinal spline. This filter is simply the derivative of the interpolation filter. We now extend their ideas to the superclass of the Cardinal splines, the BC-splines. The resulting derivative filter can be written as (for some examples see Fig. 4b):

$$d(t)=\begin{cases}\left(\frac{1}{2}B+3C\right)t^2+(2B+10C)t+(2B+8C) & -2<t\le -1\\ \left(-6+\frac{9}{2}B+3C\right)t^2+(-6+4B+2C)t & -1<t\le 0\\ \left(6-\frac{9}{2}B-3C\right)t^2+(-6+4B+2C)t & 0\le t<1\\ \left(-\frac{1}{2}B-3C\right)t^2+(2B+10C)t+(-2B-8C) & 1\le t<2\\ 0 & 2\le |t|\end{cases}$$

In Section 4.2, we show that, for this derivative filter as well, $2C + B = 1$ is optimal and produces a 2EF.

4.1 Evaluation of the BC-spline Interpolation Filter

The cubic filter has a window size of two, i.e., has an overall extent of four. Therefore, four weights have to be considered. We have computed these weights and show them in the first row in Table 3 (see Appendix). We use the filter weights to compute the coefficients a_n^h in (3). A few algebraic manipulations yield the coefficients shown in Table 1.

For a_0^h, we compute the sum of the filter weights and find that this coefficient is exactly one. This is an important outcome, for it tells us that BC-spline interpolation filters do not need to be normalized. Due to this circumstance, it is not necessary to accumulate the sum of weights and divide the convolution sum by it, which makes this class of filters more efficient and more attractive for practical applications. This was also one of the design criteria for Mitchell and Netravali [14] and was to be expected.

In order for the filter to be as accurate as possible, we desire that subsequent coefficients evaluate to zero. Recall that the number of coefficients that evaluate to zero yields the class of the filter (classification phase). With respect to a_1^h, we find that there are three cases where this coefficient can be zero (keeping in mind that $\tau < 1$). These are:

- Case 1: $\tau = 0$,
- Case 2: $2\tau - 1 = 0$, (i.e., $\tau = 0.5$) and
- Case 3: $2C + B = 1$.

If none of these three cases apply, the cubic interpolation filter represents a linear order filter (1EF), and a_1^h substituted in (9) (or (8)) represents an error bound for this class of cubic filters. To demonstrate the error behavior in this specific case, the error coefficients are drawn in Fig. 6a. The error coefficients in Fig. 6a correspond to the same interpolation filters that are drawn in Fig. 4a. Note that, in these and all subsequent error plots, we have assumed T (the sampling distance of our underlying function that is to be reconstructed) to be equal to one. Also, we have only drawn the normalized error coefficient $a_{N+1}^w(\tau)/a_n^w(\tau)$. That gives us an idea about the general error behavior, independent of f. For a careful analysis of a certain application f (the function to be reconstructed) should possibly be incorporated.

Cases one and two are rather special cases, and can lead to 2EF and 4EF filters, respectively. The results of our analysis are listed in Table 3 (see Appendix). Let us now have a closer look at the interesting third case.

4.1.1 Case 3: $2C + B = 1$

This is the class of cubic BC-splines that was also found by Mitchell and Netravali [14] to provide the best solution numerically. This class includes the Catmull-Rom spline and the cubic B-spline. The filter coefficients of this filter can be expressed in the following way:

$$h(t)=\begin{cases}\left(\frac{1}{2}+2C\right)|t|^3+(-1-3C)|t|^2+\left(\frac{2}{3}+\frac{2}{3}C\right) & 0\le |t|<1\\ \left(-\frac{1}{6}-\frac{2}{3}C\right)|t|^3+(1+3C)|t|^2+(-2-4C)|t|+\left(\frac{4}{3}B+\frac{4}{3}C\right) & 1\le |t|<2\\ 0 & 2\le |t|\end{cases}$$

Some representatives of this filter are drawn in Fig. 7a. In order to find the convergence rate of this filter, the other Taylor coefficients need to be explored. The coefficient a_2^h in Table 1 simplifies to $\left(T^2/6\right)B$. The error is shown graphically in Fig. 8a. The only way that this error coefficient can be zero, is for the case that $B = 0$. That leads us to $C = 0.5$ which gives rise to the Catmull-Rom spline. The resulting filter weights are shown in Table 3 (see Appendix) and the filter is drawn in Fig. 5a. Keys [9] found this filter to be a 3EF as well, but he did not investigate the actual magnitude of the error. The next coefficient, a_3, in Table 1, will now

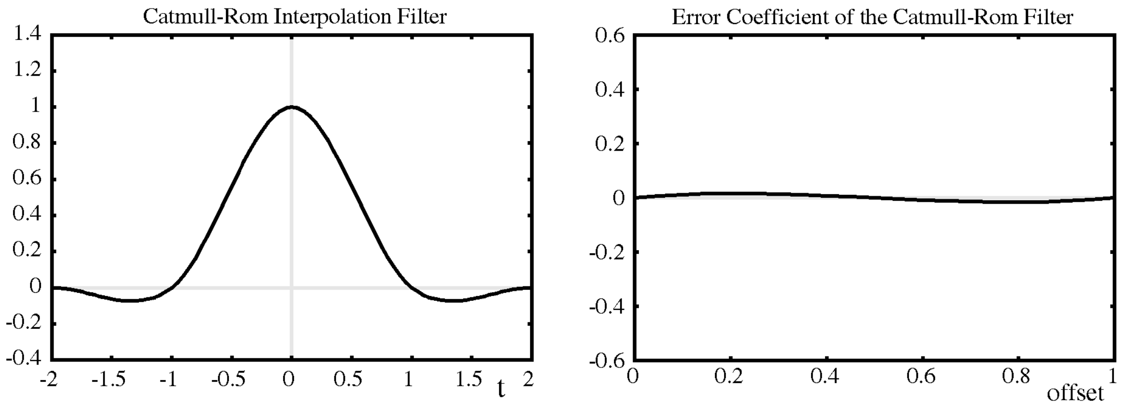


Fig. 5. (a) The optimal interpolation filter and (b) its error coefficient. We kept the scale of the *y* axis the same as for the other filters we have shown to better illustrate how the absolute errors compare.

simplify to $\left(T^3/6\right)\tau(2\tau-1)(\tau-1)$. According to our error analysis in Section 3.4 and (9), this represents an approximation of the error magnitude. The error is plotted in Fig. 5b.

We further realize that this error is zero in the specific case of $\tau = 0.5$. (see Table 3 in Appendix).

4.2 Evaluation of the BC-spline Derivative Filter

Bentum et al. [1] introduced the use of cubic derivative filters for gradient estimation, but applied that idea only to Cardinal splines, a subgroup of the BC-splines. The authors provide a limited analysis, but no analytical comparison of this filter with different parameters. In our earlier paper [15], we offer a complete spatial analysis of this filter group. Although this filter is really just a quadratic filter, we prefer to call it "cubic derivative" filter, due to its derivation from a cubic filter, the BC-splines. The four relevant weights for the BC-spline derivative filter are:

$$d(\tau-2) = \left(\tfrac{1}{2}B + 3C\right)\tau^2 - 2C\tau$$

$$d(\tau-1) = \left(-6 + \tfrac{9}{2}B + 3C\right)\tau^2 + (6 - 5B - 4C)\tau + \left(\tfrac{1}{2}B + C\right)$$

$$d(\tau) = \left(6 - \tfrac{9}{2}B - 3C\right)\tau^2 + (-6 + 4B + 2C)\tau$$

$$d(\tau+1) = \left(-\tfrac{1}{2}B - 3C\right)\tau^2 + (B + 4C)\tau + \left(-\tfrac{1}{2}B - C\right)$$

These are the derivatives of the weights for the BC-spline interpolation filter (given in Table 3). As in the previous section, we compute the coefficients, shown in Table 2, using (3).

To compute a_0^d, we sum the filter weights and find that a_0^d is zero. In Section 3.1, we observed that this is a requirement for good derivative filters. We thus conclude that the BC-splines form a class of very well behaved filters.

As was shown in Section 3.2, we need to normalize the other coefficients by a_1^d. The normalization step is simplified for filters that satisfy $2C + B = 1$, or at those reconstruction locations where $1 - 6\tau + 6\tau^2 = 0$. In these cases, we only have to divide by a constant, the sample distance T.

We note that, for $a_1^d = 0$, the filter computes a higher order derivative. In fact, for almost every τ, we can find infinitely many pairs (B, C) for which the filter is a higher order derivative filter. Specifically, when $C = 1.5$, $B = 0$, and

TABLE 2
COEFFICIENTS FOR THE BC-SPLINE DERIVATIVE FILTER

a_0^d	0
a_1^d	$T\left(1 + (2C+B-1)\left(1-6\tau+6\tau^2\right)\right)$
a_2^d	$3T^2(2C+B-1)\,\tau(1-\tau)\,(2\tau-1)$
a_3^d	$\frac{T^3}{6}\left(\left(12C\tau^2-12C\tau+1\right)+(2C+B-1)\left(18\tau^4-36\tau^3+24\tau^2-6\tau+1\right)\right)$
a_4^d	$\frac{T^4}{12}\tau(1-\tau)\,(2\tau-1)\left((12C+1)+(2C+B-1)\left(6\tau^2-6\tau+4\right)\right)$
a_5^d	$\frac{T^5}{120}\left(\left(60\tau C\left(2\tau^3-4\tau^2+3\tau-1\right)+15\tau^2(\tau-1)^2+1\right)+(2C+B-1)\left(30\tau^6-90\tau^5+135\tau^4-120\tau^3+51\tau^2-6\tau+1\right)\right)$

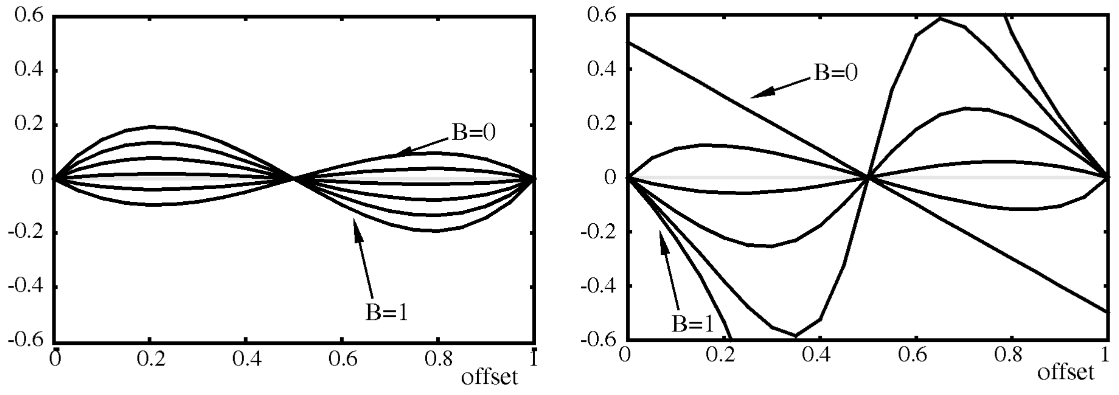


Fig. 6. Normalized error coefficients of the same filters that are shown in Fig. 4. (a) Errors for the cubic BC-spline interpolation filters and (b) errors for the cubic derivative filters. For our example, we chose B = C and varied B from 0 to 1 in steps of 0.2.

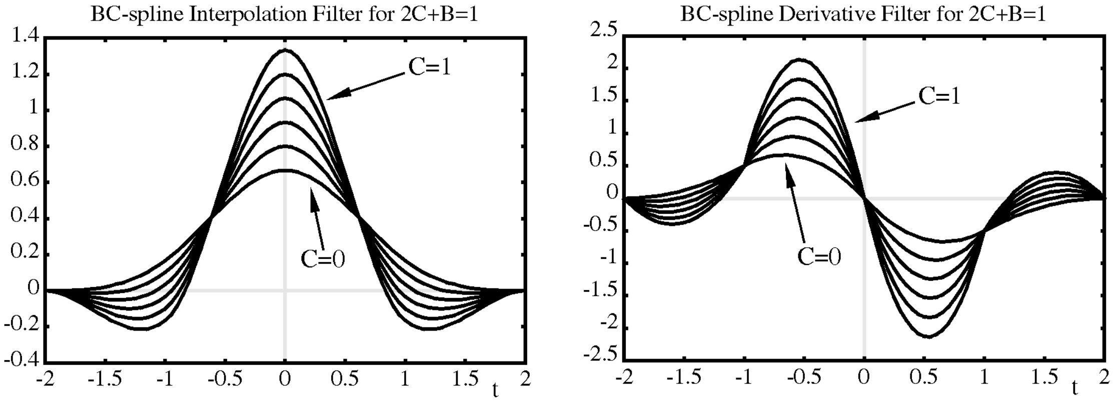


Fig. 7. Representatives of the class of (a) BC-spline interpolation filters and (b) BC-spline derivative filters with the condition $2C + B = 1$. Note these filters have a better error behavior than the general cubic splines. For our example, we varied C from 0 to 1 in steps of 0.2.

$\tau = 0.5$, we not only find a_1^d to be zero, but also a_2^d. This means that we recover at least the third derivative of the underlying function. Thus, not every combination of *B*s and *C*s results in a first derivative filter.

In order to evaluate the quality of the filters (classification), we again need to examine the other coefficients of the Taylor series expansion. With regards to a_2^d, we find that there are again three cases where this coefficient vanishes. These are the same cases as for the filters considered in Section 4.1. If none of these three conditions holds, we substitute the normalized coefficient a_2^d into (9) and find an error bound. To get an idea of the error behavior in this specific case, we have drawn the normalized error coefficient in Fig. 6b.

Case one ($\tau = 0$) and case two ($\tau = 0.5$) are again special cases that can lead to 2EF or 4EF filters. The results of our analysis are listed in Table 4 (see Appendix).

4.2.1 Case 3: $2C + B = 1$

This case is probably the most interesting one: It doesn't depend on the reconstruction offset, and is therefore the most general case. It also leads us to a new 3EF derivative filter. Note that the normalization step is again a simple division by the sample distance *T*, which makes this class of filters useful for fast and accurate convolution operations. The actual filter coefficients can also be found in Table 4.

Some members of this family of filters are drawn in Fig. 7b. The normalized coefficient a_3^d simplifies to $T^2\left(2C\tau^2 - 2C\tau + 1/6\right)$ and represents an upper bound of the derivative reconstruction error (9). The error behavior in relation to the sampling offset τ is illustrated in Fig. 8b.

The coefficient a_3^d will be zero for the case in which

$$C = \frac{1}{12\tau(1-\tau)} \quad \text{and, therefore,} \quad B = 1 - \frac{1}{6\tau(1-\tau)}$$

This relation, of course, is only meaningful when we are not trying to reconstruct at a sample point, i.e., $\tau \neq 0$ (remember, that $\tau < 1$). Table 4 lists the weights for this new filter for the case $\tau \neq 0$. A discussion what weights to use when $\tau = 0$ is given in Section 4.3. For this specific filter, we find that the normalized coefficient a_4^d / a_1^d simplifies to $\left(T^3/12\right)(2\tau - 1)\left(1 + \tau - \tau^2\right)$. Hence, this new filter is a 3EF. Since $0 < 1 + \tau - \tau^2$ for $\tau \in [0, 1]$, this coefficient can only be zero for the special case of $\tau = 0.5$. The normalized filter coefficients in this special case would be:

$$Td(\pm 1.5) = \pm \tfrac{1}{24}$$
$$Td(\pm 0.5) = \mp \tfrac{9}{8}$$

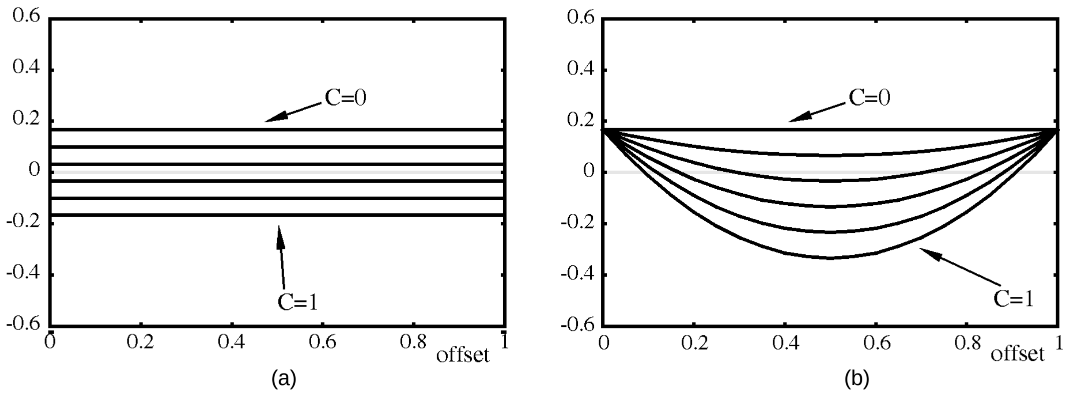


Fig. 8. Normalized error coefficients of the filters of Fig. 7. Errors for the (a) BC-spline interpolation filters and (b) BC-spline derivative filters are shown on the right. Because we applied the constrained $2C + B = 1$, these filters have better asymptotic error behavior (shown in Section 4.1.1 and Section 4.2.1). For our example, we varied C from 0 to 1 in steps of 0.2.

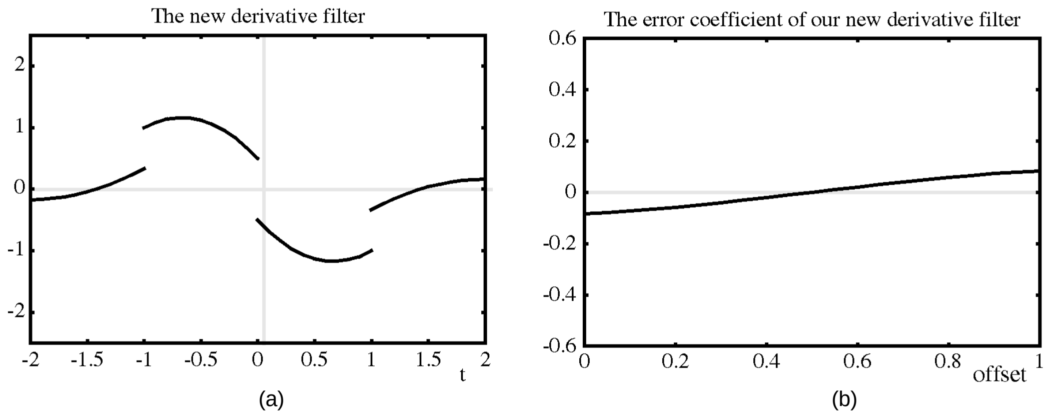


Fig. 9. (a) New optimal continuous derivative filter and (b) its error coefficient.

Therefore, in the case of $\tau = 0.5$, a_5^d / a_1^d (normalized) simplifies to $-3T^4/640$, and a 4EF results.

Let us now summarize the results of this section. Table 4, at the end of this paper, contains a comprehensive listing of all possible cases within the family of BC-spline based derivative filters. In the most general case, when $2C + B \neq 1$, we find that both interpolation and derivative filters are linear error filters (1EF). In Fig. 6, we plot the coefficients of the error term (given in (9)) for both of these filters. For the interpolation filter, we plot a_1^h, and for the derivative filter, we plot a_2^d / a_1^d. Finally, in Fig. 8, we plot the coefficients of the error terms when $2C + B = 1$. A knowledge of the relative behavior of interpolation and derivative filters is useful. For example, to determine the derivative at a point, it is conceivable that a cubic derivative be directly applied. Or, the derivatives can be determined at sample locations and then interpolated. The total error is now determined by the contributions of the derivative and interpolation filters. Both in Fig. 6 and Fig. 8, the derivative filters show a larger error.

Errors in functions and their derivatives cannot be compared directly, since they exist in different metric spaces. However, since the filters themselves are dimensionless in nature, it is possible to compare them. It is an important observation that one needs to apply more sophisticated derivative filters than interpolation filters if one wants to obtain derivative estimates as reliable as the interpolated signal. Furthermore, if we use the same parameters B and C for interpolation and derivative filter, as was done by Marschner and Lobb [13], the error introduced by the derivative filter will dominate the error due to interpolation. This is evidenced by our experiments as well (see Fig. 11).

4.3 Our New Derivative Filter

In this section, we take a closer look at the new derivative filter that we found through our analysis in Section 4.2.1. The weights of the new filter are given in Table 4 (see Appendix). Its normalized error coefficient is $a_4^d / a_1^d = \left(T^3/12\right)(2\tau - 1)\left(1 + \tau - \tau^2\right)$. Fig. 9 illustrates the new filter in the spatial domain. Note that this filter is not a member of the BC-spline family. Although the weights at each position in the filter are determined by a particular combination of the parameters B and C, this combination is different for each sampling offset τ. Recall from Fig. 8b that, depending on the offset τ, different combinations of B and C yielded the lowest reconstruction error. This behavior is exactly what this new derivative filter attempts to capture.

Recall that our new filter is not defined for $\tau = 0$. Therefore, we need to use a different filter at these locations. Since the error behavior of the central difference filter is only quadratic (as opposed to cubic for our new filter), we don't recommend using central differences in combination with our new filter. To preserve the global error behavior, we suggest the use of a high order discrete derivative filter

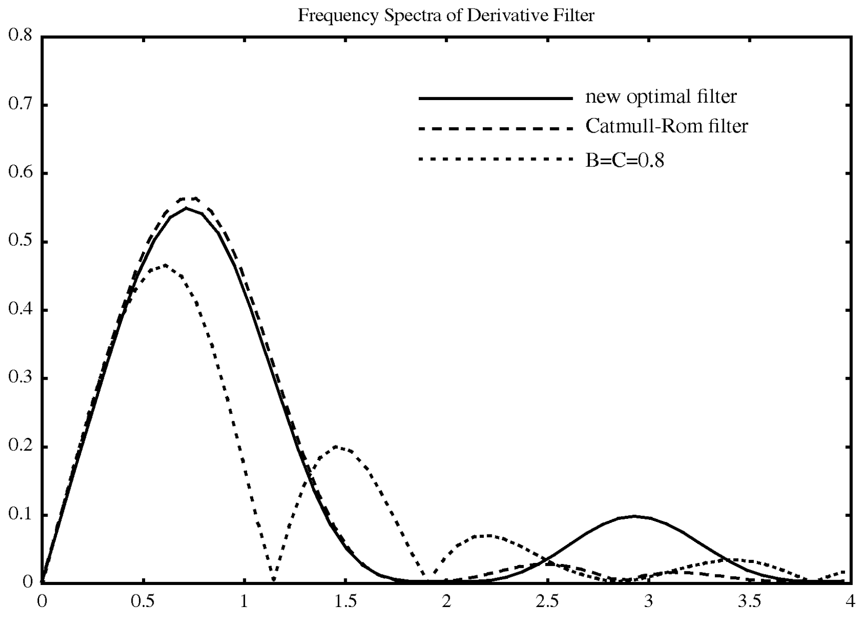


Fig. 10. Frequency plots of the derivative filters. The x and y axis are normalized by π. The straight line represents our new derivative filter. The dashed line represents the Catmull-Rom derivative filter (B = 0 and C = 0.5), and the dotted line represents a derivative filter for B = C = 0.8.

at the sample points as well. A natural choice would be to use the average of the left and right side limits at the discontinuities, which gives rise to the following filter weights:

$$Td(\pm 2) = \pm 1/12$$
$$Td(\pm 1) = \mp 2/3$$
$$Td(0) = 0$$

By analyzing this derivative filter (at $\tau = 0$), we found it to be a 4EF with an error coefficient of $T^4/30$. Note, however, that for $\tau > 0$, the filter is still 3-EF.

In order to get a more complete picture of this filter, we plotted its frequency response in Fig. 10, along with two other filters in the class of cubic derivative filters. We see that its frequency behavior is very similar to the Catmull-Rom spline. Differences in form of a hump arise near π and 3π. This is the location where the high frequencies of the first and second signal replica meet. Since, in practice, the amount of the signal content that falls into this high frequency range is rather small, the effect of that discrepancy should be negligible. At the important places near zero and 2π (where the DC of the replica is located), our new filter performs just as well, if not better, than the Catmull-Rom spline. A thorough frequency analysis, for instance, in the spirit of the one conducted by Park and Schowengerdt [16], to proof these assumptions more rigorously are beyond the scope of this current paper, but currently underway. What we can learn from the frequency plots is, however, that our new filter is not expected to perform much worse than the Catmull-Rom spline. We actually expect that, in some cases, its behavior is superior.

One deficiency of our new filter is that its error coefficient (drawn in Fig. 9b) is discontinuous. If the location we are reconstructing at is just a small distance away from the sampling grid, τ will be either close to one or to zero. We observe that the error coefficient for τ close to zero approaches $-1/12$, and for τ close to one, it approaches $1/12$. Hence, there exists a discontinuity to which the human visual system is very sensitive (see Fig. 15). However, it should be noted that this effect becomes relevant only for densely resampled images, i.e., in situations where the resampling rate is a lot higher than the original sampling rate. If the resampling rate is low in proximity of the discontinuity (e.g., if we only place two or three resampling points within a sample interval), then this artifact is likely not to be visible. Mathematically speaking, we desire filters such that:

$$a_n^+(\tau) - a_n^-(\tau) = 0,$$

where $a_n^+(\tau)\left(a_n^-(\tau)\right)$ denotes the right (left) limit at this coefficient. We also have to keep in mind the special case where $\tau = 0$, i.e., we need to require $a_n^+(0) - a_n^-(1) = 0$. (This exact case causes problems for our new derivative filter.) This results in the following condition:

$$\frac{T^n}{n!}\sum_{k=-M}^{M}(k-\tau)^n\left(w^+(\tau-k) - w^-(\tau-k)\right) = 0. \quad (10)$$

From here it does not necessarily follow that the actual filter w has to be continuous. However, all continuous filters fulfill (10). We want to make clear, however, that one should not discard discontinuous filters just because they happen to be discontinuous. For instance, our new filter can produce better images than the Catmull-Rom derivative filter (see Figs. 11, 13, and 14). Discontinuities of the filter *only* become a (visual) problem, if (10) doesn't hold, *and* if we densely sample near a discontinuity of the error coefficient. (see Fig. 15).

In order to avoid these problems of the discontinuity, one could employ additional filter design techniques, such as windowing. Jittering would also be a way to get rid of undesired aliasing effects.

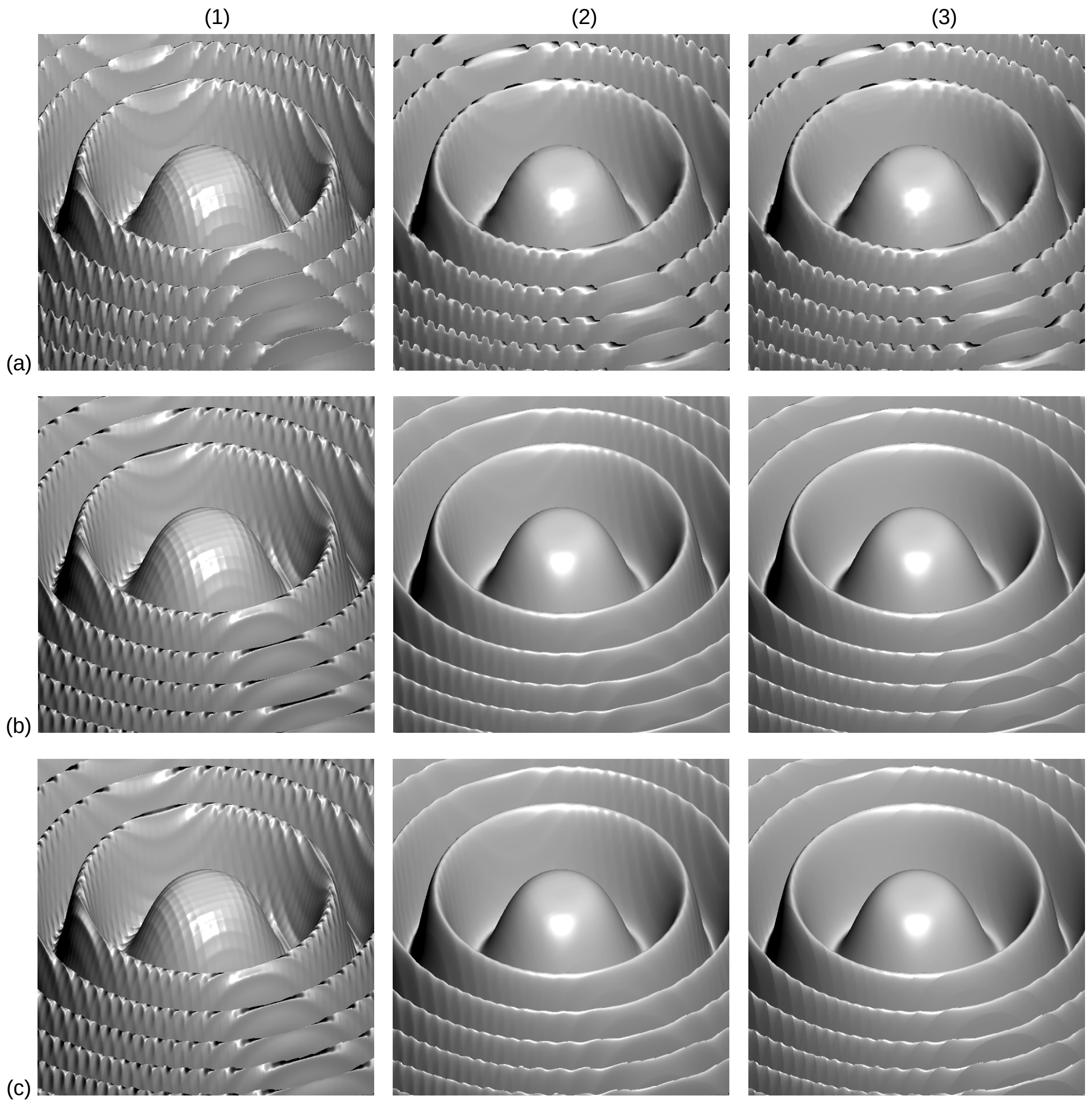


Fig. 11. Marschner Lobb data set. Column 1: Derivative: $B = C = 0.8$ (1EF), Column 2: Derivative: $B = 0$; $C = 0.5$ (2EF), Column 3: Derivative: our new filter (3EF), Row a: Interpolation: $B = C = 0.8$ (1EF), Row b: Interpolation: $B = 0.6$; $C = 0.2$ (2EF), Row c: Interpolation: Catmull-Rom (3EF).

5 Experimental Results

The images were rendered employing a simple raycaster to find the isosurfaces. The volumes were sampled at an interval of 0.1 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 preset isovalue, the derivative filter was used to compute the 3D gradient. Shading was then performed using the traditional Phong lighting model [7] 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 Bentum et al [1].

For our experiments, we used an analytic data set and an MRI data set. The synthetic data set is derived from the same function as the one used by Marschner and Lobb [13]. In Fig. 11, we rendered this data set with varying B and C for interpolation and derivative filters. Along the rows, we change the derivative filter. For the first column, we use the parameters $B = C = 0.8$ (a 1EF filter), the second column has $B = 0$ and $C = 0.5$ (a 2EF filter), and the last column uses our new derivative filter (a 3EF filter). Along the columns, we change the interpolation filter. The first row uses the parameters $B = C = 0.8$ (a 1EF filter), the second row uses $B = 0.6$ and $C = 0.2$ (a 2EF filter), and the third column uses the Catmull-Rom filter (a 3EF filter). We find that the differences between rows are not as striking as the differences between columns. That demonstrates our findings in Section 4.2, where we concluded that the derivative filter has more influence on image quality than the interpolation filter. In order to be able to compare interpolation filters, we suggest to fix the derivative filter, preferably one with error that is negligible compared to the errors in the interpolation filter. In order to better visualize the influence of the filters, we drew the angular error images of the last row of Fig. 11. For each reconstructed normal, we computed the actual normal and recorded their angular difference. The gray value of 255 was displayed for an angular error of 5 degrees.

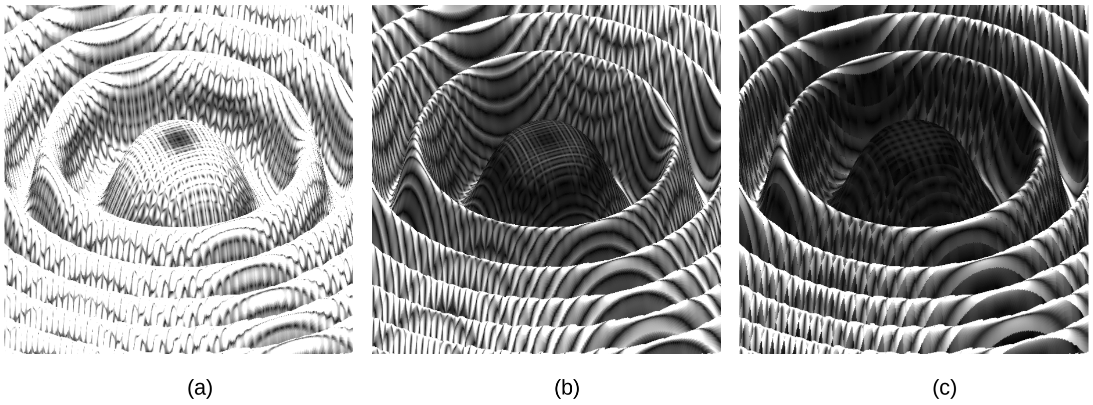


Fig. 12. Error images of the last row of Fig. 11. We compute the angular difference between the computed normal and the actual normal. The gray value of 255 represents an angular error of five degrees

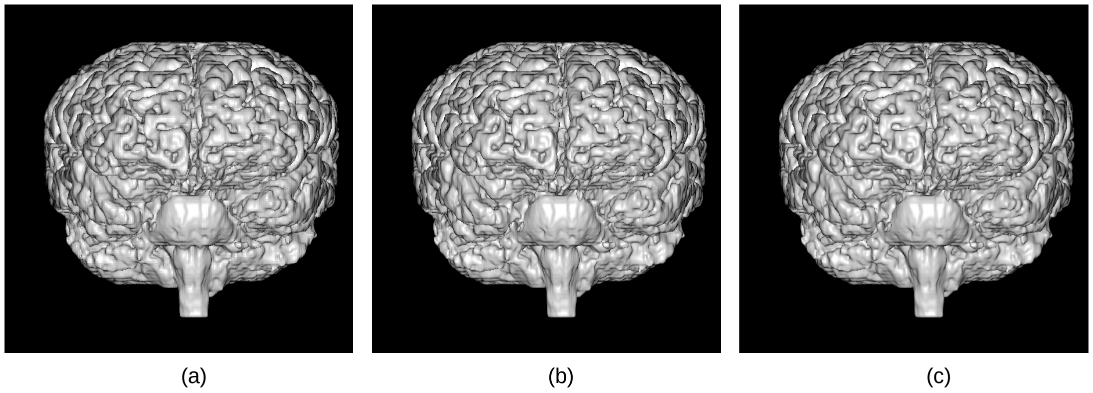


Fig. 13. Brain data set ((a) 1EF, (b) 2EF, (c) 3EF).

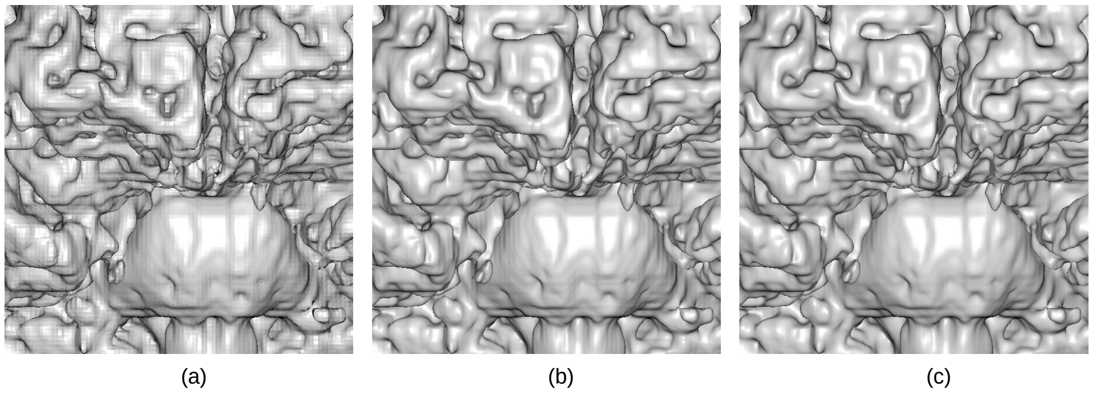


Fig. 14. Close up of the brain data set (eight rays per grid spacing, (a) 1EF, (b) 2EF, (c) 3EF).

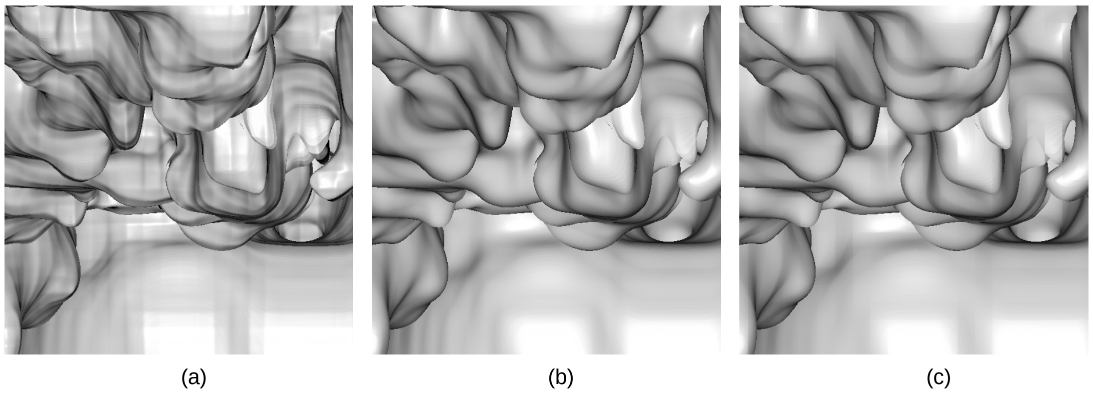


Fig. 15. Severe close up of the brain data set (30 rays per grid spacing). Artifacts become visible for our new 3EF filter ((a) 1EF,(b) 2EF, (c) 3EF).

The second data set is an MRI of a human brain. Here, we fixed the interpolation filter to the optimal cubic filter (Catmull-Rom) and varied the derivative filter in the same way as we did for the Marschner Lobb images. In Fig. 13, we find that the best image is achieved for our new derivative filter. That is exactly what we expected from our analysis in Section 4.2. Fig. 14 shows the same set of filters applied to a small section of the brain and rendered from a close-up view. Fig. 15 shows the same data set, where we zoom into a 20 by 20 voxel region. Since we have many pixels (rays) within one voxel cell, we resample the grid very fine around the original sample points. As explained in Section 4.3, this results in a discontinuous error behavior that becomes visible in Fig. 15.

6 Future Goals

In many applications, especially volume rendering, we want to both reconstruct the underlying function and / or its derivative, and also resample it on a new grid. Therefore, it is necessary to study the overall error expressed in the L_2 error norm. We are working on developing better tools to study this error. Eventually, we want to come up with techniques similar to the ones presented in this paper that will allow us to classify different filters, and also to quantify them efficiently computing their L_2 error.

In terms of filter design, we can use our tools to design filters of arbitrary order. Setting the coefficients in (2) to zero for a given N, we end up with N linear equations for $2M + 1$ coefficients. We can solve this linear system, matching N and M appropriately, and we find an NEF. It would be interesting to study how this filter behaves in terms of the offset τ, for it would enable us to construct filters of arbitrary accuracy. Similar to the adaptive filter design of Machiraju and Yagel [11], we can use these different filters adaptively in different areas of the function. Knowing the error caused by convolving a particular filter with a particular application, we want to find ways to adapt the type of filter we use according to a given error tolerance.

We plan to extend the techniques described in Section 3.1 through Section 3.4 for higher order derivatives and to the evaluation of filters, others than the BC cubic splines, such as windowed sinc and other optimal filters [4].

It is also important to study the behavior of different filters when applied to rapidly changing functions. (Such behavior could be caused by noise.) Our analysis remains valid, except for the error estimation in (9), where we assumed slowly changing functions.

Appendix

TABLE 3
FILTER WEIGHTS AND ERROR COEFFICIENTS OF THE CUBIC INTERPOLATION FILTERS UNDER VARIOUS RESTRICTIONS OF ITS PARAMETERS B AND C

condition	filter coefficients	error
none	$h(\tau-2) = \left(\frac{1}{6}B + C\right)\tau^3 - C\tau^2$ $h(\tau-1) = \left(-2 + \frac{3}{2}B + C\right)\tau^3 + \left(3 - \frac{5}{2}B - 2C\right)\tau^2 + \left(\frac{1}{2}B + C\right)\tau + \frac{1}{6}B$ $h(\tau) = \left(2 - \frac{3}{2}B - C\right)\tau^3 + (-3 + 2B + C)\tau^2 + \left(1 - \frac{1}{3}B\right)$ $h(\tau+1) = \left(-\frac{1}{6}B - C\right)\tau^3 + \left(\frac{1}{2}B + 2C\right)\tau^2 + \left(-\frac{1}{2}B - C\right)\tau + \frac{1}{6}B$	$T(2C + B - 1)\tau(2\tau - 1)(\tau - 1)$
$2C + B - 1 = 0$	$h(\tau-2) = \left(\frac{1}{6} + \frac{2}{3}C\right)\tau^3 - C\tau^2$ $h(\tau-1) = \left(-\frac{1}{2} - 2C\right)\tau^3 + \left(\frac{1}{2} + 3C\right)\tau^2 + \frac{1}{2}\tau + \left(\frac{1}{6} - \frac{1}{3}C\right)$ $h(\tau) = \left(\frac{1}{2} + 2C\right)\tau^3 + (-1 - 3C)\tau^2 + \left(\frac{2}{3} + \frac{2}{3}C\right)$ $h(\tau+1) = \left(-\frac{1}{6} - \frac{2}{3}C\right)\tau^3 + \left(\frac{1}{2} + C\right)\tau^2 - \frac{1}{2}\tau + \left(\frac{1}{6} - \frac{1}{3}C\right)$	$\frac{T^2}{6}B$
$\tau = 0.5$	$h(\pm 0.5) = \frac{1}{2} - \frac{1}{48}B + \frac{1}{8}C$ $h(\pm 1.5) = \frac{1}{48}B - \frac{1}{8}C$	$\frac{T^2}{24}(B - 6C + 3)$
$\tau = 0$	$h(\pm 2) = 0$ $h(\pm 1) = B/6$ $h(0) = 1 - B/3$	$\frac{T^2}{6}B$
$B = 0$ $C = 0.5$	$h(\tau-2) = \frac{1}{2}\tau^3 - \frac{1}{2}\tau^2$ $h(\tau-1) = -\frac{3}{2}\tau^3 + 2\tau^2 + \frac{1}{2}\tau$ $h(\tau) = \frac{3}{2}\tau^3 - \frac{5}{2}\tau^2 + 1$ $h(\tau+1) = -\frac{1}{2}\tau^3 + \tau^2 - \frac{1}{2}\tau$	$\frac{T^3}{6}\tau(2\tau - 1)(\tau - 1)$
$\tau = 0.5$ $B - 6C + 3 = 0$	$h(\pm 0.5) = 9/16$ $h(\pm 1.5) = -1/16$	$-\frac{3T^4}{128}$

TABLE 4
FILTER WEIGHTS AND ERROR COEFFICIENTS OF THE CUBIC DERIVATIVE FILTERS
UNDER VARIOUS RESTRICTIONS OF ITS PARAMETERS B AND C

condition	filter coefficients	error
$6\tau(1-\tau) \neq \frac{2C+B}{2C+B-1}$	$Td(\tau-2) = \frac{\left(\frac{1}{2}B+3C\right)\tau^2 - 2C\tau}{1+(2C+B-1)\left(1-6\tau+6\tau^2\right)}$ $Td(\tau-1) = \frac{\left(-6+\frac{9}{2}B+3C\right)\tau^2 + (6-5B-4C)\tau + \left(\frac{1}{2}B+C\right)}{1+(2C+B-1)\left(1-6\tau+6\tau^2\right)}$ $Td(\tau) = \frac{\left(6-\frac{9}{2}B-3C\right)\tau^2 + (-6+4B+2C)\tau}{1+(2C+B-1)\left(1-6\tau+6\tau^2\right)}$ $Td(\tau+1) = \frac{\left(-\frac{1}{2}B-3C\right)\tau^2 + (B+4C)\tau + \left(-\frac{1}{2}B-C\right)}{1+(2C+B-1)\left(1-6\tau+6\tau^2\right)}$	$\frac{3T(2C+B-1)\tau(1-\tau)(2\tau-1)}{1+(2C+B-1)\left(1-6\tau+6\tau^2\right)}$
$2C+B-1 = 0$	$Td(\tau-2) = \left(\frac{1}{2}+2C\right)\tau^2 - 2C\tau$ $Td(\tau-1) = \left(-\frac{3}{2}-6C\right)\tau^2 + (1+6C)\tau + \frac{1}{2}$ $Td(\tau) = \left(\frac{3}{2}+6C\right)\tau^2 + (-2-6C)\tau$ $Td(\tau+1) = \left(-\frac{1}{2}-2C\right)\tau^2 + (1+2C)\tau - \frac{1}{2}$	$T^2\left(2C\tau^2 - 2C\tau + 1/6\right)$
$\tau = 0.5$ $3-2C-B \neq 0$	$Td(\pm 0.5) = \pm\frac{-12+7B+2C}{12-4B-8C}$ $Td(\pm 1.5) = \pm\frac{-B+2C}{12-4B-8C}$	$\frac{T^2}{6}\left(\frac{3+5B-14C}{12-4B-8C}\right)$
$\tau = 0, 1$ $2C+B \neq 0$	$Td(-2) = Td(0) = 0$ $Td(\pm 1) = \mp\frac{1}{2}$	$\frac{T^2}{6}$
$B = 1 - \frac{1}{6\tau(1-\tau)}$ $C = \frac{1}{12\tau(1-\tau)}$ $\tau \neq 0,1$	$Td(\tau-2) = \frac{1}{2}\tau^2 - \frac{1}{6}$ $Td(\tau-1) = -\frac{3}{2}\tau^2 + \tau + 1$ $Td(\tau) = \frac{3}{2}\tau^2 - 2\tau - \frac{1}{2}$ $Td(\tau+1) = -\frac{1}{2}\tau^2 + \tau - \frac{1}{3}$	$\frac{T^3}{12}(2\tau-1)\left(1+\tau-\tau^2\right)$
$\tau = 0.5$ $3+5B-14C = 0$	$Td(\pm 0.5) = \pm\frac{-81+100C}{72-96C}$ $Td(\pm 1.5) = \pm\frac{3-12C}{72-96C}$	$-\frac{3T^4}{640}$

7 CONCLUSION

In this paper, we applied a Taylor series expansion to the convolution sum. This resulted in an alternative representation of the convolution sum which lead to a qualitative and quantitative comparison of both reconstruction and derivative filters. From this new representation of the convolution sum, we derived four new filter evaluation criteria—analysis, normalization, classification, and error estimation. We found that the normalization of the filter coefficients is important for accurate reconstruction.

We then applied these techniques to the class of cubic BC-splines. We derived several special filters which are numerically optimal within this class. Specifically, we concentrated our efforts on interpolation and derivative filters and found that, when both are applied to a function, the errors introduced by derivative filters are more significant than those caused by interpolation filters.

We expect the techniques developed here to be applicable to the design and evaluation of other reconstruction and high order derivative filters.

ACKNOWLEDGMENTS

We thank Ed Overman of the Department of Mathematics at Ohio State University for providing useful background information and important references. We thank Wayne Carlson and the Advanced Computing Center for the Arts and Design for the use of their computing facilities, and Prof. Robert Moorehead of the U.S. National Science Foundation Engineering Research Center, Mississippi State

University, for providing encouragement and support. This project was partially supported by the U.S. Department of Defense USAMRDC 94228001, and by the U.S. Advanced Research Projects Agency Contract DABT63-C-0056. The MRI data was provided by Scott E. Lukas, PhD Chief, Clinical Neuropsycopharmacology Laboratory, Alcohol and Drug Abuse Research Center, McLean Hospital, Harvard Medical School. Thanks to Naeem Shareef for the segmentation of this data set.

REFERENCES

[1] M.J. Bentum, T. Malzbender, and B.B. Lichtenbelt, "Frequency Analysis of Gradient Estimators in Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 3, pp. 242-254, Sept. 1996.

[2] R.N. Bracewell, *Two Dimensional Imaging*. Englewoods Cliffs, N.J.: Prentice Hall, 1995.

[3] I. Daubechies, "Ten Lectures on Wavelets," *Proc. CBMS-NSF Regional Conf., SIAM*, Philadelphia, 1992.

[4] G. Deslauriers and S. Dubuc, "Symmetric Iterative Interpolation Processes," *Constructive Approximation*, vol. 5, no. 1, pp. 49-68, 1989.

[5] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics*, vol. 22, no. 4, pp. 51-58, Aug. 1988.

[6] S.C. Dutta Roy and B. Kumar, "Digital Differentiators," *Handbook of Statistics*, N.K. Bise and C.R. Rao, eds., vol. 10, pp. 159-205, 1993.

[7] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics, Principles and Practice*, second edition. Reading, Mass.: Addison-Wesley, 1990.

[8] M.E. Goss, "An Adjustable Gradient Filter for Volume Visualization Image Enhancement," *Proc. Graphics Interface '94*, pp. 67-74, Toronto, Ont., 1994.

[9] R.G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1,153-1,160, Dec. 1981.

[10] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 29-37, May 1988.

[11] R.K. Machiraju and R. Yagel, "Reconstruction Error Characterization and Control: A Sampling Theory Approach," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 4, pp. 364-376, Dec. 1996.

[12] J.L. Mannos and D.J. Sakrison, "The Effects of a Visual Fidelity Criterion on the Encoding of Images," *IEEE Trans. Information Theory*, vol. 20, no. 3, pp. 525-536, July 1974.

[13] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *Proc. Visualization '94*, pp. 100-107, IEEE CS Press, Oct. 1994.

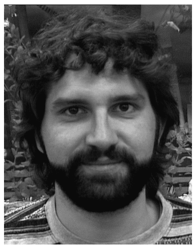
[14] D.P. Mitchell and A.N. Netravali, "Reconstruction Filters in Computer Graphics," *Computer Graphics*, vol. 22, no. 4, pp. 221-228, Aug. 1988.

[15] T. Möller, R.K. Machiraju, K. Mueller, and R. Yagel, "Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering," *Proc. 1996 Symp. Volume Visualization*, pp. 71-78, Oct. 1996.

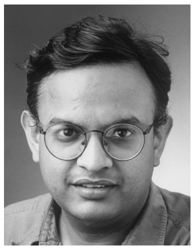
[16] S.K. Park and R.A. Schowengerdt, "Image Reconstruction by Parametric Cubic Convolution," *Computer Vision, Graphics, and Image Processing*, vol. 23, pp. 258-272, 1983.

[17] J.A. Parker, R.V. Kenyon, and D.E. Troxel, "Comparison of Interpolating Methods for Image Resampling," *IEEE Trans. Medical Imaging*, vol. 2, no. 1, pp. 31-39, Mar. 1983.

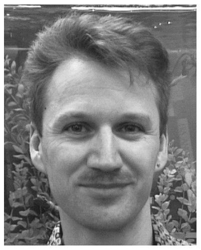
[18] A. Savitzky and M.J.E. Golay, "Analytical Chemistry," vol. 36, pp. 1,627-1,639, 1964.



Torsten Möller received a Vordiplom (BSc degree) in mathematical computer science from Humboldt University in Berlin, Germany, in 1992, and an MSc degree in computer and information science from the Ohio State University in 1993. Previous employment includes summer employment by the Lawrence Livermore National Laboratories Scientific Visualization Group and an internship at Mental Images, Berlin, Germany. He is presently working on a PhD degree in computer and information science at the Ohio State University, where he also holds a graduate research appointment. His interests reside in the field of computer graphics, in particular in applying mathematical methods and ideas from signal processing to evaluate the rendering process. His current focus is on the development of algorithms for fast and accurate rendering of unstructured three-dimensional data sets.



Raghu Machiraju received his PhD in computer science from the Ohio State University in August 1996, and has been at Mississippi State University since then. He is an assistant professor in the Department of Computer Science and the U.S. National Science Foundation Engineering Research Center at Mississippi State University. Earlier, he worked for Control Data Corporation and the Ohio Supercomputer Center as a programmer. His research interests include filter design and wavelet methods for visualization algorithms.



Klaus Mueller received a BSc degree in electrical engineering from the Polytechnic University of Ulm, Germany, in 1987, and a MSc degree in biomedical engineering from the Ohio State University in 1990. Previous employment includes an internship at Mercedes Benz, Germany, and research and development engineer positions at Medical Devices, Minneapolis, and Bopp and Reuther, Mannheim, Germany. Mueller is presently working on a PhD degree in computer and information science at the Ohio State University, where he also holds a graduate research appointment funded by General Electric. Mr. Mueller's interests reside in the field of computer graphics, in particular volume graphics as applied for the visualization of medical image data. His current focus is on the development of iterative algorithms for fast and accurate 3D and 4D reconstruction from limited sets of 2D projections.

Roni Yagel received his PhD in 1991 from the State University of New York at Stony Brook, where he was also a researcher in the Department of Anatomy and the Department of Physiology and Biophysics. He received his BSc (cum laude) and MSc (cum laude) from the Department of Mathematics and Computer Science at Ben Gurion University of the Negev, Israel, in 1986 and 1987, respectively. He is an assistant professor in the Department of Computer and Information Science and the Advanced Computing Center for the Arts and Design at the Ohio State University. His research and technical publications deal mainly with a variety of topics in volume graphics and visualization. His research interests also include algorithms for graphics, imaging, and animation, error control in image generation, and visualization.