Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering

Torsten Möller[1,3], Raghu Machiraju[1,3], Klaus Mueller[2], Roni Yagel[1,2,3]

[1]Department of Computer and Information Science
[2]Biomedical Engineering Center
[3]The Advanced Computing Center for the Arts and Design
The Ohio State University, Columbus, Ohio

Abstract

We describe a new method for analyzing, classifying, and evaluating filters, which can be applied to interpolation filters, and derivative filters. Our analysis is based on the Taylor series expansion of a convolution sum and some assumptions on the behavior of the data function. As a result of our analysis, we derive the need and the method for normalization of derivative filter coefficients. As an example, we demonstrate the utilization of our methods to the analysis of the class of cardinal cubic filters. Since our technique is not restricted to interpolation filters, we can show that the Catmull-Rom spline filter and its derivative are the most accurate reconstruction and derivative filter among this class of filters. We show that the derivative filter has a much higher impact on the rendered volume than the interpolation filter. We demonstrate the use of these optimal filters for accurate interpolation and gradient estimation in volume rendering.

1 Introduction

Reconstruction of a continuous function and possibly 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 data set at arbitrary locations to evaluate the rendering integral. The gradient (or first derivative of the function) is important in classifying the volume and applying a proper illumination model [5][12].

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 have only f_k, which is the discrete image we need to manipulate. An important assumption we make is that the continuous signal f is sampled at or above the *Nyquist frequency* [17][21]. Inherent to this assumption is that the underlying function is bandlimited and hence analytic, i.e., all derivatives exist at all points. In fact, signals commonly found in volume visualization are bandlimited because, during the process of acquiring digital images, acquisition devices (e.g., cameras, scanners) perform a filtering operation and bandlimit the function. Images generated by numerical simulations of physical phenomena (common in disciplines such as computational fluid dynamics) are also bandlimited because, 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, typically employ a filtering step that bandlimits the image. Malzbender presents similar observations for volumes obtained through medical acquisition devices (e.g CT, MRI) [14]. Much has been written about the reconstruction of sampled data sets in the fields of signal processing [17] (1D data) and image processing [3][6] (2D data), applied numerical mathematics [1][10][21].

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. Before we can classify or apply any shading method to our data set, we need to reconstruct the derivative of *f(t)* from the known samples f_k. We denote the derivative of the continuous function *f(t)* by $f'(t)$ and the reconstructed derivative by $f_r^d(t)$. Here, d denotes the high-pass derivative filter.

The filters h and d are usually chosen without much thought towards the adverse effects of the filter performance. The trilinear and central difference filters are often used for the reconstruction of the underlying function and its derivative, because they are inexpensive. However, the use of the trilinear filter results in blurring and aliasing in the final image, while the application of the central difference filter results in the loss of fine details.

With recent advances in hardware it is now possible to consider the use of better, albeit computational expensive filters. As a result, there exists a need for quantitative and qualitative methods to evaluate the goodness of the interpolation and derivative filters. Quantitative methods are useful since they provide an error metric to compare and contrast filters. Also, they can lead to the selection of optimal filters. On the other hand, qualitative methods allow the classification of the filters into categories and may lead to the application of further metrics e.g perceptual.

For our evaluation methods, we also require that the function is included in the evaluation. Also, the evaluation should be conducted in the spatial domain instead of the more cumbersome frequency domain.

With this background and assumptions in mind we summarize, in Section 2, what previous research has been done in this field. In Section 3 we introduce our concept of Taylor series expansion of the convolution sum. Because of their importance, we single out the case of interpolation and derivative filters. In Section 4, we illustrate an application of our general methods to the group of cardinal cubic interpolation and derivative filters. In Section 5 we

appeared in Proceedings of the 1996 Symposium on Volume Visualization, pp. 71-78, October 1996

show 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 frequency domain. One of the earliest comparative studies of interpolation filters for image resampling was done by Parker et al [19]. They compared nearest neighbor, linear, cubic B-Spline, and two members of the class of cardinal cubic splines, through a discussion of their respective frequency spectra. They found the Catmull-Rom spline to be superior.

A thorough study of cardinal cubic splines in frequency domain was performed by Park and Schowengerdt [18]. They found that the optimal interpolation filter of this class highly depends on the signal to which it will be applied. For most applications, the parameter α will be around -0.6 or close to the Catmull-Rom spline, for which the parameter is -0.5. Keys [11] showed that the latter filter is optimal, within the class of cardinal splines, in the sense that it interpolates the original function with the smallest spatial error. By using a Taylor series expansion of the convolution sum, he found that the Catmull-Rom spline interpolation filter has an error proportional to the cube of the sampling distance.

Mitchell and Netravali [16] introduce a more general class of cubic splines, which we refer to as BC-splines. Cardinal cubics are a subclass of these cubics. Mitchell and Netravali conducted a study of more than 500 sample images, classifying the parameter space into different regions of dominating image artifacts such as blurring, ringing, and anisotropy. They also found, by using a Taylor series expansion, that filters in which $B + 2C = 1$ are most accurate numerically and have an error proportional to the square of the sampling distance. Neither Keys nor Mitchell and Netravali approximate the actual error of their filters.

A recent comparative study by Marschner and Lobb [15] proposes the use of different metrics for different image artifacts. Specifically, they introduce metrics in the frequency domain to measure the smoothing, postaliasing, and overshoot of an interpolation filter and found the windowed *Sinc* filter to behave the best. Unfortunately, their metrics do not depend on the actual function to be reconstructed, an issue that Park and Schowengerdt found to be crucial for frequency analysis.

All the aforementioned approaches neglect to take derivative filters into account in their studies, a precondition to compare rendered and shaded images. A good survey of existing derivative filters can be found in the paper by Dutta Roy and Kumar [7] in which they describe the design of maximal linear filters in frequency domain. Their filter design can be easily adapted to various frequency ranges, an important consideration for practical applications.

Goss [9] extends the idea of windowed filters to derivative filters. He uses a Kaiser window for the ideal derivative filter, which is shown, e.g. in [2], to be the derivative of the *Sinc* filter and which we denote as the *Cosc* filter, but does not explain why this windowing is necessary and why a Kaiser window will work reasonably well. The work by Bentum [2] uses the cardinal cubics as a basis to develop derivative filters. Although he shows different plots of these filters for different parameters, he does not analytically compare these different filters.

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

In this work we develop tools for the spatial analysis of interpolation and derivative filters. We show the importance of the normalization of the filter coefficients and how this step is performed. Specifically for the class of cardinal splines, we derive known and new results in Section 4 using our new spatial method.

3 Taylor Series Expansion of the Convolution Sum

To reconstruct a continuous function *f(t)* or its derivative $f'(t)$ from a set of sampled points f_k, we compute a weighted average of these samples. This process is also known as convolution of the sampled signal with a filter which determines the weights. If we convolve the samples with a continuous interpolation filter *h*, we reconstruct the original function *f(t)*. Similarly, if we convolve the samples with a continuous derivative filter *d*, we can reconstruct the *n*th derivative of the original function. We denote the result by $f_r^w(t)$, where *w* denotes the kind of filter used. This can be written as:

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

Now f_k represents the samples of the original function *f(t)* at the values *kT*, where *T* is the sampling distance. Since we assume that all the derivatives of *f(t)* exist, we can expand $f_k = f(kT)$ in a Taylor series in *f* around *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 Equation (1) and reordering the terms according to the derivatives of *f(t)*, we can rewrite Equation (1) as:

$$f_r^w(t) = \sum_{n=0}^{N} a_n^w(t) f^{(n)}(t) + r_N^w(t) \tag{2}$$

where the coefficients $a_n^w(t)$ and the remainder term $r_N^w(t)$ are:

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

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

This gives us the impression that the values of the coefficients a_n^w and the remainder term r_N^w depend on *t*. This is somewhat misleading. Let us first have a look at the coefficients a_n^w. For practical reconstructions of signals, we do not use an infinitely long filter. Therefore, a filter *w* has a finite filter length that we call *M*. The filter *w* is defined to be zero outside the interval *[-M, M]*. Next, we observe that we are evaluating the filter *w* at points exactly one unit length apart. These weights are applied to an *n*-th degree polynomial $(kT-t)^n$ sampled also at points exactly *T* apart, centered at *t*. Therefore, it makes sense to rewrite *t* in terms of an offset τ from *T*, written as:

$$t = (i+\tau)T \text{ , where } 0 \le \tau < 1 \text{ , and } i \in Z$$

Therefore, the coefficients a_n^w can be expressed in terms of this offset:

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

which can be simplified to:

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

Here, we clearly see that a_n^w depends not on t itself, but rather on the distance to the next sampling point (expressed by τ). This is because τ tells us how 'far away' a reconstructed value is from an already known sampled value. Therefore, we can quantify how 'hard' the reconstruction process really is. Since $a_{n,\,i,\,T}^w$ does not depend on i and, T is set by the data acquisition step (and therefore cannot be changed during the reconstruction process), we will drop the appropriate subscripts in the expression for a_n^w.

If we do the same analysis for the remainder term r_N^w, we find:

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

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

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

Our objective is to quantify and classify the error occurring during the reconstruction process. 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 the coefficients a_n^w evaluate to zero, with the exception of a_0^w for interpolation filters and a_m^w for *m*-th order derivative filters. This coefficient should evaluate to one, since we want to reconstruct the continuous function or its *m*-th order derivative respectively. Choosing the value of *N* in this way, the reconstruction error is simply the remainder term r_N^w. We observe that the coefficients depend solely on the underlying filter *w*. This leads us to a conceptually ideal way to compare and classify different filters. We can put all the filters characterized by the same *N* in one class. The reconstruction error r_N^w is of the order $O(T^{N+1})$. That means that for typical applications, $\left|r_{N_1}^w\right|$ will be smaller than $\left|r_{N_2}^w\right|$, iff $N_1 > N_2$. Therefore, in general, we prefer filters in a class with largest *N*. The filters in class *(N-1)* we simply call *N*-th degree error filters (*N*-EF) to comply with standard nomenclature in numerical mathematics. 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 strongly influence the choice of a filter for a given application. A *N*-EF will reconstruct a polynomial of *(N-1)*th or lower degree (the original function as well as its derivative) without any errors. In many applications, the underlying data can be sufficiently modeled with lower degree polynomials. Therefore, a 3EF or 4EF may be sufficient.

This classification scheme is important in determining the sufficient and necessary resolution of voxelization (discretization) algorithms. One restriction might occur, when the researcher cannot change the resolution, e.g. the resolution of an MRI scanner. In such cases, one needs to consider the error terms quantitatively only, because the placement of a filter in a *N*-EF group depends on the *asymptotic* behavior of the filter error. Therefore, one can find examples where, for some *T*, a specific filter in the 2EF group will perform better than a specific filter of the 3EF group.

3.1 Continuous Interpolation Filters

In the case of an interpolation filter *h*, we need to require that a_0^h be exactly one. This requirement is also recommended by Mitchell and Netravali [16]. To achieve this, we simply normalize the filter coefficients by dividing them by a_0^h. Without this normalization step, we cannot rely on the accuracy of the interpolated function.

3.2 Continuous Derivative Filters

As with interpolation filters, we also need to normalize a derivative filter *d*. For simplicity, we will concentrate on the derivative filter for the reconstruction of the first derivative $f'(t)$ of the original function. Higher order derivative filters can be treated similarly.

We are reconstructing the derivative of the function instead of the function itself and therefore need to set the coefficient in front of $f'(t)$ to one. That means we must normalize it by dividing the filter coefficients by a_1^d. This is less known, yet very important for a reliable normal estimation. The reconstruction of derivatives is still more complicated than this; we also need to require that a_0^d is zero! In fact, if this condition does not hold, the result is useless. As an 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* [2]. Therefore, we have

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

The *Cosc* filter (like the *Sinc* filter) is an infinite filter and therefore not applicable. In the case of the *Sinc* filter, we could use a truncated *Sinc* filter, which is equal to the *Sinc* filter for all $|t| \le M$ and zero outside of this interval. In the case of the *Cosc* filter, 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. This is a rather expensive filter. In Figure 1 we plotted the normalized coefficient $a_0^d(\tau)/a_1^d(\tau)$. We found this function to be varying between -0.4 and 0.4. Notice the behavior of this specific truncated *Cosc* filter on a linear function as shown in Figure 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 on *f(t)*. In order to get a correct

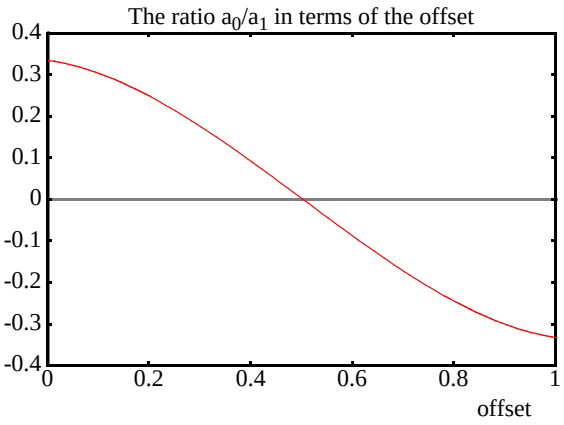

FIGURE 1. We computed the normalized coefficient a_0^d in terms of the offset to the next sampled value. We find it only to be zero for $\tau = 0.5$.

result, we need to subtract $(a_0^d(\tau)/a_1^d(\tau))f(t)$ from the reconstructed value. That means 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 be inefficient. A proposed way around this problem is to window the truncated *Cosc* filter [9].

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

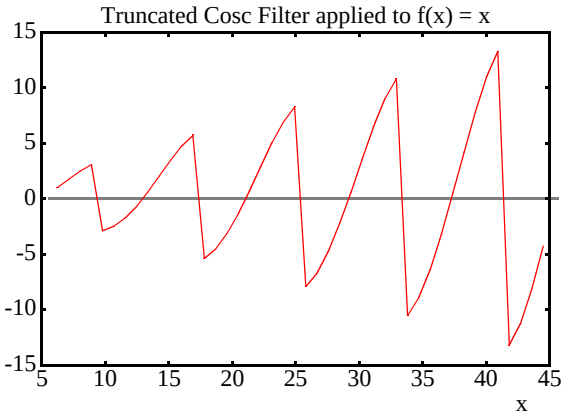

FIGURE 2. We truncated the Cosc filter at *M=3*. Since the coefficient a_0^d is not zero, we end up with artifacts that make the filter useless. (The expected result is the constant function one.)

3.3 An Approximation of the Error r_N^w

The error as computed in Equation 4 is more a theoretical result than a practical one, because we do not know the $\xi_{k,i}$. To get a fair idea about the behavior of the actual error, we suggest an approximation of r_N^w. In practical applications the length *M* of the used filter *w* is usually small, since a larger filter width *M* results in reduced efficiency. Therefore, we conclude that the interval $[(i-M)T, (i+M)T]$, in which all the $\xi_{k,i}$ are to be found, is relatively small. In addition, for MRI and CT data sets we found a behavior of the data sets in the frequency spectrum corresponding to the function $1/\omega^r$ for small *r*. Therefore, we do not expect a lot of high frequencies, which would result in a fast changing function within a short interval. It is reasonable to assume that especially $f^{(N+1)}(t)$ will not change much on a small interval of length *2M*. We conclude, that:

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

or,

$$r_{N,i}^w(\tau) \le \left(\max_{\xi \in [(i-M)T,\,(i+M)T]} (f^{(N+1)}(\xi)) \right) \left| a_{N+1}^w(\tau) \right| \qquad (5)$$

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_{N+1}^w to get an idea about the scale of the error. How well this error-bound approximates the actual error can be seen in Figure 3, where we used the derivative of the cubic interpolation filter to compute the derivative of a quintic polynomial. People usually model practical data sets with cubic polynomials locally. We have chosen a quintic polynomial since it is a supergroup of cubic polynomials and has higher variations and faster changing derivatives.

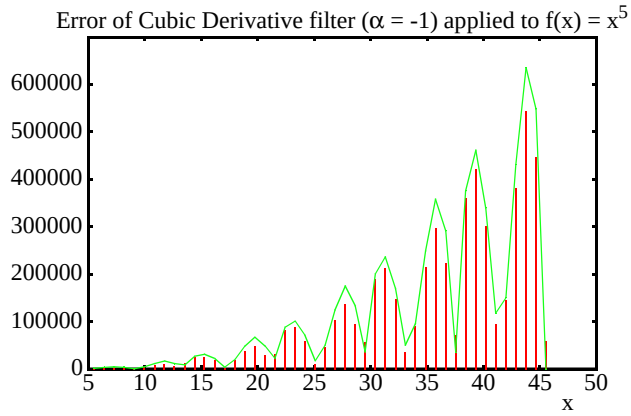

FIGURE 3. The absolute error of the cubic derivative filter with the parameter α set to -1 applied to the polynomial of 5th degree. Also we have *computed* the error bound according to Equation 5 using the result from Table 1 in Section 4.2. We can see that our error bound is very tight and almost exact. The actual error is shown as error bars and the computed error-bound is drawn as a connected curve.

4 Optimal Cubic Filter

Now, let us apply the methods we developed in Section 3 to the group of cubic filters. The group of cubic filters are described as:

$$h(t) = \begin{cases} (\alpha+2)|t|^3 - (\alpha+3)|t|^2 + 1 & 0 < |t| < 1 \\ \alpha|t|^3 - 5\alpha|t|^2 + 8\alpha|t| - 4\alpha & 1 < |t| < 2 \\ 0 & 2 < |t| \end{cases}$$

This class of filters is derived by Keys [11]. He shows that for $\alpha = -0.5$ this filter is a 3EF. He also uses a Taylor series expansion to derive this result. In Section 4.1 we will derive this result again, to demonstrate the power of our method. Bentum [2] developed a continuous derivative filter originating from the cubic interpolation filter. This filter is simply the derivative of the cubic interpolation filter, and can be written as:

$$d(t) = \begin{cases} -3\alpha t^2 - 10\alpha t - 8\alpha & -2 < t < -1 \\ -3(\alpha+2)t^2 - 2(\alpha+3)t & -1 < t < 0 \\ 3(\alpha+2)t^2 - 2(\alpha+3)t & 0 < t < 1 \\ 3\alpha t^2 - 10\alpha t + 8\alpha & 1 < t < 2 \\ 0 & 2 \le |t| \end{cases}$$

In section Section 4.2 we show that for this derivative filter as well, $\alpha = -0.5$ is optimal and produces a 2EF.

4.1 The cubic interpolation filter

The cubic filter has a window size of two, i.e. an overall extent of four. Therefore, we have four weights to consider. These are:

$$\begin{aligned} h(\tau) &= (\alpha+2)\tau^3 - (\alpha+3)\tau^2 + 1 \\ h(\tau-1) &= -(\alpha+2)\tau^3 + (2\alpha+3)\tau^2 - \alpha\tau \\ h(\tau+1) &= \alpha\tau^3 - 2\alpha\tau^2 + \alpha\tau \\ h(\tau-2) &= -\alpha\tau^3 + \alpha\tau^2 \end{aligned} \tag{6}$$

We are using these filter weights to compute the coefficients a_n^h from Equation 3. The results are summarized in Table 1.

TABLE 1. Coefficients for the cubic interpolation filter

a_0^h	1
a_1^h	$T(2\alpha+1)\tau(1-2\tau)(\tau-1)$
a_2^h	$2T^2(2\alpha+1)\tau^2(\tau-1)^2$
a_3^h (evaluated only for $\alpha = -0.5$)	$\frac{T^3}{6}\tau(1-2\tau)(1-\tau)$

For a_0^h we compute their sum and find that this coefficient is exactly one. This is an important result, for it tells us that cubic interpolation filters do not need to be normalized. This saves many computations and makes this class of filters more efficient and more attractive for practical applications.

Considering a_1^h we find, as Keys [11] did, that the cubic filter with $\alpha = -0.5$, also known as the Catmull-Rom-Spline, has the best behavior in terms of numerical accuracy and makes this cubic interpolation filter at least a 2EF. Additionally, we find, that in the special case of $\tau = 0.5$ the cubic interpolation filter is, for any choice of α, at least a 2EF too. This means we interpolate exactly in the middle of two sampling points. If we neither choose α to be -0.5 nor interpolate in the middle of two sampling points, the cubic interpolation filter represents a linear order filter (1EF) and a_1^h substituted in Equation 5 represents an error bound for this class of cubic filters.

For further analysis in the case of $\alpha = -0.5$ or $\tau = 0.5$, we evaluate a_2^h. For $\tau = 0.5$ it will not be zero, and therefore substituted in Equation 5 presents an error approximation. In the case of $\alpha = -0.5$, a_2^h will evaluate to zero and in order to approximate the error, we need to analyze a_3^h.

Computing a_3^h for $\alpha = -0.5$, we find that this coefficient is not zero and therefore, substituted in Equation 5, represents an error-bound for the Catmull-Rom cubic filter. Thus this filter is a 3EF filter. We conclude that the choice $\alpha = -0.5$ results in an optimal cubic interpolation filter. This filter is optimal in terms of numerical accuracy. Keys presented this same result, except, that he did not provide an error-bound and did not mention the special case in which the offset τ is set to -0.5.

With τ set to -0.5, we find that also a_3^h evaluates to zero and we get an even better filter. This special discrete interpolation filter has the following filter coefficients:

$$\begin{aligned} h(\pm 0.5) &= 9/16 \\ h(\pm 1.5) &= -1/16 \end{aligned}$$

For this filter, we find similarly $a_4^h(0.5) = -3T^4/128$, which, again, we plug into our error approximation of Equation 5. Therefore, when both conditions hold ($\tau = 0.5$ and $\alpha = -0.5$), we have a 4EF filter.

4.2 The cubic derivative filter

Bentum [2] introduced the cubic derivative filter, but did not provide an analysis nor an analytical comparison of its performance for different values of the parameter α. Although this filter is really just a quadratic filter, we prefer to call it 'cubic derivative' filter, since its parent is the cubic interpolation filter, which is a very well known and commonly used class of filters. The four relevant weights in this case are:

$$\begin{aligned} d(\tau) &= 3(\alpha+2)\tau^2 - 2(\alpha+3)\tau \\ d(\tau-1) &= -3(\alpha+2)\tau^2 + 2(2\alpha+3)\tau - \alpha \\ d(\tau+1) &= 3\alpha\tau^2 - 4\alpha\tau + \alpha \\ d(\tau-2) &= -3\alpha\tau^2 + 2\alpha\tau \end{aligned}$$

These are the derivatives of the weights for the interpolation filter in Equation 6. As in the previous section, we compute the coefficients in Table 2 using Equation 3.ex

To compute a_0^d we compute the sum of these weights and find that a_0^d is indeed zero. In Section 3.2 we found that this is a requirement for good derivative filters, and we also saw that it is not obvious this condition holds. Here, we note once again that cubic filters are a well-defined class of filters. As shown in Section 3.2, we need to normalize the other coefficients by a_1^d. Note that for

$\alpha = -0.5$, the normalization step is a simple division by the sample distance *T*. This again saves time and the convolution becomes more efficient, making this class of filters useful for fast, accurate volume rendering. Especially in the case of $T = 1$ this division does not need to be performed. In the given literature we could not find a mention of this necessary normalization step. We also want to point out that in the case of $a_1^d = 0$, we really have a filter which computes a higher order derivative. This is the case, e.g. if $\alpha = -1.5$ and $\tau = 0.5$. We not only find a_1^d to be zero but also a_2^d. That means that we actually recover at least the third derivative of the underlying function. A more thorough analysis, where one would compute a_3^d or higher, is beyond the scope of this paper.

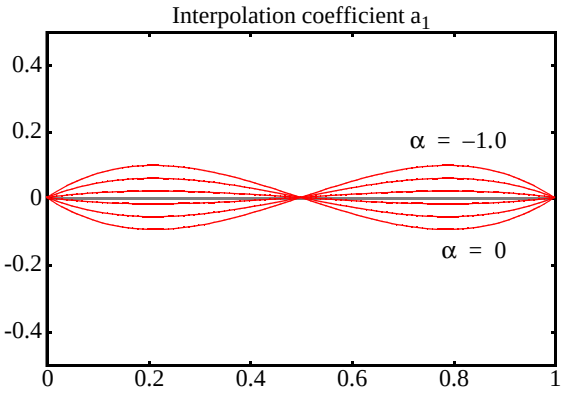

FIGURE 4. Here we plot the coefficient a_1^h of the cubic interpolation filter for varying α. α is 0, -0.2, -0.4, -0.6, -0.8 and -1. We set *T* to one.

As is the case for the cubic interpolation filter, we discover that for $\alpha = -0.5$ or $\tau = 0.5$ the coefficient a_2^d is zero, leading us to a derivative filter of higher order error. If none of these two conditions hold, we substitute the normalized a_2^d into Equation 5 and find an error bound. In the case that $\alpha = -0.5$, we substitute a_3^d/T into Equation 5 for an error bound. Since we normalize by *T*, this filter is a 2NF.

Again, we conclude that for the cubic derivative filter $\alpha = -0.5$ is the optimal filter in terms of numerical accuracy. Note that we lose one degree of accuracy for the first derivative filter. The cubic interpolation filter is a 3EF. However, the cubic derivative filter is only a 2EF.

In the case that $\alpha \neq -0.5$, we find that both filters are linear error filter (1EF). For a better perspective we plotted the error coefficients of the error term in Equation (5). For the interpolation filter we plotted a_1^h for different α in Figure 4 and for the derivative filter we plotted a_2^d/a_1^d in Figure 5. We find that the actual error term is larger in magnitude for the derivative filter.

TABLE 2. Coefficients for the cubic derivative filter.

a_0^d	0
a_1^d	$T(1 + (2\alpha + 1)(-6\tau^2 + 6\tau - 1))$
a_2^d	$3T^2(2\alpha + 1)\tau(1 - \tau)(1 - 2\tau)$
a_3^d (evaluated only for $\alpha = -0.5$)	$\frac{T^3}{6}(6\tau^2 - 6\tau + 1)$

These are important observations, for in order to obtain as reliable data for derivatives as for the interpolated signal, we need to apply more sophisticated filters. Furthermore, if we use the same α for interpolation and derivative filter as Marschner and Lobb [15], the error introduced by the derivative filter is larger.

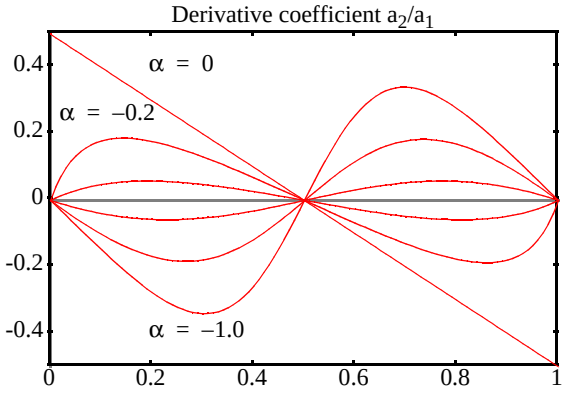

FIGURE 5. We plot the normalized coefficient a_2^d/a_1^d of the cubic derivative filter for varying α. α is 0, -0.2, -0.4, -0.6, -0.8 and -1. We set *T* to one.

5 Experimental Results

For our experiments we used an analytic data set and a MRI data set. The synthetic data set is the same function as the one used by Marschner and Lobb [15], sampled into a 40 x 40 x 40 volume lattice. A high definition ray caster, sampling the volume at a step size of 0.05 voxel lengths, was employed to display the function's opaque isosurface at 0.5 of the maximum function value. In contrast to the images given in [15], Figure 6 focuses on the center section of the function, where the distinct effects induced by the various interpolation-derivative filter pairings are most apparent.

In Figure 6.1 through Figure 6.9 we rendered this data set with varying α for interpolation and derivative filters. Along the rows we decrease α for the interpolation filter from top ($\alpha = -0.2$) to bottom ($\alpha = -1$). Along the columns we decrease α for the derivative filter from left ($\alpha = -0.2$) to right ($\alpha = -1$). 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 deriva-

tive filter, preferably one with error that is negligeable compared to the errors in the interpolation filter.

The second data set is an MRI of a human head, also used by Bentum [2]. Here, we fixed the interpolation filter to the optimal cubic filter and varied α for the derivative filter. In Figure FIGURE 7..1 through Figure FIGURE 7..3, we find that the best image is achieved when α is set to -0.5. That is exactly what we expected from our analysis in Section 4.2. Figure FIGURE 8..1 through Figure FIGURE 8..3 show the same set of filters applied to a small section of the brain and rendered from a close-up view.

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 Equation 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 a *N*EF. 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 et al [13], 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 describes in Section 3.2 for higher order derivatives and to the evaluation of filters others than the cardinal cubic splines, such BC splines [16] 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 Equation 5, where we assumed slowly changing functions.

7 Summary

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. We found that the normalization of the filter coefficients is important for accurate reconstruction.

We then applied these techniques to the analysis of the class of cardinal cubic splines. We derived several special filters, which are numerically optimal within this class. Especially, we concentrated our efforts on interpolation and derivative filters and found that, when both are applied to a function, the error 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 for providing useful background information and important references. We thank Wayne Carlson and the Advanced Center for Arts for the use of their computing facilities and Mark Fontana for reviewing the manuscript. This project was partially supported Department of Defense USAMRDC 94228001, and by the Advanced Research Projects Agency Contract DABT63-C-0056.

8 References

[1] Baker G., Overman E., “The Art of Scientific Computing”, Draft Edition, The Ohio State University Bookstore. 1995

[2] Bentum M.J., “Interactive Visualization of Volume Data”, Ph.D. Thesis, University of Twente, Enschwede, The Netherlands, 1995.

[3] Bracewell, R.N., *Two Dimensional Imaging*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1995.

[4] Deslauriers G., Dubuc S., “Symmetric Iterative Interpolation Processes, *CONAP*, 5(1):49-68,1989.

[5] Drebin R.A., Carpenter L., Hanrahan P., “Volume Rendering”, *Computer Graphics*, 22(4):51-58, August 1988.

[6] Dudgeon D.E., Russell M.M., *Multidimensional Digital Signal Processing*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1984.

[7] Dutta Roy S. C. and B. Kumar, “Digital Differentiators”, in *Handbook of Statistics*, N. K. Bise and C. R. Rao eds., Vol. 10:159-205, 1993.

[8] Glassner A., *Principles of Digital Image Synthesis*, Morgan Kaufmann, 1995.

[9] Goss M.E., “An Adjustable Gradient Filter for Volume Visualization Image Enhancement”, *Proceedings of Graphics Interface’94*:67-74, Toronto, Ontario, 1994.

[10] Hamming R.W., *Numerical Methods for Scientists and Engineers*, Dover Publications Inc., Mineola, N.Y., 1986.

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

[12] Levoy M., “Display of Surfaces from Volume Data”, *IEEE Computer Graphics and Applications*, 8(5):29-37, May 1988.

[13] Machiraju R., Yagel R., “Reconstruction Error Characterization and Control: A Sampling Theory Approach”, submitted for publication *IEEE Transactions on Computer Graphics and Visualization*.

[14] Malzbender T., “Fourier Volume Rendering”, *ACM Transactions on Computer Graphics*, 12(3):233-250, July 1993.

[15] Marschner S.R. and Lobb R.J., “An Evaluation of Reconstruction Filters for Volume Rendering”, *Proceedings of Visualization ‘94*, IEEE CS Press:100-107, October 1994.

[16] Mitchell D.P. and Netravali A.N., “Reconstruction Filters in Computer Graphics”, *Computer Graphics*, 22(4):221-228, August 1988.

[17] Oppenheim A.V., Schafer R.W., *Discrete-Time Signal Processing*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1989.

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

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

[20] Simoncelli E.P., “Design of Multi-Dimensional Derivative Filters”, presented at the *IEEE International Conference on Image Processing*, Austin TX, November 1994. Also found at http://www.cis.upenn.edu/~eero/publications.html.

[21] Stenger F., *Numerical Methods Based on Sinc and Analytic Functions*, Springer Verlag, 1993.

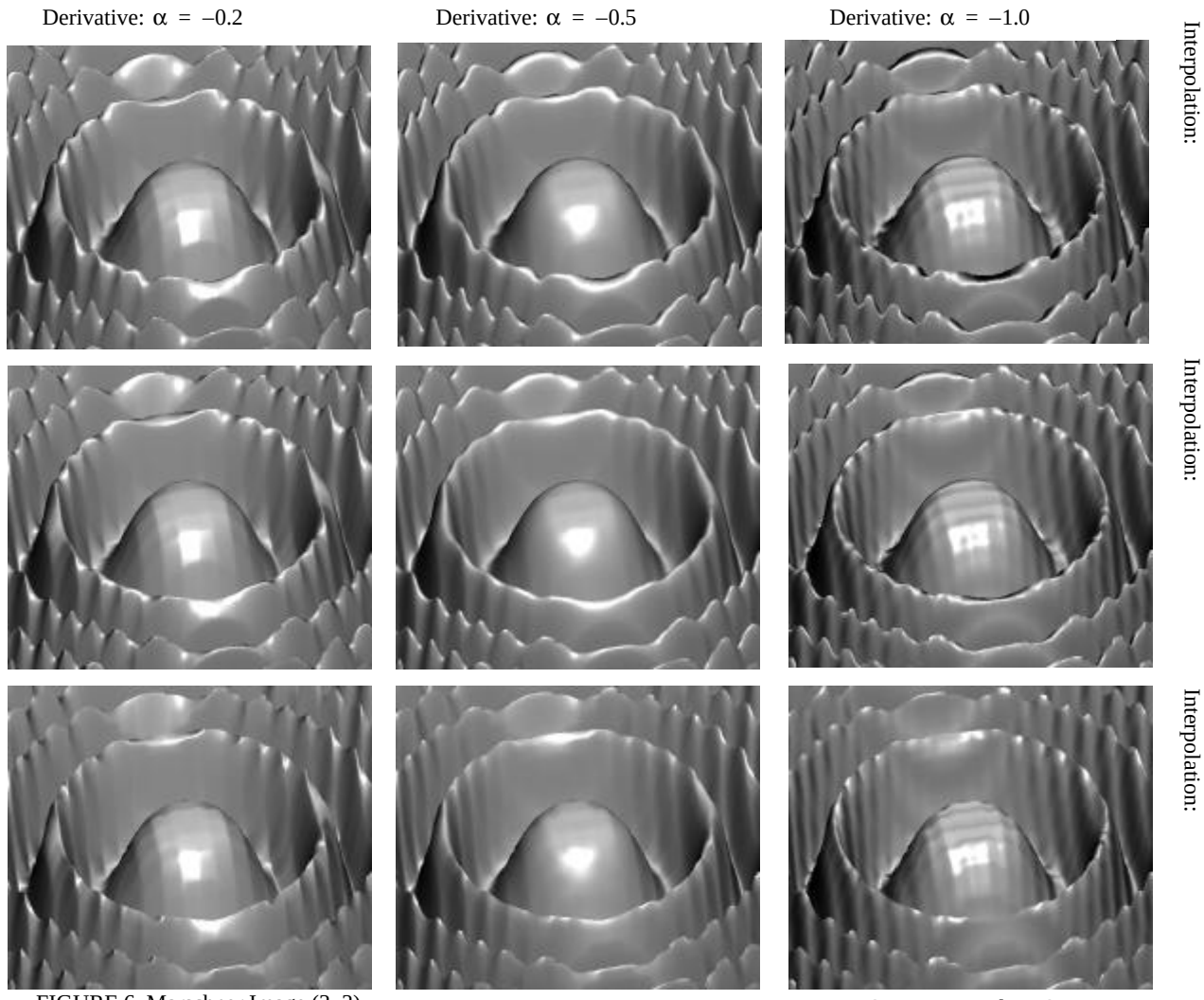

FIGURE 6. Marschner Image (3x3)

FIGURE 7. Head 1,2,3

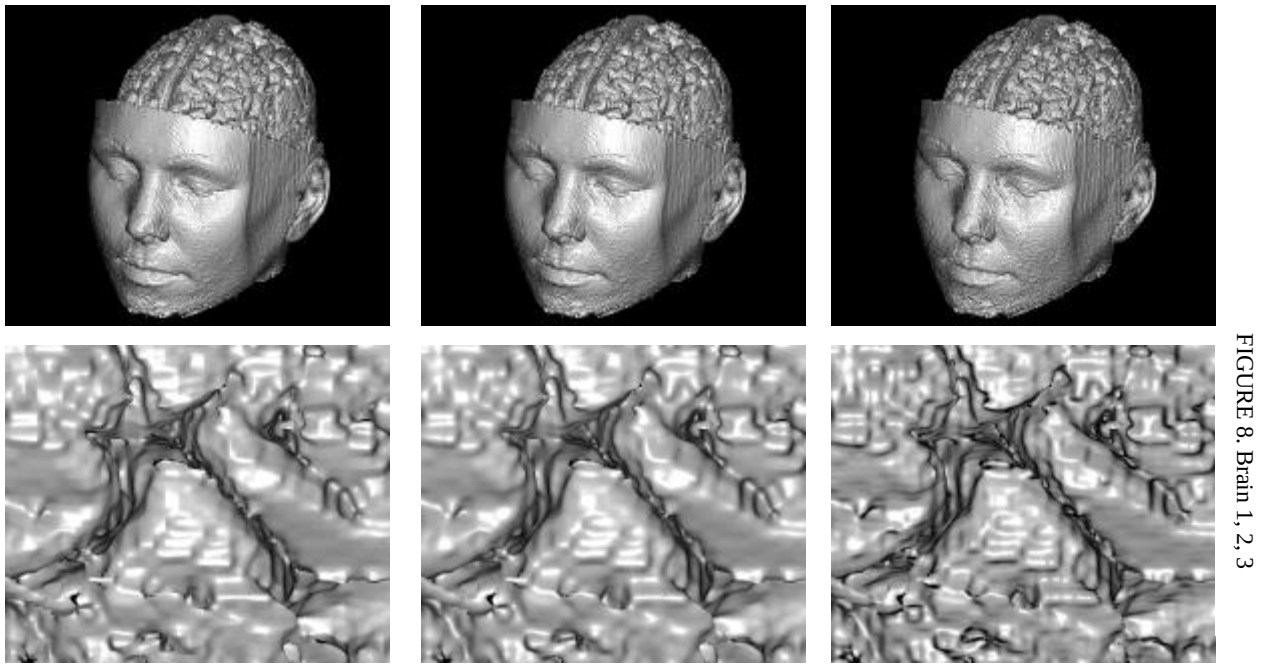

FIGURE 8. Brain 1, 2, 3