Toward High-Quality Gradient Estimation on Regular Lattices

Zahid Hossain, Usman R. Alim, and Torsten Möller, *Member*, *IEEE*

Abstract—In this paper, we present two methods for accurate gradient estimation from scalar field data sampled on regular lattices. The first method is based on the multidimensional Taylor series expansion of the convolution sum and allows us to specify design criteria such as compactness and approximation power. The second method is based on a Hilbert space framework and provides a minimum error solution in the form of an orthogonal projection operating between two approximation spaces. Both methods lead to discrete filters, which can be combined with continuous reconstruction kernels to yield highly accurate estimators as compared to the current state of the art. We demonstrate the advantages of our methods in the context of volume rendering of data sampled on Cartesian and Body-Centered Cubic lattices. Our results show significant qualitative and quantitative improvements for both synthetic and real data, while incurring a moderate preprocessing and storage overhead.

Index Terms—Approximation theory, Taylor series expansion, normal reconstruction, orthogonal projection, body-centered cubic lattice, box splines.

1 Introduction

Volumetric data, typically given on a discrete lattice, are perceived as a continuous data type, and therefore, require the algorithms working on them to model the data as if it were given in a continuous domain. Hence, interpolation and reconstruction are the key aspects of any volumetric manipulation and have a tremendous impact on the quality and efficiency of the underlying visualization task. While there has been a large body of work on interpolation and reconstruction filter design, in many tasks, we also need secondary information of the volumetric data, such as histograms for data exploration [16], gradients for shading [23] or higher order gradients for illustrative rendering [18], and feature detection [17]. One could simply just take an interpolation filter and consider its analytical derivative as a proper derivative filter. However this unnecessarily constrains the conditions on accuracy and smoothness. Hence, a separate design of gradient estimation schemes can lead to much better results. Therefore, we will consider the design of gradient estimation schemes in this paper. In particular, we consider two competing designs. On one hand, we explore a design based on a Taylor series expansion, which leads to computationally efficient discrete derivative kernels that can be used on the fly without precomputing gradients in a gradient volume. On the other hand, we consider the idea of an approximation space: a Hilbert space spanned by the shifts of a generating function (reconstruction filter) on a lattice. A function is approximated by projecting it onto this space. The projection operation amounts to applying a prefilter to the sampled data in a preprocessing step. This typically yields tremendous improvement of image quality, but requires the results to be stored in a gradient volume in order to be computationally feasible.

Our focus is on improved shading, with the assumption that poor gradient estimation may overshadow the performance of a superior scalar data reconstruction filter, particularly when perceptual metrics are employed. For this reason, we postulate that it is just as important to improve shading as it is to improve the interpolation of the underlying function.

2 Previous Work

There is a vast body of work on function interpolation and reconstruction. There are really two philosophies—1) improving the numerical accuracy based on Taylor series expansions and 2) considering the space of bandlimited functions. The former stems from a local argument and is typically pursued in numerical mathematics and focuses on accuracy in terms of asymptotic error behavior, while the latter is a global constraint grounded in signal processing theory and focuses on smoothness properties. Strang and Fix [26] were the first to try to reconcile these two viewpoints. Later, Unser [30] introduced the framework of reconstruction in shift-invariant spaces and removed the restriction of bandlimited functions so that more general basis functions can be employed. This resulted in the emergence of an elegant unified framework for combining smoothness and accuracy constraints.

In rendering, we are often concerned with the smoothness of the reconstruction. Toward this end, Möller et al. [22] provide a general filter design scheme that extends a purely numerical approach based on a Taylor series expansion by incorporating smoothness constraints. In a different work [23], they contrast two possible approaches to gradient estimation—using a combination of discrete derivative filter with a continuous interpolation filter or simply computing the derivative of the interpolation filter. While in the former

• *The authors are with the School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada. E-mail: {zha13, ualim, torsten}@cs.sfu.ca.*

Manuscript received 25 Aug. 2009; revised 21 Dec. 2009; accepted 30 Dec. 2009; published online 11 Feb. 2010.
Recommended for acceptance by H. Pfister.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2009-08-0187.
Digital Object Identifier no. 10.1109/TVCG.2010.37.

case, one has much better control over smoothness and accuracy, the latter case is simply more attractive since it creates the exact gradient of the interpolated function. Here, we argue that for rendering applications, it is not paramount to compute the exact derivative of the interpolated function, but it is preferable to compute the derivative that is closest to the *true* underlying function. This allows us to incorporate smoothness constraints as well.

Despite these fundamental insights on function reconstruction, not much work has focused on derivative reconstruction, and to the best of our knowledge, no work has been done on designing proper derivative filters for arbitrary lattices. While Unser's approach [30] has been exploited in the visualization community to demonstrate the tremendous impact prefiltering can have on both image quality as well as reconstruction accuracy [3], [14], it remains to be seen how effective prefiltering is for derivative reconstruction. Similarly, for volume visualization tasks, the Body Centered Cubic (BCC) lattice has been shown to outperform the Cartesian Cubic (CC) lattice [24], [28]. However, it is not clear whether the advantages of the BCC lattice extend to gradient reconstruction as well. This paper addresses this issue too.

Outside the visualization community, there has been some progress in the way of gradient estimation on regular lattices. Hamers et al. [15] derive discrete filters for gradient estimation using Lagrange polynomials on a CC lattice. In fact, their discrete filters happen to be particular solutions of the general solution space in our Taylor series approach (Section 3). Sun et al. [27] develop a fourth-order gradient estimation scheme for the hexagonal lattice in 2D only. Hertog et al. [7] compare different discrete derivative filters in the presence of noise. However, assuming a Lagrange polynomial fit for the gradient estimation and using a different basis, for example, tricubic B-splines for data reconstruction may not yield the most optimal surface shading. We address this in our second method (Section 4), where we take the interpolation kernel into account and derive discrete derivative filters that are optimized for the interpolation kernel.

One track of research has focused on designing digital derivative filters in the Fourier domain. These techniques inherently assume an underlying bandlimited signal. Most methods have focused on designing filters in 1D, where derivative reconstruction corresponds to a multiplication with a unit slope ramp in the frequency domain. The ideal discrete derivative filter in that case is the infinite impulse response (IIR) sinc' sampled at the grid points. The continuous derivative can then be recovered by using the sinc as an interpolation kernel on the filtered signal. However, most methods seek to recover the derivative at the grid points only for which a digital filtering solution suffices. Because of the slow decay, sinc' is rarely used in practice and many approximations have been proposed. These approximations proceed by appropriately choosing a design criterion in the frequency domain and then optimizing it to yield either IIR or finite impulse response (FIR) filters in the spatial domain. For example, Dutta Roy and Kumar [9] design 1D FIR filters that are maximally linear over a specified frequency band, and therefore, attempt to match the unit slope ramp as closely as possible within the band. Farid and Simoncelli [13] choose the rotation invariance of the gradient operator in higher dimensions as an optimality criterion to design separable FIR filters. We are unaware of any Fourier domain techniques that design nonseparable derivative filters for arbitrary sampling lattices in higher dimensions.

In comparison to Fourier domain techniques, our proposed methods are different for two main reasons. First, we are interested in volume visualization as a primary application, and for that, we need to accurately estimate derivatives everywhere and not just at the sample points. Second, we are not tied to the bandlimitedness assumption, this allows us to employ approximation-theoretic techniques in our design methodology. The principle that unifies our proposed approaches is the approximation order that governs how the error behaves asymptotically. The Taylor framework (Section 3) gives us a pointwise bound on the error whereas the orthogonal projection (OP) framework (Section 4) yields an error bound on the L_2 norm of the difference between the original function and its approximation.

3 Taylor Series Approach Toward Filter Design

We use the notation $\mathcal{L}$ to characterize an arbitrary d-dimensional lattice generated by the matrix

$$\boldsymbol{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_d], \tag{1}$$

where $\boldsymbol{l}_i$ are the column vectors. Lattice sites of $\mathcal{L}$ are given by the product $\boldsymbol{Lk}$, where the vector $\boldsymbol{k} = (k_1, k_2, \ldots, k_d)^{\mathrm{T}}$ indicates the lattice index. In this section, we derive the Taylor series expansion of a convolution sum in $\mathbb{R}^d$. We follow the 1D analysis of Möller et al. [21], [22] and extend it to multiple dimensions.

3.1 Taylor Expansion of Convolution Sum over a Lattice $\mathcal{L}$

We can decompose a multidimensional function f at the point $\boldsymbol{v} \in \mathbb{R}^d$ about $\boldsymbol{x} \in \mathbb{R}^d$ using the Taylor series as

$$f(\boldsymbol{v}) = \sum_{\boldsymbol{n} \geq \boldsymbol{0}} \frac{(\boldsymbol{v} - \boldsymbol{x})^{\boldsymbol{n}}}{\boldsymbol{n}!} D^{\boldsymbol{n}} f(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{n} \in \mathbb{N}^d$ and $D^{\boldsymbol{n}}$ is a cascaded partial differential operator defined as

$$D^{\boldsymbol{n}}(\cdot) := \left(\frac{\partial^{n_1}}{\partial {x_1}^{n_1}} \frac{\partial^{n_2}}{\partial {x_2}^{n_2}} \cdots \frac{\partial^{n_d}}{\partial {x_d}^{n_d}} \right)(\cdot). \tag{3}$$

Vector factorial and vector exponent have the usual multi-index interpretation, i.e., $\boldsymbol{n}! := \prod_{k=1}^{d} n_k!$, and $\boldsymbol{v}^{\boldsymbol{n}} := \prod_{k=1}^{d} v_k^{n_k}$. With this setup, the function at the lattice sites, i.e., letting $\boldsymbol{v} = \boldsymbol{Lk}$, is given by

$$f(\boldsymbol{Lk}) = \sum_{\boldsymbol{n}} \frac{(\boldsymbol{Lk} - \boldsymbol{x})^{\boldsymbol{n}}}{\boldsymbol{n}!} D^{\boldsymbol{n}} f(\boldsymbol{x}). \tag{4}$$

To capture a varying sampling rate, we uniformly scale the lattice L by a scalar factor h. Intuitively, higher h means a lower sampling rate and vice versa. Denoting $f_r^w(\boldsymbol{x})$ as the result of convolving the function, sampled at the scaled lattice points $h\boldsymbol{Lk}$, with the filter w, defined for the lattice L, we can write

$$f_r^w(\boldsymbol{x}) = \sum_{\boldsymbol{k}} f(h\boldsymbol{Lk}) \cdot w\left(\frac{\boldsymbol{x} - h\boldsymbol{Lk}}{h} \right). \tag{5}$$

By substituting (4) into (5), we obtain

$$f_r^w(\boldsymbol{x}) = \sum_{\boldsymbol{n}} D^{\boldsymbol{n}} f(\boldsymbol{x}) \cdot a_{\boldsymbol{n}}^w(\boldsymbol{x}), \tag{6}$$

where $a_{\boldsymbol{n}}^w(\boldsymbol{x})$, hereinafter referred to as Taylor coefficient, is defined as

$$a_{\boldsymbol{n}}^w(\boldsymbol{x}) := \sum_{\boldsymbol{k}} \frac{(h\boldsymbol{L}\boldsymbol{k} - \boldsymbol{x})^{\boldsymbol{n}}}{\boldsymbol{n}!} \cdot w\left(\frac{\boldsymbol{x} - h\boldsymbol{L}\boldsymbol{k}}{h}\right). \tag{7}$$

This is very similar to what Möller et al. [22] arrived at with their 1D analysis, and therefore, following their approach, we also introduce a continuous variable $\boldsymbol{\tau}$. Let $\boldsymbol{x} = h\,\boldsymbol{L}(\boldsymbol{k}_0 + \boldsymbol{\tau}) : \boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_d)^{\mathrm{T}}, \text{and } \forall i, \tau_i \in [0, 1)$, and where $\boldsymbol{k}_0$ is a lattice site coordinate such that the above condition on $\boldsymbol{\tau}$ is satisfied for any $\boldsymbol{x}$. Rewriting (7) with the new variable $\boldsymbol{\tau}$ and replacing $\boldsymbol{m} = \boldsymbol{k} - \boldsymbol{k}_0$ yields

$$a_{\boldsymbol{n}}^w(\boldsymbol{\tau}) = \frac{h^{\boldsymbol{n}}}{\boldsymbol{n}!} \sum_{\boldsymbol{m}} (\boldsymbol{L}(\boldsymbol{m} - \boldsymbol{\tau}))^{\boldsymbol{n}} \cdot w(\boldsymbol{L}(\boldsymbol{\tau} - \boldsymbol{m})), \tag{8}$$

which is very similar to the 1D counterpart [22]. Finally, we can rewrite the convolution sum in (6) as

$$f_r^w(\boldsymbol{x}) = \sum_{\boldsymbol{n}} D^{\boldsymbol{n}} f(\boldsymbol{x}) \cdot a_{\boldsymbol{n}}^w(\boldsymbol{\tau}), \tag{9}$$

where $\boldsymbol{\tau} = (\frac{1}{h}\boldsymbol{L}^{-1}\boldsymbol{x} - \boldsymbol{k}_0)$ and $\boldsymbol{k}_0 = \lfloor \frac{1}{h}\boldsymbol{L}^{-1}\boldsymbol{x} \rfloor$. Again, the multi-index operator $\lfloor \cdot \rfloor$ has the usual interpretation of taking componentwise floor.

3.2 Classification

A filter w can be classified based on its Taylor coefficients $a_{\boldsymbol{n}}^w(\boldsymbol{\tau})$ given by (8). For example, in 3D, an ideal interpolation filter will have $a_{\boldsymbol{n}}^w(\boldsymbol{\tau}) = 1$ for $\boldsymbol{n} = \boldsymbol{0}$ and $a_{\boldsymbol{n}}^w(\boldsymbol{\tau}) = 0$ for $\boldsymbol{n} \neq \boldsymbol{0}$. Likewise, an ideal first derivative filter along x in 3D will have $a_{\boldsymbol{n}}^w(\boldsymbol{\tau}) = 1$ for $\boldsymbol{n} = (1, 0, 0)$ and 0 otherwise. However, in practical settings, $a_{\boldsymbol{n}}^w(\boldsymbol{\tau})$ may not be equal to zero for all combinations of $\boldsymbol{n}$ and this characterizes the error behavior of a filter, which can be used in filter classification. Before introducing a classification, we introduce a λ-order set η_d^λ that contains all the vectors in $\mathbb{N}^d$ whose l_1-norm is a constant λ. More formally, with $\lambda \in \mathbb{N}$ and d dimensions, a *λ-order* set η_d^λ is defined as

$$\eta_d^\lambda := \left\{ \boldsymbol{v} \in \mathbb{N}^d : \|v\|_1 = \lambda \right\}. \tag{10}$$

Analogously, we define an *$[a, b]$-order* set $\eta_d^{[a,b]}$ as

$$\eta_d^{[a,b]} := \bigcup_{k=a}^{b} \eta_d^k, \tag{11}$$

where $a, b \in \mathbb{N}$ and $0 \leq a \leq b$. Contrary to [22], we define an *n-order filter* (n-OF) and a *k-error filter* (k-EF) separately. The definition of k-EF is the same as that of Möller et al. [22] and depends on the asymptotic order of h in (8). We define an n-order filter to be a filter capable of reconstructing the desired derivative (or function) perfectly for any underlying polynomial of degree n or less. Shortly, we will show that n-OF and k-EF are related to each other depending on the type of derivative we wish to reconstruct.

Given a vector $\boldsymbol{u}_0 \in \eta^{[0,N]}$, which we refer to as a *derivative vector*, a filter w is n-OF if the following is satisfied:

$$a_{\boldsymbol{n}}^w(\boldsymbol{\tau}) = \begin{cases} 0, & \boldsymbol{n} \in \eta^{[0,n]} \quad \text{and} \quad \boldsymbol{n} \neq \boldsymbol{u}_0, \\ \alpha, & \alpha \neq 0, \quad \boldsymbol{n} = \boldsymbol{u}_0. \end{cases} \tag{12}$$

In light of this definition, it is easy to see from (9) that the filter will recover $D^{\boldsymbol{u}_0} f(\boldsymbol{x})$. Therefore, we shall also refer to w as a $\boldsymbol{u}_0$-derivative filter. In 3D, for example, when $\boldsymbol{u}_0 = \boldsymbol{0}$, the filter is actually an interpolation filter.

3.3 Designing First Derivative Filters in $\mathbb{R}^3$

In this section, we focus mostly on designing first derivative filters in 3D for the BCC lattice, and hence, we set $d = 3$ and drop subscripts from the notations accordingly. In this case, $\boldsymbol{u}_0 \in \eta_3^1$.

Möller et al. [21] compared various normal estimation schemes showing that using the analytic derivative of the interpolation filter to estimate gradients for any arbitrary point may not always be superior to using a combination of a discrete filter and a continuous interpolation filter. In this paper, we always prefiltered our discrete scalar data before applying any gradient estimation filter, and therefore, we will use the notation F_{P_H} to denote a discrete function $f[k]$ which is appropriately prefiltered for the interpolation filter H. It is important to note that this prefilter is tied to the interpolation filter and is performed only once as a preprocessing step. Now, denoting D and H to be the discrete derivative and continuous interpolation operators, respectively, we can rephrase their insight as: $F_{P_H}H'$ may not always be numerically superior to $(F_{P_H}D)H$. The numerical accuracy depends largely on the underlying data and also on the type of the interpolation filter in question. We have observed rendering artifacts (Figs. 4 and 5) specially around high-frequency regions using the method $F_{P_H}H'$ with quintic box splines [12] on BCC. In [21], it has also been shown that the schemes $(F_{P_H}D)H$, $(F_{P_H}H)D$ and $F_{P_H}(DH)$ are all numerically equivalent. However, $(F_{P_H}D)H$ is more useful when caching is employed as gradients can be estimated on the same grid as F_{P_H} and the previously computed gradients can be reused irrespective of the type (orthographic or perspective) and step size of the ray traversal in volume rendering. On the other hand, with $(F_{P_H}H)D$, previously computed scalar values cannot be cached to estimate derivatives accurately in perspective projection. Finally, $F_{P_H}(DH)$ may not be trivial to compute in higher dimensions, for example, when using box splines [6], [12]. Therefore, for the rest of this section, we seek to develop a high-quality discrete derivative filter to be used in the normal estimation scheme $(F_{P_H}D)H$. On BCC grids, we shall utilize quintic box spline interpolation [12], and on CC grids, we use tricubic B-spline interpolation.

Considering (8) for a discrete filter Δ implies $\boldsymbol{\tau} = \boldsymbol{0}$. Hence, we drop $\boldsymbol{\tau}$ and write the Taylor coefficients as

$$a_{\boldsymbol{n}}^{\Delta} = \frac{h^{\boldsymbol{n}}}{\boldsymbol{n}!} \sum_{\boldsymbol{m}} (\boldsymbol{L}\boldsymbol{m})^{\boldsymbol{n}} \cdot \Delta(\boldsymbol{L}(-\boldsymbol{m})). \tag{13}$$

The definition (12) is also applicable to any discrete filter.

Möller et al. [22] have shown that a normalization step is necessary before the actual derivative of the function can be extracted properly. This step is very important as it ensures that $a_{\boldsymbol{u}_0}^w(\boldsymbol{\tau})$ and $a_{\boldsymbol{u}_0}^{\Delta}$ evaluate to 1 for the desired derivative $D^{\boldsymbol{u}_0}$. The normalization is performed by simply dividing the filter weights by $a_{\boldsymbol{u}_0}^w(\boldsymbol{\tau})$ or $a_{\boldsymbol{u}_0}^{\Delta}$. With proper normalization, the asymptotic order of the error in terms of h is given by

$O(h^{n+1-m})$ [22], where n is the order of the filter as defined in (12) and m is the l_1-norm of the vector $\boldsymbol{u}_0$. In case of first derivative filters, the error is bound by $O(h^n)$. Hence, we refer to such a filter as n-EF. An n-OF filter that computes first derivatives is also an n-EF filter.

3.4 Linear System for Designing Discrete Filters for BCC

Equation (13) forms a system of linear equations, where we set values for $a^{\Delta}_{\boldsymbol{n}}$ a priori and seek to find the unknowns $\Delta(\boldsymbol{L}(-\boldsymbol{m}))$. The two parameters for this system are as follows:

1. Number of equations: This is determined by how many $a^{\Delta}_{\boldsymbol{n}}$ are fixed, which, in turn, is given by the size of the set $\eta^{[0,n]}$. This parameter decides the order of the filter in definition (12).
2. Number of unknowns: This is given by how many different $\boldsymbol{m}$ vectors, i.e., neighborhood of the filter, we want to restrict the filter to. This parameter therefore decides the support of the discrete filter.

The polynomial approximation order of the interpolating quintic box spline has been shown to be 4, where the interpolation constraint is met by applying a suitable digital prefilter [4], [11]. Thus, to design a good gradient estimator that uses the $(F_{P_H}D)H$ method, we need a discrete derivative filter that has a polynomial order of at least 4. Therefore, in this section, we design a 4-OF discrete derivative filter along x that is suitable for the BCC lattice. For this, we have to impose the following conditions:

- $a^{\Delta}_{\boldsymbol{n}} = 0, \forall \boldsymbol{n} \in \eta^{[0,4]}$, and $\boldsymbol{n} \neq [1,0,0]$.
- $a^{\Delta}_{\boldsymbol{n}} = 1, \boldsymbol{n} = [1,0,0]$.

This leads to 35 equations, and hence, we must have at least 35 unknowns to form a general solution. Interestingly, if we take the vectors $\boldsymbol{m}$ such that $\boldsymbol{L}(-\boldsymbol{m})$ include all the BCC lattice points upto fourth-order neighbors (inclusive) we have 35 unknown weights $\Delta(\boldsymbol{L}(-\boldsymbol{m}))$. However, the linear system formed with this setting is not full rank and produces a five-dimensional solution space with all the five free variables (filter weights) belonging to the set of fourth-order neighbors. Interesting patterns, however, can be seen in the general solution space among the weights of the first-order neighbors. Particularly, we observe a total of 12 symmetries and antisymmetries. We impose similar symmetries on the five free variables, which reduces the solution space to 1D and the resulting filter shows an overall antisymmetry along the x-axis and symmetries along the y and z-axes. At this point, we seek to reduce the error in the 5th polynomial order, and for that, we choose an error metric which weights all the Taylor coefficients equally and is given by

$$E = \sum_{\boldsymbol{n} \in \eta^5} \left(a^{\Delta}_{\boldsymbol{n}}\right)^2. \tag{14}$$

Minimizing this error with respect to the one free variable (filter weight) yields a discrete filter Δ, which has 26 nonzero weights. The supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG. 2010.37, provides the weights of this 26 sample discrete 4-OF first derivative filter (*OPT26*) along the x-axis. Filters for y and z derivatives follow analogously. These filter weights are not unique as they are derived from a general solution space, and hence, an alternate metric that minimizes the support of the filter can be employed. This yields a filter having 16 nonzero weights (*OPT16*, see Table 1). In a similar fashion, we can derive filters for CC and BCC with different orders and support. The properties of some of these filters are listed in Table 1. *BCD* is a 2-OF discrete derivative filter that takes into account the closest 8 first-order neighbors, which form the corners of a box, while *SOCD* uses the two axis-aligned second-order neighbors that are further away in euclidean distance.

TABLE 1
Taylor Series Filters

Filter Type	Filter Name	Approx. Order (OF)	Filter Size
BCC	*SOCD*	2	2
	BCD	2	8
	OPT16	4	16
	OPT26	4	26
CC	*2-cd*	2	2
	4-cd	4	4

The filters are grouped by lattice type and approximation order with corresponding filter size, in number of nonzero weights. The first row shows the BBC filters: Second-Order Central Differencing (SOCD). Box Central Differencing (BCD), Support-Optimal-16 (OPT16), and Error-Optimal-26 (OPT26). The second row provides the CC filters and the prefix in the name denotes approximation order while cd stands for Central Differencing. The weights are given as supplementary material.

3.5 Combination of Discrete and Continuous Filter

We briefly present the effectiveness of combining a discrete first derivative filter and a continuous interpolation filter to approximate first derivatives globally. Following Möller et al. [22], we extend their analysis to arbitrary dimensions and reaffirm the convolution relationship between the Taylor coefficients of the discrete filter and the continuous interpolation filter. Denoting the combined filter by $\Delta w := \Delta * w$, the Taylor coefficient of Δw is given by

$$a^{\Delta w}_{\boldsymbol{n}}(\boldsymbol{\tau}) = \sum_{0 \leq \boldsymbol{i} \leq \boldsymbol{n}} a^{\Delta}_{\boldsymbol{n}-\boldsymbol{i}} \cdot a^{w}_{\boldsymbol{i}}(\boldsymbol{\tau}), \tag{15}$$

where $\boldsymbol{i}$ is the index vector such that $\forall k, 0 \leq i_k \leq n_k$. Möller et al. [21] showed a similar convolution relationship in 1D.

Further, we analyze the polynomial order (OF) of the combined filter. Given a discrete n-OF first derivative filter and an ϵ-OF continuous interpolation filter, the order of the combined filter is given by

$$\min(\epsilon + 1, n). \tag{16}$$

It is rather interesting to note that given $\epsilon < k$, i.e., continuous interpolating filter being worse than the discrete first derivative filter, the combined filter has an order one larger than that of the continuous filter, but can never exceed the quality of the discrete derivative filter.

4 Orthogonal Projections

4.1 Preliminaries

In this section, we shall restrict attention to real-valued, measurable functions that belong to the Hilbert space $L_2(\mathbb{R}^3)$. We refer the reader to [8] for a review of multi-

dimensional signal processing and to [30] for a Hilbert space formulation of 1D signal processing.

We denote by $\langle \cdot, \cdot \rangle$, the inner product associated with $L_2(\mathbb{R}^3)$. It is given by

$$\langle f, g \rangle := \int_{\mathbb{R}^3} f(\boldsymbol{x}) g(\boldsymbol{x}) d\boldsymbol{x},$$

where f and g are functions in $L_2(\mathbb{R}^3)$. We are concerned with seeking an approximation of a function that can be conveniently represented as a linear combination of basis functions on a 3D lattice that are obtained by the appropriate scaling and translation of a generating function $\varphi(\boldsymbol{x})$. Toward this end, let $\mathcal{L}_h$ denote a scaled version of lattice $\mathcal{L}$. We denote a family of basis functions associated with the lattice $\mathcal{L}_h$ by

$$\mathcal{L}_h(\varphi) := \left\{ \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) = \varphi\left(\frac{\boldsymbol{x}}{h} - \boldsymbol{L}\boldsymbol{k}\right) : \boldsymbol{k} \in \mathbb{Z}^3 \right\}.$$

We drop the subscript h when it is unity and the subscript $\boldsymbol{k}$ when talking about a function centered at the origin. We denote the approximation space associated with $\mathcal{L}_h$ by

$$V_{\mathcal{L}_h}(\varphi) := \left\{ s(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^3} c[\boldsymbol{k}] \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) : c[\boldsymbol{k}] \in l_2(\mathbb{Z}^3) \right\},$$

where $c[\boldsymbol{k}]$ is a discrete finite-energy coefficient sequence associated with $\mathcal{L}_h$, and may be quite different from the sample values of f.

The generating function φ must be carefully chosen so as to ensure that the approximation error vanishes as $h \to 0$. The admissibility of φ is determined by the Strang-Fix conditions, which provide a convenient way to characterize the approximation order of φ in terms of the number of zero crossings of $\hat{\varphi}$ (the Fourier transform of φ) on the reciprocal lattice in the frequency domain [25].

The optimal approximation of a given input function f in $V_{\mathcal{L}_h}(\varphi)$, in the least squares sense, is obtained by the orthogonal projection of f onto $V_{\mathcal{L}_h}(\varphi)$ [30], which can be written as

$$\mathrm{P}_{V_{\mathcal{L}_h}(\varphi)}(f) := \sum_{\boldsymbol{k} \in \mathbb{Z}^3} \langle f, \mathring{\varphi}_{h,\boldsymbol{k}} \rangle \varphi_{h\boldsymbol{k}}, \tag{17}$$

where $\mathring{\varphi}$ denotes the dual of φ and $\mathring{\varphi}_{h,\boldsymbol{k}}(\cdot) := h^{-3} \mathring{\varphi}\left(\frac{\cdot}{h} - \boldsymbol{L}k\right)$ is the corresponding scaled and translated version. The dual basis functions satisfy the biorthogonality constraint, namely $\langle \mathring{\varphi}_{h,\boldsymbol{k}}, \varphi_{h,\boldsymbol{l}} \rangle = \delta_{\boldsymbol{k}-\boldsymbol{l}}$, and have the Fourier domain representation

$$\hat{\mathring{\varphi}}(\boldsymbol{\omega}) := \frac{\hat{\varphi}(\boldsymbol{\omega})}{\hat{R}_{\varphi}(\boldsymbol{\omega})} = \frac{\hat{\varphi}(\boldsymbol{\omega})}{\sum_{\boldsymbol{k}} r_{\varphi}[\boldsymbol{k}] \exp(-j\boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{L}k)}, \tag{18}$$

where $\hat{R}_{\varphi}$, as defined above, is the Fourier transform of $r_{\varphi}[\cdot]$, the autocorrelation sequence of φ. This sequence is given by $r_{\varphi}[\boldsymbol{k}] := (\bar{\varphi} * \varphi)(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{L}k}$, where $*$ denotes the continuous convolution operation and $\bar{\varphi}(\boldsymbol{x}) := \varphi(-\boldsymbol{x})$ is the reversed version of φ.

This projection interpretation of sampling is more general as compared to Shannon's view as it encompasses a class of functions that is much larger than the class of bandlimited functions. In fact, the combination of ideal low-pass filtering followed by sampling is equivalent to a projection onto the space of bandlimited functions. In particular, when $\varphi = \mathring{\varphi} = \mathrm{sinc}_{\mathcal{L}}$, the self-dual *sinus cardinalis* function associated with the lattice $\mathcal{L}$, the *analysis function* $\mathring{\varphi}$ plays the role of an ideal low-pass filter, and the *synthesis function* φ plays the role of an ideal interpolator. If it so happens that the given function f is already bandlimited, then the coefficient sequence obtained through (17) is exactly the same as the sequence obtained through point sampling f at the lattice sites. In that case, we can replace $\mathring{\varphi}(\boldsymbol{x})$ with Dirac's delta $\delta(\boldsymbol{x})$ in (17), and the familiar sampling theorem of Shannon follows.

4.2 Gradient Approximation

Typically in visualization applications, a sequence consisting of the sample values of an unknown function f is given to us and we have no control over the choice of the analysis function. Our goal is to accurately estimate ∇f, the gradient of f from the given sampled sequence. This can be accomplished using a two-stage procedure as suggested in [29]. In the first stage, we approximate f in an auxiliary approximation space $V_{\mathcal{L}_h}(\psi)$, and in the second stage, we orthogonally project the gradient of the approximation of f onto another approximation space $V_{\mathcal{L}_h}(\varphi)$. The generating functions ψ and φ can be chosen according to the needs of the application. When accuracy is of prime importance, ψ should be chosen to have a higher approximation order as compared to φ. On the other hand, when visual quality and efficiency are important, ψ can be chosen so that it has comparable smoothness properties. Note that $V_{\mathcal{L}_h}(\psi)$ is an intermediate approximation space that, as we shall soon see, governs the order and size of the discrete derivative filter that is to be applied to the samples of f. The final gradient approximation lies in $V_{\mathcal{L}_h}(\varphi)$, where φ plays the role of an interpolation filter.

Let $f[\boldsymbol{k}] = f(h\boldsymbol{L}k)$ denote the given sampled sequence and $f_1(\boldsymbol{x}) = \sum_{\boldsymbol{k}} c_1[\boldsymbol{k}] \psi_{h,\boldsymbol{k}}(\boldsymbol{x})$ denote the first-stage approximation of f. Since we do not have any knowledge of the underlying function f other than its sample values, we consider a design based on consistency, i.e., the coefficient sequence c_1 should be such that the first-stage approximation should be able to exactly interpolate f at the sample locations. In other words, $f_1(h\boldsymbol{L}k) = \sum_{\boldsymbol{k}} c_1[\boldsymbol{k}] \psi_{h,\boldsymbol{k}}(h\boldsymbol{L}k) = f[\boldsymbol{k}]$. If the basis functions $\psi_{h,\boldsymbol{k}}$ are interpolating (i.e., $\psi_{h,\boldsymbol{k}}(h\boldsymbol{L}m) = \delta_{\boldsymbol{k}-\boldsymbol{m}}$), the coefficient sequence c_1 is exactly equivalent to the sampled sequence f. However, if the basis functions are not interpolating, c_1 can be obtained from the sampled sequence f by applying a suitable digital prefilter [30]. If we denote the prefilter by $p_1[\cdot]$ and its Fourier transform by $\hat{P}_1(\cdot)$, then the first-stage approximation can be written as

$$f_1(\boldsymbol{x}) = \sum_{\boldsymbol{k}} c_1[\boldsymbol{k}] \psi_{h,\boldsymbol{k}}(\boldsymbol{x}) = \sum_{\boldsymbol{k}} (f * p_1)[\boldsymbol{k}] \psi_{h\boldsymbol{k}}(\boldsymbol{x}), \tag{19}$$

where $*$, in this context, denotes the discrete convolution operation and the prefilter p_1 is given in the Fourier domain by

$$\hat{P}_1(\boldsymbol{\omega}) = \frac{1}{\sum_{\boldsymbol{k}} \psi(\boldsymbol{L}\boldsymbol{k}) \exp(-j\boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{L}\boldsymbol{k})}. \tag{20}$$

This prefiltering step not only makes the basis functions interpolating, it is also necessary to utilize the full approximation power of the basis function [2].

In the second stage, we project ∇f_1 onto $V_{\mathcal{L}_h}(\varphi)$. This is tantamount to performing three orthogonal projections, one for each component of the gradient. Let $\partial_i f$ denote the partial derivative $\frac{\partial f}{\partial x_i}$, $i \in \{1,2,3\}$. Using (17) and (19), the second-stage approximation of f is given by

$$\begin{aligned} f_{2,i}(\boldsymbol{x}) &:= \mathrm{P}_{V_{\mathcal{L}_h}(\varphi)}(\partial_i f_1) \\ &= \sum_{\boldsymbol{k}} \langle \partial_i f_1, \mathring{\tilde{\varphi}}_{h,\boldsymbol{k}} \rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{k},\boldsymbol{m}} c_1[\boldsymbol{m}] \langle \partial_i \psi_{h,\boldsymbol{m}}, \mathring{\tilde{\varphi}}_{h,\boldsymbol{k}} \rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{k},\boldsymbol{m}} c_1[\boldsymbol{m}] \langle \partial_i \psi_h, \mathring{\tilde{\varphi}}_{h,\boldsymbol{k}-\boldsymbol{m}} \rangle \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{k}} (c_1 * \mathring{d}_i)[\boldsymbol{k}] \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}), \end{aligned} \tag{21}$$

where $\mathring{d}_i$ is a digital derivative filter given by the inner product

$$\mathring{d}_i[\boldsymbol{n}] := \langle \partial_i \psi_h, \mathring{\tilde{\varphi}}_{h,\boldsymbol{n}} \rangle. \tag{22}$$

4.3 Examples

Once ψ and φ have been chosen, the remaining key step in the above scheme is the evaluation of the inner product (22) that yields the discrete derivative filter $\mathring{d}_i$. Here, we focus on the CC and BCC lattices and show how the 1D B-splines can be used to design derivative filters that implement the above two-stage approximation scheme. In particular, on the CC lattice, we work with generating functions formed by tensor product B-splines, and for the BCC lattice, we employ the family of box splines introduced by Entezari et al. [12].

Let $\beta^n(x)$ denote the centered 1D B-spline of degree n. It is given by [31]

$$\beta^n(x) := \sum_{j=0}^{n+1} \frac{(-1)^j}{n!} \binom{n+1}{j} \left(x + \frac{n+1}{2} - j \right)_+^n, \tag{23}$$

where $(x)_+^n = \max(0,x)^n$ is the one-sided power function. We also make use of the noncentered 1D B-splines that we denote by $\beta_{\triangleright}^n(x)$. These have support in the interval $[0, n+1]$ and are related to the centered B-splines through $\beta_{\triangleright}^n(x) := \beta^n(x - \frac{n+1}{2})$.

A very useful property of the B-splines that we shall exploit is that their derivatives can be expressed in terms of lower degree B-splines. In particular,

$$\begin{aligned} \acute{\beta}^n(x) &:= \frac{d\beta^n(x)}{dx} = \beta^{n-1}\left(x + \frac{1}{2}\right) - \beta^{n-1}\left(x - \frac{1}{2}\right), \text{ and} \\ \acute{\beta}_{\triangleright}^n(x) &:= \frac{d\beta_{\triangleright}^n(x)}{dx} = \beta_{\triangleright}^{n-1}(x) - \beta_{\triangleright}^{n-1}(x-1). \end{aligned} \tag{24}$$

Before delving into the specifics of CC and BCC lattices, we mention the convolution interpretation of (22) that will aid us in our design. Due to the shift-invariance of the basis functions, the inner product in (22) can be seen as a sampled convolution, i.e.,

$$\langle \partial_i \psi_h, \mathring{\tilde{\varphi}}_{h,\boldsymbol{n}} \rangle = (\partial_i \psi_h * \bar{\mathring{\tilde{\varphi}}}_h)(\boldsymbol{x})\big|_{\boldsymbol{x}=h\boldsymbol{L}\boldsymbol{n}} = \frac{1}{h}\big(\partial_i \psi * \bar{\mathring{\tilde{\varphi}}}\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{L}\boldsymbol{n}}.$$

The latter convolution can be expressed in one of the three equivalent forms, $(\partial_i \psi * \bar{\mathring{\tilde{\varphi}}}) = (\psi * \partial_i \bar{\mathring{\tilde{\varphi}}}) = \partial_i (\psi * \bar{\mathring{\tilde{\varphi}}})$, which can be easily verified in the Fourier domain. Furthermore, when $\mathring{\tilde{\varphi}}$ is symmetric (i.e., $\bar{\mathring{\tilde{\varphi}}} = \mathring{\tilde{\varphi}}$), as is the case with tensor-product-centered B-splines on CC and the box splines on BCC, this convolution can be simplified by expanding the dual $\mathring{\tilde{\varphi}}$ in terms of the primal basis functions in $\mathcal{L}(\varphi)$. The digital derivative filter in (22) can then be written as $\mathring{d}_i = (d_i * p_2)$, where d_i is obtained from the primal φ through

$$d_i[\boldsymbol{n}] := \frac{1}{h}\big(\partial_i(\psi * \varphi)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{L}\boldsymbol{n}}, \tag{25}$$

and p_2 is a digital postfilter that has the Fourier transform $\hat{P}_2(\boldsymbol{\omega}) := 1/\hat{R}_{\varphi}(\boldsymbol{\omega})$ (cf., (18)).

The orthogonal projection scheme (21) that approximates the first partial derivative can now be compactly written as

$$f_{2,i}(\boldsymbol{x}) = \sum_{\boldsymbol{k}} (p_1 * f * d_i * p_2)[\boldsymbol{k}] \varphi_{h,\boldsymbol{k}}(\boldsymbol{x}). \tag{26}$$

4.3.1 CC Lattice

The CC lattice $\mathcal{I} = \mathbb{Z}^3$ is generated by the matrix $\boldsymbol{I} := \mathrm{diag}(1,1,1)$. The 1D centered B-splines are easily extended to the CC lattice via the tensor product. Due to their separability, tensor product B-splines are easy to manipulate and provide a convenient way to design a derivative filter according to (25).

Let us denote the $(n+1)$-EF tensor product trivariate B-spline as

$$b^n(\boldsymbol{x}) := \prod_{i=1}^{3} \beta^n(x_i). \tag{27}$$

We choose $\mathcal{I}_h(b^m)$ and $\mathcal{I}_h(b^n)$ as the bases for the first and second approximation stages, respectively. The first-stage prefilter is given by the samples of b^m at the lattice sites as given in (20). Since the B-splines are symmetric, we use (25) to obtain the digital derivative filter. Due to the fact that the convolution of the two B-splines yields another B-spline of a higher degree [31], the CC derivative filter takes the form

$$\begin{aligned} d_i[\boldsymbol{k}] &= \frac{1}{h}\big(\partial_i (b^m * b^n)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{k}} = \frac{1}{h}\big(\partial_i b^{m+n+1}\big)(\boldsymbol{k}) \\ &= \frac{1}{h} \acute{\beta}^{m+n+1}(k_i) \prod_{j \neq i} \beta^{m+n+1}(k_j). \end{aligned} \tag{28}$$

This filter only needs to be evaluated once, derivative filters for other directions are given by appropriate permutations. For instance, if $d_1[\boldsymbol{k}]$ is known, d_2 and d_3 are given by

$$d_2[\boldsymbol{k}] = d_1[k_2, k_1, k_3] \text{ and } d_3[\boldsymbol{k}] = d_1[k_3, k_2, k_1]. \tag{29}$$

Finally, the postfilter is obtained from the autocorrelation sequence $r_{b^n}[\cdot]$ which, due to the symmetry of B-splines, consists of samples of $(b^n * b^n) = b^{2n+1}$ at the lattice sites.

4.3.2 BCC Lattice

The BCC lattice $\mathcal{H}$ is generated by the matrix

$$\boldsymbol{H} := \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix},$$

and is a sublattice of $\mathcal{I}$ consisting of those points that have the same parity (coordinates are either all odd or all even). The Voronoi cell of each lattice site of $\mathcal{H}$ is a truncated octahedron having a volume of 4.

The four-directional BCC box splines, as introduced by Entezari et al. [12], generate bases that satisfy the Strang-Fix relations and have mathematical properties that are very similar to the tensor product B-splines. Box splines, in general, have various equivalent definitions that make use of the generating direction vectors either in the spatial domain or in the Fourier domain [6], [12]. Here, we follow a somewhat different approach by using the projection interpretation of the BCC box splines as it allows us to easily extend the B-spline framework to BCC.

A BCC box spline can be constructed by projecting a 4D tensor product B-spline along the antipodal axis of the supporting tesseract [12]. Let $\boldsymbol{\xi}$ be the 4D column vector $\boldsymbol{\xi} := (\boldsymbol{x}, t)^{\mathrm{T}} = (x_1, x_2, x_3, t)^{\mathrm{T}}$ and $\boldsymbol{\Theta}$ be the 4D rotation matrix

$$\boldsymbol{\Theta} := \frac{1}{2}[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4] = \frac{1}{2}\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

that rotates 4D space so that the antipodal axis of the tesseract is parallel to the t-axis [12]. We can now define the BCC box spline of order n as the projection of a 4D tensor product function consisting of dilated noncentered B-splines of degree $(n/2-1)$. We write this as

$$\Xi^n(\boldsymbol{x}) := \frac{1}{4}\int_{t=0}^{2n} \prod_{j=1}^{4} \beta_{\triangleright}^{\frac{n}{2}-1}\left(\frac{1}{4}\boldsymbol{\theta}_j^{\mathrm{T}}\boldsymbol{\xi}\right) dt, \tag{30}$$

where $n \in 2\mathbb{Z}_+$ and the leading factor of $1/4$ ensures that the box splines are normalized to have an integral of 4 (volume of the Voronoi cell of $\mathcal{H}$) over their support. In contrast to the definition (27) of tensor product B-splines, n is the approximation order of the box spline rather than the degree of the constituent B-splines. The support of Ξ^n is a rhombic dodecahedron that has its 14 vertices at the lattice sites $(\pm n/2, \pm n/2, \pm n/2)$, $(\pm n, 0, 0)$, $(0, \pm n, 0)$, and $(0, 0, \pm n)$. Since the B-splines are piecewise polynomials, the integral in (30) can be analytically evaluated for arbitrary $\boldsymbol{x} \in \mathbb{R}^3$.

Analogous to the CC case, let us choose the first- and second-stage approximation bases as $\mathcal{H}_h(\Xi^m)$ and $\mathcal{H}_h(\Xi^n)$, respectively. The first-stage prefilter is related to the samples of Ξ^m through (20). Like a tensor product B-spline, a BCC box spline is symmetric and can be represented as a convolution of lower order BCC box splines. The BCC derivative filter (25) therefore becomes

$$d_i[\boldsymbol{\kappa}] = \frac{1}{h}\big(\partial_i(\Xi^m * \Xi^n)\big)(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{H}\boldsymbol{\kappa}} = \frac{1}{h}\big(\partial_i \Xi^{m+n}\big)(\boldsymbol{H}\boldsymbol{\kappa}),$$

which, after using (30) and the derivative relation in (24), simplifies to

$$d_i[\boldsymbol{\kappa}] = \frac{1}{8h}\int_{t=0}^{4(\tilde{n}+1)} \sum_{j=0}^{4} \Theta_{ji} \acute{\beta}_{\triangleright}^{\tilde{n}}\left(\frac{1}{4}\boldsymbol{\theta}_j^{\mathrm{T}}\begin{bmatrix}\boldsymbol{H}\boldsymbol{\kappa} \\ t\end{bmatrix}\right) \prod_{k\neq j} \beta_{\triangleright}^{\tilde{n}}\left(\frac{1}{4}\boldsymbol{\theta}_k^{\mathrm{T}}\begin{bmatrix}\boldsymbol{H}\boldsymbol{\kappa} \\ t\end{bmatrix}\right) dt, \tag{31}$$

where $\tilde{n} := \frac{m+n}{2} - 1$. The integrand above is also a piecewise polynomial and can be analytically evaluated. It is easy to verify that the permutation relation (29) is also applicable here.

TABLE 2
Orthogonal Projection Derivative Filters

ψ \ φ	b^1	b^3
b^1	*ll* 18	
b^3	*cl* 100	*cc* 294
b^5	*ql* same as *cc*	*qc* 648

(a)

ψ \ φ	Ξ^2	Ξ^4
Ξ^2	*LL* 10	
Ξ^4	*QL* 52	*QQ* 150
Ξ^6	*NL* same as *QQ*	*NQ* 328

(b)

(a) CC. (b) BCC. The filters are grouped according to the approximation order of the first- and second-stage basis functions. Filter sizes, in terms of the number of nonzero filter weights, are shown. The filters are named according to the degree of the polynomials that make up the basis functions; on CC, l-trilinear, c-tricubic and q-triquintic; and on BCC, L-linear, Q-quintic, and N-nonic. Filter weights are provided as supplementary material.

Finally, the autocorrelation sequence $r_{\Xi^n}[\cdot]$ is needed for the postfilter. It is obtained by sampling the box spline $(\Xi^n * \Xi^n) = \Xi^{2n}$ at the lattice sites.

5 Applicability Analysis

5.1 Fourier Domain Analysis

When working with signals that are either bandlimited or sufficiently oversampled, the quality of a filter can be characterized in the Fourier domain in terms of its deviation from the ideal filter. On the Cartesian lattice, it is a common practice to design one-dimensional filters and then extend them to higher dimensions via a tensor product. In this section, we compare the frequency behavior of some of our derivative filters for the CC lattice. To simplify the analysis, we focus on those filters that are to be used in combination with the cubic B-spline (see Table 2).

Derivatives in higher dimensions are usually computed by using a one-dimensional derivative filter along a canonical direction. For such a scenario, it suffices to compare the 1D frequency profiles of the filters. Fig. 1 shows the response of the 1D versions of our OP filters *qc* (eight nonzero weights) and *cc* (six nonzero weights) along with the responses of some other digital derivative filters that are combined with the prefiltered cubic B-spline. Roy et al.'s filter, despite its largest kernel size, is inferior to the OP filters which have the best passband behavior. However, this gain comes at the expense of a deteriorated postaliasing. ϵ-*cd* fares similarly in the passband but has the worst postaliasing performance which attests the fact that taking the analytical derivative of the interpolation function may not be the best choice.

Applying 1D filters that have no off-axis contribution is not the only way to compute gradients in higher dimensions. As we have seen, the OP framework when applied to the CC lattice leads to derivative filters that have nonzero components along all directions. These filters can be expressed as a tensor product of a 1D antisymmetric filter and a 1D symmetric filter (see (28)). The resulting Fourier transform is also separable. Farid and Simoncelli [13] have also developed similar filters for the Cartesian lattice by optimizing the rotation invariance of the gradient operator. In Fig. 2, we compare the performance of our OP filter *cc* with one of

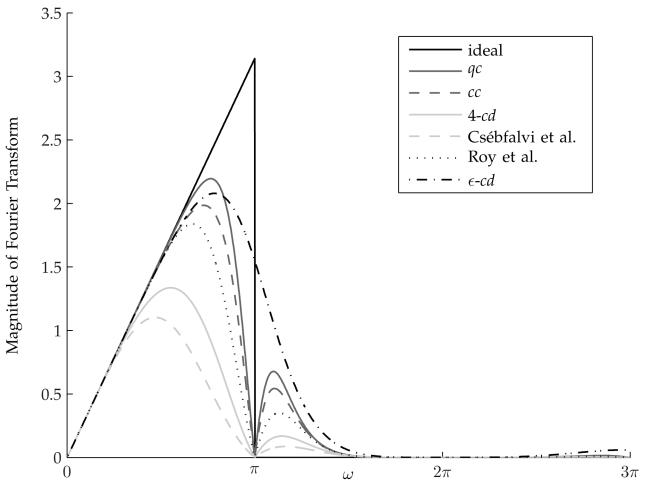

Fig. 1. Frequency response of various 1D derivative filters used in combination with the cubic B-spline. The 4 nonzero weight, 4-EF filter of Csébfalvi and Domonkos [5] is combined with the cubic B-spline with no prefiltering. With prefiltering, it leads to the same weights as our 4-*cd* filter. Roy and Kumar's 14 nonzero weight filter is designed to be maximally linear in the passband [9]. ϵ-*cd* is the analytical derivative of the prefiltered cubic B-spline.

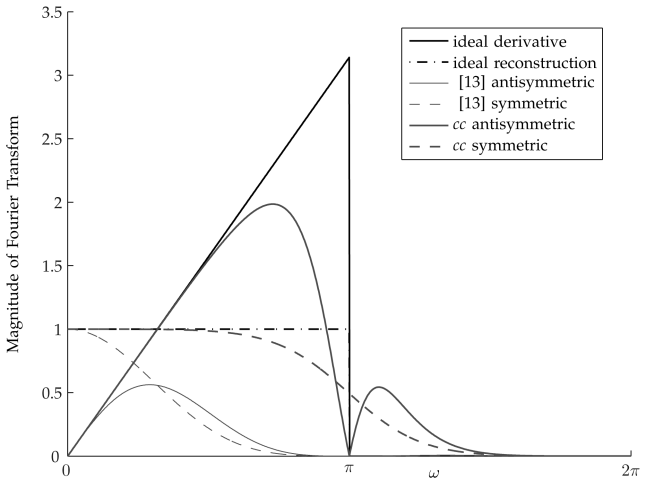

Fig. 2. Farid and Simoncelli's [13] 294 ($6 \times 7 \times 7$) nonzero weight tensor product derivative filter combined with the prefiltered cubic B-spline as compared to our derivative filter *cc*. The right half of the antisymmetric filter is $[0, -0.193091, -0.125376, -0.018708]$ while the right half of the symmetric filter is $[0.361117, 0.245410, 0.069321, 0.004711]$, the first component is at the origin.

their first-order derivative filters that has the same kernel size. It is clear that the OP framework yields filters that are not only optimal in the L_2 sense but also closest to the ideal in the passband. The rotation-invariant criterion leads to filters that reconstruct a spectrally reshaped signal [13].

Since nonseparable lattices such as the BCC lattice have been shown to improve scalar reconstruction quality [10], [20], [28], we believe that the OP framework when extended to the BCC lattice should improve gradient reconstruction quality as well.

5.2 Runtime and Storage Analysis

We proposed two different gradient estimation frameworks wherefrom practitioners can choose an appropriate one considering the lattice type, memory space available, and the extent of numerical accuracy required. While the focus of our paper is the evaluation of accuracy, in this section, we consider the time-space requirement of both the frameworks.

Consideration for the OP framework (Section 4) is rather straightforward. Discrete filters developed using this method are often large, and therefore, practical implementation is feasible when gradients are precomputed, for example, as three separate gradient volumes (one for each component) with each having the same number of elements as the data volume itself. If all the four volumes (including the data volume) fit into memory, then using OP will not only yield the most accurate gradients but will also be the fastest; as for every sample we just need to perform four interpolations (one for the data and three for the gradient). Data streaming techniques can be employed when storage space is not the limiting factor but RAM is.

The primary advantage of the Taylor series framework lies in the fact that filters can be designed with sufficient compactness along with choosing polynomial order (OF). This keeps the filter size small enough for gradients to be computed on the fly without storing them beforehand. A typical implementation is shown in Algorithm 1. Implementation can be further optimized if caching is employed carefully. In most visualization applications (for example, ray tracing), sampling is performed in a sequential manner and often the sampling step is very small compared to the grid spacing. For such a setting, step 4 in Algorithm 1 can be optimized considerably by employing a cache whereby previously computed gradients at the lattice sites are reused.

Algorithm 1. Compute the gradient at an arbitrary point $\boldsymbol{x}$, using an interpolation filter w and discrete derivative filters Δ_x, Δ_y and Δ_z, one for each component. G denotes a set of lattice site gradients, that are computed on the fly.

Require $\boldsymbol{x}, w, \Delta_x, \Delta_y$ and, Δ_z

Ensure $\boldsymbol{v}$ is the gradient at $\boldsymbol{x}$

1: $G \leftarrow \emptyset$
2: **for all** {$\boldsymbol{k}$: Lattice sites that are within the support of the interpolation filter w} **do**
3: $\Omega \leftarrow$ {All the data within the support of $\Delta_x, \Delta_y,$ and $\Delta_z,$ centered at $\boldsymbol{k}$}
4: $\boldsymbol{g} \leftarrow$ Compute the lattice site gradient at $\boldsymbol{k}$ using Ω and the discrete derivative filters $\Delta_x, \Delta_y,$ and Δ_z
5: $G \leftarrow G \cup \boldsymbol{g}$
6: **end for**
7: $\boldsymbol{v} \leftarrow$ Compute the gradient using the set G and the filter w
8: **return** $\boldsymbol{v}$

In this section, we compare the complexity of gradient estimation on different lattices and for different filters. For this comparison, we compute the total number of multiplications and data accesses, as indicators of computational and data fetch overheads, respectively, incurred by a discrete derivative (Taylor series filters) and interpolation filter combination. For simplicity, we will restrict our analysis to 3D and assume that the gradient will be computed on the fly for a single arbitrary point, without reusing results from any previously computed gradients, using Algorithm 1.

Let H and D denote the support size of an interpolation filter and a discrete derivative filter, respectively. In CC, due to the separable nature of the discrete derivative filters, the

TABLE 3
Efficiency Analysis

	b^1		b^3	
	DA	M	DA	M
2-*cd*	48	72	384	576
4-*cd*	96	120	768	960

(a)

	Ξ^2		Ξ^4	
	DA	M	DA	M
SOCD	24	36	192	288
BCD	32	108	256	864
OPT16	64	204	512	1632
OPT26	136	324	1088	2592

(b)

(a) CC. (b) BCC. Number of data accesses (DA) and multiplications (M) for various discrete and interpolation filter combinations.

total number of data accesses incurred in step 3 of Algorithm 1 is simply given by $3 \times D$, i.e., the size of the set Ω in that step. Therefore, a total number of $3 \times D \times H$ data accesses will be required to compute the gradient at $\boldsymbol{x}$. In step 4 (Algorithm 1), the number of multiplications will also be given by $3 \times D$. Considering that three separate interpolations are required in step 7 (one for each component of $\boldsymbol{v}$), the total number of multiplication incurred can therefore be given by $3 \times D \times H + 3 \times H = 3 \times H(D+1)$. So, for example, in CC, with the 4-*cd* convolved with the tricubic B-spline (we will use the notation $4\text{-}cd * b^3$ to denote this convolution), the total number of data accesses will be $3 \times D \times H = 3 \times 4 \times 64 = 768$; and similarly, the total number of multiplications will be $3 \times H(D+1) = 3 \times 64(4+1) = 960$.

In BCC, on the other hand, due to the nonseparable structure, the discrete derivative filters Δ_x, Δ_y, and Δ_z (see supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.37) access a lot of common data, if not exactly the same, in step 3 of Algorithm 1. For instance, with the $OPT16$, the size of the set Ω in step 3 will be just 16 because all the three filters Δ_x, Δ_y, and Δ_z use exactly the same data. But in case of the $OPT26$, due to the geometrical structure of the filters, a total number of 34 data points are accessed. Therefore, in BCC, the total number of data accesses will be given by $|\Omega| \times H$, where $|\Omega|$ denotes the size of the set Ω in step 3. However, the number of multiplications will still be given by the same formula as that of the CC, i.e., $3 \times H(D+1)$, where now D is the support of a discrete derivative filter for BCC. This is because the filter coefficients (Δ_x, Δ_y, and Δ_z) are different from one another even though they may access the same set of data. Now it is easy to verify that the $OPT16$ convolved with the quintic box spline Ξ^4 ($OPT16 * \Xi^4$) will access 512 data points and incur 1,632 multiplications to compute a gradient.

We summarize our analysis in Table 3, which shows that the $OPT16 * \Xi^4$ has 1.5 times lower data access overhead compared to the $4\text{-}cd * b^3$. This is interesting because, even though the 1D separable 4-*cd* has a much smaller support, the overall process ends up accessing more data compared to its BCC counterpart. However, $OPT16 * \Xi^4$ has 1.7 times higher computational overhead compared to $4\text{-}cd * b^3$. On the other hand, Table 4 reveals that the error difference between *OPT16* and *OPT26* is rather small while the former has a much smaller support. All these make *OPT16* a good practical purpose 4-OF filter for BCC. Similar trends can also be seen between $2\text{-}cd * b^3$ and a BCC counterpart, $BCD * \Xi^4$, where the latter accesses 1.5 times less data but at the same time incurs 1.5 times more multiplications. Again, Table 4 reveals that *BCD* is superior among all the 2-OF filters and this makes it a good practical choice as a 2-OF filter in BCC.

TABLE 4
Quantitative Results

	f_{test}			ML		
	l	θ	time	l	θ	time
2-*cd*	19.0	35.3	1.59/35.81	3.13	49.4	2.23/22.56
4-*cd*	17.9	31.8	2.55/36.06	2.79	44.1	2.64/21.77
ϵ-*cd*	11.9	22.4	0.85/35.63	1.52	25.7	0.86/19.10
ll	18.6	34.1	0.03/34.77	2.76	32.4	0.06/18.67
cl	17.6	32.2	0.08/33.91	2.20	29.8	0.04/19.99
ql	17.4	31.6	0.06/34.02	2.09	29.9	0.05/19.87
cc	16.1	27.7	1.23/34.07	1.78	23.8	1.69/19.94
qc	15.9	27.6	1.31/35.54	1.69	24.5	1.63/20.79

(a)

	f_{test}			ML		
	l	θ	time	l	θ	time
SOCD	22.0	70.3	1.43/35.58	3.78	68.9	1.84/21.58
BCD	15.8	18.7	1.89/34.57	2.88	39.2	2.37/21.44
OPT16	13.7	17.0	2.39/35.96	2.42	28.7	2.65/22.07
OPT26	13.5	16.1	3.00/34.44	2.42	28.7	4.01/23.17
ϵ-*CD*	9.8	12.0	0.88/34.95	1.61	26.1	0.97/20.55
LL	15.1	22.9	0.17/32.96	2.50	29.3	0.20/18.87
QL	13.1	22.8	0.10/33.60	1.91	28.0	0.11/18.80
NL	12.5	22.9	0.13/33.11	1.80	29.1	0.14/18.96
QQ	10.5	13.0	1.15/33.39	1.38	19.4	1.51/20.87
NQ	9.7	12.5	1.16/34.63	1.29	21.9	1.56/20.42

(b)

(a) CC. (b) BCC. RMS length of the error vector (l) and RMS angular deviation (θ in degrees) on the visible isosurface. Normal computation time (in seconds) versus total render time is indicated. The comparison is performed on the 0.4 isosurface of f_{test} and the 0.5 isosurface of ML. For f_{test}, $\epsilon = 0.005$ and for ML, $\epsilon = 0.003$. All images were rendered at a resolution of 800×800 pixels.

6 Results and Discussion

We followed the recipes presented in Sections 3 and 4 to design gradient estimation filters of different orders for both the CC and BCC lattices. A summary of our Taylor filters is given in Table 1 while the orthogonal projection filters are summarized in Table 2.

The gradient filters presented in Table 1 are defined in the spatial domain and have compact support. We therefore implemented them so that the gradients are estimated on the fly using Algorithm 5.2. On the other hand, the OP gradient filters in Table 2 have comparatively larger kernels and need to be combined with pre and postfilters ((20) and (18), respectively) that are defined in the frequency domain. To efficiently implement these filters, we employed the multidimensional discrete Fourier transform (MDFT) in a preprocessing step to yield a gradient volume. On the CC lattice, the MDFT can be evaluated using a tensor product fast Fourier transform (FFT). The MDFT on the BCC lattice is nonseparable, but can still be efficiently evaluated using the FFT as recently shown by Alim and Möller [1]. We also used the MDFT to prefilter scalar data when interpolating with

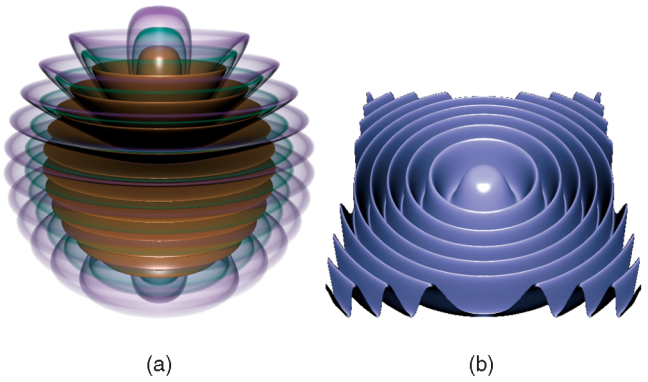

Fig. 3. Isosurfaces of the unsampled synthetic functions. (a) The modified test function, $\alpha = 0.25$, $\gamma = 2$, and $f_m = 6$, showing the isovalues 0.4 (rendered opaque), 0.5 (green), and 0.6 (purple). (b) The ML test function, $\text{isovalue} = 0.5$ and other parameters as given in [19].

either the tricubic B-splines on CC or the quintic box spline on BCC in order to fully exploit the approximation power. The cost of this prefiltering step is negligible as compared to the cost of the subsequent rendering operations.

6.1 Implementation

For OP filters, we used two separate volumes to store prefiltered scalar values and precomputed gradients, whose components were interleaved in memory, respectively. On the other hand, we only stored the prefiltered scalar volume in memory for all the Taylor series filters and always computed gradients on the fly. We evaluated ray-casting integrals in two modes:

- **Isosurface Rendering (ISR)**: A given isosurface is extracted along a ray in the volume using a Linear Bisection technique, and once the isosurface is found, the gradient is estimated at that point and shaded accordingly. Only scalar interpolation is performed during the isosurface extraction stage. Keeping the underlying interpolation filter the same allows us to investigate how the quality of the rendered images changes as a result of different gradient estimation schemes.
- **Direct Volume Rendering (DVR)**: For every sample taken along the ray, scalar interpolation is performed and a transfer function is evaluated. The gradient is estimated only when the transfer function is nonzero.

We implemented our volume ray caster as a single-threaded application and ran all our experiments on an Intel Core 2 Duo (2.40 GHz on each core) machine with 4 GB RAM running Linux. We also optimized our codes using compiler (GCC version 4.4.1) level optimization flags (`-march=core2 -O6 -ffast-math -funroll-all-loops -ftree-vectorize`) turned on. All reported timing data are obtained using these codes.

6.2 Synthetic Data

We used the popular synthetic function proposed by Marschner and Lobb (ML) [19]. This function has a form that allows one to correctly point sample it so as to ensure that there is no aliasing of the spectrum in the frequency domain. In practice, however, one may have little to no knowledge of the underlying frequency content of a sampled signal in which case, a reconstruction that attempts to minimize the L_2 norm of the error is a more desirable one. Furthermore, isosurfaces of the ML function are not closed manifolds and error is introduced near the boundaries of the sampling window since data outside the window need to be fetched to accurately reconstruct the function or its gradient. For these reasons, we also employed an appropriately modified version of the ML function so that the isosurfaces are closed manifolds that radiate spherically outward with increasing isovalue (Fig. 3). The resulting function can be written in Cartesian coordinates as

$$f_{\text{test}}(\boldsymbol{x}) := \gamma\|\boldsymbol{x}\| - \alpha\cos\left(2\pi f_m \frac{x_3}{\|\boldsymbol{x}\|}\right), \tag{32}$$

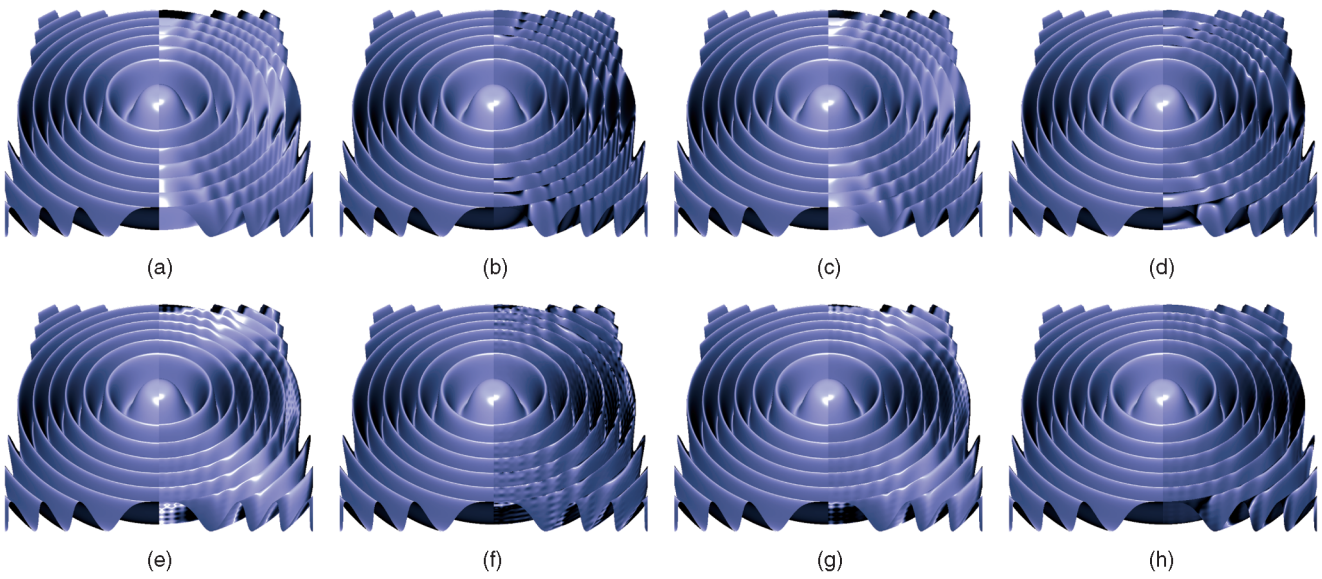

Fig. 4. Isosurface of the ML function shaded using different normal estimation schemes; top row, CC, and bottom row, BCC. The analytic form of the ML function is used to compute the isosurface and the sampled data are used for normal estimation. To facilitate comparison, the left half of each image shows the truth. For (b) and (f), $\epsilon = 0.003$. (a) *2-cd*. (b) ϵ-cd. (c) *4-cd*. (d) *qc*. (e) *BCD*. (f) ϵ-CD. (g) *OPT16*. (h) *NQ*.

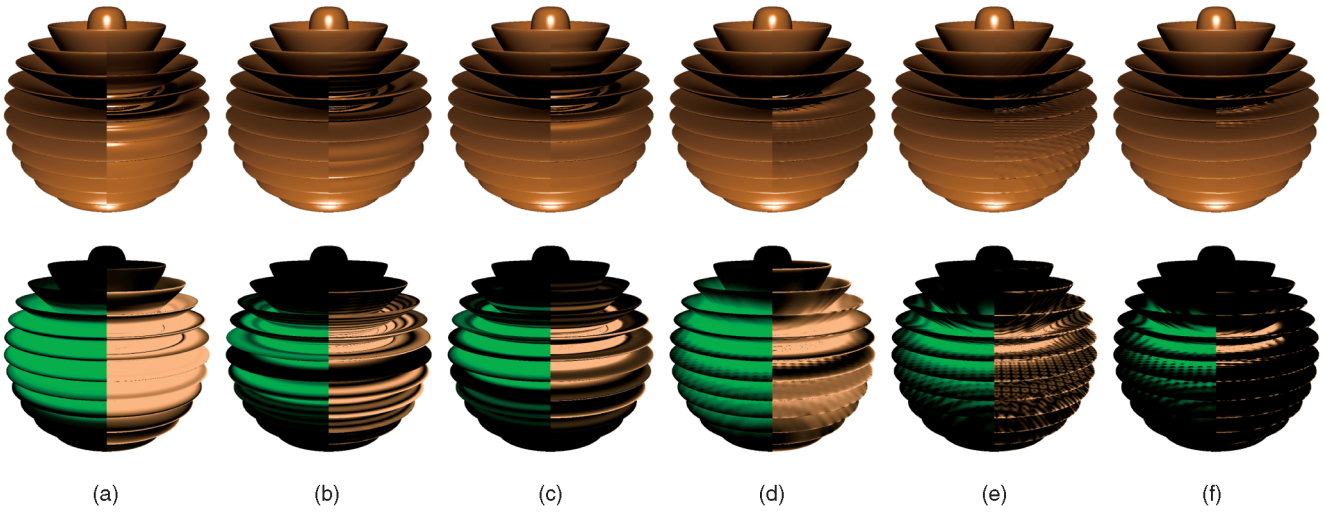

Fig. 5. Reconstructed isosurface of f_{test} shaded using different gradient estimation schemes. The top row shows the rendered images as compared to the truth while the bottom row shows the corresponding error images. The left half of an error image illustrates the l_2-norm of the error vector, where a value of 15 or more is mapped to the brightest green. The right half illustrates the angular deviation, where an angle of 15 degrees or more is mapped to the brightest orange. For (b) and (e), $\epsilon = 0.005$. (a) *4-cd*. (b) ϵ-*cd*. (c) *cc*. (d) *OPT16*. (e) ϵ-CD. (f) *QQ*.

where $\boldsymbol{x} \in \mathbb{R}^3$ $(\boldsymbol{x} \neq 0)$ and γ, α, and f_m are positive real parameters. The cosine frequency modulation form akin to the ML function can be obtained by expressing the above equation in spherical coordinates. As $\boldsymbol{x} \to \mathbf{0}$, the oscillation frequency of this function tends to infinity. Thus, for any finite sampling rate, one can always choose an isosurface that would be a demanding test for any reconstruction filter.

We point sampled both the ML function and f_{test} within a $(-1, 1)^3$ window on CC and BCC lattices. For the ML function, we used the parameters given in [19] and sampled the function on a $41 \times 41 \times 41$ CC grid and on an equivalent $32 \times 32 \times 64$ BCC grid. For f_{test}, we used the parameters shown in Fig. 3 and sampled it on CC and BCC grid sizes of $101 \times 101 \times 101$ and $80 \times 80 \times 160$, respectively.

We performed ISR experiments using both test functions. For the ML function, we chose an isovalue of 0.5, whereas for f_{test}, we chose an isovalue of 0.4. We used the analytic form of the functions to compute isosurface intersections and used the sampled versions solely for normal estimation. This ensures that the underlying shape of the isosurface is the same for both lattice types. For the Taylor filters, gradients were computed on the fly using the prefiltered sampled data and combined with either tricubic B-spline or quintic box spline interpolation depending on the lattice. On the other hand, for the OP filters, the stored gradient volume was used to interpolate gradients at nongrid points. The interpolation filter used is governed by the basis function used in the second stage as indicated by the second letter of the filter name.

Fig. 4 shows the isosurface of the ML function shaded with different gradient estimation schemes. The terms ϵ-*cd* (on CC) and ϵ-CD (on BCC) refer to estimating the gradient locally at the point of intersection by computing the gradient of the interpolated function using central differences in the axial directions with a step size of ϵ. As ϵ goes to zero, we recover the analytic derivative of the interpolated function. The superiority of the BCC lattice is clearly evident; the BCC filters *BCD* and *OPT16* do a better job at reconstructing the normals than their CC counterparts *2-cd* and *4-cd*. The difference between *OPT16* and *BCD* is also more apparent than the difference between *4-cd* and *2-cd*. Additionally, we observe that ϵ central differencing yields better normal estimates as compared to the second and fourth-order Taylor filters. However, ϵ-*CD* on BCC gives rise to rippling artifacts, which are absent in ϵ-*cd* on CC. The OP filters *qc* and *NQ* outperform all the other filters in their respective categories. With *NQ*, the artifacts introduced by ϵ-*CD* are removed and the appearance of the isosurface is closest to the truth.

We also used the test functions to quantify the performance of the filters and measured the Root Mean Square (RMS) l_2-norm of the difference between the true gradient and the estimated gradient, as well as the RMS angular deviation from the truth, on the visible isosurface. Besides numerical accuracy, we also measured the time taken to estimate gradients. The resulting data are tabulated in Table 4 and some of the isosurface renderings of f_{test} along with the error distributions are shown in Fig. 5. Note that we do not report the scalar interpolation time because scalar values were evaluated from the analytic functions.

Our numerical results corroborate the fact that the advantages of BCC sampling extend to gradient reconstruction as well. The BCC filters yield significantly lower RMS error values as compared to their CC cousins. We observe the same trend quantitatively that we qualitatively saw in Fig. 4; ϵ-central differencing is better than the second and fourth-order Taylor filters, and the higher order OP filters are comparable in accuracy to ϵ-central differencing. On the BCC side, however, orthogonal projection seems to have a clear advantage which is further substantiated by the corresponding images in Fig. 5. We see no rippling artifacts in *QQ* and the error image is mostly black.

Timing results show that all the OP filters perform similarly as gradients are all precomputed. For the Taylor series filter, which are used to compute normals on the fly, runtime increases with increasing filter order (OF). Since most of the time in ISR mode is spent evaluating the analytic functions, the total runtimes of all the experiments

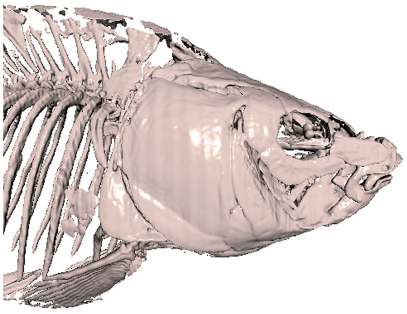

Fig. 6. An isosurface of the high-resolution carp fish data set.

are fairly close to each other. With compiler optimizations turned on, we observed that quintic box spline evaluation is fairly similar, if not marginally faster sometimes, to that of tricubic B-Spline on Intel Core 2 Duo. But on AMD Opteron, with the same compiler optimizations, we noticed that quintic box spline evaluation is usually marginally slower than that of tricubic B-Spline. On the other hand, with no compiler optimizations, we observed that this fact is quite the opposite and BCC performs twice faster than CC on both platforms. However, we do not report times from unoptimized codes in this paper.

6.3 Real Data

To assess the practical impact of our filters on the visualization of volumetric data, we rendered isosurface images of the carp and bunny data sets in ISR mode. The original CC data sets have grid sizes of $512 \times 512 \times 512$ and $512 \times 512 \times 361$, respectively. Fig. 6 depicts the original high-resolution carp's skull reconstructed with prefiltered tricubic B-spline interpolation and shaded using the OP filter *cc* in conjunction with the tricubic B-spline.

The first row of Fig. 7 shows the carp's skull reconstructed using prefiltered tricubic B-spline scalar interpolation on a downsampled CC grid and shaded using four different gradient estimation schemes. The second row analogously shows the results for a downsampled BCC grid. The images follow the same trend as that in Fig. 4. With the second-order filters (2-*cd* and *BCD*), the image appears smoothed out and sharp features are largely absent. It becomes progressively better with the fourth order filters (4-*cd* and *OPT16*) and ϵ-central differencing. The rippling artifacts that we observed in the case ϵ-CD earlier can be seen here as well. As before, the OP filters (*qc* and *NQ*) reveal the lost features by enhancing high-frequency details.

In a typical ISR setting, the majority of the time is spent performing scalar interpolations and this fact is reaffirmed in the timings of Fig. 7. Note that the gradient estimation time for ISR is very small compared to the overall time and is therefore susceptible to time measurement noises.

We also rendered DVR images, Fig. 8, of the carp data-set to study the visual artifacts and the runtime of different gradient estimation techniques combined with different scalar interpolations on CC and BCC. For these images, we used the same data set as Fig. 7. The zoomed-in insets of Fig. 8 clearly show that BCC renditions are visually superior in terms of scalar interpolation as the bones were reconstructed better. Normals estimated using the OP framework, in both CC and BCC, enhance details compared to the Taylor series filters. However, *OPT16* in BCC produces better contrast compared to the CC counterpart, *4-cd*.

The time measurements in Fig. 8 show that gradient estimation using the OP framework is the fastest, which is of no surprise as the gradients are precomputed. Also, gradient estimation with *OPT16*, for example, is slower than that of *4-cd* by a factor of about 1.04. This agrees with our analysis in Section 5, where we have argued that the discrete convolution

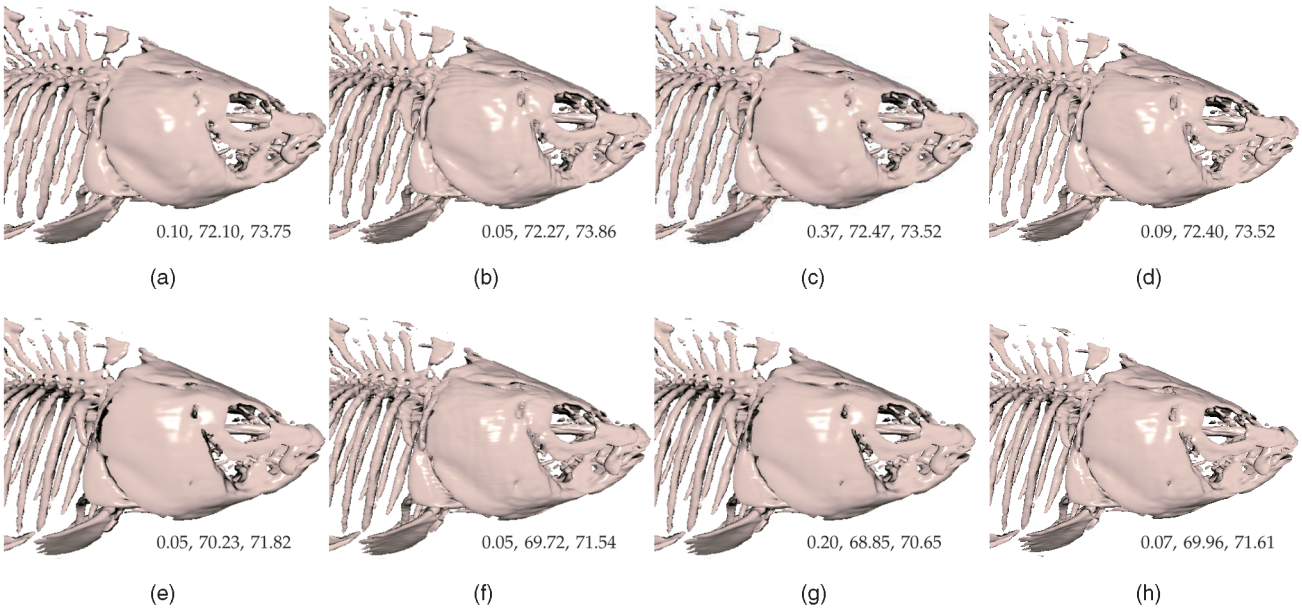

Fig. 7. Carp data set downsampled to a $160 \times 160 \times 160$ CC grid (a-d) and a $126 \times 126 \times 252$ BCC grid (e-h) and prefiltered appropriately for interpolation filters on the respective grids. Isosurface reconstructed and shaded using tricubic B-spline interpolation on CC and quintic box spline interpolation on BCC. The timing data (in seconds) indicate the normal computation time, the scalar interpolation time, and the total render time, respectively. All images were rendered at a resolution of 512×512. (a) 2-*cd*. (b) ϵ-*cd*. (c) 4-*cd*. (d) *qc*. (e) *BCD*. (f) ϵ-CD. (g) *OPT16*. (h) *NQ*.

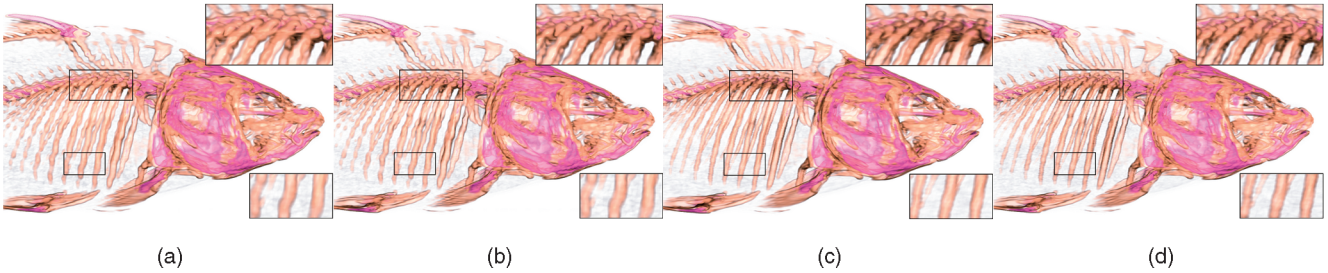

Fig. 8. DVR images of the downsampled carp CC (a,b) and BCC (c,d) data sets (prefiltered for the respective interpolation filters) rendered at a resolution of 600×390 pixels. Normal computation time, scalar interpolation time, and the total render time are indicated. (a) 4-*cd* (56.35, 69.89, 129.00). (b) *qc* (35.44, 69.68, 108.33). (c) *OPT16* (58.75, 68.38, 130.22). (d) *NQ* (29.15, 69.67, 101.55).

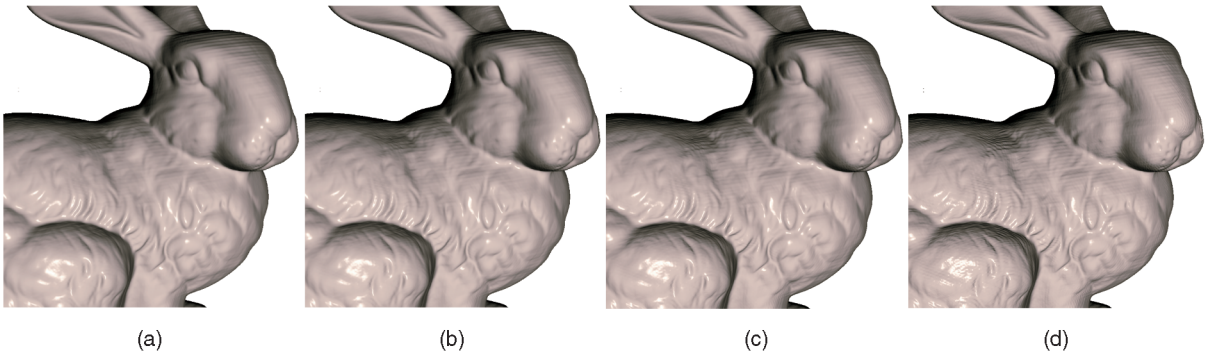

Fig. 9. An isosurface of the high-resolution bunny data set. Trilinear interpolation is used for both the scalar data and the gradient. (a) *2-cd*. (b) *4-cd*. (c) *ll*. (d) *ql*.

of *OPT16*, if not similar, should be marginally slower than *4-cd* despite the former having a much larger support.

Finally, Fig. 9 illustrates the result of combining the normal estimation schemes with trilinear interpolation. We used the high-resolution CC bunny data set for this purpose. As before, in comparison to *2-cd*, *4-cd* enhances the details slightly especially in the high-frequency regions. These details are enhanced even further with the OP filters *ll* and *ql*. Subtle features on the surface of the bunny, which are smoothed out in the *2-cd* rendition, are much more clearly visible. At the same time, however, ringing artifacts due to an imperfect CT reconstruction are also appreciably enhanced. This suggests that the higher order OP filters enhance high-frequency details and should therefore be used with caution in the presence of noise.

7 Conclusion

In this paper, we have presented two gradient estimation methods to extend the state of the art. We believe that these methods have a broad range of applicability as they give the practitioner the flexibility to design filters to suit their needs. We extended a 1D Taylor series framework to multiple dimensions and used it to design compact filters that can be computed on the fly with little runtime overhead. We also considered the idea of prefilters and derived high-quality filters using tensor product B-splines on CC and box splines on BCC. Our results show that, when accuracy and quality are crucial, a filter based on the Hilbert space framework can be employed with some storage overhead to appreciably improve image quality. Additionally, our methods can easily be extended to design filters that compute higher order derivatives on arbitrary sampling lattices.

In future, we plan to extend the Taylor series framework to design compact stencils for the numerical solution to partial differential equations. We also plan to investigate the error behavior of the orthogonal projection scheme in the Fourier domain in terms of a frequency error kernel as proposed by Blu and Unser [2]. Their formulation has the advantage that it is applicable to a class of functions that is much richer than the class of bandlimited functions. It also enables us to incorporate smoothness constraints so that the L_2 approximation error is guaranteed for functions that are sufficiently regular. Furthermore, the error kernel can be used to design suboptimal projection schemes that are not only computationally more efficient but also achieve the same rate of decay of error as the orthogonal projection scheme.

Acknowledgments

The carp and bunny data sets are courtesy of the Volume Library, http://www9.informatik.uni-erlangen.de/External/vollib/. The authors would like to thank Dr. Alireza Entezari for providing the BCC interpolation code as well as Dr. Dimitri Van De Ville and Dr. Laurent Condat for many fruitful discussions. This work has been funded in part by the Natural Science and Engineering Research Council of Canada.

References

[1] U.R. Alim and T. Möller, "A Fast Fourier Transform with Rectangular Output on the BCC and FCC Lattices," *Proc. Eighth Int'l Conf. Sampling Theory and Applications (SampTA '09)*, May 2009.

[2] T. Blu and M. Unser, "Quantitative Fourier Analysis of Approximation Techniques: Part I—Interpolators and Projectors," *IEEE Trans. Signal Processing*, vol. 47, no. 10, pp. 2783-2795, Oct. 1999.

[3] B. Csebfalvi, "An Evaluation of Prefiltered Reconstruction Schemes for Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 2, pp. 289-301, Mar./Apr. 2008.
[4] B. Csébfalvi and B. Domonkos, "Pass-Band Optimal Reconstruction on the Body-Centered Cubic Lattice," *Proc. Conf. Vision, Modeling, and Visualization 2008,* p. 71, Oct. 2008.
[5] B. Csébfalvi and B. Domonkos, "Prefiltered Gradient Reconstruction for Volume Rendering," *J. WSCG,* vol. 17, nos. 1-3, pp. 49-56, 2009.
[6] C. de Boor, K. Höllig, and S. Riemenschneider, *Box Splines.* Springer Verlag, 1993.
[7] D. den Hertog, R. Brekelmans, L. Driessen, and H. Hamers, "Gradient Estimation Schemes for Noisy Functions," technical report, 2003.
[8] D.E. Dudgeon and R.M. Mersereau, *Multidimensional Digital Signal Processing,* first ed. Prentice-Hall, Inc., 1984.
[9] S.C. Dutta Roy and B. Kumar, "Digital Differentiators," *Handbook of Statistics,* vol. 10, pp. 159-205, Elsevier Science Publishers B.V., 1993.
[10] A. Entezari, R. Dyer, and T. Möller, "Linear and Cubic Box Splines for the Body Centered Cubic Lattice," *Proc. IEEE Conf. Visualization,* pp. 11-18, Oct. 2004.
[11] A. Entezari, M. Mirzargar, and L. Kalantari, "Quasi Interpolation on the Body Centered Cubic Lattice," *Computer Graphics Forum,* vol. 28, pp. 1015-1022, 2009.
[12] A. Entezari, D. Van De Ville, and T. Möller, "Practical Box Splines for Reconstruction on the Body Centered Cubic Lattice," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 2, pp. 313-328, Mar./Apr. 2008.
[13] H. Farid and E. Simoncelli, "Differentiation of Discrete Multi-Dimensional Signals," *IEEE Trans. Image Processing,* vol. 13, no. 4, pp. 496-508, Apr. 2004.
[14] B. Finkbeiner, U.R. Alim, D.V.D. Ville, and T. Möller, "High-Quality Volumetric Reconstruction on Optimal Lattices for Computed Tomography," *Computer Graphics Forum,* vol. 28, no. 3, pp. 1023-1030, 2009.
[15] H. Hamers, R. Brekelmans, L. Driessen, and D. den Hertog, "Gradient Estimation Using Lagrange Interpolation Polynomials," technical report, 2003.
[16] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. IEEE Symp. Volume Visualization,* pp. 79-86, 1998.
[17] G. Kindlmann, X. Tricoche, and C.-F. Westin, "Anisotropy Creases Delineate White Matter Structure in Diffusion Tensor MRI," *Proc. Ninth Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI '06),* pp. 126-133, Oct. 2006.
[18] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller, "Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications," *Proc. IEEE Conf. Visualization,* pp. 513-520, Oct. 2003.
[19] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *Proc. IEEE Conf. Visualization,* pp. 100-107, Oct. 1994.
[20] T. Meng, B. Smith, A. Entezari, A.E. Kirkpatrick, D. Weiskopf, L. Kalantari, and T. Möller, "On Visual Quality of Optimal 3D Sampling and Reconstruction," *Proc. Conf. Graphics Interface,* pp. 265-272, May 2007.
[21] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "A Comparison of Normal Estimation Schemes," *Proc. IEEE Conf. Visualization,* pp. 19-26, Oct. 1997.
[22] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "Evaluation and Design of Filters Using a Taylor Series Expansion," *IEEE Trans. Visualization and Computer Graphics,* vol. 3, no. 2, pp. 184-199, Apr.-June 1997.
[23] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel, "Design of Accurate and Smooth Filters for Function and Derivative Reconstruction," *Proc. Symp. Volume Visualization,* pp. 143-151, Oct. 1998.
[24] N. Neophytou and K. Mueller, "Space-Time Points: 4D Splatting on Efficient Grids," *Proc. 2002 IEEE Symp. Volume Visualization and Graphics (VVS '02),* pp. 97-106, 2002.
[25] G. Strang and G.J. Fix, "A Fourier Analysis of the Finite Element Variational Method," *Constructive Aspects of Functional Analysis,* pp. 796-830, 1971.
[26] G. Strang and G.J. Fix, *An Analysis of the Finite Element Method.* Prentice Hall, 1973.
[27] H. Sun, N. Kang, J. Zhang, and E.S. Carlson, "A Fourth-Order Compact Difference Scheme on Face Centered Cubic Grids with Multigrid Method for Solving 2D Convection Diffusion Equation," *Math. and Computers in Simulation,* vol. 63, no. 6, pp. 651-661, 2003.
[28] T. Theußl, T. Möller, and E. Gröller, "Optimal Regular Volume Sampling," *Proc. IEEE Conf. Visualization 2001,* pp. 91-98, Oct. 2001.
[29] M. Unser, "A General Hilbert Space Framework for the Discretization of Continuous Signal Processing Operators," *Proc. SPIE Conf. Math. Imaging: Wavelet Applications in Signal and Image Processing III,* pp. 51-61, Part I, July 1995.
[30] M. Unser, "Sampling-50 Years after Shannon," *Proc. IEEE,* vol. 88, no. 4, pp. 569-587, 2000.
[31] M. Unser, A. Aldroubi, and M. Eden, "B-Spline Signal Processing: Part I—Theory," *IEEE Trans. Signal Processing,* vol. 41, no. 2, pp. 821-833, Feb. 1993.

Zahid Hossain received the BSc degree in computing science with a minor in telecommunication from North South University, Bangladesh, in 2006. He is currently working toward the MSc degree at the School of Computing Science, Simon Fraser University, Canada. His research interests include approximation and application of signal processing techniques in computer graphics and visualization. He is also interested in illumination and GPU methods.

Usman R. Alim received the BS and BA degrees in physics and mathematics, respectively, from the University of Rochester, and the MS degree in computer science from Rochester Institute of Technology. He is currently working toward the PhD degree at the School of Computing Science at Simon Fraser University. His research interests span the fields of computer graphics and visualization and include global illumination, physically based modeling, and the applications of optimal sampling lattices.

Torsten Möller received the vordiplom (BSc) degree in mathematical computer science from Humboldt University of Berlin, Germany, and the PhD degree in computer and information science from the Ohio State University in 1999. He is an associate professor at the School of Computing Science at Simon Fraser University. His research interests include the fields of visualization and computer graphics, especially the mathematical foundations thereof. He is the director of Vivarium, codirector of the Graphics, Usability and Visualization Lab (GrUVi), and serves on the Board of Advisors for the Centre for Scientific Computing at Simon Fraser University. He is the appointed vice chair for publications of the IEEE Visualization and Graphics Technical Committee (VGTC). He has served on a number of program committees (including the Eurographics and the IEEE Visualization conferences) and has been papers cochair for the IEEE Visualization, EuroVis, Graphics Interface, and the Workshop on Volume Graphics as well as the Visualization track of the 2007 International Symposium on Visual Computing. He has also co-organized the 2004 Workshop on Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration at the Banff International Research Station, Canada. He is currently serving on the steering committee of the Symposium on Volume Graphics. Further, he is an associate editor for the *IEEE Transactions on Visualization and Computer Graphics* (TVCG) as well as the *Computer Graphics Forum*. He is a member of the IEEE, the IEEE Computer Society, the ACM, the Eurographics, and the Canadian Information Processing Society (CIPS).