EDGE AWARE ANISOTROPIC DIFFUSION FOR 3D SCALAR DATA ON REGULAR LATTICES

by

Zahid Hossain
BSc. Honours, North South University 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

© Zahid Hossain 2011
SIMON FRASER UNIVERSITY
Spring 2011

All rights reserved. However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for Fair Dealing. Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

APPROVAL

Name: Zahid Hossain

Degree: Master of Science

Title of thesis: Edge Aware Anisotropic Diffusion for 3D Scalar Data on Regular Lattices

Examining Committee: Dr. Greg Mori
Chair

Dr. Torsten Möller,
Senior Supervisor
Associate Professor, Computing Science

Dr. Steven J. Ruuth,
Supervisor Professor, Applied and Computational Mathematics

Dr. Hao-Min Zhou,
Examiner
Associate Professor, School of Mathematics
Georgia Institute of Technology, USA

Date Approved: _____

Abstract

We present a novel anisotropic diffusion model targeted for 3D scalar field data. Our model preserves material boundaries as well as fine tubular structures while noise is smoothed out. One of the major novelties is the use of the directional second derivative to define material boundaries instead of the gradient magnitude for thresholding. This results in a diffusion model that has much lower sensitivity to the diffusion parameter and smoothes material boundaries consistently compared to gradient magnitude based techniques. We analyze the stability and convergence of the proposed diffusion and demonstrate its de-noising capabilities for both analytic and real data. We also discuss applications in the context of volume rendering.

We extend our algorithm to non-Cartesian lattices such as Body Centric Cubic (BCC). The key to such an extension is a method to estimate derivatives reliably. Therefore, we present a general framework to estimate derivatives on arbitrary regular lattices. With this framework a user can design filters with compact support and specify a polynomial order of accuracy.

Keywords: anisotropic diffusion; PDE; de-noising; scale-space; principal curvatures; Taylor series expansion; derivative estimation; Body-Centred Cubic (BCC) lattice;

Subject terms: 3D image processing; level-set method; visualization; computer graphics; de-noising

To my parents

"The more I learn, the more I learn how little I know"

— *Unknown*

Acknowledgments

It was a breathtaking journey through my masters degree at Simon Fraser University and for every single bit of my stay there I would like to thank my senior supervisor Dr. Torsten Möller. It was Dr. Möller who introduced me to the intriguing world of non-standard lattices and asked me to solve one of the open problems of derivative estimation in such lattices. Since then, not only got I curious about these lattices but also faced massive challenges to learn the intense mathematics involved therein. However, his patience with me in the beginning and relentless academic and moral support by simply saying "you can do it, you are good" never let me give up even in the most despairing moments. While learning more about these lattices, Dr. Möller introduced me to one of his PhD students, Usman Raza Alim. Usman was a true mentor who would be working right next to me both literally and metaphorically all the time. He was one of my "go to" persons for almost everything during my entire masters degree. Had it been not for Usman, my first paper, where Usman was a co-author, would not have been polished and rigorous.

In the second year of my masters degree I took a course with Dr. Steven Ruuth. Beside being a wonderful researcher as we know, Dr. Ruuth is a brilliant teacher as I learned so much about numerical partial differential equations that my second paper on anisotropic diffusion spawned from this class. Soon after that, I was lucky to have Dr. Ruuth himself as my supervisor who gave me lots of interesting insights about analyzing stabilities of PDEs.

I would also like to thank Steven Bergner and Bernard Finkbeiner of the Graphics, Usability, and Visualization (GrUVi) lab for helping me out in several occasions with research and deep insights. Finally, I would also like to thank Dr. Alireza Entezari, a GrUVi alumni, for his invaluable suggestions from time to time. My two and half years of stay at GrUVi was made exceptionally special by all the great people working there.

This thesis is based on the following two papers

- Zahid Hossain and Torsten Möller, "Edge Aware Anisotropic Diffusion for 3D Scalar Data," *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010)*, vol. 16, no. 6, pp. 1375–1384, Nov.-Dec. 2010.

- Zahid Hossain, Usman R. Alim and Torsten Möller, "Toward High-Quality Gradient Estimation on Regular Lattices," *IEEE Transactions on Visualization and Computer Graphics*, to appear, TVCG 16: 2011.

However, parts of the papers that I was not directly involved with were removed from this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The core of volume visualization and volume rendering has been the extraction and presentation of the salient features in the volume. Often times, the data at hand has been corrupted by noise (e.g. Ultrasound [56], MRI, or data range scanners [47]) or the salient features of interest are fine structures, like the tubular vessel structures [29] or the cell walls in microscopy [36]. Usually, these types of data can not be properly processed by a volume rendering pipeline, since both transfer function based approaches [48] as well as topological approaches [9] will break down and not yield a comprehensible view of the data. In all of these cases, a pre-processing step is needed in order to prepare the data for visualization and analysis purposes. A very powerful framework for this purpose is diffusion.

A simple Gaussian blur usually destroys a lot of features together with noise artifacts in an isotropic / indiscriminate way. Hence, the concept of anisotropic diffusion has been introduced by Perona and Malik in 1990 [40] (summarized in Section 2) and has become one of the most popular techniques in image and volume processing. Many different variants of anisotropic diffusion have since been introduced. One of the core drawbacks, however, of any diffusion model has been the non-intuitive setting of some parameters attached with it. This is typically rooted in the fact that the diffusion is controlled by the gradient magnitude of the underlying function. In most practical cases the distribution of gradient strength of salient boundaries is not known a priori. In fact a single gradient magnitude threshold rarely defines all the salient features within the data. Hence, the diffusion algorithm often needs to be re-fined for each new data set or each new application.

In this thesis we attempt to address these problems of the existing non-linear diffusion and propose a noble anisotropic diffusion. We also develop a generic framework to compute

various derivatives on arbitrary lattices so that our proposed anisotropic diffusion could be implemented on other lattices too. Our main contributions are:

- We derive an anisotropic diffusion equation with the following features (see Section 3.1.1):
  - No diffusion is performed along the gradient direction.
  - Diffusion is stopped around the edge locations.
  - Diffusion is performed anisotropically along the direction of the minimum curvature.
  - Isotropic diffusion on the tangent plane of the normal in regions where the local iso-surface is isotropic in shape.
- We create a stopping function, that is based on the second derivative in the gradient direction, which allows us to create a robust diffusion algorithm, insensitive to parameter tuning (see Section 3.1.1).
- We develop a generic framework for designing discrete derivative estimation filters for arbitrary lattices whereby users can specify design criteria like polynomial order. With this framework in hand one can implement our proposed anisotropic diffusion on any regular lattice.

In Chapter 3 we will introduce our novel anistropic diffusion and show an efficient way to compute the diffusion equation using the principal curvatures and thereby reducing the computational burden inherit in the scheme. In Section 3.3 we will discuss some properties of our proposed diffusion model along with aptly demonstrating its advantage over a gradient based diffusion method [29]. We will follow this by showing the de-noising property of our model in Section 3.4 and support its significance by critically comparing it with a very recent anisotropic diffusion based de-noising filter [28]. Before the comparison, however, we will analyze the stability of our diffusion both theoretically and empirically in Section 3.3.5. Finally, in Section 3.5 we will discuss some potential applications in volume visualization.

Volumetric data can be given in various kinds of discrete lattices other than the commonly used Cartesian Cubic (CC) lattice. Often times, therefore, its is desirable to apply image processing techniques like de-nosing and other PDE based methods directly on the given lattice without converting them to CC, which may introduce undesirable errors and may even be non-trivial. Therefore, in Chapter 4 we will develop a generic framework to

design discrete filters to compute various derivatives, which are required to implement our proposed anisotropic diffusion, on arbitrary lattices. Along this line we will introduce the Taylor Series approach towards designing filters along with other design criteria in Section 4.1. Finally, in Section 4.2, we will compare our results in light of gradient (first derivative) estimation across two different regular lattices namely Cartesian Cubic (CC) and Body Centric Cubic (BCC) lattices. We will also compare our framework with the existing gradient estimation technique on the non-standard BCC lattice which only uses filters from the CC lattice without considering the special geometry of the BCC.

Chapter 2

Previous Work

Throughout the thesis we will use the notation f and t to denote a real valued scalar function and the time dimension respectively.

2.1 Anisotropic Diffusion

2.1.1 The Perona and Malik Model

To alleviate the problem of isotropic diffusion, which is similar to Gaussian blurring, Perona and Malik [40] proposed an anisotropic diffusion scheme, which we will refer to as the PM model for brevity, given by the following:

$$\frac{\partial f}{\partial t} = \mathbf{div}\left(h(\|\nabla f\|)\nabla f\right) \tag{2.1}$$

The function $h(\|\nabla f\|)$, termed *stopping function*, is usually a monotonically decreasing function with function values around one for smaller arguments. Perona and Malik [40] also proposed two such h functions and one of them is:

$$h(\alpha) = e^{-\frac{1}{2}\left(\frac{\alpha}{k}\right)^2}, \tag{2.2}$$

With such an $h(\cdot)$ function this approach does tend to preserve certain edges given the parameter k in Equation (2.2) is chosen carefully which is often not trivial and data dependent.

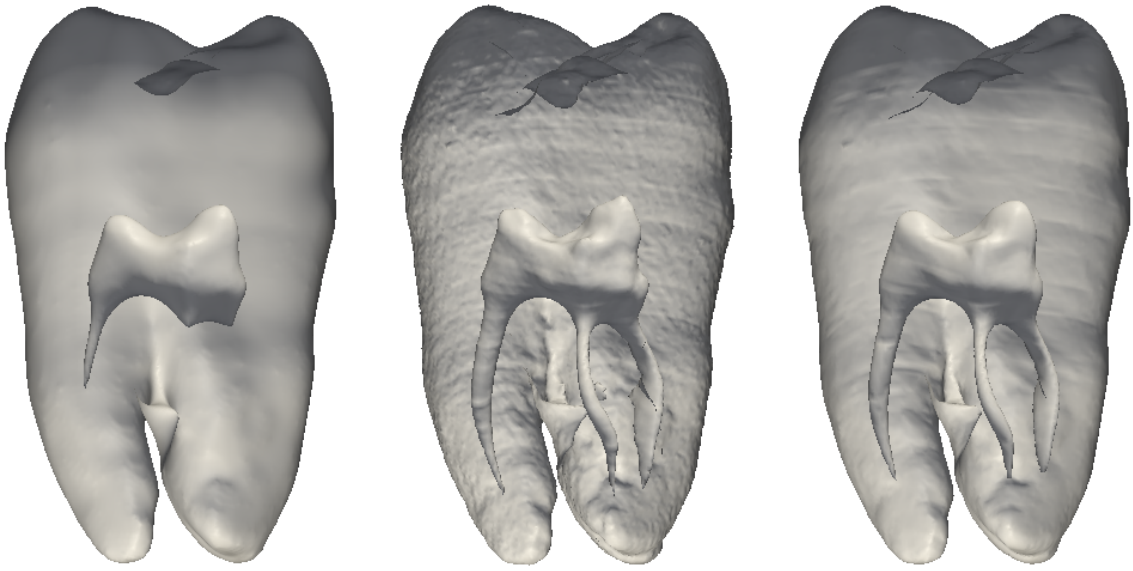
2.1.2 Generalizations in 2D

Carmona et al. [8] generalized the classical PM [40] model (in 2D) in a more intuitive way - performing diffusions along the gradient and the orthogonal direction to the gradient as given by the following:

$$\frac{\partial f}{\partial t} = c(\cdot)\big(a(\cdot) f_{\mathbf{nn}} + b(\cdot) f_{\mathbf{vv}}\big) \tag{2.3}$$

Here a and b are some scalar functions modulating diffusion along the gradient direction $\mathbf{n} = \nabla f / \|\nabla f\|$ and the orthogonal direction to the gradient, $\mathbf{v} = \mathbf{n}^{\perp}$ respectively. Here, c is a scalar function, usually called the *stopping function*, that modulates the overall diffusion. The notation $f_{\mathbf{nn}}$ and $f_{\mathbf{vv}}$ denotes the directional second derivative along the gradient direction $\mathbf{n}$ and the orthogonal direction $\mathbf{v}$ respectively.

2.1.3 Extensions to 3D



(a) Without curvatures (b) Original (c) With curvatures

Figure 2.1: Iso-surface (iso-value=100) rendering of the tooth data showing effects of taking principal curvatures into account during diffusion. (a) Diffused isotropically on the tangent plane of gradients without taking principal curvatures into account. (b) Without any diffusion. (c) Diffused anisotropically taking principal curvatures into account.

A straightforward extension of the PM model to 3D, as it was generalized by Gerig et

al. [18], remains isotropic on the tangent plane of the gradient. This isotropic behavior on the tangent plane may destroy fine tubular structures which are vital, for example, in 3D medical imaging. Figure 2.1 clearly shows how tubular structures get destroyed when diffusion is performed isotropically on the tangent plane without taking principal curvatures[1] into account. Therefore Weickert [54] classifies the PM model and higher order diffusion processes as *non-linear* rather than *anisotropic*. Thus far, the majority of previous work on these higher order PDEs are based on 2D solutions. On the other hand, a true anisotropic diffusion in arbitrary dimension is usually derived from the *diffusion tensor* notation [55] and has the following form:

$$\frac{\partial f}{\partial t} = \mathbf{div}(\mathbf{D}\nabla f) \tag{2.4}$$

were $\mathbf{D}$ is a matrix known as *diffusion tensor*.

To address this, Krissian et al. [29] proposed a true anisotropic diffusion model (referred to as KM model in the remainder of the thesis) whereby diffusion would be performed primarily along the direction of the minimum curvature. However, their underlying formulation was based on that of the gradient based PM model. They have a gradient threshold parameter k and the edges in a volume get implicitly defined by locations where $\|\nabla f\| > k$. However, the tuning of this parameter is difficult and very much data and application dependent. Different values for this parameter can lead to drastically different smoothing effects as we will demonstrate later. Therefore, the KM model inherits similar problems related to this threshold parameter as the classical PM model. Nonetheless, the KM model based on principal curvatures is truly anisotropic in nature. In a later work, Krissian [25] proposed a flux-based anisotropic diffusion which is based on a directional first derivative, i.e. the gradient measured along a direction vector, while the author himself acknowledged the difficulty of choosing a correct threshold parameter for this directional first derivative.

Recently Mosaliganti et al. [36] reaffirm the problems of the gradient threshold based *stopping function* as found in PM or KM models and proposed a new anisotropic diffusion in 3D that is able to automatically detect and enhance specifically plate like structures in a 3D microscopy image of cell membranes. Beside being specific to a particular problem domain, i.e. detecting cell membranes which are largely planar, their method has at least four different user tunable parameters which makes it hard to apply in a practical setting.

[1] Principal curvatures, measured at a point, are the minimum and the maximum curvatures of the level set surface passing through that point.

Therefore, the problem of developing a robust general purpose anisotropic diffusion that respects edges in a 3D volume in a meaningful way remains open.

On the other hand, several interesting works have been done on anisotropic diffusion using level sets, for example Nemitz et al. [37] evolved a separate level set function to capture the tubular structures of 3D angiography data. This work is different in the sense that the level set function attempts to restore tubular structure using shape priors even when they may be broken. Other interesting level set methods were proposed by Preusser et al. [42] and Tasdizen et al. [47] where diffusion is performed on a level set and the definition of an *edge* is based on *curvature* that is measured on the surface of the level set. This is different from our method where an *edge* is defined by the directional second derivative along a gradient and is measured across level sets.

2.2 De-noising

A variant of anisotropic diffusion, also known as SRAD, has been developed to specifically de-noise speckle noise in 2D by Yu and Action [57], which was then extended to 3D by Krissian et al. [26] and Sun et al. [46]. Both SRAD and 3D SRAD use a local statistical measure to define the *stopping function*. Very recently Krissian and Aja-Fernández [28] proposed a noise-driven anisotropic diffusion that is able to remove *Rician* noise from a 3D MRI volume, and this method too uses statistical measures similar to that of the 3D SRAD [26]. Both of these methods require the user to specify a *region of interest* for the estimation of noise.

State-of-the art image de-noising techniques are often based on techniques such as *bilateral filtering* [50], *mean-shift* filtering [11], or *non-local means* [7]. These techniques are often related to diffusion processes. Barash et al. [5] showed that bilateral filtering, mean-shift, and *non-linear diffusion* are indeed equivalent and use the gradient magnitude to decide on the amount of diffusion/smoothing.

2.3 Derivative Estimation

There is a vast body of work on function interpolation and reconstruction. There are really two philosophies - a) improving the numerical accuracy based on Taylor series expansions and b) considering shift-invariant function spaces. 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 [44] were the first to try to reconcile these two viewpoints. Later, Unser [51] 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. Towards this end, Möller et al. [34] 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 [35], they contrast two possible approaches to gradient estimation - using a combination of a discrete derivative filter with a continuous interpolation filter or simply computing the derivative of the interpolation filter. While in the former 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, but is typically more expensive.

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. Theoretically, BCC is the optimal sampling lattice for band limited functions as its dual in the frequency domain happens to be the Face Centric Cubic (FCC) lattice, which has the tightest packing ratio. From a signal processing perspective; a tighter pack in the frequency domain corresponds to sparser sampling in the spatial domain and therefore less number of samples are theoretically required to recover the original function. This theoretical fact of the BCC lattice being optimal for sampling over CC lattice has been shown by Theußl et al. [49] and Neuphytou et al. [38] using volume splatting. Later Entezari et al. [16] developed box splines for scalar interpolation on BCC lattices and showed how both numerical and run-time performances outperformed the CC lattice, which is remarkable. On the other hand, this superiority of the BCC over CC was also observed by Alim et al. [3] in a totally different context of fluid simulation. However, it is not clear whether the advantages of the BCC lattice extend to derivative reconstruction as well. This thesis addresses this issue.

Hamers et al. [20] 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 4.1). Sun et al. [45] develop

a fourth order gradient estimation scheme for the hexagonal lattice in 2D only. Hertog et al. [13] compare different discrete derivative filters in the presence of noise.

One track of research has focused on designing digital first derivative (gradient) filters in the Fourier domain. These techniques inherently assume an underlying band-limited signal. Most methods have focused on designing filters in 1D where gradient reconstruction corresponds to a multiplication with a unit slope ramp in the frequency domain. The ideal discrete gradient 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 - like numerical non-linear PDEs - 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 et al. [14] 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 et al. [17] 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 non-separable derivative filters for arbitrary sampling lattices in higher dimensions.

Chapter 3

Edge Aware Anisotropic Diffusion

3.1 Proposed Anisotropic Diffusion

Kindlmann and Durkin [22] used the directional second derivative along the normal direction as a measure for edge locations. They pointed out that, for a simple 1D case, an edge could be modeled using the *error function* [24] as plotted in Figure 3.1.

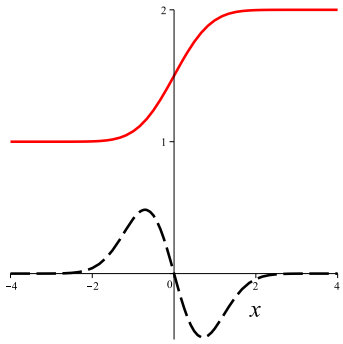


Figure 3.1: The red solid curve is the *error function*, scaled and shifted for clarity, while the dashed black curve is the second derivative of it. Note the second derivative crosses zero at the edge of the *error function*.

This is a fair assumption as often measuring devices are band-limited and so sharp edges get convolved with a Gaussian type point spread function upon sampling. Therefore, an edge location can be defined by the zero-crossing of the second derivative, a technique commonly used in computer vision [30]. The same idea can be applied in 3D by measuring

the directional second derivative along the normal direction and checking for zero-crossings to define the edge/boundary locations.

For the rest of the thesis we will restrict our attention to a 3D scalar function $f : \mathbb{R}^3 \to \mathbb{R}$. We will use the notation $f_{\mathbf{v}}$ to denote a directional first derivative along a unit vector $\mathbf{v}$ which is simply given by $f_{\mathbf{v}} = \langle \nabla f, \mathbf{v} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. Similarly, we will use the notation $f_{\mathbf{vv}}$ to denote a directional second derivative measured along a unit vector $\mathbf{v}$ and this is given by $f_{\mathbf{vv}} = \mathbf{v}^\top \mathbf{H} \mathbf{v}$, where $\mathbf{H}$ is the 3D Hessian (see Appendix A.1 for details). Therefore, using the notation $\mathbf{n} = \nabla f / \|\nabla f\|$ as the normal vector we will denote the directional second derivative along the normal with $f_{\mathbf{nn}}$.

In the following subsection we will derive a PDE with the following objectives in mind:

O-1 No diffusion will be performed along the gradient direction. This is one of the major differences our proposed diffusion model has with that of the classical ones [29, 40]. An edge in a 3D volume will be a surface which is perpendicular to the normal $\mathbf{n}$. Not diffusing along $\mathbf{n}$ prevents blurring across an edge.

O-2 Diffusion will be stopped around the edge locations. Diffusion can be stopped in locations where $f_{\mathbf{vv}} = 0$, a condition which will be satisfied in both constant homogeneous regions and edge locations. However, stopping diffusion in constant homogeneous regions creates no problem as diffusion in such regions would not have any effect.

O-3 Diffusion will be performed anisotropically along the direction of the minimum curvature. In accordance to the work of Krissian et al. [29], this motivation was derived from the fact that fine tubular structures, e.g. blood vessels in a CT scan, get preserved.

O-4 Diffuse isotropically on the tangent plane of the normal n in regions where the local iso-surface has similar principal curvatures. On the surface of a sphere, for example, where both the principal curvatures are fairly close to each other, it makes more sense to diffuse isotropically on the tangent plane of $\mathbf{n}$, rather than choosing one direction, which might lead to undesirable artifacts, as is the case with Krissian et al. [29].

3.1.1 Our PDE Model

Consider a simple 1D heat equation as follows:

$$\frac{\partial f}{\partial t} = c f_{\mathbf{xx}} \tag{3.1}$$

The solution of the above Equation (3.1) can be approximated very well by convolving the function locally with a 1D Gaussian kernel. We will use this insight to design our new anisotropic diffusion in 3D with the goals described in the previous subsection in mind.

In 3D, we can use the directional derivatives along the directions given by three orthonormal basis vectors, $\mathbf{r}_1, \mathbf{r}_2$, and $\mathbf{n}$ at a point and write our diffusion equation as the following:

$$\frac{\partial f}{\partial t} = h(\cdot) f_{\mathbf{r}_1\mathbf{r}_1} + g(\cdot) f_{\mathbf{r}_2\mathbf{r}_2} + w(\cdot) f_{\mathbf{nn}} \tag{3.2}$$

where $h(\cdot), g(\cdot)$, and $w(\cdot)$ are some scalar functions and the vector $\mathbf{n}$ is the normal direction. We emphasize that the PDE model given in Equation (3.2) is different from the diffusion tensor model (2.4). At this point we will take the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ to be the directions associated with the minimum curvature, κ_1, and maximum curvature, κ_2, respectively such that $|\kappa_1| \leq |\kappa_2|$. By definition, the vectors $\mathbf{r}_1, \mathbf{r}_2,$ and $\mathbf{n}$ form an orthonormal bases and thus fit our proposed diffusion model. We will also set the scalar functions such that $g(\cdot) = \tau h(\cdot)$ and $w(\cdot) = \eta h(\cdot)$ where $\tau, \eta \in [0, 1]$. With this setup, Equation (3.2) can be re-written:

$$\frac{\partial f}{\partial t} = h(\cdot)\left(f_{\mathbf{r}_1\mathbf{r}_1} + \tau f_{\mathbf{r}_2\mathbf{r}_2} + \eta f_{\mathbf{nn}}\right) \tag{3.3}$$

Without referring to the exact argument of the scalar function $h(\cdot)$ yet, Equation (3.3) models an anisotropic diffusion which can be intuitively thought of as the summation of the local convolutions of three different 1D Gaussian kernels oriented along the three associated vector fields (compare each term of the diffusion with Equation (3.1)). The amount of diffusion along the maximum curvature direction $\mathbf{r}_2$, and the normal direction $\mathbf{n}$ is modulated by τ and η respectively, while diffusion is always performed along the minimum curvature direction $\mathbf{r}_1$ and finally the overall diffusion is modulated by the scalar function $h(\cdot)$, which we will call the *stopping function*.

We will set $\eta = 0$ for the rest of the thesis to achieve objective **O-1**. To fulfill objectives **O-3** and **O-4** together we will replace the notation τ with τ_ρ which is defined by the

following:

$$\tau_\rho = \begin{cases} \left(\frac{\kappa_{1,\rho}}{\kappa_{2,\rho}}\right)^{2\lambda} & \text{where } |\kappa_{2,\rho}| > 0, \lambda \in \mathbb{N} \\ 1 & \kappa_{2,\rho} = 0 \end{cases} \tag{3.4}$$

where the quantities, $\kappa_{1,\rho}$ (the minimum curvature) and $\kappa_{2,\rho}$ (the maximum curvature), are computed from a smoothed version of the data f_ρ with a Gaussian filter having a small variance ρ^2. The technique of taking measurements from a smoothed volume f_ρ is common in many PDE based methods specially under noisy conditions and we will discuss the effect of having a small ρ in section Section 3.4. We will use the notation $\rho = 0$ to imply that f was used to compute the curvatures instead of the smoothed version f_ρ. With the above definition, τ_ρ drops quickly to values very close to 0 whenever the maximum curvature $|\kappa_{2,\rho}|$ is larger than the minimum curvature $|\kappa_{1,\rho}|$, i.e. the surface is not isotropic. In this case, diffusion is performed primarily along the minimum curvature direction $\mathbf{r}_1$. For an isotropic surface where $|\kappa_{1,\rho}| = |\kappa_{2,\rho}|$ and a degenerate case, where $|\kappa_{2,\rho}| = 0$ (note that $|\kappa_{1,\rho}| \leq |\kappa_{2,\rho}|$), τ_ρ gets assigned to 1 which amounts to performing simple isotropic diffusion on the tangent plane of $\mathbf{n}$. The exponent of Equation (3.4) is always an even integer which makes sure that we are comparing only the absolute values of the curvatures.

Objective **O-2** can be addressed by computing the second derivative along the gradient direction, $f_{\mathbf{nn}}$ and *stop* diffusion whenever $f_{\mathbf{nn}} \approx 0$. To model this we can define the function $h(\cdot)$ such that it is continuous and approaches 0 for an argument close to 0 and 1 otherwise. For this we can simply modify the functions proposed by PM [40] as follows:

$$h(\alpha) = 1 - e^{-\ln\left(\frac{10}{9}\right)\left(\frac{\alpha}{\sigma}\right)^2} = 1 - (0.9)^{\left(\frac{\alpha}{\sigma}\right)^2}, \quad \sigma \in \mathbb{R} \tag{3.5}$$

The scaling factor of $\ln(10/9)$ (Equation (3.5)) is there so that we have $h(\alpha) < 0.1$, which we considered to be very little diffusion, whenever $|\alpha| < \sigma$. This allows a more intuitive relationship between the argument α and the parameter σ. However, for a different purpose, this scaling factor could be changed or just simply be omitted. Using $f_{\mathbf{nn}}$ as the input to $h(\cdot)$ we essentially fulfill all four objectives we had set for ourselves. We finally present our anisotropic diffusion PDE by the following equation:

$$\frac{\partial f}{\partial t} = h(f_{\mathbf{nn}})(f_{\mathbf{r}_1\mathbf{r}_1} + \tau_\rho f_{\mathbf{r}_2\mathbf{r}_2}) \tag{3.6}$$

3.2 Implementation

Krissian et al. [29] have shown that we could skip computing the principal curvature directions, which usually involves expensive eigenvalue decomposition of some matrix [23, 52], altogether and compute the quantities $f_{\mathbf{r}_1\mathbf{r}_1}$ and $f_{\mathbf{r}_2\mathbf{r}_2}$ directly using the following relationships:

$$f_{\mathbf{r}_1\mathbf{r}_1} = -\|\nabla f\|\kappa_1, \qquad f_{\mathbf{r}_2\mathbf{r}_2} = -\|\nabla f\|\kappa_2 \tag{3.7}$$

In the above, κ_1 and κ_2 can be computed in a straightforward fashion by the following

$$\kappa_i = K \pm \sqrt{K^2 - G}, \quad i \in \{1, 2\} : |\kappa_1| \leq |\kappa_2| \tag{3.8}$$

where G and K are the Gaussian and Mean curvatures respectively. These can be computed directly from the first and second derivatives as given by Goldman [19]:

$$\begin{aligned} G &= \frac{1}{\|\nabla f\|^4} \nabla f^{\mathsf{T}} \mathbf{H}_c \nabla f \\ K &= \frac{1}{2\|\nabla f\|^3} \left(\nabla f^{\mathsf{T}} \mathbf{H} \nabla f - \|\nabla f\|^2 \operatorname{trace}(\mathbf{H}) \right) \end{aligned}$$

where $\mathbf{H}_c$ (see Appendix A.1 for details) is the co-factor matrix of the Hessian $\mathbf{H}$. Computation of both G and K can be implemented very efficiently without performing the actual matrix multiplications by expanding the equations as summations first (see Appendix A.2 for details). It should be noted that the formulae given by Goldman [19] are based on the convention that if a surface *turns* in the direction of the normals then it would be said to have a *positive* curvature. In simple terms; if a sphere is defined implicitly by a level-set of a function f, which has higher values inside the sphere while lower outside (i.e normals are pointing inside the sphere) then the curvature on the surface of the sphere will be *positive*.

Taking the relations given by (3.7), Equation (3.6) can be written more compactly and in matrix form as the following:

$$\frac{\partial f}{\partial t} = -h(\mathbf{n}^{\mathsf{T}} \mathbf{H} \mathbf{n}) \left(\|\nabla f\| (\kappa_1 + \tau_\rho \kappa_2) \right) \tag{3.9}$$

Note that κ_1 and κ_2 in the above equation are measured from f whereas τ_ρ is measured from f_ρ where ρ is usually 1 when diffusion is used for the purpose of de-noising (see Section 3.4) and 0 otherwise.

3.3 Results and Discussion

In this section we will first demonstrate the shortcomings of the classical anistropic diffusion, proposed by Krissian et al. [29] - referred to as KM model for brevity - which in turn was based on the gradient magnitude, i.e. the PM model [40], and compare the results with our novel approach. Next we will show the impact of the parameter σ, in Equation (3.5), of our method on the final output. We will then follow the discussion by showing diffusion results using higher order derivative filters and finally provide empirical analysis of stability and convergence of our PDE.

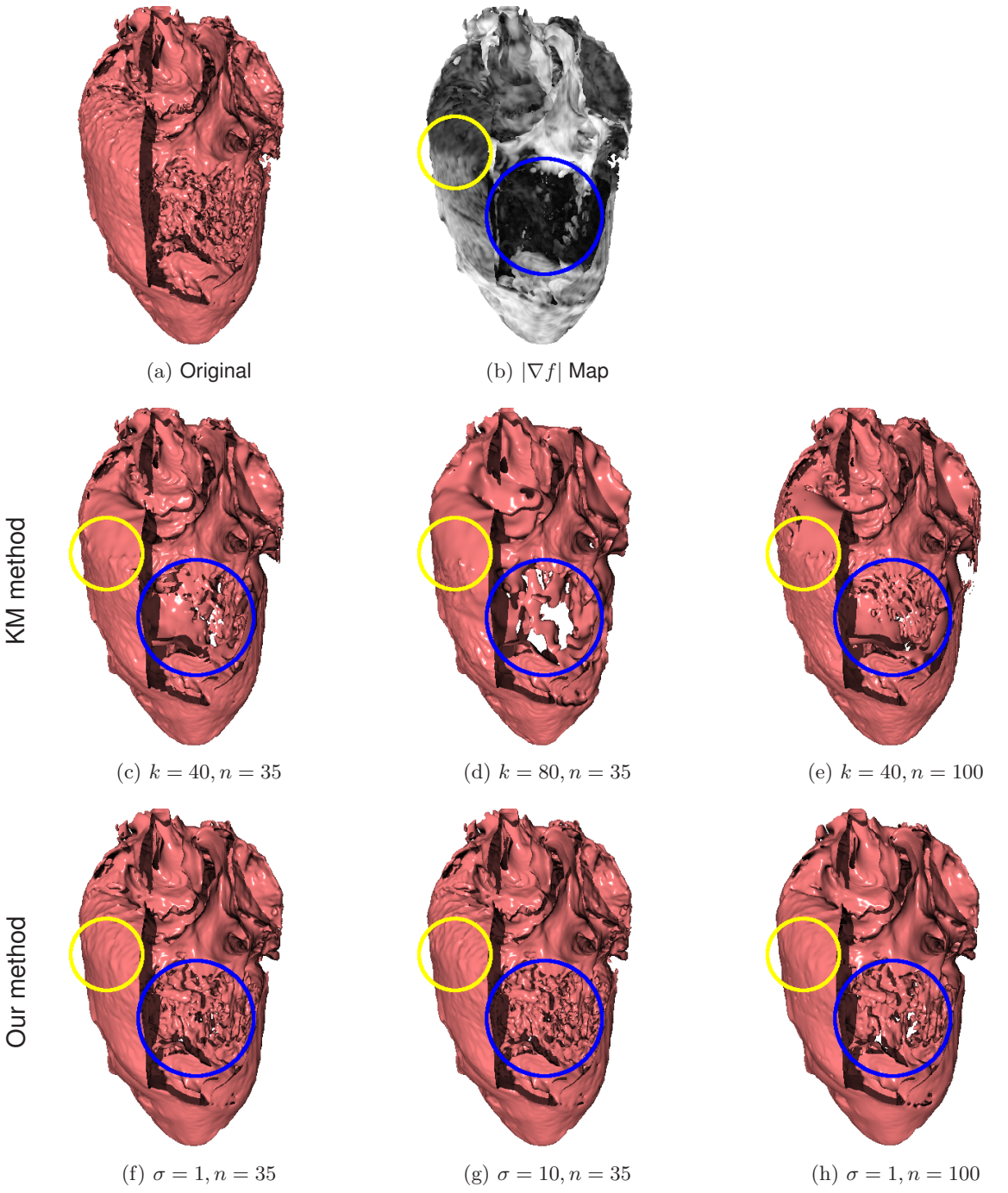
3.3.1 Parameter Settings

For all the diffusion experiments performed in this section we have set $\lambda = 2$ in Equation (3.4). Voxel spacing was assumed to be 1 in all directions and scalar values, which ranged between $[0, 255]$, and the 3D volumes were not scaled. For the time step we have chosen $\Delta t = 0.05$, and $\eta = 0$ (no diffusion along the gradient direction) for the rest of the thesis. We have used f without smoothing, i.e. $\rho = 0$ while we investigated different properties of our diffusion, as discussed in Subsections 3.3.2, 3.3.3, and 3.3.4. However, we have set $\rho = 1$ when we analyzed the stability and applied our diffusion in real world problems, as discussed in Section 3.3.5 onward.

For all derivative estimations we have used the standard second-order stencils unless specified otherwise.

3.3.2 The Impact of the Stopping Function

In this section we will use the Sheep's Heart dataset [43] and demonstrate the sensitivity of the KM method to the parameter chosen. Likewise, we will show the robustness of our novel method with respect to its parameter and yet give a compelling example of its importance. In our experiment, we chose Equation (2.2) and Equation (3.5) as the stopping functions in the KM method and in our new proposed method respectively, and use 35 iterations for both diffusion models. Figure 3.2 demonstrates the sensitivity of the KM method to its parameter k. The yellow circle marks an area where regions of small gradient magnitude merge with regions of higher gradient magnitude (see Figure 3.2b). Figure 3.2c, which was diffused with the KM model at $k = 40$ shows the selective nature of this method. Regions with a low gradient magnitude diffused much more compared to regions with higher gradient



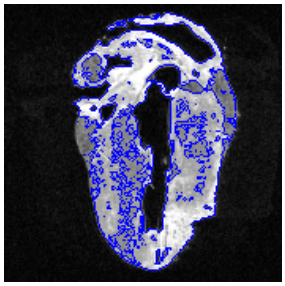
(a) Original (b) $|\nabla f|$ Map

(c) $k = 40, n = 35$ (d) $k = 80, n = 35$ (e) $k = 40, n = 100$

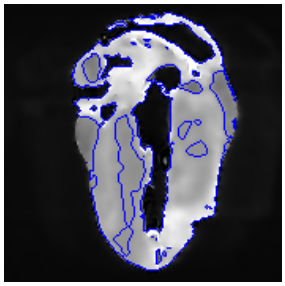
(f) $\sigma = 1, n = 35$ (g) $\sigma = 10, n = 35$ (h) $\sigma = 1, n = 100$

Figure 3.2: Iso-surface (iso-value=153) rendering of the Sheep's Heart dataset. Except for (b) all the other images were rendered with shadows. The image (b) is the gradient magnitude map where 0 is mapped to black while 130 is mapped to white. The blue circle shows a region of low gradient magnitude, while the yellow circle shows a region where medium and high gradient magnitude meet. The second and the third row show results from the KM method and our method respectively. The corresponding parameter values, k for the KM method and σ for our method along with the number of iterations n are provided for each experiment.

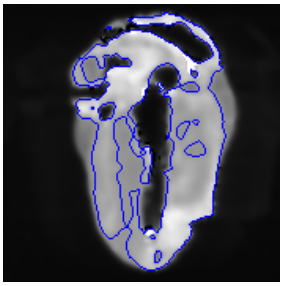
magnitude. On the other hand, regions having very low gradient magnitude (the blue circle) got diffused the most. As the value of k was increased to 80 (Figure 3.2d) a dramatically different output was produced. Here, regions inside the blue circle got diffused to the point of losing structure while those with higher gradient magnitude (the lower part of the yellow circle) just started to get diffused. We also refer readers to Figure 3.2e to see this selective nature of the KM method based on gradient magnitude and the resulting artifacts after 100 iterations were performed. A single iso-surface rendering may not tell the full story sometimes and for this reason we provide 2D slices of the same experiment (as performed for Figure 3.2) in Figure 3.3.



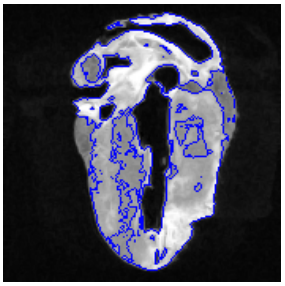
(a) Original



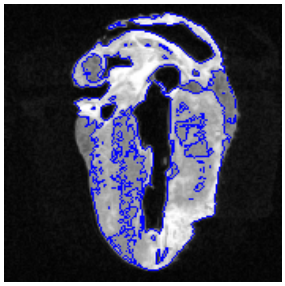
(b) KM method: $k = 40$



(c) KM method: $k = 80$



(d) Our method: $\sigma = 1$



(e) Our method: $\sigma = 10$

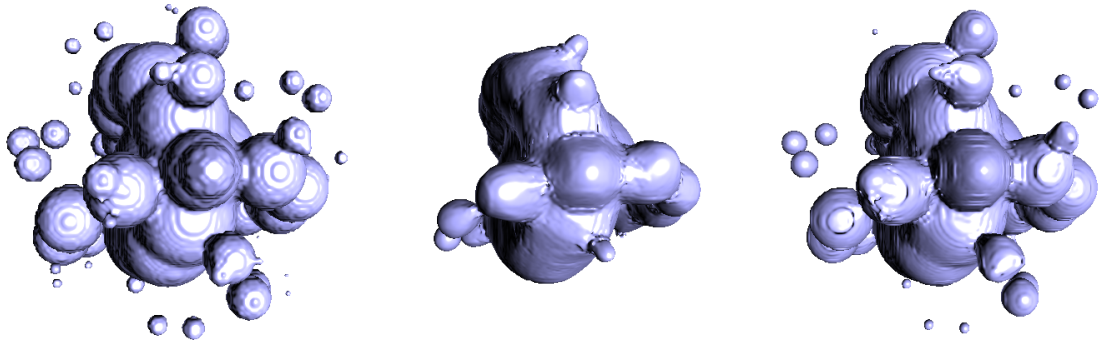
Figure 3.3: The 2D slice ($z = 65$) of the Sheep's Heart dataset, after diffusing with the KM method and our method with 100 iterations as used in Figure 3.2e and 3.2h. We have also marked the same iso-surface of 153 with blue lines.

On the other hand, our method has been found to be more robust with respect to its parameter σ. Unlike the KM model, smoothing of an iso-surface in our approach is performed consistently without any discrimination based on the gradient magnitudes. Since the parameter σ in our model is tied to the directional second derivative, $f_{\mathbf{nn}}$, it has a

more intuitive impact on the overall diffusion process; i.e. decreasing σ amounts to smoothing a larger range of $f_{\mathbf{nn}}$. This can be thought of removing high frequencies from the 3D volume except near the boundaries between relatively homogeneous regions, where $f_{\mathbf{nn}}$ approaches zero and the stopping function $h(f_{\mathbf{nn}})$ approaches zero too, making the diffusion stop (see Equation (3.9)). The sensitivity of parameters in both the KM and our proposed model is further illustrated in two separate animations we provide as supplementary materials (`KM_K_effect.avi` and `sigma_effect.avi` respectively). For both animations, we only changed the parameters k and σ for the KM and our model respectively and ran 35 iterations of diffusion while all the other parameters were kept the same.

3.3.3 Significance of σ

In our previous examples, we have demonstrated the robustness of our new diffusion model with regards to the parameter σ. However, this robustness is observed for points away from zero. In this section, we argue with an appropriate example that the role of σ around zero is critical.



(a) Original (b) $h(\cdot) = 1$ (c) $\sigma = 40$

Figure 3.4: Iso-surface rendering of a sampled phantom data with spheres. The value inside the spherical regions were 255 and 0 elsewhere. The volumes in (b) and (c) were diffused with 300 iterations. (a) The original volume (b) Diffused with $h(\cdot) = 1$ (c) Diffused with Equation (3.5) set as $h(\cdot)$ with $\sigma = 40$.

A quick look at Equation (3.9) immediately reveals that having the stopping function $h(\cdot)$ always evaluate to 1 with $\eta = 0$ makes the diffusion similar to the well known *mean curvature motion* [32] (assuming $\tau \approx 1$, i.e. on isotropic surfaces, like a sphere). *Mean curvature motion* is a well studied diffusion scheme where spherical structures shrink until

they disappear, which may not be desirable when preserving structures in a 3D volume.

To verify this we created a phantom dataset with several spherical regions of different sizes and diffused it by making $h(\cdot)$ a constant 1 as well as by using Equation (3.5) with $\sigma = 40$. Figure 3.4 aptly demonstrates the significance of the parameter σ. When the phantom data was diffused with $\sigma = 40$ (Figure 3.4c) the basic structure of the original data, Figure 3.4a, was retained except for the very small spheres. On the other hand when the stopping function $h(\cdot)$ was set to a constant 1, Figure 3.4b, the overall diffusion converged to a simple *mean curvature motion* and the structure was destroyed. Our supplemental material includes animations that show the full evolution of the diffusions as given in Figure 3.4b (`sigma_unity.avi`), and 3.4c (`sigma_40.avi`).

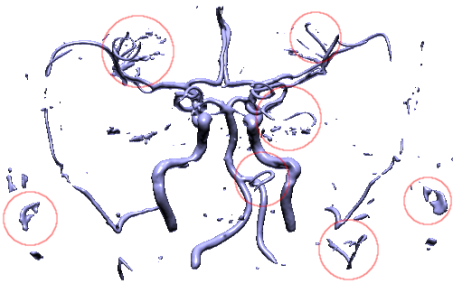
3.3.4 Impact of Derivative Estimation Filters

To implement Equation (3.9) we need to compute the principal curvatures which require second order derivatives for the Hessian $\mathbf{H}$ in addition to the gradient components. Since we are computing all these quantities from sampled data the quality of the derivative estimation filters plays an important role.

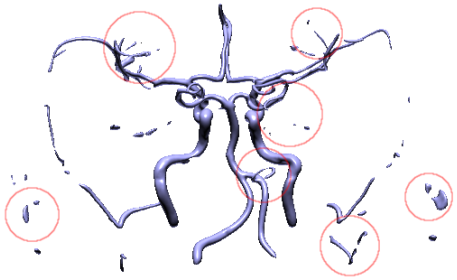
We used the Taylor Series based framework, which is discussed in more details in Chapter 4, to construct discrete derivative estimation filters of error order 2-EF and 4-EF [1]. Usually for real data where the polynomial order of the underlying function is not known apriori, a 4-EF filter has been found to yield more accurate results than 2-EF.

For our experiment, we used an Angiography dataset [6] in which the blood vessels were the focus of the study. We used the same diffusion parameters ($\sigma = 1$) and only varied the order of the derivative filters. For the derivative filters, we used 2-*cd* (see Table C.1a) and 4-*cd* (see Table C.1b) as the second and fourth order filters respectively (consult Chapter 4 for more details). Figure 3.5 demonstrates that the higher order filter (4-EF) preserves more details in the blood vessels while still removing some of the spurious elements.

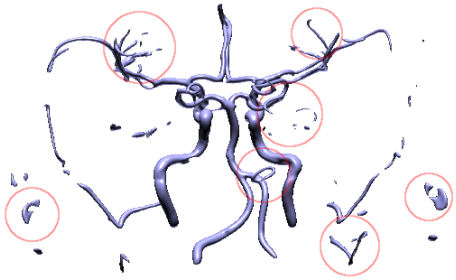
[1] A filter is called n-EF, where EF stands for *Error Filter*, if it estimates a given derivative with error bounded by $O(h^n)$, where h is the grid spacing.



(a) Original



(b) 2-EF



(c) 4-EF

Figure 3.5: Direct volume rendering of an angiography dataset of a human head. The red circles mark the regions where the images had noticeable differences. (a) The original dataset. (b) Derivatives estimated using 2-EF filters. (c) Derivatives estimated using 4-EF filters.

3.3.5 Stability and Convergence

Setting $\eta = 0$ and rearranging terms renders Equation (3.9) into the following:

$$\frac{\partial f}{\partial t} + h(\cdot)\left(\kappa_1 + \tau\kappa_2\right)\mathbf{N}\cdot\nabla f = 0 \tag{3.10}$$

where $\mathbf{N} = \nabla f/\|\nabla f\|$ is the normal of the scalar field f at every point. Equation (3.10) is of the form $\frac{\partial f}{\partial t} + \mathbf{V}\cdot\nabla f = 0$ where $\mathbf{V}$ is the velocity, at a point, during the evolution of the PDE of the level set passing through the same point. Hence, Equation (3.10) describes an evolution whereby the level sets would be moving along the directions of the normals of the level sets. Due to the notational convention adopted in this thesis, as discussed in Section 3.2, this propagation of the level-sets along the direction of the normals is consistent and similar to the *Mean Curvature Motion* (MCM) as discussed in the book of Osher and

Fedkiw [39].

In this section we will use the *Courant-Friedrichs-Lewy* (CFL) condition to provide a necessary but not sufficient condition on Δt to be used in numerical methods to evolve Equation (3.9).

The CFL requires that the distance travelled by a level set during the time interval Δt must be less than the minimum distance between two neighboring grid points. Mathematically this means

$$s\Delta t < d \tag{3.11}$$

Where d is the minimum distance between two neighboring grid points and s is the speed, i.e the magnitude of the velocity $\mathbf{V}$. In case of a Cartesian Cubic (CC) Lattice, $d = \min\{dx, dy, dz\}$ where dx, dy, and dz are the grid spacings along x, y, and z axes respectively. In Equation (3.10), $s = |h(\cdot)(\kappa_1 + \tau\kappa_2)|$ and therefore, from the CFL point of view, the following must be satisfied at every point to keep the numerical evolution stable.

$$\begin{aligned} |h(\cdot)(\kappa_1 + \tau\kappa_2)|\,\Delta t &< d \\ \Delta t &< \frac{d}{|h(\cdot)(\kappa_1 + \tau\kappa_2)|} \end{aligned} \tag{3.12}$$

Note that both $h(\cdot)$ and τ are bounded between 0 and 1, i.e.

$$0 \le h(\cdot) \le 1, \qquad 0 \le \tau \le 1, \tag{3.13}$$

Therefore,

$$\begin{aligned} \frac{d}{|h(\cdot)(\kappa_1 + \tau\kappa_2)|} &\ge \frac{d}{|(\kappa_1 + \tau\kappa_2)|}, && 0 \le h(\cdot) \le 1 \\ &\ge \frac{d}{|\kappa_1| + |\tau\kappa_2|}, \\ &\ge \frac{d}{|\kappa_1| + |\kappa_2|}, && 0 \le \tau \le 1 \end{aligned}$$

By definition, $|\kappa_1| \le |\kappa_2|$, and hence

$$\ge \frac{d}{2\,|\kappa_2|} \tag{3.14}$$

An important observation to be made here is that the largest meaningful curvature that can be captured in a grid must be bounded by $\kappa_2 \leq 1/d$, as argued by Osher and Fedkiw [39]. This is because the smallest sphere that can be sampled in a grid must have a radius greater than d. Hence, we can write:

$$\frac{d}{2\left|\kappa_2\right|} \geq \frac{d}{2\frac{1}{d}} = \frac{d^2}{2} \tag{3.15}$$

Therefore the right hand side of Equation (3.12) is bounded from the bottom by the following:

$$\frac{d^2}{2} \leq \frac{d}{\left|h(\cdot)\left(\kappa_1 + \tau\kappa_2\right)\right|} \tag{3.16}$$

Hence a safe choice for Δt must satisfy the following relationship:

$$\Delta t \leq C\frac{d^2}{2} \tag{3.17}$$

where C is the Courant number which is a constant and is bounded by $0 \leq C \leq 1$ and is typically assigned a value between 0.5 and 0.9.

To prescribe a value for the Courant number C in Equation (3.17) we performed empirical analysis of stability and convergence. For this analysis we only varied Δt for each diffusion experiment keeping all the other parameters the same. Let $L_2(i,j)$ denote the l_2 norm between the volumes $f(\mathbf{x},i)$ and $f(\mathbf{x},j)$ at iterations i and j respectively during the evolution. This is given by the following:

$$L_2(i,j) = \sqrt{\sum_{\mathbf{x}\in\mathbb{R}^3}\left(f(\mathbf{x},i) - f(\mathbf{x},j)\right)^2} \tag{3.18}$$

The *Root Mean Squared Difference* RMSD between two volumes at iterations i and j, which is just the scaled l_2 norm, can now be given by:

$$RMSD(i,j) = \frac{L_2(i,j)}{\sqrt{V}} \tag{3.19}$$

where V is the total number of voxels and is a constant for a given dataset. Considering the volume $f(\mathbf{x},n)$ as a V dimensional point in $\mathbb{R}^V$, the RMSD can be thought of as the *Euclidean distance* between the volume at iterations i and j, scaled by a constant $1/\sqrt{V}$.

We define a quantity $D(n)$, that measures the RMSD of the volume at iteration $n \in$

$\{0 \dots N\}$ from the original volume, i.e. $f(\mathbf{x}, 0)$, as given below:

$$D(n) = RMSD(n, 0) = \frac{L_2(n, 0)}{\sqrt{V}} \tag{3.20}$$

Finally we define the rate of change of the *Euclidean distance*, scaled by the constant $1/\sqrt{V}$, with respect to time t between two successive volumes at iterations $n-1$ and n by the following:

$$S(n) = \frac{RMSD(n, n-1)}{\Delta t} = \frac{L_2(n, n-1)}{\Delta t \sqrt{V}} \tag{3.21}$$

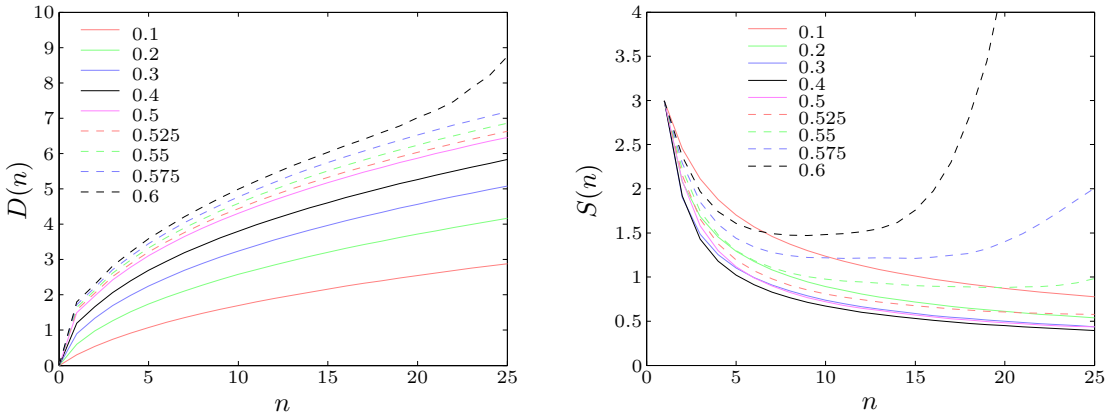
Note that the quantity $S(n)$ is nothing but a numerical approximation of the instantaneous *speed*, scaled by the $1/\sqrt{V}$, of the evolution of the volume $f(\mathbf{x}, n) \in \mathbb{R}^V$ at iteration n.

Now, for every Δt we ran N iterations of diffusion on a dataset and measured $D(n)$ and $S(n)$. The quantity $D(n)$ will show how a volume evolves with respect to the original volume in an l_2 norm sense and as well provide evidence of de-noising, which we will show later. On the other hand $S(n) \to 0$ as $n \to \infty$ will provide evidence of convergence of our proposed PDE.

We used a simulated structural MR data, obtained from the BrainWeb database [10] for this study. This data is noise free yet realistic with many details and variation. Therefore, it is a good candidate for our test scenario. The structural MR data contains 256 gray levels and has a size of $181 \times 217 \times 181$.

In the first phase of the experiment we kept all the parameters the same as Section 3.3.1, except we set $\rho = 1$ (in voxel units), and only varied Δt. Therefore, the other parameters were: $\lambda = 2$, $\sigma = 1$, and voxel spacing was assumed to be 1 in all directions and scalar values ranged between $[0, 255]$. The 3D volumes were not scaled and a simple central differencing 2-EF filter was applied for all the derivative estimations. For each Δt we ran $n = 25$ iterations. Figure 3.6 plots $D(n)$ and $S(n)$ for different values of Δt as indicated by the legend. With the noise-free MRI data, Figure 3.6b shows that for $\Delta t \leq 0.4$ the PDE behaves well. However for $\Delta t \geq 0.5$, the *speed* $S(n)$ drops less quickly until $\Delta t \geq 0.55$ when the PDE becomes unstable. However, the plot for $D(n)$ (Figure 3.6a) does not reveal anomalies until $\Delta t \geq 0.6$.

In the second phase of the experiment we added Gaussian noise with zero mean and a variance of 0.01 (in a normalized scale) to the synthetic MR data to add random high frequency variation to pose a more challenging test for our proposed diffusion PDE in terms



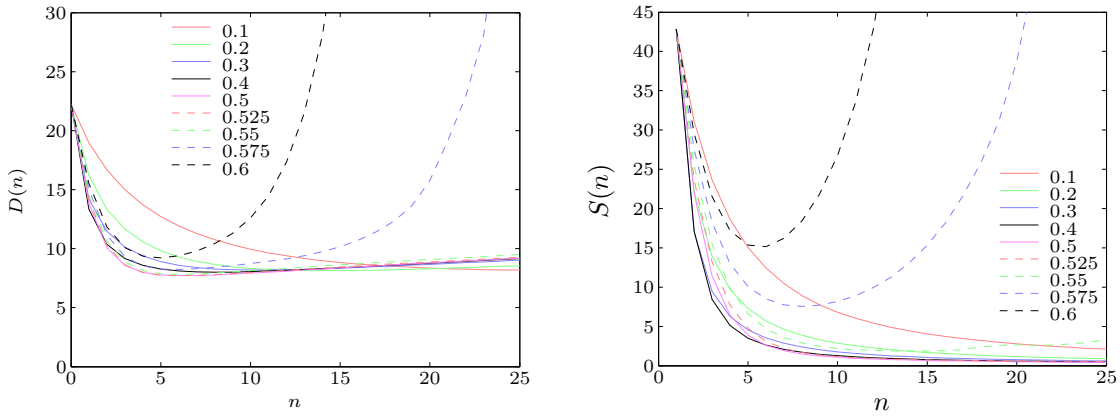
(a) $D(n)$ (b) $S(n)$

Figure 3.6: Plot of (a) $D(n)$ and (b) $S(n)$ for each Δt (indicated by the legend), with the noise-free MRI data.

of stability. This noisy volume has an SNR of 11.3 dB. All other parameters were kept the same. For this experiment $D(n)$ was measured from the original noise-free MRI data. In Figure 3.7a, the slope of $D(n)$ is negative initially because the $D(n)$ was measured from the original noise-free MRI data and diffusion would bring the noisy data closer to the original in an l_2 sense with each iteration, i.e. the l_2 norm would progressively get reduced. This is an indication of de-noising taking place. This time however, both Figure 3.7a and Figure 3.7b indicate that for $\Delta t \geq 0.55$ the PDE becomes unstable. It is noteworthy that even a bad SNR of 11.3 dB did not drastically change the stability from the one we found with the noise-free MRI data.

In the third phase, we used a $40 \times 40 \times 40$ data volume of a random signal uniformly distributed for values in $(0, 255)$. This poses an even more challenging test of stability and convergence. Figure 3.8b shows that even in the case of this random noisy volume the stability of the PDE did not change for $\Delta t < 0.55$. For $0.4 < \Delta t < 0.55$, although the PDE eventually converged, Figure 3.8a reveals some oscillation in $D(n)$ in the first few iterations.

In all three experiments, the plot of $S(n)$ showed that the PDE converged for $\Delta t \leq 0.4$ with $\Delta t = 0.4$ yielding the fastest convergence. On the other hand, the plot of $D(n)$ in all three experiments revealed that for $\Delta t \leq 0.4$ the PDE evolves without any oscillation. This led us to believe that for the parameter settings used in Section 3.3.1, augmented with $\rho = 1$, our proposed PDE is stable for $\Delta t \leq 0.4$ for most practical purposes.



(a) $D(n)$ (b) $S(n)$

Figure 3.7: Plot of (a) $D(n)$ and (b) $S(n)$ for each Δt (indicated by the legend), with the MRI data corrupted with Gaussian noise.

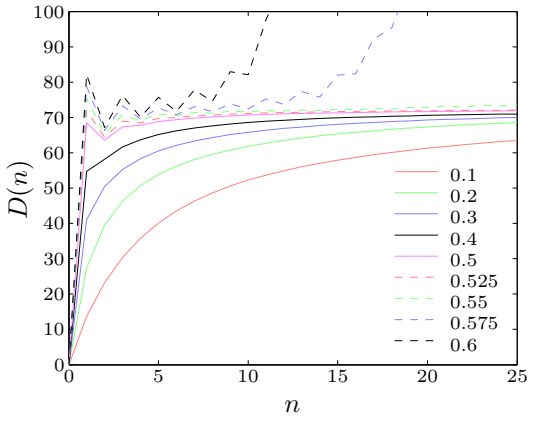
The choice of $\rho = 1$, which is common in many diffusion methods, was made only to have a better smoothing behavior in a relatively homogeneous region under noise, and this has little effect on the stability. This parameter ρ is typically useful for the purpose of de-noising and the value of 1 yields best results for most noise types as we will demonstrate in the next section.

From this empirical analysis we conclude that a Courant number C of 0.8 in Equation (3.17) is stable for most practical purposes and therefore prescribe the following condition for stability:

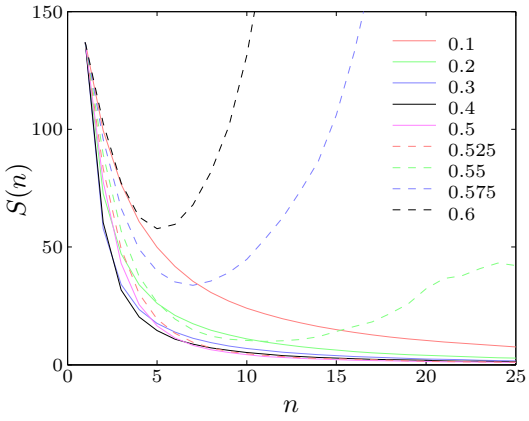
$$\Delta t \leq \frac{0.8d^2}{2} = 0.4d^2 \tag{3.22}$$

where d is the minimum distance between two neighboring grid points.

Finally, we validated the above Courant number by performing two more experiments with grid spacing set to 0.5 and 0.25. According to Equation (3.22) a time stepping of $\Delta t \leq 0.1$ and $\Delta t \leq 0.025$ should be stable for the grid spacing of 0.5 and 0.25 respectively. The experimental results are shown in Figure 3.9 which corroborates with this prediction. In both cases (grid spacing of 0.5 and 0.25) the evolution becomes unstable above the prescribed time steppings.

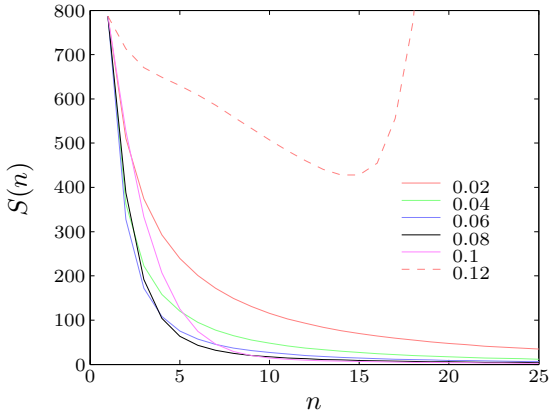


(a) $D(n)$

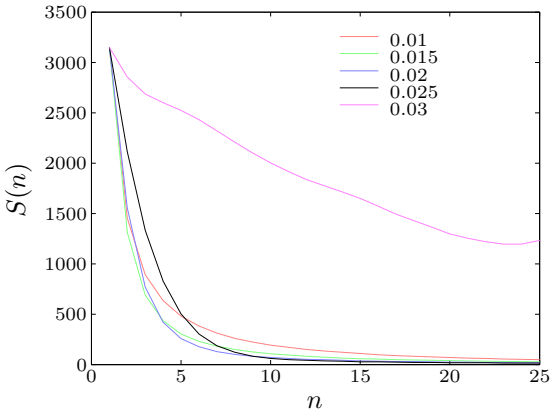


(b) $S(n)$

Figure 3.8: Plot of (a) $D(n)$ and (b) $S(n)$ for each Δt (indicated by the legend), with a random data volume $(40 \times 40 \times 40)$ uniformly distributed for $[0, 255]$.



(a) Grid spacing $= 0.5$. Stable for $\Delta t \leq 0.1$



(b) Grid spacing $= 0.25$. Stable for $\Delta t \leq 0.025$

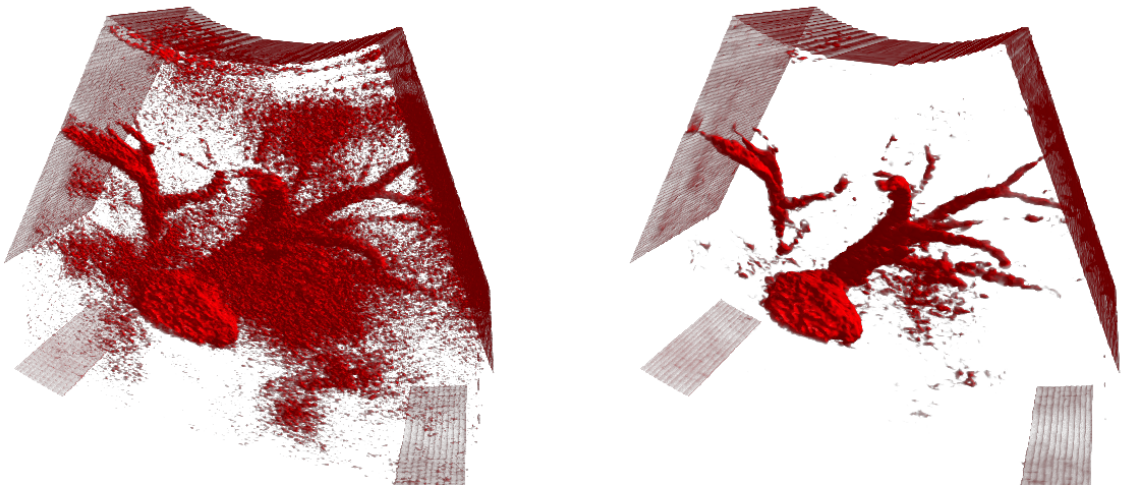
Figure 3.9: Plot of $S(n)$ for each Δt (indicated by the legend) for a grid spacing of (a) 0.5 and (b) 0.25 , with a random data volume $(40 \times 40 \times 40)$ uniformly distributed for $[0, 255]$.

3.4 De-noising Properties of Our Proposed Model

Although our proposed PDE model is a general purpose smoothing technique to evolve a 3D volume over time removing small high frequency details while preserving edges it exhibits de-noising properties too. On the other hand, general purpose de-noising techniques have been formulated previously, e.g. simple *bi-lateral filtering*, which has been shown to be a variant of gradient magnitude based *non-linear diffusion* by Barash and Comaniciu [5]. But the drawbacks of the gradient magnitude based models have been aptly demonstrated in Section 3.3.2 and therefore equivalent de-noising operators will inherit similar problems. Further, anistropic diffusion in 3D based on the KM model is very sensitive to the parameter choice. Therefore, none of these existing techniques would be well suited for de-noising in 3D without facing difficulties.

In this subsection we will discuss, both qualitatively and quantitatively, the de-noising properties our proposed PDE model has in the context of the four common noise types: additive *Gaussian* noise, additive *Poisson* noise, multiplicative *Speckle* noise and *Salt and Pepper* noise. We chose the Tooth dataset [43] which is relatively noise-free and diffused it using our proposed method after adding a particular type of noise. For all the experiments we have set the parameters as described in Section 3.3.5.

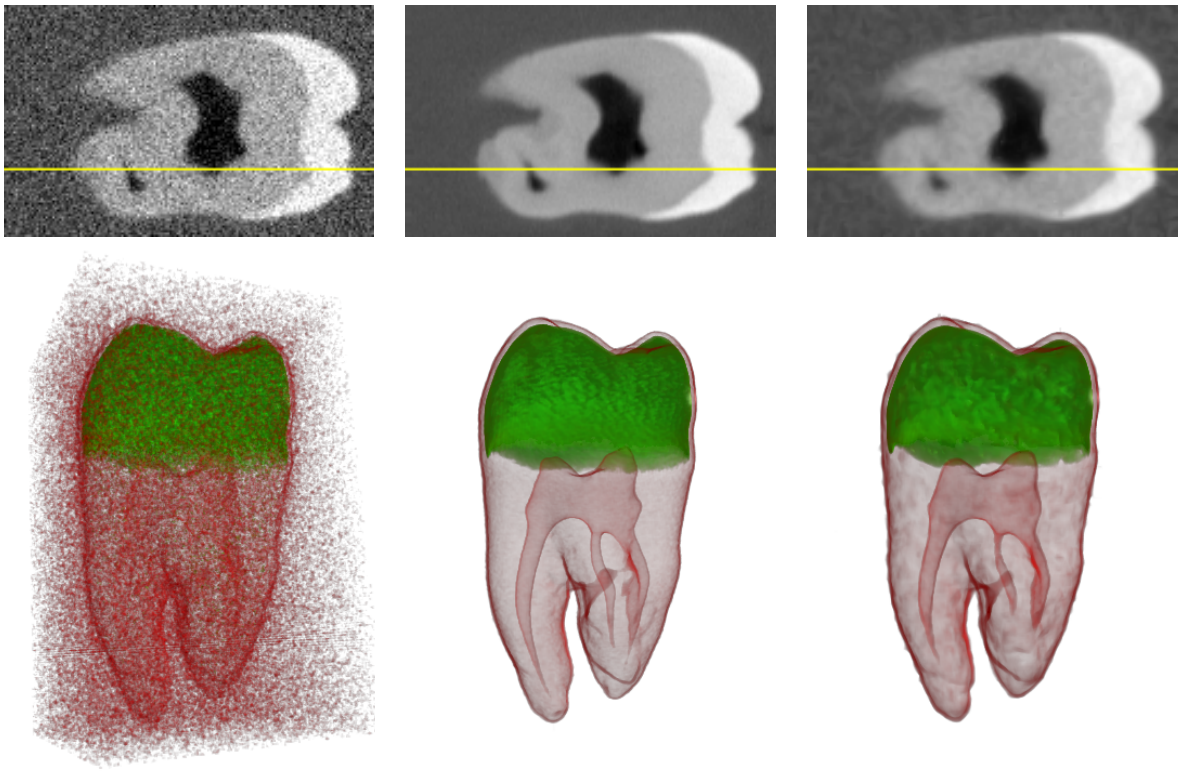
Figure 3.11–3.14 summarizes the de-noising properties of our proposed anistropic diffusion model and it shows that our proposed method could de-noise the tooth data quite well in all four cases. Our method works best with *Salt and Pepper* noise which is of no surprise because *Salt and Pepper* noise introduces random and very local blob type artifacts in the volume that get removed immediately and remarkably well. For *Gaussian* noise and *Poisson* noise, our method performed similarly well on both occasions. *Speckle* noise turned out to be the hardest to tackle of all, which is not surprising, and yet our method performed well achieving an SNR of over 24.5 dB (see Figure 3.13). We also provide animation sequences to show the de-noising process for each noise type listed in Figure 3.11–3.14 as supplementary materials: `gaussian.avi`, `poisson.avi`, `speckle.avi`, and `salt-pepper.avi`. In Figure 3.10 we present a qualitative result of our diffusion model applied to a 3D Ultrasound data of human liver [41] with a size of $247 \times 208 \times 86$, which has been sub-sampled from the original data by a factor of two in each dimension only to speed up computations. Ultrasound data are usually contaminated with *speckle* noise and it is noteworthy how the noise was lessened keeping all the vital structures intact even for a relatively low resolution



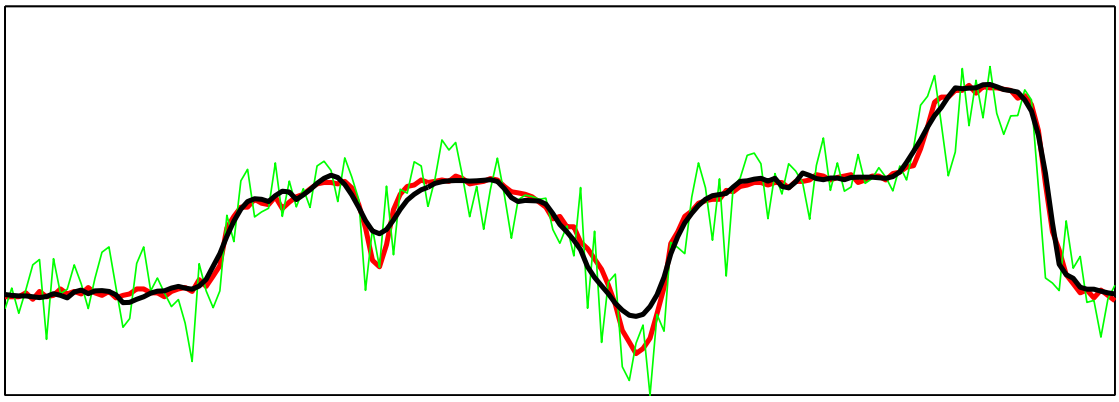
(a) Original (b) Diffused

Figure 3.10: A 3D Ultrasound data diffused with our method using $\sigma = 1$ and 5 iterations. (a) The original volume (b) The diffused volume using the same transfer function.

volume. Note how the tubular structures, which were barely discernible in Figure 3.10a, stick out clearly in Figure 3.10b. When we attempt to apply the KM model on this same dataset we face real challenges to pick a right value for the parameter k as we had no prior knowledge about the gradient magnitude distribution around the tubular structures and this itself speaks in favor of our method where we could pick within a wide range of values for σ and still get some decent and consistent results.

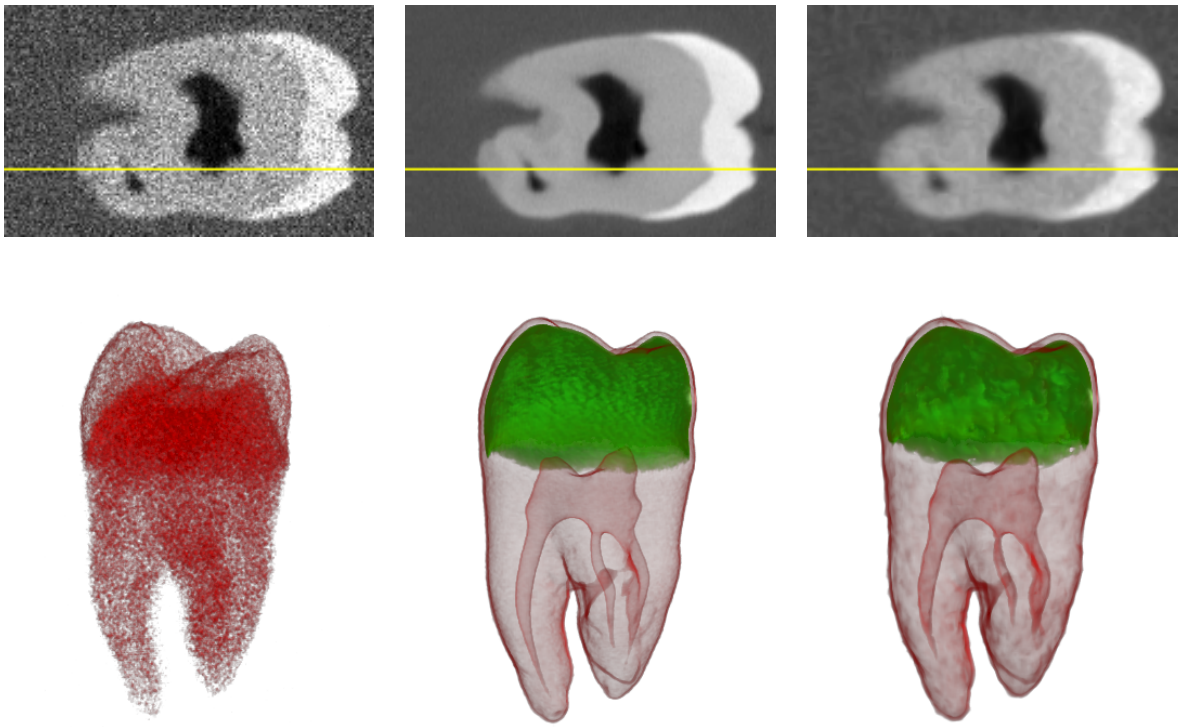


Noisy, SNR=12.89 Original Our method, SNR=26.12

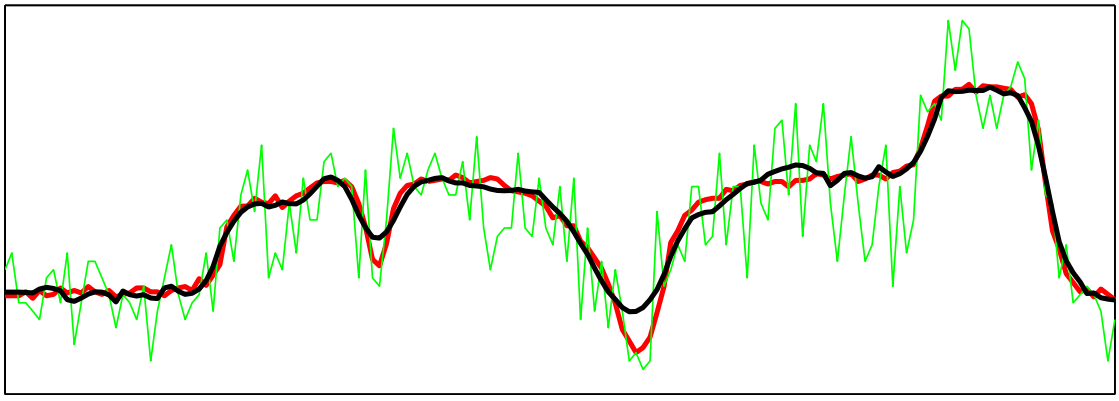


Profile

Figure 3.11: Gaussian noise: $\lambda = 2, \Delta t = 0.4, \sigma = 1, \rho = 1,$ iterations $= 25$.

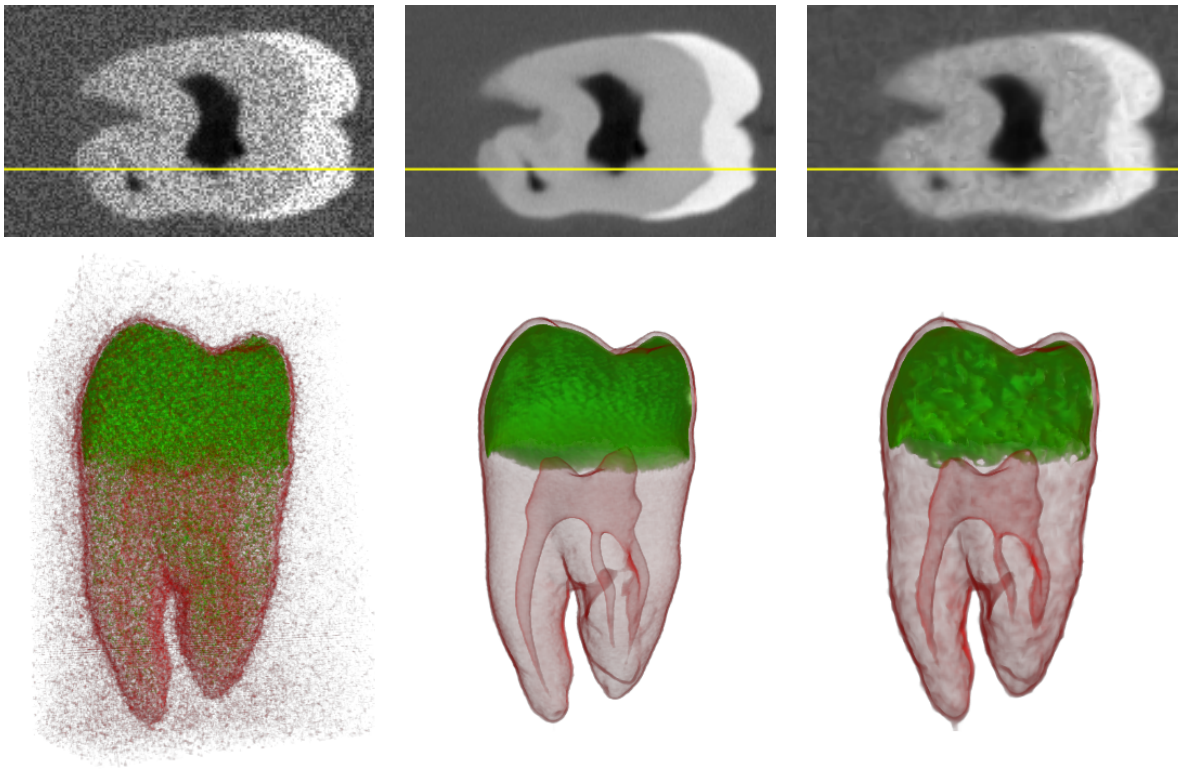


Noisy, SNR=12.80 Original Our method, SNR=25.76

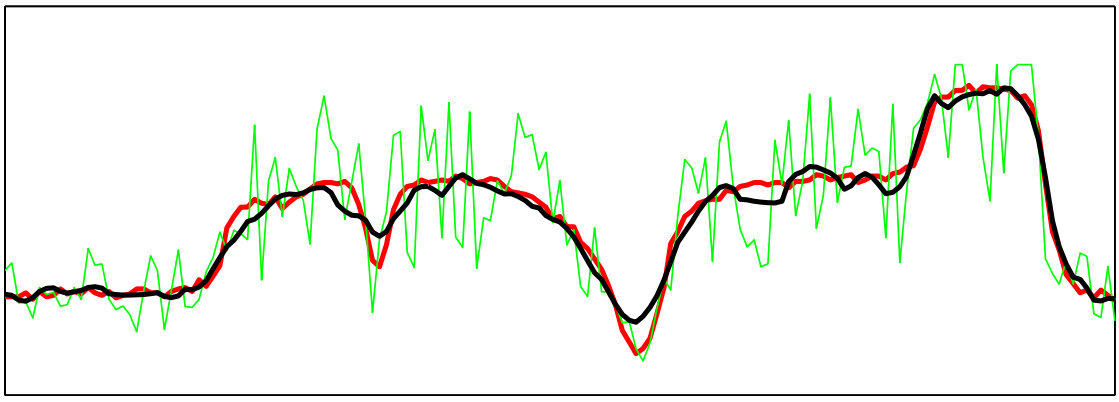


Profile

Figure 3.12: Poisson noise: $\lambda = 2, \Delta t = 0.4, \sigma = 1, \rho = 1,$ iterations $= 25$

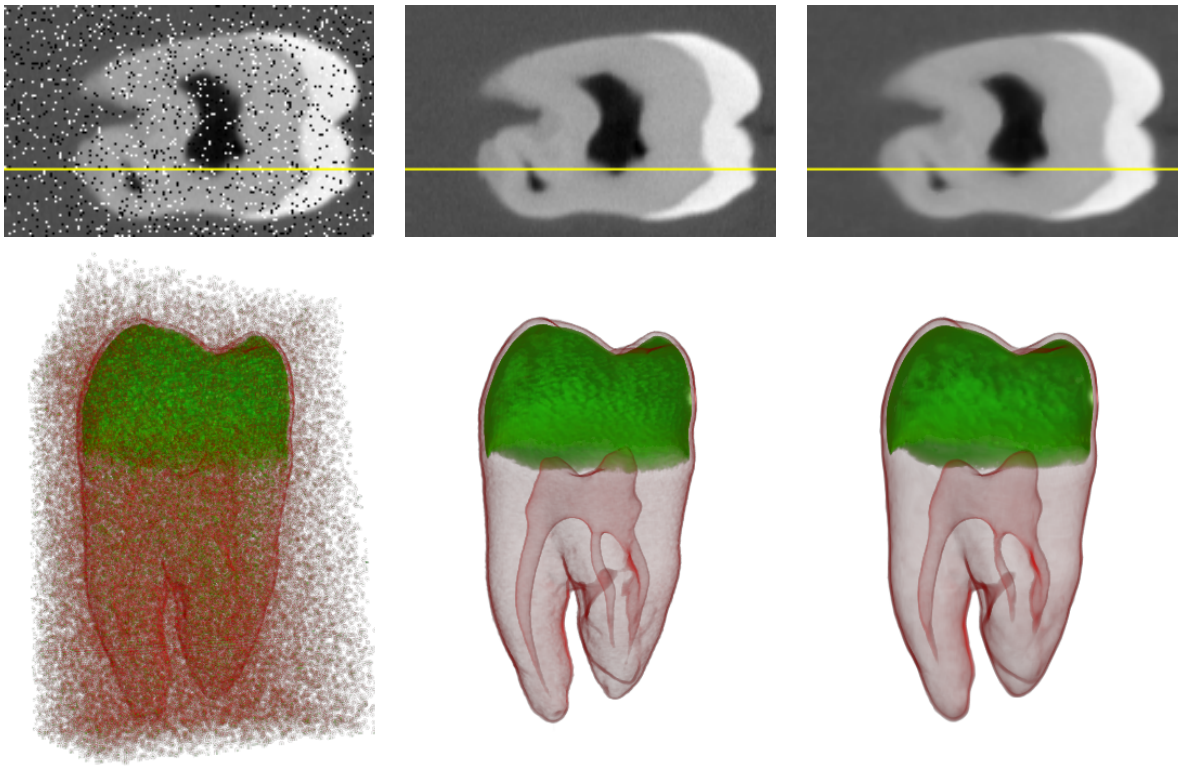


Noisy, SNR=12.99 Original Our method, SNR=24.68

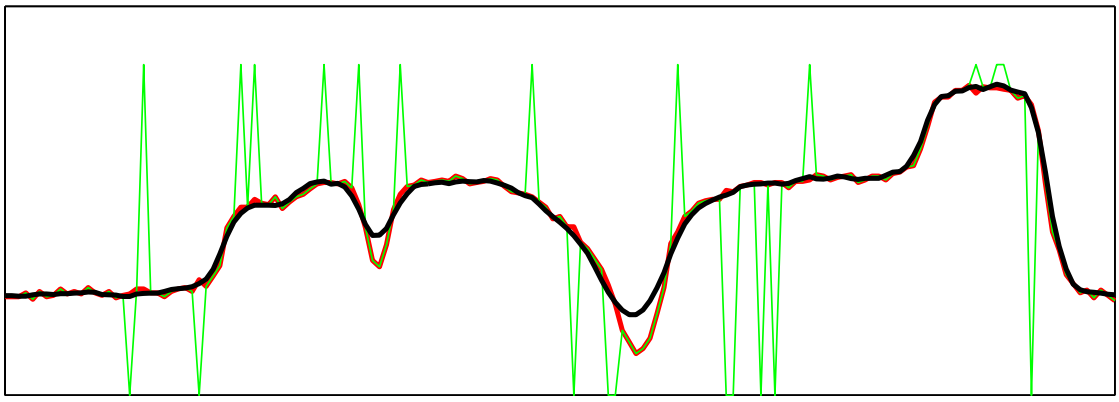


Profile

Figure 3.13: Speckle noise: $\lambda = 2, \Delta t = 0.4, \sigma = 1, \rho = 1,$ iterations $= 25$



Noisy, SNR=8.17 Original Our method, SNR=30.06



Profile

Figure 3.14: Salt and Pepper noise: $\lambda = 2, \Delta t = 0.4, \sigma = 1, \rho = 1,$ iterations $= 25$

3.4.1 Comparison with other de-noising methods

In this section we will compare our method with two very recent PDE based de-noising filters, namely *Scalar Rician Noise-Reducing Anisotropic Diffusion* SRNRAD and *Oriented Rician Noise-Reducing Anisotropic Diffusion* ORNRAD, proposed by Krissian and Aja-Fernández [28]. These filters were designed to de-noise specifically Rician noise in 3D MRI data. At a higher SNR, a Rician distribution converges to a Gaussian distribution. According to Aja-Fernández et al. [2], the noise estimator used by Krissian and Aja-Fernández [28] is based on the variance estimation of additive Gaussian noise. This is, however, not an unfair mismatch because the Rician noise converges to additive Gaussian noise after few iterations. Therefore, we will also apply their filters on additive Gaussian noise. It is worth mentioning that both SRNRAD and ORNRAD require users to manually specify a sub-volume for the noise estimator. Further, to demonstrate the versatility of our proposed method, we will apply all three methods on other types of noise and show that our method can be applied equally well in most types of naturally occurring noise types using exactly the same set of parameter values.

For comparison we have chosen the same dataset as [28]: the simulated structural MR data [10], and used the same quantitative metrics - namely *Mean Squared Error* (MSE), *Structural Similarity Index* (SSIM) [53], and *Quality Index Based on Local Variance* (QILV) [1] - for quality assessment. In accordance to Krissian and Aja-Fernández [28], we also discarded the background, where the original noise-free volume is zero, from any assessments. It is noteworthy to mention that we have used the exact same Matlab script to compute these metrics as the authors of [28] and for SRNRAD and ORNRAD we have used their own C/C++ implementations in the AMILab software [27].

Table 3.1 summarizes the performance of the three diffusion techniques and we provide the corresponding images in Figure 3.15–3.19. In terms of MSE, our method performed significantly better than both ORNRAD and SRNRAD for all noise types except *Speckle* and *Poisson*, where the differences are close for the MSE metric. For the SSIM metric, we find that our method again performed better than both ORNRAD and SRNRAD for all noise types except *Speckle* where the numerical difference is only in the third decimal place. It is rather intriguing to find that our method tied with ORNRAD and actually performed better than SRNRAD for the SSIM metric in the case of *Rician* noise, for which those two filters were specifically designed for. Using the QILV metric, our method performed only

	Gaussian			Rician		
	MSE	SSIM	QILV	MSE	SSIM	QILV
Noise	558.750	0.540	0.503	450.558	0.561	0.712
SRNRAD	162.017	0.816	**0.886**	232.459	0.792	**0.913**
ORNRAD	153.873	0.819	0.859	226.405	**0.795**	0.889
Our	**75.878**	**0.900**	0.860	**173.851**	**0.795**	0.820

	Poisson			Salt & Pepper			Speckle		
	MSE	SSIM	QILV	MSE	SSIM	QILV	MSE	SSIM	QILV
Noise	482.905	0.678	0.492	2202.452	0.476	0.022	493.013	0.714	0.455
SRNRAD	91.766	0.917	**0.914**	1262.760	0.625	0.033	84.301	**0.929**	**0.906**
ORNRAD	78.731	0.924	0.899	1114.028	0.642	0.101	**78.308**	**0.929**	0.863
Our	**70.671**	**0.930**	0.857	**33.210**	**0.968**	**0.924**	85.129	0.922	0.838

Table 3.1: Performance evaluation of different diffusion methods: our method, SRNRAD and ORNRAD, on different types of naturally occurring noise. The best performing result for each metric is highlighted with a **bold number** while the second best is underlined. For the Rician noise we used a standard deviation of 20 similar to [28]. Note that there can be ties.

marginally different from ORNRAD for all the noise types except *Salt and Pepper* noise for which case our method clearly outperformed both ORNRAD and SRNRAD by a large margin for all the three metrics. However, we acknowledge that ORNRAD and SRNRAD were designed for *Rician* noise (and should also work well for *Gaussian*) but this experiment reveals that our proposed method, unlike many de-noising methods, can be applied generally for all common noise types and still produces decent results if not better in some cases and without re-tuning parameters.

Figure 3.15–3.19 corroborates our numerical results although we see the results yielded by ORNRAD and SRNRAD are visually smoother in homogeneous regions. On the other hand we argue that the fine fiber like structures as seen in the bottom left region of the original volume were preserved better by our method.

Our method was implemented in Matlab where the values of $\tau_\rho, \kappa_1, \kappa_2$ and $\mathbf{n}^\mathsf{T}\mathbf{H}\mathbf{n}$ in Equation (3.9) were computed using MEX files (C/C++ extension for Matlab) using only a single thread. Other computations including all first and second derivative estimations and convolutions were performed using Matlab scripts. On the other hand SRNRAD and ORNRAD were implemented as multi-threaded C/C++ codes in AMILab as mentioned

in [28]. Table 3.2 reports average time (in seconds) each method took per iteration when we ran them on an Intel Core^{TM}2 Duo (2.4 GHz on each core) based system using the MRI dataset. Table 3.2 shows that despite the Matlab implementation our method performs $\approx$ 14.4 times faster than ORNRAD, which was implemented in C/C++ using multiple threads. On the other hand, though SRNRAD - which was implemented in C/C++ using multiple threads too - runs faster, a clear winner is not yet obvious as most computations in our method were performed in Matlab.

	Our	SRNRAD	ORNRAD
Time/Iteration (seconds)	7.52	3.75	108.48

Table 3.2: Average time (in seconds) for each method per iteration while diffusing the MRI dataset ($181 \times 217 \times 181$).

The proposed method is not only superior in run-time performance, but is superior or comparable in both qualitative (Figure 3.15–3.19) and quantitative (Table 3.1) measures. In addition, we would like to re-emphasize the ease of parameter choice in our method as we achieved all of these performances using exactly the same parameter settings without re-tuning.

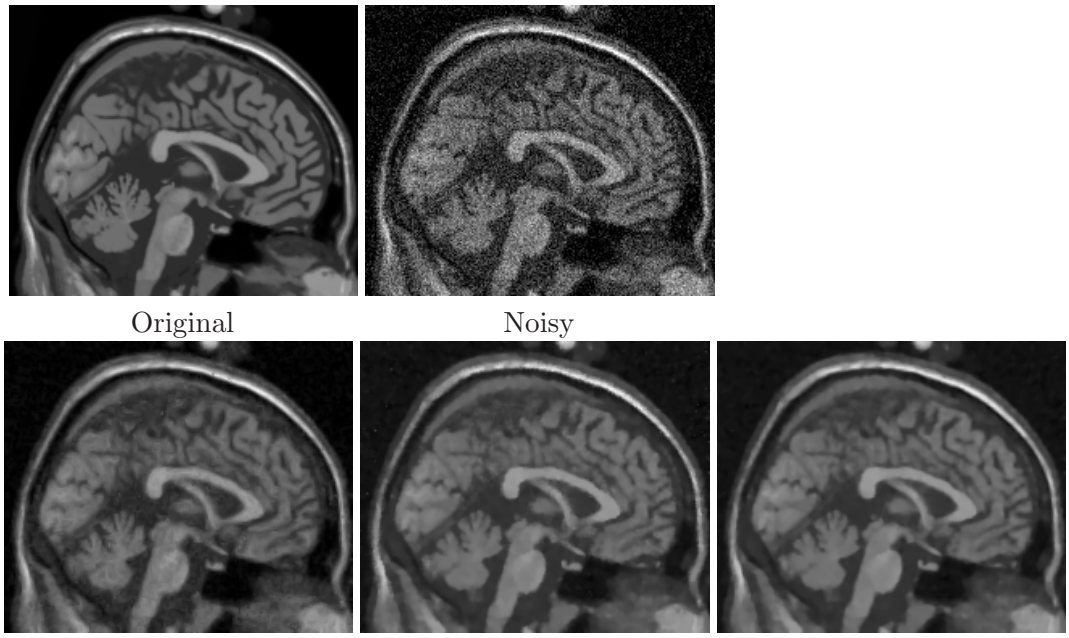


Figure 3.15: Gaussian noise ($x = 90$ slice of the synthetic MRI data)

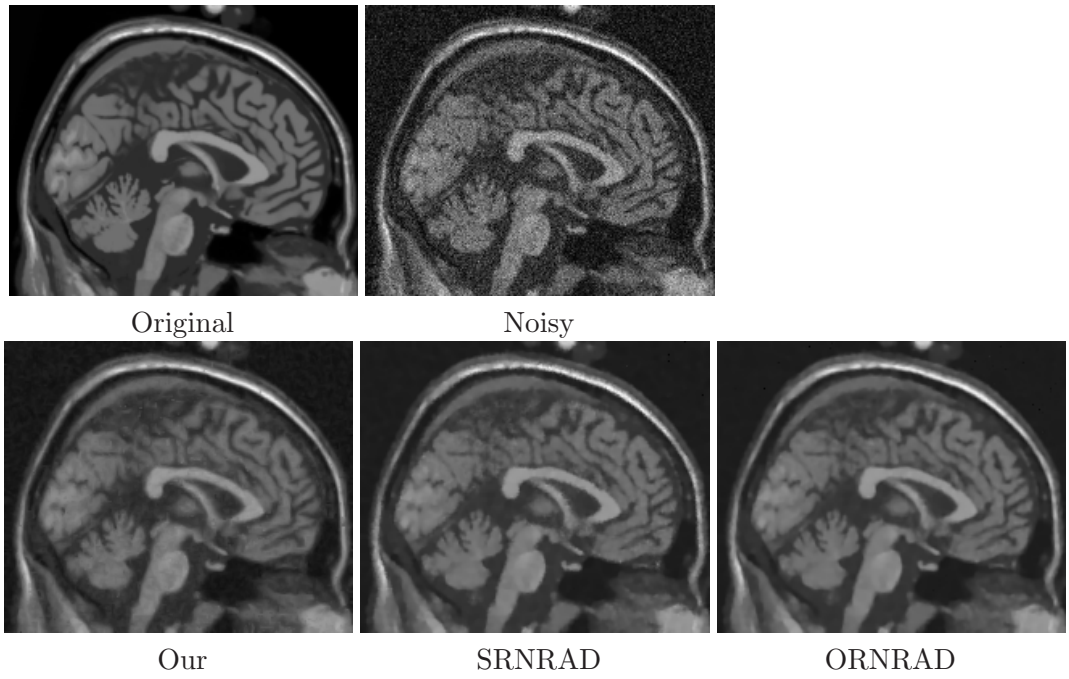


Figure 3.16: Rician noise ($x = 90$ slice of the synthetic MRI data)

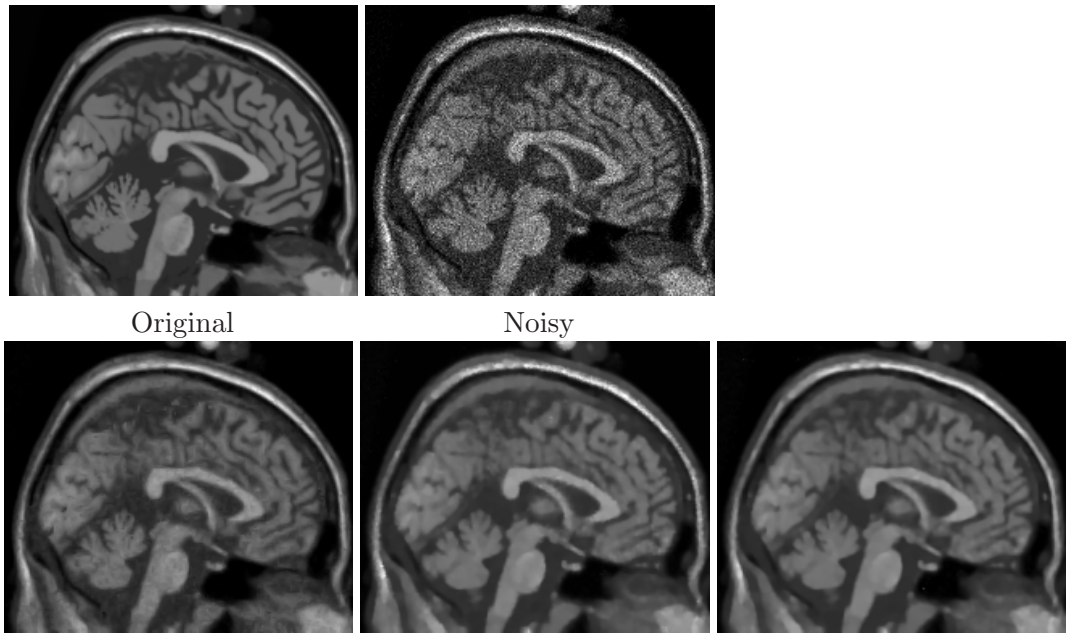


Figure 3.17: Poisson noise ($x = 90$ slice of the synthetic MRI data)

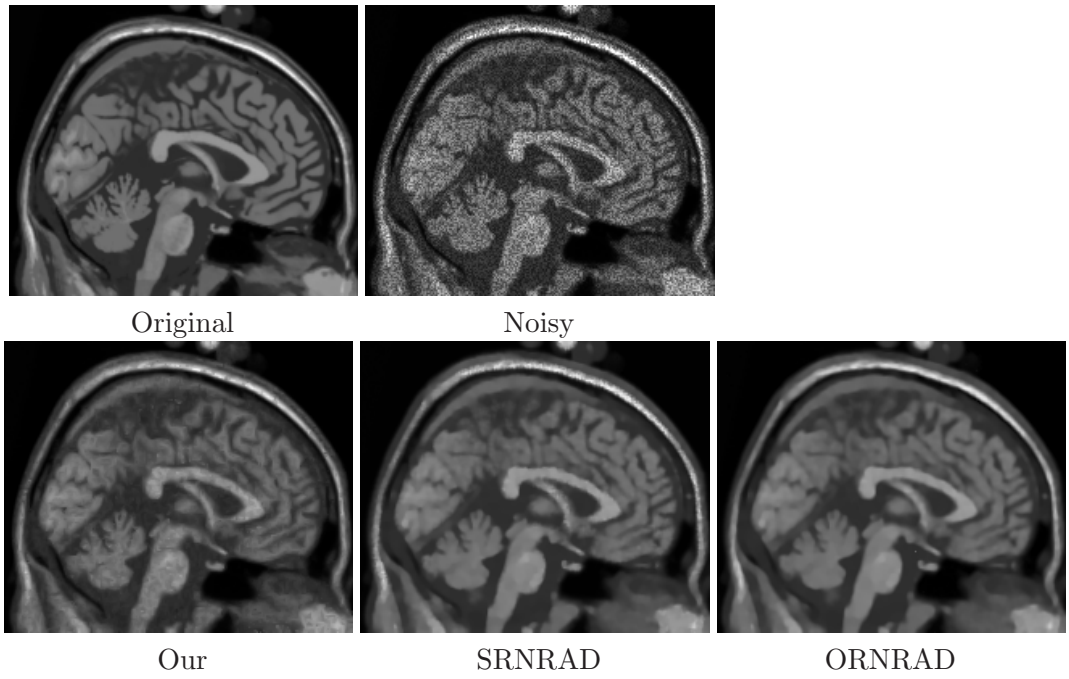


Figure 3.18: Speckle noise ($x = 90$ slice of the synthetic MRI data)

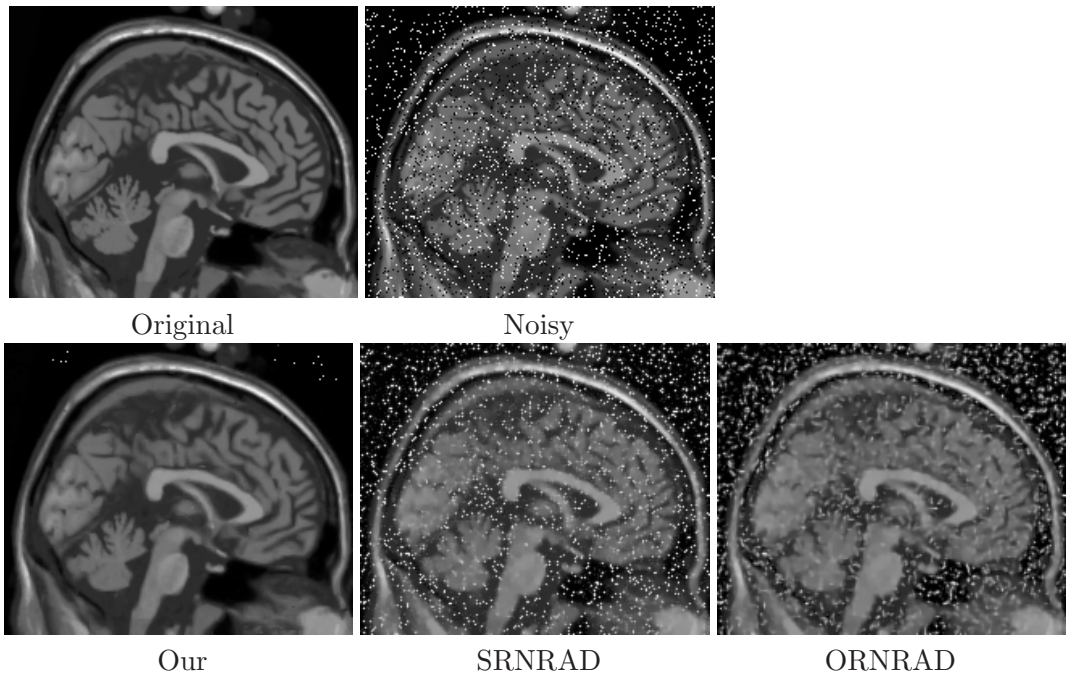
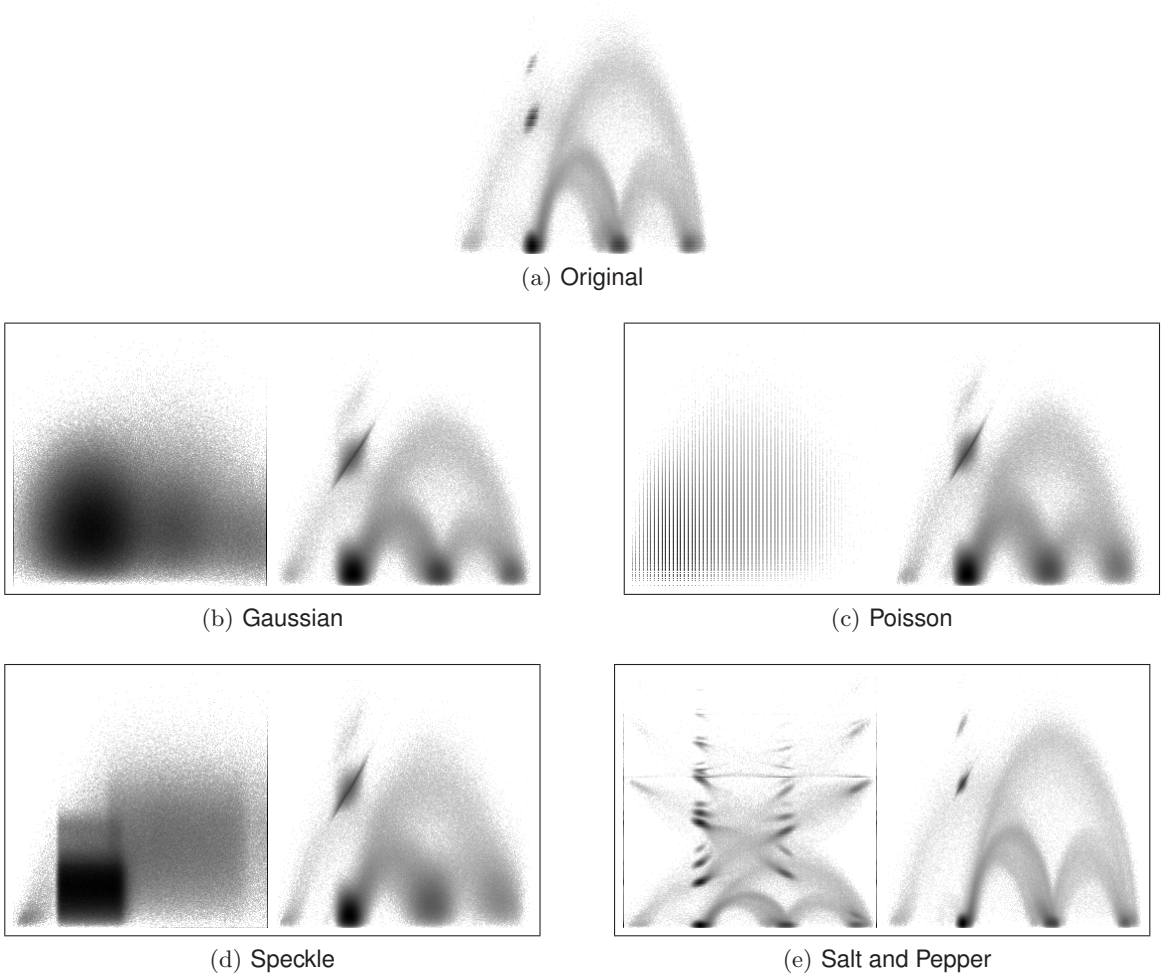


Figure 3.19: Salt and Pepper noise ($x = 90$ slice of the synthetic MRI data)

3.5 Impact on Visualization: 2D Transfer Function

Kindlmann and Durkin [22] proposed a 2D transfer function which proved to be very powerful at classifying homogeneous regions and boundaries in a 3D volume. In their method a 2D histogram would be generated where the horizontal axis would be the function value and the vertical axis would be the directional first derivative along the gradient f_n, a quantity that happens to be just the gradient magnitude, i.e. $f_v = \|\nabla f\|$. For a clean 3D scalar data this histogram will have arc like patterns for every unique boundary in the volume. Because of its importance in visualization, we show the utility of our novel diffusion method in the context of this 2D transfer function. The counts in the histograms are compressed with the function $log_{10}(x+1)$ - where x is the actual count - before plotting them as pixels. Darker pixels denote higher count. For brevity, we will omit the axes labels from the 2D histogram images in this section. In Figure 3.20 we present the 2D histogram before and after diffusion for two different noise types while a reference histogram of the original dataset is provided in Figure 3.20a and also in Figure 3.21a. It is remarkable to see how each noise type changes the 2D histogram so dramatically while our diffusion method brings the original pattern back to a recognizable form. A closer inspection of Figure 3.20e reveals that our diffusion not only removes the *Salt and Pepper* noise but also enhances the pattern in the 2D histogram. This indicates that running our diffusion on a clean dataset for the purpose of smoothing will also enhance its 2D histogram. To verify this, we diffused the tooth data without adding any noise and computed the 2D histogram in Figure 3.21 which shows that the patterns in the histogram are indeed enhanced. This makes sense because our diffusion was modeled in a way such that edges and tubular structures are well preserved while small, high frequency details are smoothed out. Figure 3.22 immediately demonstrates an even greater problem with the gradient magnitude based KM model where the patterns in the 2D histogram get virtually destroyed whereas the patterns get enhanced with our method instead.



(a) Original

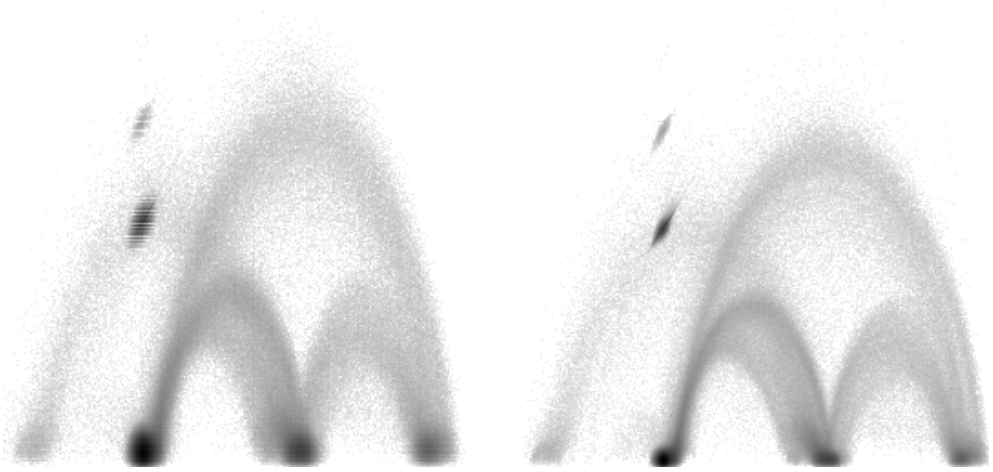
(b) Gaussian

(c) Poisson

(d) Speckle

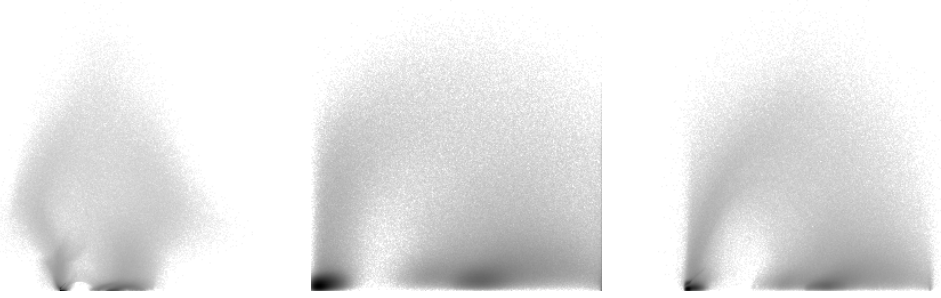
(e) Salt and Pepper

Figure 3.20: 2D histogram of the tooth dataset with four different noise types. (a) is the 2D histogram of the original dataset. After that, each pair of images correponds to a noise type as indicated by the caption. The histogram on the left column of each pair was computed after adding the noise and that of the right column after performing diffusion.



(a) Original (b) Diffused

Figure 3.21: Tooth data diffused without adding any noise. (a) 2D histogram of the original tooth (b) Histogram of the tooth data after diffusion.



(a) KM model (b) Original (c) Our method

Figure 3.22: 2D histogram - (a), (b) and (c) - of the Sheep's heart dataset corresponding to Figure 3.2e, 3.2a, and 3.2h respectively.

Chapter 4

Derivative Estimation on Regular Lattices

4.1 Taylor Series Approach Towards Filter Design

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

$$\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_d], \tag{4.1}$$

where the $\mathbf{l}_i$ are column vectors. Lattice sites of $\mathcal{L}$ are given by the product $\mathbf{Lk}$, where the integer vector $\mathbf{k} = (k_1, k_2, \cdots, k_d)^\mathsf{T}$ 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. [33, 34] and extend it to multiple dimensions.

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

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

$$f(\mathbf{v}) = \sum_{\mathbf{n} \geq \mathbf{0}} \frac{(\mathbf{v} - \mathbf{x})^{\mathbf{n}}}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{x}), \tag{4.2}$$

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

$$D^{\mathbf{n}}(\cdot) := \left(\frac{\partial^{n_1}}{\partial {x_1}^{n_1}} \frac{\partial^{n_2}}{\partial {x_2}^{n_2}} \cdots \frac{\partial^{n_d}}{\partial {x_d}^{n_d}} \right) (\cdot). \tag{4.3}$$

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

$$f(\mathbf{Lk}) = \sum_{\mathbf{n}} \frac{(\mathbf{Lk} - \mathbf{x})^{\mathbf{n}}}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{x}). \tag{4.4}$$

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 $f_r^w(\mathbf{x})$ as the result of convolving the function, sampled at the scaled lattice points $h\mathbf{Lk}$, with the filter w, defined for the lattice L, we can write

$$f_r^w(\mathbf{x}) = \sum_{\mathbf{k}} f(h\mathbf{Lk}) \cdot w\left(\frac{\mathbf{x} - h\mathbf{Lk}}{h}\right) \tag{4.5}$$

Substituting (4.4) into (4.5) we obtain

$$\begin{aligned} f_r^w(\mathbf{x}) &= \sum_{\mathbf{k}} f(h\mathbf{Lk}) \cdot w\left(\frac{x - h\mathbf{Lk}}{h}\right) \\ &= \sum_{\mathbf{k}} \left[\sum_{\mathbf{n}} \frac{(h\mathbf{Lk} - \mathbf{x})^{\mathbf{n}}}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{x})\right] \cdot w\left(\frac{x - h\mathbf{Lk}}{h}\right) \\ &= \sum_{\mathbf{n}} \left[\sum_{\mathbf{k}} \frac{(h\mathbf{Lk} - \mathbf{x})^{\mathbf{n}}}{\mathbf{n}!} \cdot w\left(\frac{x - h\mathbf{Lk}}{h}\right)\right] D^{\mathbf{n}} f(\mathbf{x}) \\ &= \sum_{\mathbf{n}} a_{\mathbf{n}}^w(\mathbf{x}) \cdot D^{\mathbf{n}} f(\mathbf{x}) \end{aligned} \tag{4.6}$$

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

$$a_{\mathbf{n}}^w(\mathbf{x}) := \sum_{\mathbf{k}} \frac{(h\mathbf{Lk} - \mathbf{x})^{\mathbf{n}}}{\mathbf{n}!} \cdot w\left(\frac{\mathbf{x} - h\mathbf{Lk}}{h}\right) \tag{4.7}$$

This is very similar to what Möller et al. [34] arrived at with their 1D analysis and therefore, following their approach, we also introduce a continuous variable τ. Let $\mathbf{x} = h\mathbf{L}(\mathbf{k}_0 + \tau) : \tau = (\tau_1, \tau_2, \ldots, \tau_d)^{\mathsf{T}}$, and $\forall i, \tau_i \in [0, 1)$, and where $\mathbf{k}_0$ is a lattice site coordinate such that the above condition on τ is satisfied for any $\mathbf{x}$. Rewriting equation (4.7) with the new variable τ and replacing $\mathbf{m} = \mathbf{k} - \mathbf{k}_0$ yields

$$a_{\mathbf{n}}^w(\tau) = \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{m}} (\mathbf{L}(\mathbf{m} - \tau))^{\mathbf{n}} \cdot w(\mathbf{L}(\tau - \mathbf{m})), \tag{4.8}$$

which is very similar to the 1D counterpart [34]. Finally we can rewrite the convolution sum in equation (4.6) as

$$f_r^w(\mathbf{x}) = \sum_{\mathbf{n}} D^{\mathbf{n}} f(\mathbf{x}) \cdot a_{\mathbf{n}}^w(\tau), \tag{4.9}$$

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

4.1.2 Classification

A filter w can be classified based on its *Taylor Coefficients* $a_{\mathbf{n}}^w(\tau)$ given by (4.8). For example, in 3D, an ideal interpolation filter will have $a_{\mathbf{n}}^w(\tau) = 1$ for $\mathbf{n} = \mathbf{0}$ and $a_{\mathbf{n}}^w(\tau) = 0$ for $\mathbf{n} \neq \mathbf{0}$. Likewise, an ideal first derivative filter along x in 3D will have $a_{\mathbf{n}}^w(\tau) = 1$ for $\mathbf{n} = (1, 0, 0)$ and $\mathbf{0}$ otherwise. However, in practical settings, $a_{\mathbf{n}}^w(\tau)$ may not be equal to zero for all combinations of $\mathbf{n}$ 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 *order*, i.e. 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

$$\eta_d^\lambda := \left\{ \mathbf{v} \in \mathbb{N}^d : \|v\|_1 = \lambda \right\}. \tag{4.10}$$

The size of this set, $|\eta_d^\lambda|$ is given by

$$|\eta_d^\lambda| = \binom{d + \lambda - 1}{\lambda}$$

The above formula could be derived easily using combinatorics by treating d as the number of container while λ as the number of similar objects and solving for how many distinct ways the objects could be distributed among the containers. A container in this case is allowed to be empty.

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

$$\eta_d^{[a,b]} := \bigcup_{k=a}^{b} \eta_d^k, \tag{4.11}$$

where $a, b \in \mathbb{N}$ and $0 \leq a \leq b$. Contrary to [34] we define an n-*order filter* (n-OF) and

a *k-error filter* (k-EF) separately. 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. Whereas, if the underlying polynomial degree is higher than n (with an n-OF filter) then the error will be bounded by $O(h^k)$ if the same filter is a k-EF filter, where $h \leq 1$ is the grid spacing. This definition of k-EF is the same as that of Möller et al. [34]. 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 $\mathbf{u} \in \eta_d^{[0,n]}$, which we refer to as a *derivative vector*, a filter w is n-OF if the following is satisfied.

$$a_{\mathbf{n}}^{w}(\tau) = \begin{cases} 0, & \mathbf{n} \in \eta_d^{[0,n]} \text{ and } \mathbf{n} \neq \mathbf{u} \\ 1, & \mathbf{n} = \mathbf{u}. \end{cases} \tag{4.12}$$

In light of this definition, it is easy to see from equation (4.9) that the filter will recover $D^{\mathbf{u}} f(\mathbf{x})$, with $f(\mathbf{x})$ having a maximum polynomial order of n. Therefore, we shall also refer to w as a $\mathbf{u}$-derivative filter. When $\mathbf{u} = \mathbf{0}$, the filter is actually an interpolation filter.

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

For rest of the thesis, 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, $\mathbf{u} \in \eta_3^1$.

Considering equation (4.8) for a discrete filter Δ implies $\tau = \mathbf{0}$. Hence, we drop τ and write the *Taylor Coefficients* as

$$a_{\mathbf{n}}^{\Delta} = \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{m}} (\mathbf{L}\mathbf{m})^{\mathbf{n}} \cdot \Delta\left(\mathbf{L}\left(-\mathbf{m}\right)\right). \tag{4.13}$$

The definition (4.12) is also applicable to any discrete filter.

Möller et al. [34] 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 $a_{\mathbf{u}}^{w}(\tau)$ and $a_{\mathbf{u}}^{\Delta}$ evaluate to 1 for the desired derivative $D^{\mathbf{u}}$. The normalization is performed by simply dividing the filter weights by $a_{\mathbf{u}}^{w}(\tau)$ or $a_{\mathbf{u}}^{\Delta}$. With proper normalization, the asymptotic order of the error in terms of h is given by $O(h^{n+1-\varsigma})$ [34], where n is the order of the filter as defined in (4.12) and $\varsigma = \|\mathbf{u}\|_1$ is the l_1-norm of the vector $\mathbf{u}$. In other

terms, the relationship between EF and OF can be given by

$$\text{EF} = \text{OF} + 1 - \varsigma \tag{4.14}$$

In case of first derivative filters, the error is bounded 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.

4.1.4 Linear System For Designing Discrete Filters for BCC

Equation (4.13) forms a system of linear equations where we set values for $a_{\mathbf{n}}^{\Delta}$ a priori and seek to find the unknowns $\Delta(\mathbf{L}(-\mathbf{m}))$. The two parameters for this system are:

1. Number of equations: This is determined by how many $a_{\mathbf{n}}^{\Delta}$ 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 (4.12)

2. Number of unknowns: This is given by how many different $\mathbf{m}$ vectors, i.e. neighborhood of the filter, we want to restrict the filter to. This parameter therefore decides the support of the discrete filter.

Later in Section 3.3 we will demonstrate our Taylor Series based framework in the light of gradient estimation and how shading is improved while rendering volumetric data in both CC and BCC lattices. While shading, we will interpolate the gradient estimated at the grid points with an appropriate interpolation filter which is typically the *Tricubic B-Spline* for CC and *Quintic Box Spline* [16] for BCC. 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 [12, 15]. Thus, to design a good gradient estimator 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.

- $a_{\mathbf{n}}^{\Delta} = 0, \forall \mathbf{n} \in \eta^{[0,4]}$ and $\mathbf{n} \neq [1,0,0]$
- $a_{\mathbf{n}}^{\Delta} = 1, \mathbf{n} = [1,0,0]$

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 $\mathbf{m}$ such that $\mathbf{L}(-\mathbf{m})$ include all the

BCC lattice points up to fourth-order neighbors (inclusive) we have 35 unknown weights $\Delta(\mathbf{L}(-\mathbf{m}))$. However, the linear system formed with this setting is not full-rank and produces a 5 dimensional solution space with all 5 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 anti-symmetries. We impose similar symmetries on the 5 free variables which reduces the solution space to 1D and the resulting filter shows an overall anti-symmetry 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

$$E = \sum_{\mathbf{n} \in \eta^5} \left(a_{\mathbf{n}}^{\Delta}\right)^2 \tag{4.15}$$

Minimizing this error with respect to the one free variable (filter weight) yields a discrete filter Δ which has 26 non-zero weights. Table C.2d in Appendix C.1.2 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 functional that could be minimized is the l_0-norm of the filter, i.e. the support of the filter. This yields a filter having 16 non-zero weights (*OPT16*, see Table 4.1). 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 4.1. *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 which are further away in Euclidean distance.

4.1.5 Combination of Discrete and Continuous Filter

In this section we present the effectiveness of combining a discrete first derivative filter and a continuous interpolation filter to approximate first derivatives globally. Following Möller et al [34], 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.

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 non-zero weights. The first row shows the BCC 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 in Appendix C.

Table 4.1: Taylor Series filters

Consider a discrete filter $\Delta[\mathbf{k}]$, which is given by Equation (4.16), and a continuous filer w defined on an unscaled lattice defined by the matrix $\mathbf{L}$.

$$\Delta[\mathbf{k}] = \Delta(\mathbf{L}\mathbf{k}) \tag{4.16}$$

Note how we use the notation $\Delta[\mathbf{k}]$ with square braces to denote discrete values. With this notation we will develop a continuous filter by convolving these two filters which we will denote as Ψ and this can be given by [1]:

$$\Psi(\mathbf{x}) = (\Delta * w)(\mathbf{x}) = \sum_{\mathbf{k}} \Delta[\mathbf{k}] \cdot w(\mathbf{x} - \mathbf{L}\mathbf{k}) \tag{4.17}$$

[1]Note both Δ and w are filters and hence their convolution can be performed on an unscaled lattice.

Writing the *Taylor Coefficient* of Ψ and substituting Equation (4.17) yields

$$\begin{aligned}
a_{\mathbf{n}}^{\Psi}(\tau) &= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{m}} \left(\mathbf{L}\left(\mathbf{m}-\tau\right)\right)^{\mathbf{n}} \cdot \Psi\left(\mathbf{L}\left(\tau-\mathbf{m}\right)\right) \\
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{m}} \left(\mathbf{L}\left(\mathbf{m}-\tau\right)\right)^{\mathbf{n}} \cdot \sum_{\mathbf{k}} \Delta[\mathbf{k}] \cdot w\left(\mathbf{L}\left(\tau-\mathbf{m}-\mathbf{k}\right)\right) \\
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \sum_{\mathbf{m}} \left(\mathbf{L}\left(\mathbf{m}-\tau\right)\right)^{\mathbf{n}} \cdot w\left(\mathbf{L}\left(\tau-\mathbf{m}-\mathbf{k}\right)\right)
\end{aligned}$$

Taking $\mathbf{p} = \mathbf{m} + \mathbf{k}$

$$\begin{aligned}
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \sum_{\mathbf{p}} \left(\mathbf{L}\left(\mathbf{p}-\mathbf{k}-\tau\right)\right)^{\mathbf{n}} \cdot w\left(\mathbf{L}\left(\tau-\mathbf{p}\right)\right) \\
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \sum_{\mathbf{p}} \left(\mathbf{L}\left(\mathbf{p}-\tau\right)+\mathbf{L}\left(-\mathbf{k}\right)\right)^{\mathbf{n}} \cdot w\left(\mathbf{L}\left(\tau-\mathbf{p}\right)\right) \\
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \sum_{\mathbf{p}} \left\{ \sum_{\mathbf{i}} \binom{\mathbf{n}}{\mathbf{i}} \left(\mathbf{L}\left(\mathbf{p}-\tau\right)\right)^{\mathbf{i}} \left(\mathbf{L}\left(-\mathbf{k}\right)\right)^{\mathbf{n}-\mathbf{i}} \right\} \cdot w\left(\mathbf{L}\left(\tau-\mathbf{p}\right)\right)
\end{aligned}$$

After re-arranging

$$= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{i}} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \left(\mathbf{L}\left(-\mathbf{k}\right)\right)^{\mathbf{n}-\mathbf{i}} \binom{\mathbf{n}}{\mathbf{i}} \sum_{\mathbf{p}} \left(\mathbf{L}\left(\mathbf{p}-\tau\right)\right)^{\mathbf{i}} \cdot w\left(\mathbf{L}\left(\tau-\mathbf{p}\right)\right)$$

Using Equation (4.8)

$$\begin{aligned}
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{i}} \sum_{\mathbf{k}} \Delta[\mathbf{k}] \left(\mathbf{L}\left(-\mathbf{k}\right)\right)^{\mathbf{n}-\mathbf{i}} \binom{\mathbf{n}}{\mathbf{i}} \left\{ \frac{\mathbf{i}!}{h^{\mathbf{i}}} \cdot a_{\mathbf{i}}^{w}(\tau) \right\} \\
&= \frac{h^{\mathbf{n}}}{\mathbf{n}!} \sum_{\mathbf{i}} \left\{ \frac{(\mathbf{n}-\mathbf{i})!}{h^{\mathbf{n}-\mathbf{i}}} \cdot a_{\mathbf{n}-\mathbf{i}}^{\Delta}(\mathbf{0}) \right\} \binom{\mathbf{n}}{\mathbf{i}} \left\{ \frac{\mathbf{i}!}{h^{\mathbf{i}}} \cdot a_{\mathbf{i}}^{w}(\tau) \right\} \\
&= \sum_{\mathbf{i}} a_{\mathbf{n}-\mathbf{i}}^{\Delta}(\mathbf{0}) \cdot a_{\mathbf{i}}^{w}(\tau)
\end{aligned} \tag{4.18}$$

Dropping the $\mathbf{0}$ from the notation for the discrete filter we can compactly write the above as

$$a_{\mathbf{n}}^{\Psi}(\tau) = \sum_{0 \leq \mathbf{i} \leq \mathbf{n}} a_{\mathbf{n}-\mathbf{i}}^{\Delta} \cdot a_{\mathbf{i}}^{w}(\tau), \tag{4.19}$$

where $\mathbf{i}$ is the index vector such that $\forall k, 0 \leq i_k \leq n_k$. Möller et al. [33] showed a similar convolution relationship in 1D.

Further, we analyze the polynomial order (OF) of the combined filter which leads to the following Theorem 4.1.1.

Theorem 4.1.1 *Given a discrete n-OF derivative, an ϵ-OF continuous interpolation filter and a required derivative order $\varsigma = \|\mathbf{u}\|_1$, (see Equation (4.12) for $\mathbf{u}$) the order (OF) of the combined filter is given by*

$$\min(\epsilon + \varsigma, n) \tag{4.20}$$

The proof of this theorem is provided in the Appendix B.

A direct consequence of the above Theorem 4.1.1 is that having an interpolation filter set with a polynomial order (OF) of ϵ, a discrete derivate filter of polynomial order (OF) anything higher than $\epsilon + \varsigma$ does not improve the overall polynomial order and hence is not very useful.

4.2 Results and Discussion

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 derivatives to be computed on the fly without storing them beforehand.

In this section we will show the effectiveness of the filters generated from our proposed framework in the light of gradient estimation and thereby shading iso-surfaces and volume rendered images. Along this line, we will compare estimation qualities, both qualitatively and quantitatively, between CC and BCC lattices.

The gradient filters presented in Table 4.1 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 1. We also used the *Multi-dimensional Discrete Fourier Transform* (MDFT), proposed by Alim and Möller [4], to prefilter scalar data when interpolating with 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. In most visualization applications (for example ray-tracing), sampling is performed in a sequential manner and often the sampling step is

Algorithm 1 Compute the gradient at an arbitrary point $\mathbf{x}$, 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: $\mathbf{x}, w, \Delta_x, \Delta_y$, and Δ_z
Ensure: $\mathbf{v}$ is the gradient at $\mathbf{x}$
1: $G \leftarrow \emptyset$
2: **for all** $\{\mathbf{k}:$ Lattice sites that are within the support of the interpolation filter $w\}$ **do**
3: $\quad \Omega \leftarrow \{$All the data within the support of Δ_x, Δ_y, and Δ_z, centered at $\mathbf{k}\}$
4: $\quad \mathbf{g} \leftarrow$ Compute the lattice site gradient at $\mathbf{k}$ using $\Omega, \Delta_x, \Delta_y$, and Δ_z
5: $\quad G \leftarrow G \cup \mathbf{g}$
6: **end for**
7: $\mathbf{v} \leftarrow$ Compute the gradient using the set G and the filter w
8: **return** $\mathbf{v}$

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.

4.2.1 Implementation

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:

- **Iso-Surface Rendering (ISR)**: A given iso-surface is extracted along a ray in the volume using a Linear Bisection technique and once the iso-surface is found, the gradient is estimated at that point and shaded accordingly. Only scalar interpolation is performed during the iso-surface 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 non-zero.

We implemented our volume ray-caster as a single threaded application and ran all our experiments on an Intel Core$^{\text{TM}}$2 Duo (2.40GHz on each core) machine with 4GB 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`). All reported timing data are obtained using these codes.

4.2.2 Synthetic Data

We used the popular synthetic function proposed by Marschner and Lobb (ML) [31]. 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 needs 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 outwards with increasing isovalue (Figure 4.1). The resulting function can be written in Cartesian coordinates as

$$f_{\mathsf{test}}(\mathbf{x}) := \gamma\|\mathbf{x}\| - \alpha\cos(2\pi f_m \frac{x_3}{\|\mathbf{x}\|}), \tag{4.21}$$

where $\mathbf{x} \in \mathbb{R}^3$ ($\mathbf{x} \neq 0$) 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 $\mathbf{x} \to \mathbf{0}$, 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 [31] 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 Figure 4.1 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. 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.

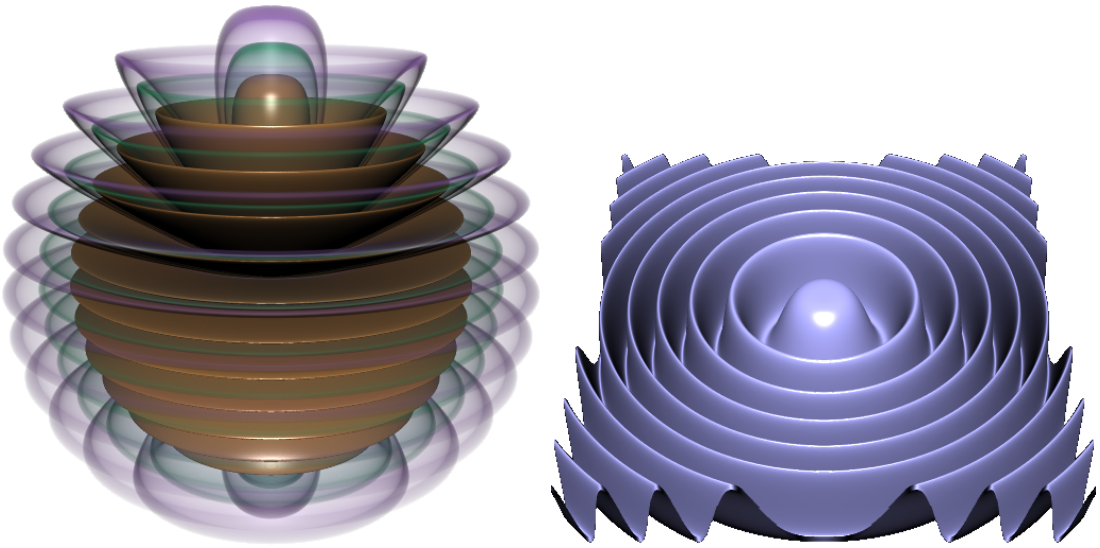
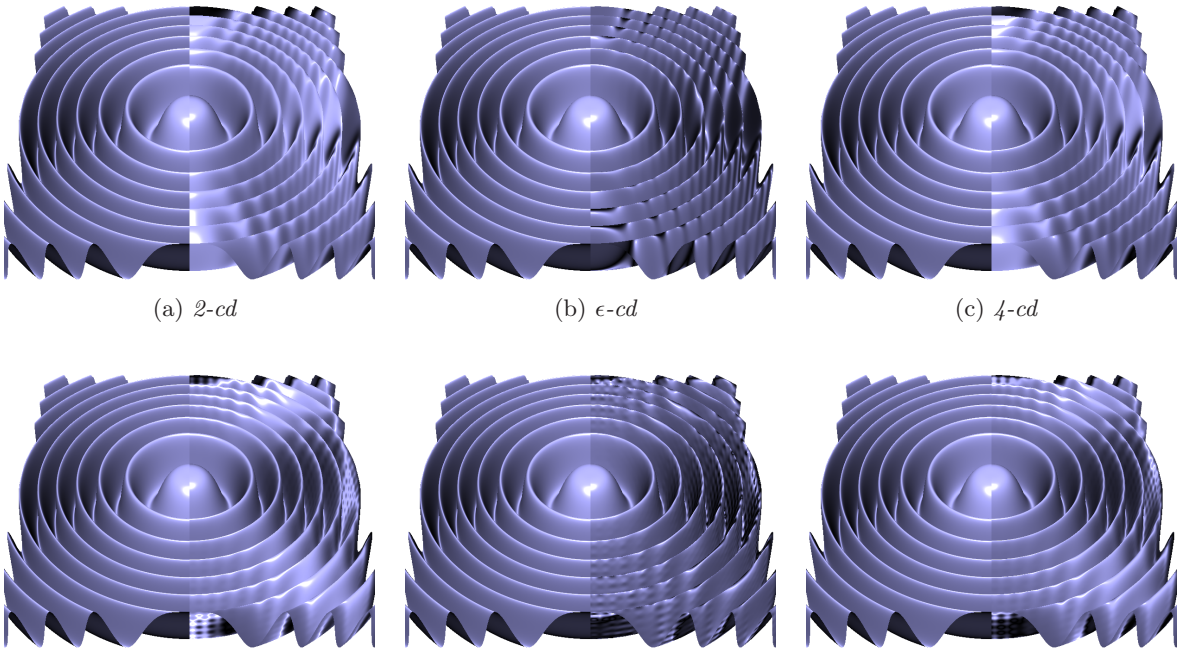


Figure 4.1: Isosurfaces of the unsampled synthetic functions. Left, the modified test function, $\alpha = 0.25$, $\gamma = 2$ and $f_m = 6$, showing the isovalues 0.4 (rendered opaque), 0.5 (green) and 0.6 (purple). Right, isovalue = 0.5 of the ML test function with $f_m = 6$ and $\alpha = 0.25$ as used in [31].

Figure 4.2 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.

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. Beside numerical accuracy, we also measured the time taken to estimate gradients. The resulting data is tabulated in Table 4.2 and some of the isosurface renderings of



(a) *2-cd* (b) ϵ-*cd* (c) *4-cd*

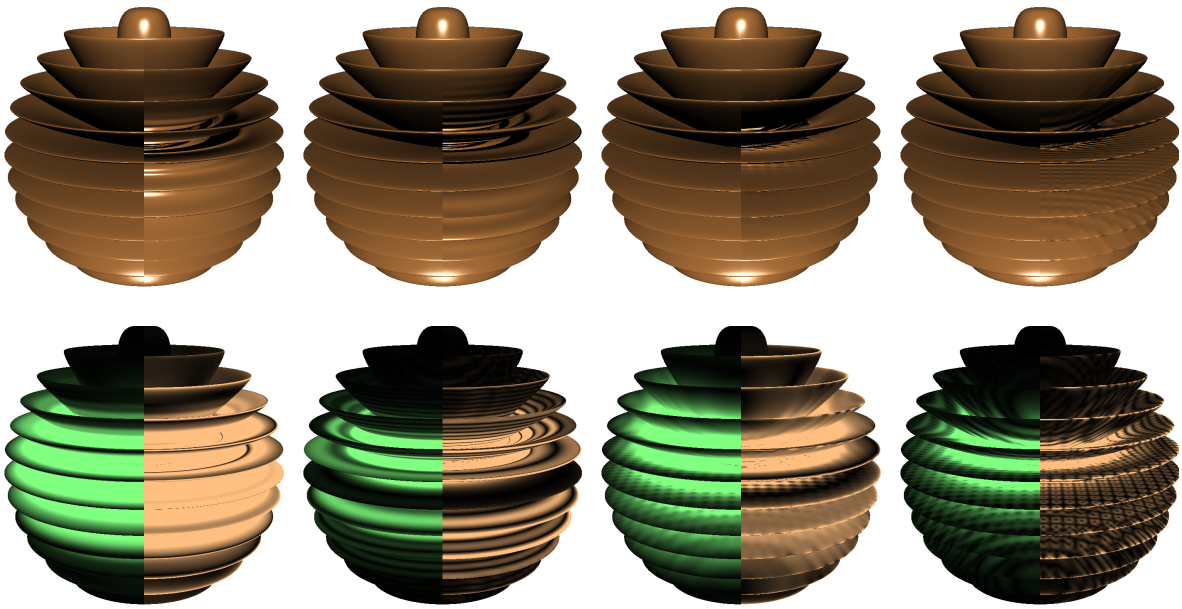
(d) *BCD* (e) ϵ-*CD* (f) *OPT16*

Figure 4.2: 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 is used for normal estimation. To facilitate comparison, the left half of each image shows the truth. For (b) and (e), $\epsilon = 0.003$.

f_{test} along with the error distributions are shown in Figure 4.3. Please 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 Figure 4.2; epsilon central differencing is better than the second and fourth order Taylor filters.

Timing results show run time increases with increasing filter order (OF). Since most of the time in ISR mode is spent evaluating the analytic functions, the total run-times of all the experiments 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



(a) *4-cd* (b) ϵ-*cd* (c) *OPT16* (d) ϵ-*CD*

Figure 4.3: 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), $\epsilon = 0.005$.

sometimes, to that of tricubic B-Spline on Intel Core$^{\text{TM}}$2 Duo. But on AMD Opteron$^{\text{TM}}$, 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 thesis.

4.2.3 Real Data

To assess the practical impact of our filters on the visualization of volumetric data, we rendered isosurface images of the carp 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. Figure 4.4 depicts the original high resolution carp's skull reconstructed with prefiltered tricubic B-spline interpolation

	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

(a) CC

	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

(b) BCC

Table 4.2: RMS length of the error vector (l) and RMS angular deviation (θ in degrees) on the visible isosurface. Normal computation time (in seconds) vs. 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.

and shaded using a very accurate *Orthogonal Projection* (OP) filter namely *cc*, as proposed by [21], in conjunction with the tricubic B-spline. This was done to generate a ground truth image because the OP framework was the best known method to produce the most accurate, yet smooth, shading of sample data at the time of writing of this thesis. However filters generated by the OP framework are typically large and can only be practically implemented in the frequency domain.

The first row of Figure 4.5 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 quality of filters is of paramount importance with volumes given in lower resolution otherwise even a mediocre filter would suffice for high resolution volumes. Futhermore, large volumes are often downsampled for practical purposes. The images follow

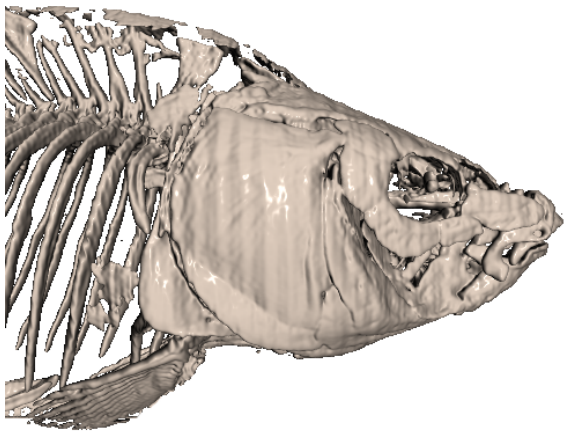
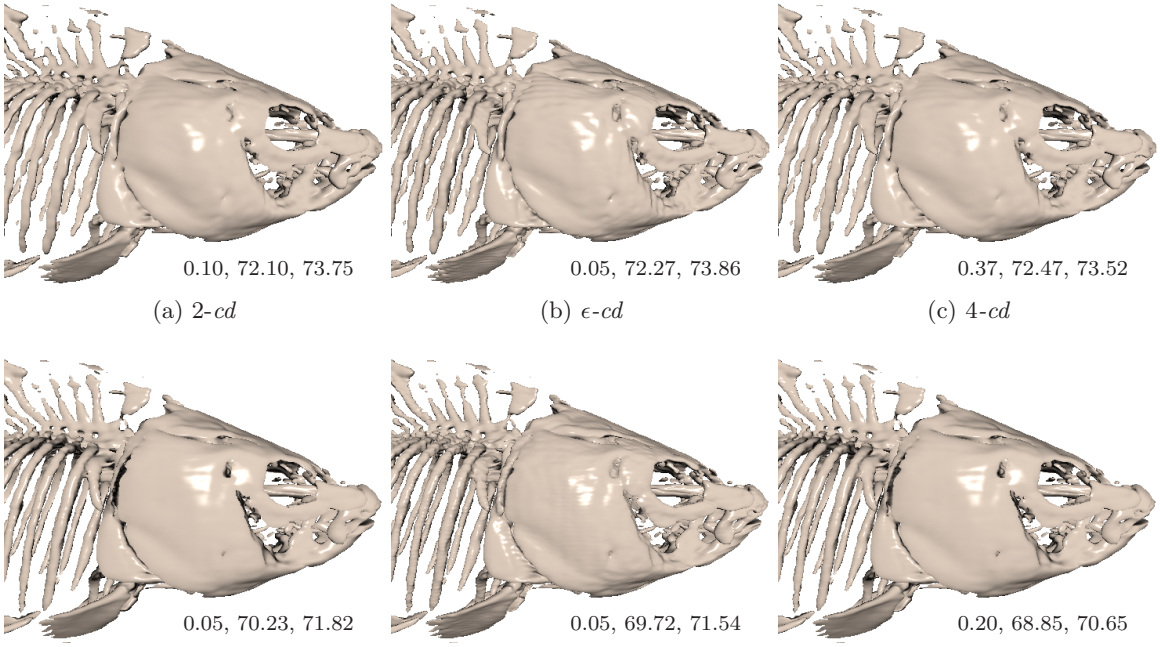


Figure 4.4: An Isosurface of the high resolution carp fish data set.

the same trend as that in Figure 4.2. 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.

In a typical ISR setting, the majority of the time is spent performing scalar interpolations and this fact is reaffirmed in the timings of Figure 4.5. Note that the gradient estimation time for ISR is very small compared to the overall time and is therefore susceptible to time measurement noises.

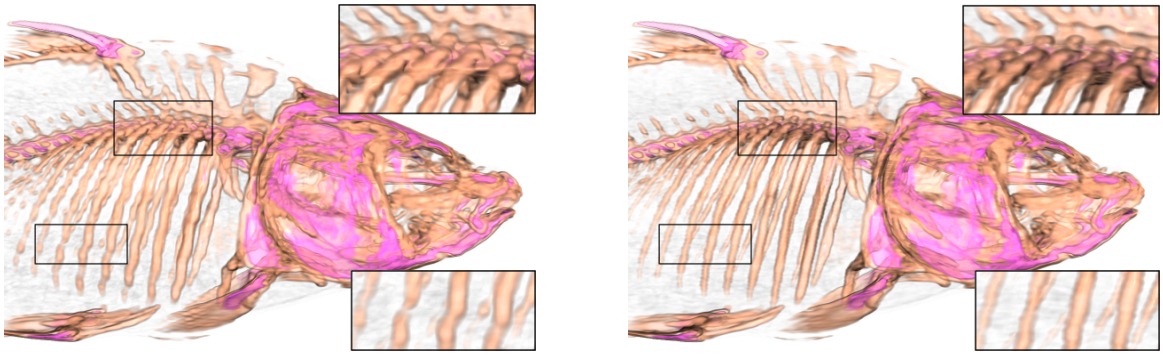
We also rendered DVR images, Figure 4.6, of the carp dataset to study the visual artifacts and the run time of different gradient estimation techniques combined with different scalar interpolations on CC and BCC. For these images, we used the same data set as Figure 4.5. The zoomed-in insets of Figure 4.6 clearly show that BCC renditions are visually superior in terms of scalar interpolation as the bones were reconstructed better. Likewise, in terms of gradient estimation, *OPT16* in BCC produces better contrast compared to the CC counterpart, *4-cd*. The time measurements in Figure 4.6 show that the gradient estimation with *OPT16*, for example, is slower than that of *4-cd* by a factor of about 1.04.



(a) 2-*cd* (b) ϵ-*cd* (c) 4-*cd*

(d) *BCD* (e) ϵ-*CD* (f) *OPT16*

Figure 4.5: Carp data set downsampled to a $160 \times 160 \times 160$ CC grid (a-c) and a $126 \times 126 \times 252$ BCC grid (d-f) 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) indicates 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) 4-*cd* (56.35, 69.89, 129.00) (b) *OPT16* (58.75, 68.38, 130.22)

Figure 4.6: DVR images of the downsampled carp CC (a) and BCC (b) datasets (prefiltered for the respective interpolation filters) rendered at a resolution of 600×390 pixels. Normal computation time, scalar interpolation time and the total time (all in seconds) are indicated.

Chapter 5

Conclusion

Often times, the salient features in 3D data are not easily detectable by volume visualization methods, especially those based on simple transfer function designs. Therefore, there is often a need to pre-process data in a reliable manner so that the salient features are preserved. Such pre-processing methods are often based on a smoothing or anisotropic diffusion framework, which requires laborious parameter tuning.

In this thesis we have presented a novel anisotropic diffusion model for 3D scalar data to address these issues. We have used an intuitive definition for edges based on the directional second derivative along the gradient. This led us to the design of a PDE with a *stopping function* that is much less sensitive to its parameter σ and smoothes data while preserving edges. We have shown that with this new *stopping function* diffusion is performed consistently on an iso-surface regardless of the gradient magnitude, which is in contrast to previous methods, like KM. Even more so, our proposed diffusion model remains second order and much simpler to implement, unlike higher order and existing de-noising PDEs.

On the other hand, our results demonstrate some remarkable de-noising properties of the proposed diffusion model. On this end, we have compared our results with a recent PDE based de-noising technique on five different noise types.

With such consistent edge preserving smoothing and de-noising properties our diffusion model has great utility in the context of visualization. We demonstrated the effect of our diffusion model on the quality of the rendered images using a variety of datasets, including both synthetic and real. Further, we showed its impact on multi-dimensional histograms, which are the basis of many volume rendering algorithms. Specifically, we are able to recover the arc patterns in the histogram even in the presence of strong noise. In the noiseless case,

our diffusion has been found to still enhance these histograms without re-tuning a parameter for each new dataset. This will eventually let the practitioner use our diffusion model as a general purpose smoothing and de-noising tool without destroying salient features or tuning a parameter.

When volumetric data are given in other types of lattices, for example BCC, practitioners need to apply de-noising and other PDE methods directly on the given lattice without having to convert them to CC. Along this line we have developed a general framework for designing derivative estimation filters on arbitrary lattices whereby users can specify polynomial orders along with filter sizes. These filters are compact and often small in size which are suitable for solving PDEs. With this framework in hand, we can now implement our proposed anisotropic diffusion and many other PDEs on regular lattices.

Finally, we conclude this thesis by providing all the filters for estimating first and second derivatives for both CC and BCC lattices in Appendix C.

Appendix A

Formulae

A.1 Co-factor matrix of the Hessian

Given a 3D scalar function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ we will use the notation f_{xy} to denote the partial derivative $\frac{\partial}{\partial x}\frac{\partial f}{\partial y}$ and the notation f_{xx} to denote the second derivative along x, i.e. $\frac{\partial^2 f}{\partial x^2}$ and so forth. Using these notations a Hessian matrix $\mathbf{H}$ can be written as the following:

$$\mathbf{H} = \begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{xy} & f_{yy} & f_{yz} \\ f_{xz} & f_{yz} & f_{zz} \end{bmatrix}$$

The co-factor of the Hessian, denoted by $\mathbf{H}_c$, is another matrix where every element is replaced by the co-factor of the corresponding element in $\mathbf{H}$, and the simplification is given below:

$$\mathbf{H}_c = \begin{pmatrix} f_{yy}f_{zz} - f_{yz}f_{yz} & f_{yz}f_{xz} - f_{xy}f_{zz} & f_{xy}f_{yz} - f_{yy}f_{xz} \\ f_{xz}f_{yz} - f_{xy}f_{zz} & f_{xx}f_{zz} - f_{xz}f_{xz} & f_{xy}f_{xz} - f_{xx}f_{yz} \\ f_{xy}f_{yz} - f_{xz}f_{yy} & f_{xy}f_{xz} - f_{xx}f_{yz} & f_{xx}f_{yy} - f_{xy}f_{xy} \end{pmatrix}$$

A.2 Gaussian and Mean Curvature

Gaussian Curvature G is given by the following

$$G = \frac{1}{\|\nabla f\|^4}\Big[f_x^2\left(f_{yy}f_{zz} - f_{yz}^2\right) + 2f_yf_z\left(f_{xz}f_{xy} - f_{xx}f_{yz}\right) + f_y^2\left(f_{xx}f_{zz} - f_{xz}^2\right) + 2f_xf_z\left(f_{yz}f_{xy} - f_{yy}f_{xz}\right) + f_z^2\left(f_{xx}f_{yy} - f_{xy}^2\right) + 2f_xf_y\left(f_{xz}f_{yz} - f_{zz}f_{xy}\right)\Big]$$

and the Mean Curvature K is given by

$$K = \frac{1}{2\|\nabla f\|^3}\Big[2f_yf_zf_{yz} - f_x^2\left(f_{yy} + f_{zz}\right) + 2f_xf_zf_{xz} - f_y^2\left(f_{xx} + f_{zz}\right) + 2f_xf_yf_{xy} - f_z^2\left(f_{xx} + f_{yy}\right)\Big]$$

where the gradient magnitude is $\|\nabla f\| = \sqrt{f_x^2 + f_y^2 + f_z^2}$.

Appendix B

Polynomial Order of Combined Filter

Lemma B.0.1 *Given two non-negative vectors* $\mathbf{u}$ *and* $\mathbf{v}$*, i.e.* $\forall i, u_i, v_i \geq 0$*, that satisfies* $\mathbf{u} \geq \mathbf{v}$*, i.e.* $\forall i, u_i \geq v_i$ *then*

$$\|\mathbf{u} - \mathbf{v}\|_1 = \|\mathbf{u}\|_1 - \|\mathbf{v}\|_1 \tag{B.1}$$

Proof

$$\begin{aligned} &\|\mathbf{u} - \mathbf{v}\|_1 \\ =&\sum_{i=1}^{d} |u_i - v_i|, \quad d = \text{ dimensions} \end{aligned}$$

But $\forall i, u_i \geq v_i$

$$\begin{aligned} =&\sum_{i=1}^{d} (u_i - v_i) \\ =&\sum_{i=1}^{d} u_i - \sum_{i=1}^{d} v_i \end{aligned}$$

Since $\forall i, u_i, v_i \geq 0$

$$
\begin{aligned}
&= \sum_{i=1}^{d} |u_i| - \sum_{i=1}^{d} |v_i| \\
&= \|\mathbf{u}\|_1 - \|\mathbf{v}\|_1 \qquad \blacksquare
\end{aligned}
$$

Theorem B.0.2 *Given a discrete n-OF derivative, an ϵ-OF continuous interpolation filter and a required derivative order $\varsigma = \|\mathbf{u}\|_1$, see Equation* (4.12) *for* $\mathbf{u}$*, the order (OF) of the combined filter is given by*

$$\min(\epsilon + \varsigma, n) \tag{B.2}$$

Proof Lets assume a discrete n-OF derivative filter Δ and a continuous ϵ-OF interpolation filter w. Also, lets assume that $\varsigma = \|\mathbf{u}\|_1 \leq n$ otherwise the discrete filter Δ would not be able to recover the derivative with a lower polynomial order (OF).

Therefore, according to Equation (4.12) the following must be satisfied for the filter w.

$$a_{\mathbf{v}}^{w}(\tau) = \begin{cases} 0, & \|\mathbf{v}\|_1 \leq \epsilon, \text{ and } \mathbf{v} \neq 0 \\ 1, & \mathbf{v} = 0. \end{cases} \tag{B.3}$$

Similarly the following relationship must hold for the discrete derivative filter Δ:

$$a_{\mathbf{v}}^{\Delta} = \begin{cases} 0, & \|\mathbf{v}\|_1 \leq n, \text{ and } \mathbf{v} \neq \mathbf{u} \\ 1, & \mathbf{v} = \mathbf{u} \text{ and } \varsigma = \|\mathbf{u}\|_1 \leq n \end{cases} \tag{B.4}$$

Now we write the convolution relationship as given by the Equation (4.19) for the combined filter $\Psi = w * \Delta$:

$$a_{\mathbf{x}}^{\Psi}(\tau) = \sum_{0 \leq \mathbf{i} \leq \mathbf{x}} a_{\mathbf{x}-\mathbf{i}}^{\Delta} \cdot a_{\mathbf{i}}^{w}(\tau), \quad \mathbf{x} \in \mathbb{N}^d$$

substituting $\mathbf{k} = \mathbf{x} - \mathbf{i}$ and re-writing we have

$$a_{\mathbf{x}}^{\Psi}(\tau) = \sum_{0 \leq \mathbf{k} \leq \mathbf{x}} a_{\mathbf{k}}^{\Delta} \cdot a_{\mathbf{x}-\mathbf{k}}^{w}(\tau), \quad \mathbf{x} \in \mathbb{N}^d \tag{B.5}$$

To analyze $a_{\mathbf{x}}^{\Psi}(\tau)$ for each $\mathbf{x}$ we will use the following three cases. For all the cases we will

assume that all the *Taylor Coefficients*, i.e. $a^w_{\mathbf{x}}(\tau)$ and $a^{\Delta}_{\mathbf{x}}$, are positive so as to create worst possible scenario by avoiding cancellation in the summations.

- **Case 1:** $\|\mathbf{x}\|_1 < \varsigma \leq n$
  According to Equation (B.4) the terms $a^{\Delta}_{\mathbf{k}}$ in Equation (B.5) will all be 0. Hence, for this case, $a^{\Psi}_{\mathbf{x}}(\tau) = 0$.

- **Case 2:** $\|\mathbf{x}\|_1 = \varsigma \leq n$
  Equation (B.5) could be written as:

$$a^{\Psi}_{\mathbf{x}}(\tau) = a^{\Delta}_{\mathbf{x}} \cdot a^w_0(\tau) + \sum_{0 \leq \mathbf{k} < \mathbf{x}} a^{\Delta}_{\mathbf{k}} \cdot a^w_{\mathbf{x}-\mathbf{k}}(\tau)$$

  Similarly to Case 1, for $\mathbf{0} \leq \mathbf{k} < \mathbf{x}$ where $\|\mathbf{x}\|_1 = \varsigma$ the terms $a^{\Delta}_{\mathbf{k}} = 0$ and therefore

$$a^{\Psi}_{\mathbf{u}}(\tau) = a^{\Delta}_{\mathbf{u}} \cdot a^w_0(\tau)$$

  Using Equation (B.3), $a^w_0(\tau) = 1$ and so we have

$$a^{\Psi}_{\mathbf{u}}(\tau) = a^{\Delta}_{\mathbf{x}} \tag{B.6}$$

  Now, using Equation (B.4)

$$a^{\Psi}_{\mathbf{x}}(\tau) = \begin{cases} 1 & \mathbf{x} = \mathbf{u} \\ 0 & \text{otherwise} \end{cases} \tag{B.7}$$

  Therefore, combined filter Ψ recovers the derivative $\mathbf{u}$ and $\mathbf{u}$ only.

- **Case 3:** $\varsigma < \|\mathbf{x}\|_1 \leq n$
  Note that for this case we are assuming $\varsigma < n$ while the situation $\varsigma = n$ is already handled in the **Case 2**.

  Assuming $\mathbf{u} < \mathbf{x}$, which is the worst case in this scenario, Equation (B.5) could be

written as:

$$a^{\Psi}_{\mathbf{x}}(\tau) = a^{\Delta}_{\mathbf{u}} \cdot a^{w}_{\mathbf{x}-\mathbf{u}}(\tau) + \sum_{\substack{0\leq\mathbf{k}\leq\mathbf{x}\\ \mathbf{k}\neq\mathbf{u}}} a^{\Delta}_{\mathbf{k}} \cdot a^{w}_{\mathbf{x}-\mathbf{k}}(\tau)$$

Using Equation (B.4), $a^{\Delta}_{\mathbf{u}} = 1$, yields

$$a^{\Psi}_{\mathbf{x}}(\tau) = a^{w}_{\mathbf{x}-\mathbf{u}}(\tau) + \sum_{\substack{0\leq\mathbf{k}\leq\mathbf{x}\\ \mathbf{k}\neq\mathbf{u}}} a^{\Delta}_{\mathbf{k}} \cdot a^{w}_{\mathbf{x}-\mathbf{k}}(\tau)$$

Now, for $\|\mathbf{x}\|_1 \leq n$ and $\mathbf{k} \neq \mathbf{u}$, the terms $a^{\Delta}_{\mathbf{k}}$ will vanish and therefore

$$a^{\Psi}_{\mathbf{x}}(\tau) = a^{w}_{\mathbf{x}-\mathbf{u}}(\tau) \tag{B.8}$$

According to Equation (B.3) the above will be 0 only if

$$0 < \|\mathbf{x} - \mathbf{u}\|_1 \leq \epsilon \tag{B.9}$$

Taking the right hand side of the above inequality (Equation (B.9)) we have

$$\|\mathbf{x} - \mathbf{u}\|_1 \leq \epsilon$$

Using Lemma B.0.1

$$\|\mathbf{x}\|_1 \leq \epsilon + \varsigma, \quad \|\mathbf{u}\|_1 = \varsigma \tag{B.10}$$

On the other hand, the left hand side of the inequality (Equation (B.9)) only reveals that $\|\mathbf{x}\|_1 > \varsigma$ which was the basic premise of this case anyway.

Hence, for this case, the term $a^{\Psi}_{\mathbf{x}}(\tau) = 0$ when $\|\mathbf{x}\|_1 \leq \varsigma + \epsilon$ and $\|\mathbf{x}\|_1 \leq n$. Another way of writing this is

$$\|\mathbf{x}\|_1 \leq \min(\varsigma + \epsilon, n) \tag{B.11}$$

Finally we write the *Taylor Coefficients* of our combined filter $\Psi = w * \Delta$ as:

$$a_{\mathbf{v}}^{\Psi}(\tau) = \begin{cases} 0, & \|\mathbf{v}\|_1 \leq \min(\epsilon + \varsigma, n), \text{ and } \mathbf{v} \neq \mathbf{u} \\ 1, & \mathbf{v} = \mathbf{u}. \end{cases} \tag{B.12}$$

which proves that the polynomial order (OF) of Ψ is $\min(\epsilon + \varsigma, n)$ ∎

We can understand the above theorem more intuitively by considering a sampled polynomial function of degree n. According to the definition of OF (see Section 4.1.2) we can apply our n-OF discrete derivative filter Δ on the samples to recover the exact derivate of the function at the grid points. Since the order of the derivative is ς, the polynomial degree of the derivative will be $n - \varsigma$. Therefore, we only need a continuous filter w of ϵ-OF, where $\epsilon = n - \varsigma$, to interpolate this sampled derivative function perfectly. Since we started out with a sampled polynomial function of degree of n and recovered its derivative perfectly, the overal polynomial order of the combined filter $\Psi = w * \Delta$ will be n or $\epsilon + \varsigma$, whichever is smaller, i.e. $\min(\epsilon + \varsigma, n)$.

Appendix C

Taylor Series Filters Weights

C.1 Filters for First Order Derivatives

The weights are for filters that compute the first order derivative along the x axis. First derivative along other axes can be inferred easily from this.

Note that the filter weights need to be divided by h_x, the scaling factor along x axis, as indicated by each of the tables listed below. For filters along other axes, the weights need to be divided by their corresponding scaling factors, i.e. h_y and h_z.

C.1.1 First Order Derivative Filters for CC

Lattice Coordinates	$h_x \cdot$ Weight
$(1, 0, 0)$	$-1/2$
$(-1, 0, 0)$	$1/2$

(a) 2-*cd* (2-OF, 2-EF)

Lattice Coordinates	$h_x \cdot$ Weight
$(1, 0, 0)$	$-2/3$
$(-1, 0, 0)$	$2/3$
$(2, 0, 0)$	$1/12$
$(-2, 0, 0)$	$-1/12$

(b) 4-*cd* (4-OF,4-EF)

Table C.1: First order derivative filters for CC

C.1.2 First Order Derivative Filters for BCC

Lattice Site Coordinates	$h_x\cdot$ Weight
$(2,0,0)$	$-1/4$
$(-2,0,0)$	$1/4$

(a) *SOCD* (2-OF,2-EF)

Lattice Site Coordinates	$h_x\cdot$ Weight
$(1,\pm1,\pm1)$	$-1/8$
$(-1,\pm1,\pm1)$	$1/8$

(b) *BCD* (2-OF,2-EF)

Lattice Site Coordinates	$h_x\cdot$ Weight
$(1,\pm1,\pm1)$	$-1/6$
$(-1,\pm1,\pm1)$	$1/6$
$(2,\pm2,\pm2)$	$1/48$
$(-2,\pm2,\pm2)$	$-1/48$

(c) *OPT16* (4-OF,4-EF)

Lattice Site Coordinates	$h_x\cdot$ Weight
$(1,\pm1,\pm1),(2,0,0)$	$-1/6$
$(-1,\pm1,\pm1),(-2,0,0)$	$1/6$
$(2,0,-2),(2,-2,0),(2,2,0),(2,0,2)$	$1/32$
$(-2,0,-2),(-2,-2,0),(-2,2,0),(-2,0,2)$	$-1/32$
$(2,\pm2,\pm2)$	$1/192$
$(-2,\pm2,\pm2)$	$-1/192$

(d) *OPT26* (4-OF,4-EF)

Table C.2: First order derivative filters for BCC

C.1.3 Taylor Coefficients Plot for First Derivative Filters

In this section we plot the Taylor Coefficients of each filter and group them by EF. For clarity we only plot for the derivative orders n, where most significant errors are made. Therefore Figure C.1 plots Taylor Coefficients of both CC and BCC filters in the third derivative order.

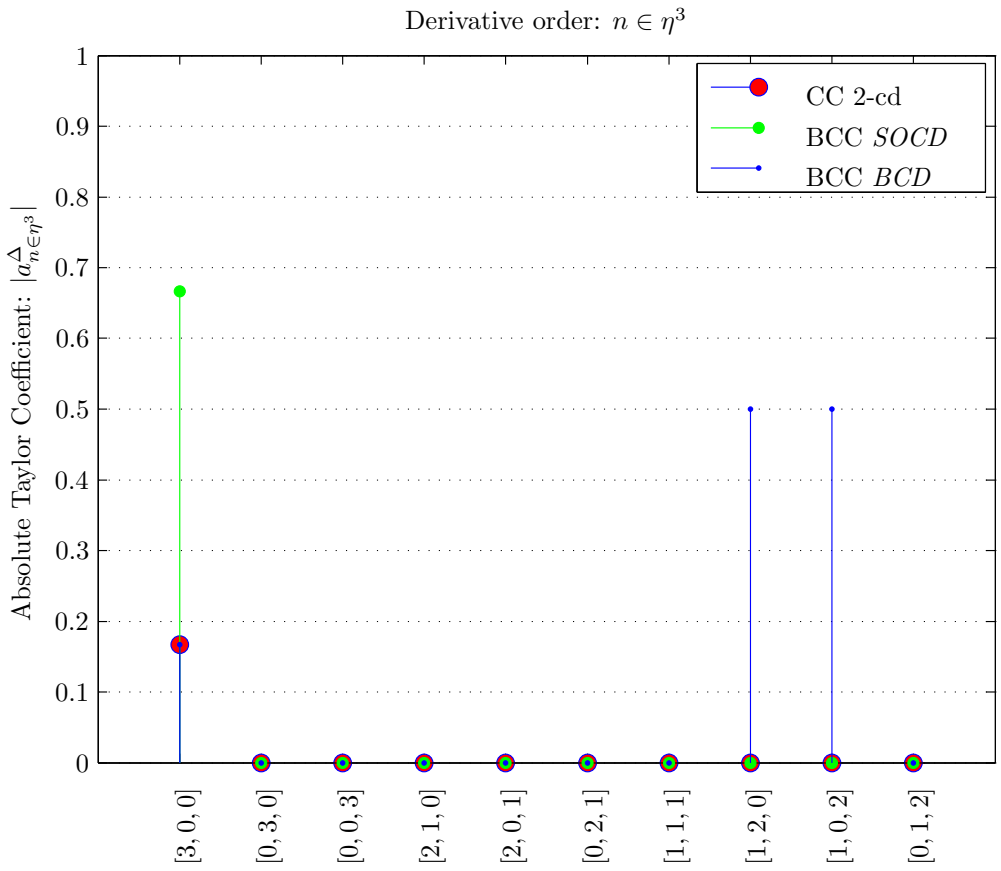


Figure C.1: Plot of absolute Taylor Coefficients in the third derivative ($n \in \eta^3$) order for different 2-EF filters in CC and BCC.

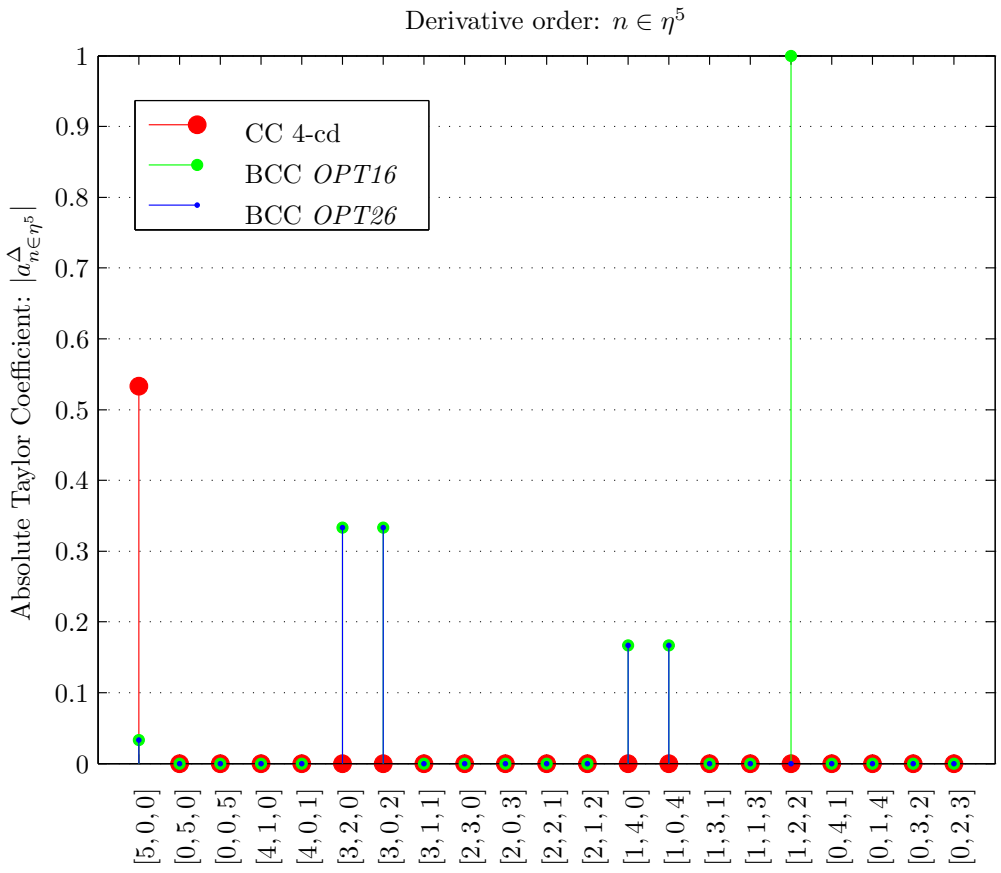


Figure C.2: Plot of absolute Taylor Coefficients in the fifth derivative ($n \in \eta^5$) order for different 4-EF filters in CC and BCC.

C.2 Filters for Second Order Derivatives

In this section we provide filters for estimating second order derivative of f_{xx} and f_{xy}. The other second order derivatives filters, like filters for estimating f_{yy}, f_{zz}, f_{xz} and f_{yz}, can be inferred easily from that of f_{xx} and f_{xy}.

For every filter we also provide a plot of Taylor Coefficients where the most significant errors are made, and h_x, h_y are the scaling factors along x and y axes respectively. Note that the filter weights in this case need to be divided by either h_x^2 or $h_x h_y$ (quadratic order), as indicated by each of the tables listed below.

In this section we will use the following naming convention for every filter.

$$\text{Filter Name} = \text{(Lattice Type)-(Derivative Type)-(EF of the Filter)}$$

Where

$$\begin{aligned} \text{(Lattice Type)} &\in \{\text{CC,BCC}\} \\ \text{(Derivative Type)} &\in \{xx, xy\} \\ \text{(EF of the Filter)} &\in \{\text{2ef,4ef}\} \end{aligned}$$

Therefore a name of CC-xx-2ef refers to a 2-EF filter in CC lattice that estimates the derivative of f_{xx}. Likewise, BCC-xy-4ef refers to a 4-EF filter in BCC lattice that estimates the derivative of f_{xy} and so forth.

C.2.1 Second Order Derivative Filters for CC

Lattice Site Coordinates	$h_x^2\cdot$ Weight
$(\pm 1, 0, 0)$	1
$(0, 0, 0)$	-2

(a) CC-xx-2ef (3-OF,2-EF)

Lattice Site Coordinates	$h_x h_y\cdot$ Weight
$(-1, -1, 0), (1, 1, 0)$	$1/4$
$(1, -1, 0), (-1, 1, 0)$	$-1/4$

(b) CC-xy-2ef (3-OF,2-EF)

Lattice Site Coordinates	$h_x^2\cdot$ Weight
$(\pm 2, 0, 0)$	$-1/12$
$(\pm 1, 0, 0)$	$4/3$
$(0, 0, 0)$	$-5/2$

(c) CC-xx-4ef (5-OF,4-EF)

Lattice Site Coordinates	$h_x h_y\cdot$ Weight
$(-2, -2, 0), (2, 2, 0)$	$1/144$
$(-1, -2, 0), (1, 2, 0)$	$-1/18$
$(1, -2, 0), (-1, 2, 0)$	$1/18$
$(2, -2, 0), (-2, 2, 0)$	$-1/144$
$(-2, -1, 0), (2, 1, 0)$	$-1/18$
$(-1, -1, 0), (1, 1, 0)$	$4/9$
$(1, -1, 0), (-1, 1, 0)$	$-4/9$
$(2, -1, 0), (-2, 1, 0)$	$1/18$

(d) CC-xy-4ef (5-OF,4-EF)

Table C.3: Second order derivative filters for CC

C.2.2 Second Order Derivative Filters for BCC

Lattice Site Coordinates	$h_x^2\cdot$ Weight
$(\pm 2, 0, 0), (2, 2, 0)$	$1/4$
$(0, 0, 0)$	$-1/2$

(a) BCC-xx-2ef (3-OF,2-EF)

Lattice Site Coordinates	$h_x h_y\cdot$ Weight
$(-1, -1, 1), (1, 1, -1), (1, 1, 1), (-1, -1, -1)$	$1/8$
$(1, -1, 1), (-1, 1, -1), (-1, 1, 1), (1, -1, -1)$	$-1/8$

(b) BCC-xy-2ef (3-OF,2-EF)

Lattice Site Coordinates	$h_x^2\cdot$ Weight
$(\pm 4, 0, 0)$	$-1/48$
$(\pm 2, 0, 0)$	$1/3$
$(0, 0, 0)$	$-5/8$

(c) BCC-xx-4ef (5-OF,4-EF)

Lattice Site Coordinates	$h_x h_y\cdot$ Weight
$(-1, -1, 1), (1, 1 - 1), (1, 1, 1), (-1, -1, -1)$	$1/6$
$(1, -1, 1), (-1, 1, -1), (-1, 1, 1), (1, -1, -1)$	$-1/6$
$(-2, -2, 2), (2, 2, -2), (2, 2, 2), (-2, -2, -2)$	$-1/96$
$(2, -2, 2), (-2, 2, -2), (-2, 2, 2), (2, -2, -2)$	$1/96$

(d) BCC-xy-4ef (5-OF,4-EF)

Table C.4: Second order deriviate filters for BCC

C.2.3 Taylor Coefficients Plot for Second Derivative Filters

In this section we plot the Taylor Coefficients of each filter and group them by EF. For clarity we only plot for the derivative orders n, where most significant errors are made. Therefore Figure C.3 plots Taylor Coefficients of both CC and BCC filters in the fourth derivative order. This is because OF of a 2-EF filters will be 3 (see Equation (4.14)) and hence all the errors will be made in the fourth order and beyond. Similarly, all the errors of a 4-EF second derivative filter will be made in the sixth order and beyond.

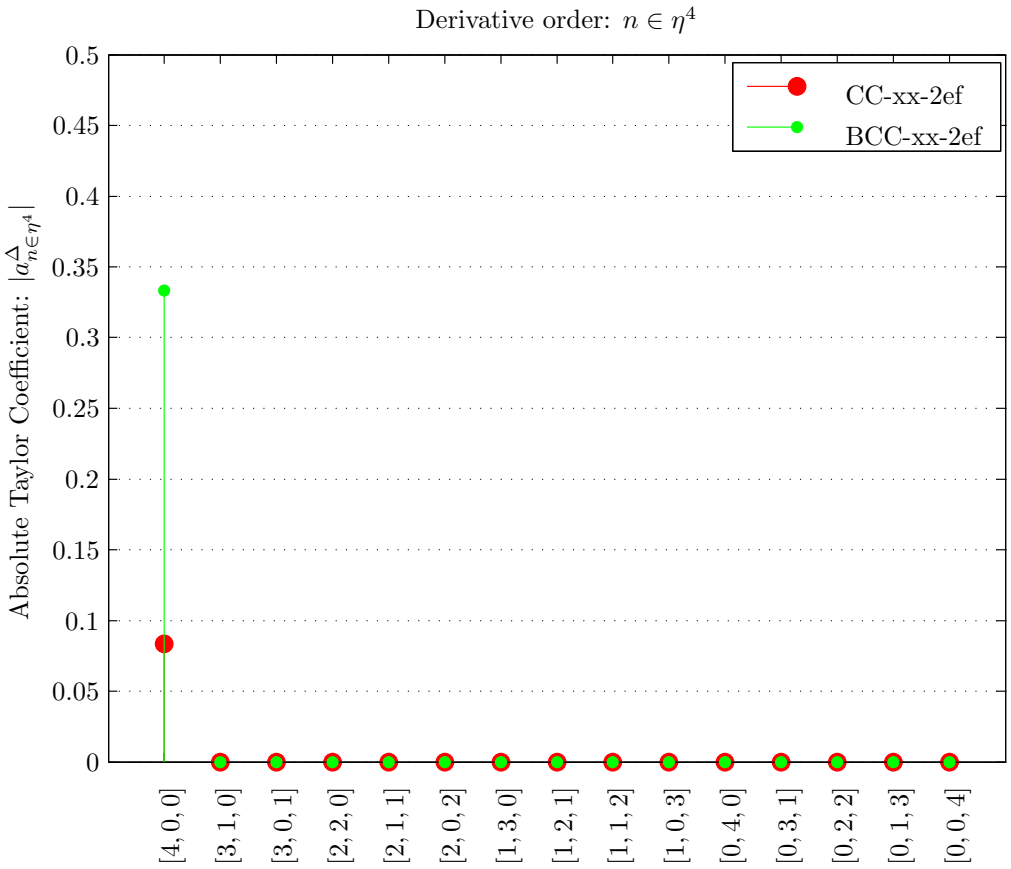


Figure C.3: Plot of absolute Taylor Coefficients in the fourth derivative ($n \in \eta^4$) order for different 2-EF f_{xx} estimating filters in CC and BCC.

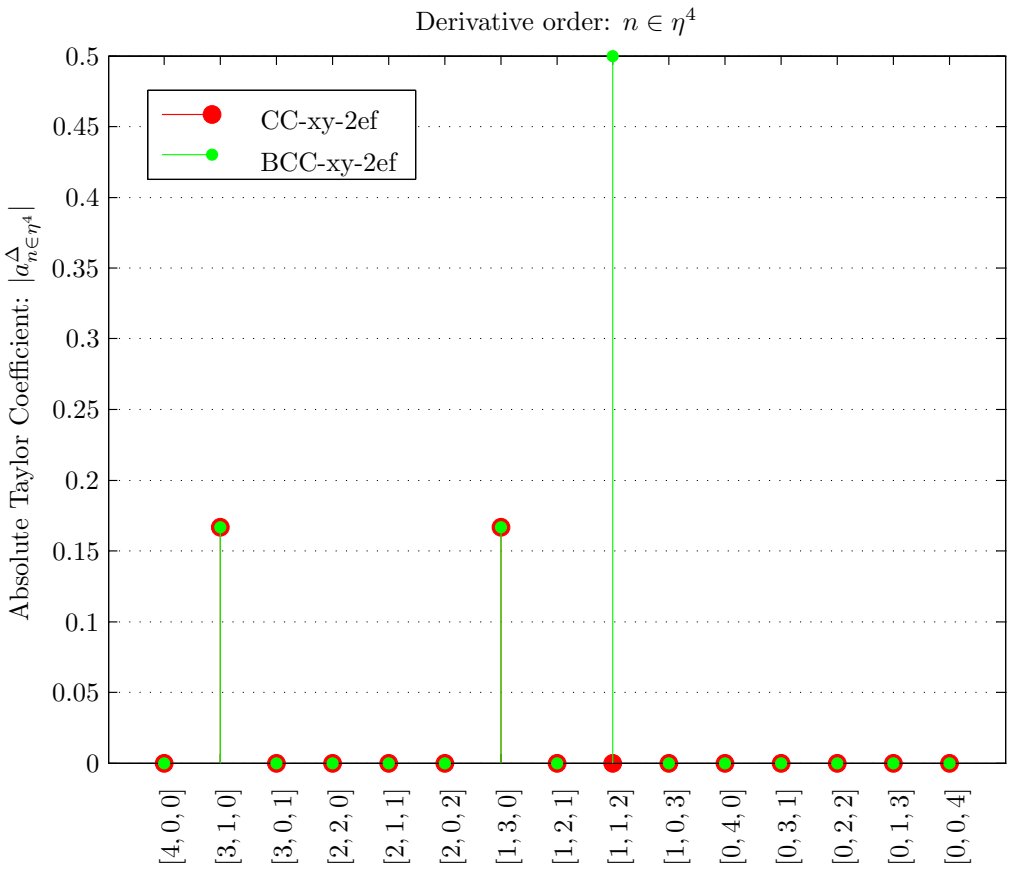


Figure C.4: Plot of absolute Taylor Coefficients in the fourth derivative ($n \in \eta^4$) order for different 2-EF f_{xy} estimating filters in CC and BCC.

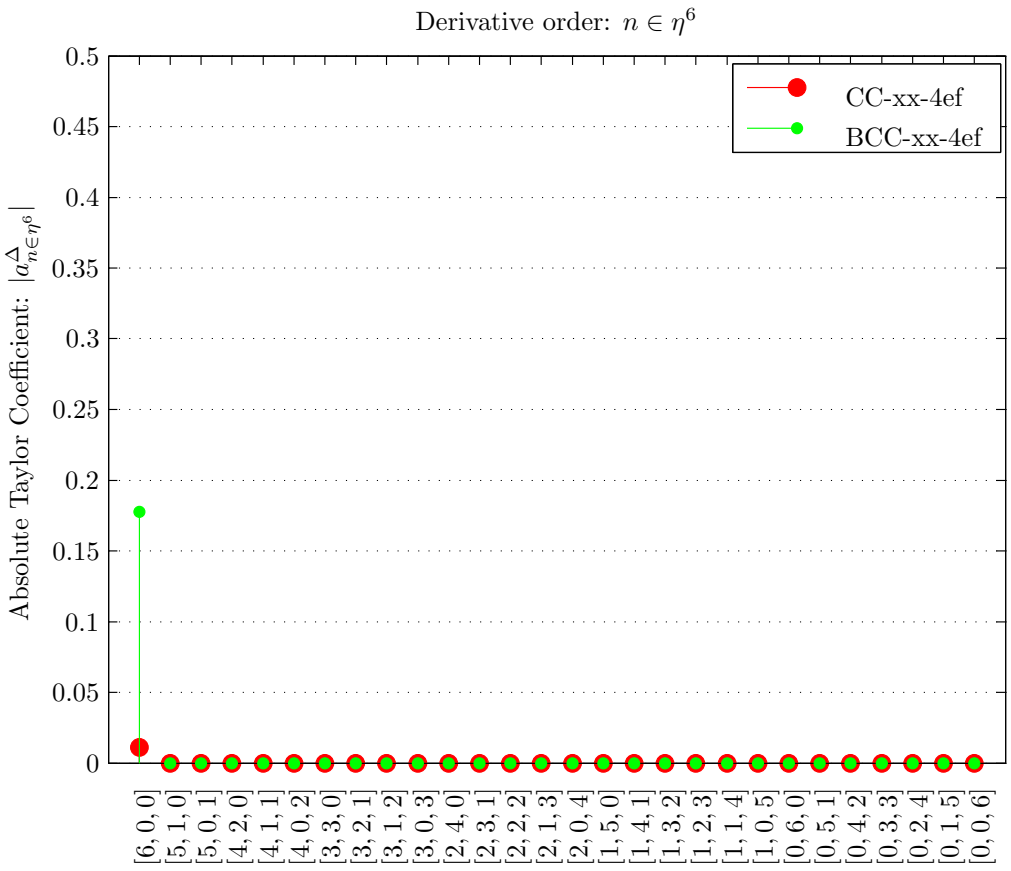


Figure C.5: Plot of absolute Taylor Coefficients in the sixth derivative ($n \in \eta^6$) order for different 4-EF f_{xx} estimating filters in CC and BCC.

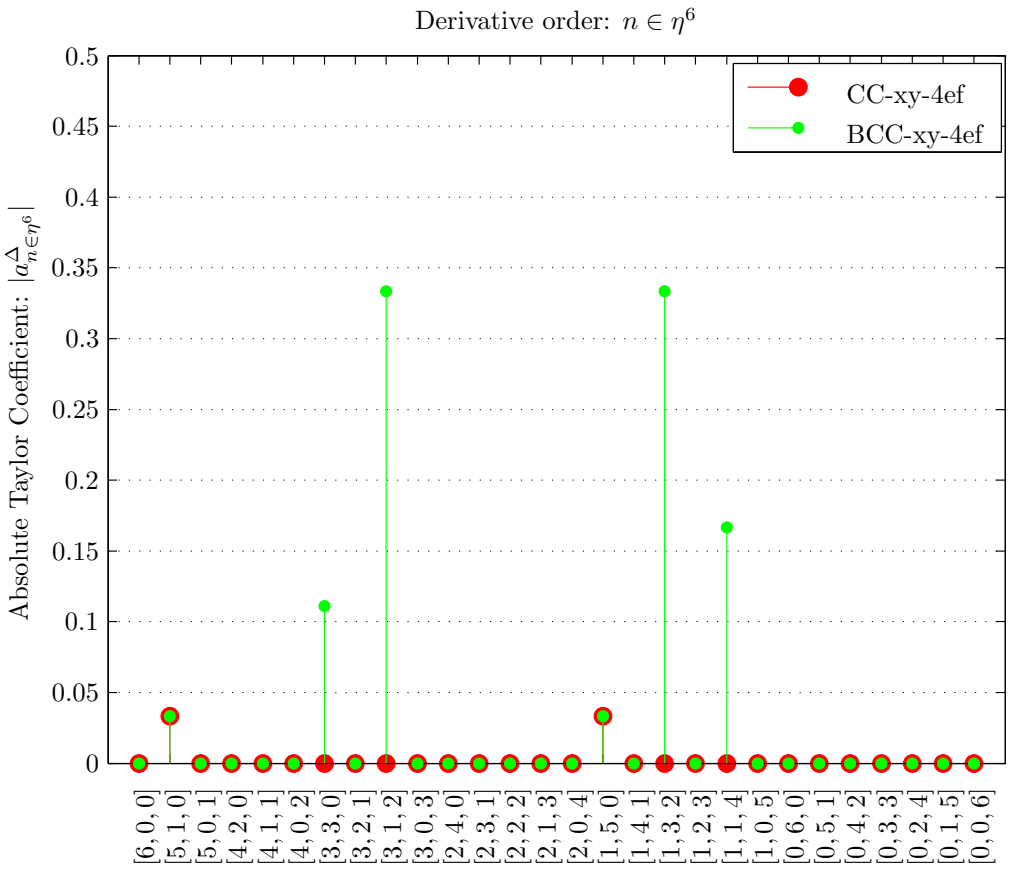


Figure C.6: Plot of absolute Taylor Coefficients in the sixth derivative ($n \in \eta^6$) order for different 4-EF f_{xy} estimating filters in CC and BCC.

Bibliography

[1] S. Aja-Fernandez, R. San-José-Estépar, C. Alberola-Lopez, and C.-F. Westin. Image quality assessment based on local variance. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pages 4815 –4818, Aug 2006.

[2] S. Aja-Fernández, G. Vegas-Sánchez-Ferrero, M. Martín-Fernández, and C. Alberola-López. Automatic noise estimation in images using local statistics. Additive and multiplicative cases. *Image and Vision Computing*, 27(6):756–770, 2009.

[3] Usman Raza Alim, Alireza Entezari, and Torsten Möller. The Lattice-Boltzmann method on optimal sampling lattices. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):630–641, 2009.

[4] Usman Raza Alim and Torsten Möller. A fast Fourier transform with rectangular output on the BCC and FCC lattices. In *Proceedings of the Eighth International Conference on Sampling Theory and Applications (SampTA'09)*, Marseille, France, May 18-22, 2009.

[5] D. Barash and D. Comaniciu. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Vision Computing*, 22(1):73–81, 2004.

[6] Dirk Bartz. Volvis, [online]. `http://www.volvis.org`. Last accessed: 21 July 2010.

[7] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[8] R.A. Carmona and S. Zhong. Adaptive smoothing respecting feature directions. *IEEE Transactions on Image Processing*, 7(3):353–358, 1998.

[9] Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proceedings of the IEEE Conference on Visualization*, pages 497–504. IEEE Computer Society, 2004.

[10] D.L. Collins, A.P. Zijdenbos, V. Kollokian, J.G. Sled, NJ Kabani, C.J. Holmes, and A.C. Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.

[11] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002.

[12] B. Csébfalvi and B. Domonkos. Pass-Band optimal reconstruction on the body-centered cubic lattice. *Vision, Modeling, and Visualization 2008: Proceedings, October 8-10, 2008, Konstanz, Germany*, pages 71–80, 2008.

[13] D. den Hertog, R. Brekelmans, L. Driessen, and H. Hamers. Gradient estimation schemes for noisy functions. *Tilburg University, Center for Economic Research, Tech. Rep*, 12, 2003.

[14] S. C. Dutta Roy and B. Kumar. *Handbook of Statistics*, volume 10, chapter Digital Differentiators, pages 159–205. Elsevier Science Publishers B. V., North Holland, 1993.

[15] A. Entezari, M. Mirzargar, and L. Kalantari. Quasi-interpolation on the body centered cubic lattice. In *Computer Graphics Forum*, volume 28, pages 1015–1022. Blackwell Publishing Ltd, 2009.

[16] Alireza Entezari, Dimitri Van De Ville, and Torsten Möller. Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):313 – 328, 2008.

[17] H. Farid and E.P. Simoncelli. Differentiation of discrete multi-dimensional signals. *IEEE Transactions on Image Processing*, 13(4):496–508, 2004.

[18] G. Gerig, O. Kubler, R. Kikinis, and F.A. Jolesz. Nonlinear anisotropic filtering of MRI data. *IEEE Transactions on Medical Imaging*, 11(2):221–232, 1992.

[19] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.

[20] H.J.M. Hamers, R.C.M. Brekelmans, L. Driessen, and D. den Hertog. Gradient estimation using Lagrange interpolation polynomials. Technical Report 101, Tilburg University, Center for Economic Research, 2003.

[21] Zahid Hossain, Usman R. Alim, and Torsten Möller. Toward high-quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics*, 99(RapidPosts), 2010.

[22] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 79–86, New York, NY, USA, 1998. ACM.

[23] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, pages 513–520. IEEE Computer Society, 2003.

[24] G.A. Korn and T.M. Korn. *Mathematical Handbook for Scientists and Engineers*. McGraw-Hill, New York, 1968.

[25] K. Krissian. Flux-based anisotropic diffusion applied to enhancement of 3D angiogram. *IEEE Transactions on Medical Imaging*, 21(11):1440–1442, 2002.

[26] K. Krissian, C.F. Westin, R. Kikinis, and K.G. Vosburgh. Oriented speckle reducing anisotropic diffusion. *IEEE Transactions on Image Processing*, 16(5):1412–1424, 2007.

[27] Karl Krissian. AMILab, [online]. `http://amilab.sourceforge.net`. Version: 2.0.4, Last accessed: 21 July 2010.

[28] Karl Krissian and Santiago Aja-Fernández. Noise-driven anisotropic diffusion filtering of MRI. *IEEE Transaction on Image Processing.*, 18(10):2265–2274, 2009.

[29] Karl Krissian, Grégoire Malandain, and Nicholas Ayache. Directional anisotropic diffusion applied to segmentation of vessels in 3D images. In *SCALE-SPACE '97: Proceedings of the First International Conference on Scale-Space Theory in Computer Vision*, pages 345–348, London, UK, 1997. Springer-Verlag.

[30] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London Series B*, 207:187–217, 1980.

[31] Steve R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of the IEEE Conference on Visualization*, pages 100–107. IEEE Computer Society Press, October 1994.

[32] B. Merriman, J. Bence, and S. Osher. *Diffusion generated motion by mean curvature*. Dept. of Mathematics, University of California, Los Angeles, 1992.

[33] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. A comparison of normal estimation schemes. *Proceedings of the IEEE Conference on Visualization*, pages 19–26, Oct. 1997.

[34] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. Evaluation and design of filters using a Taylor series expansion. *Visualization and Computer Graphics, IEEE Transactions on*, 3(2):184–199, Apr-Jun 1997.

[35] Torsten Möller, Klaus Mueller, Yair Kurzion, Raghu Machiraju, and Roni Yagel. Design of accurate and smooth filters for function and derivative reconstruction. *Proceedings of the Symposium on Volume Visualization*, pages 143–151, Oct 1998.

[36] K. Mosaliganti, F. Janoos, A. Gelas, R. Noche, N. Obholzer, R. Machiraju, and S. Megason. Anisotropic plate diffusion filtering for detection of cell membranes in 3D microscopy images. In *Proceedings of the 2010 IEEE International Conference on Biomedical Imaging: From Nano to Macro*, pages 588 –591, April 2010.

[37] O. Nemitz, M. Rumpf, T. Tasdizen, and R. Whitaker. Anisotropic curvature motion for structure enhancing smoothing of 3D MR angiography data. *Journal of Mathematical Imaging and Vision*, 27(3):217–229, 2007.

[38] Neophytos Neophytou and Klaus Mueller. Space-time points: 4D splatting on efficient grids. In *VVS '02: Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics*, pages 97–106, Piscataway, NJ, USA, 2002.

[39] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, 2003.

[40] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

[41] Richard Prager, Gee Andrew, and Graham Treece. Stradx, [online]. `http://mi.eng.cam.ac.uk/~rwp/stradx/`. Version: 7.4g, Last accessed: 21 July 2010.

[42] T. Preusser and M. Rumpf. A level set method for anisotropic geometric diffusion in 3D image processing. *SIAM Journal on Applied Mathematics*, 62(5):1772–1793, 2002.

[43] Stefan Roettger. The volume library, [online]. `http://www9.informatik.uni-erlangen.de/External/vollib/`. Last accessed: 21 July 2010.

[44] Gilbert Strang and George J. Fix. *An Analysis of the Finite Element Method*. Prentice Hall, 1973.

[45] Haiwei Sun, Ning Kang, Jun Zhang, and Eric S. Carlson. A fourth-order compact difference scheme on face centered cubic grids with multigrid method for solving 2D convection diffusion equation. *Math. Comput. Simul.*, 63(6):651–661, 2003.

[46] Q. Sun, J.A. Hossack, J. Tang, and S.T. Acton. Speckle reducing anisotropic diffusion for 3D ultrasound images. *Computerized Medical Imaging and Graphics*, 28(8):461–470, 2004.

[47] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. *Proceedings of the IEEE Conference on Visualization*, pages 125–132, 2002.

[48] Shivaraj Tenginakai, Jinho Lee, and Raghu Machiraju. Salient iso-surface detection with model-independent statistical signatures. In *Proceedings of the IEEE Conference on Visualization*, pages 231–238, 2001.

[49] Thomas Theußl, Torsten Möller, and Eduard Gröller. Optimal regular volume sampling. In *Proceedings of the IEEE Conference on Visualization*, pages 91–98, Oct 2001.

[50] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 839–846. IEEE Computer Society, 1998.

[51] M. Unser. Sampling-50 years after Shannon. *Proceedings of the IEEE*, 88(4):569–587, 2000.

[52] L. Velho, L. H. de Figueiredo, and J. Anthony Gomes. *Implicit Objects in Computer Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[53] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[54] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. B.G. Teubner, Stuttgart, Germany, 1998.

[55] Joachim Weickert. Coherence-enhancing diffusion filtering. *Int. J. Comput. Vision*, 31(2-3):111–127, 1999.

[56] Ross Whitaker. Volumetric deformable models: Active blobs. In *Proceedings of SPIE*, pages 122–134, 1994.

[57] Y. Yu and S.T. Acton. Speckle reducing anisotropic diffusion. *IEEE Transactions on Image Processing*, 11(11):1260–1270, 2002.