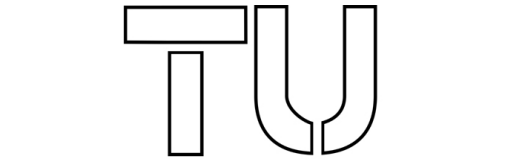


TECHNISCHE UNIVERSITÄT WIEN

Dissertation

Derivatives and Eigensystems for Volume-Data Analysis and Visualization

ausgeführt
zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller,
Institut für Computergraphik und Algorithmen,

eingereicht
an der Technischen Universität Wien,
Fakultät für Technische Naturwissenschaften und Informatik,

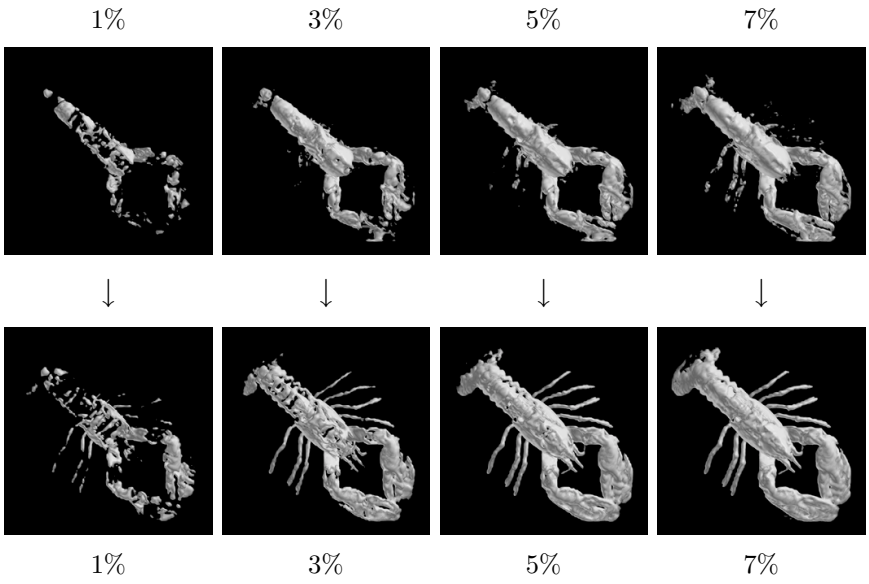
von
Jiří Hladůvka,
Matrikelnummer 9826903,
Rötzergasse 20/5,
A-1170 Wien, Österreich,
geboren am 19. April 1972 in Bratislava.

Wien, im Dezember 2001.

Jiří Hladůvka

Derivatives and Eigensystems for Volume-Data Analysis and Visualization

Ph.D. Thesis



a hypertext PDF version available via
http://www.cg.tuwien.ac.at/research/theses/

To my father

Abstract

Volume data refer to sampled three-dimensional spatial signals. The tools which handle them can broadly be divided into two categories: visual tools which aim at an output interpretable by a human user and analytic tools which prepare the data for further machine processing. Although it would be natural that the two related disciplines, i.e., *volume visualization* and *volume processing* closely collaborate, they are still rather separated. The work presented here contributes to bridge the gap in between.

This thesis addresses classification of volume samples based on observations of how the scalar values vary in their vicinity. We investigate the first three terms of a Taylor series expansion of the corresponding scalar field at the inspected points. An important issue arising with such an analysis in higher dimensions are the directions to be examined. In order to find an answer to this problem we study the *eigensystems* of algebraic structures composed of the first- and second-order partial *derivatives*.

After a survey on derivative-based classification in volume visualization and processing we present new contributions which apply to three distinct problems: specification of transfer functions, content-based retrieval of volume-data features, and shape-based interpolation.

Kurzfassung

Der Begriff Volumsdaten beschreibt diskrete drei-dimensionale räumlichen Signale. Verfahren, welche auf diesen Daten arbeiten, können grob in zwei Kategorien unterteilt werden: Visuelle Verfahren, welche das Ziel haben, eine für den Menschen interpretierbare Ausgabe zu liefern. Analytische Verfahren bereiten die Daten für weitere maschinelle Bearbeitungen auf. Obwohl beide Disziplinen (Volumsvisualisierung und Volumsverarbeitung) große Ähnlichkeiten aufweisen, werden Themenstellungen oft getrennt behandelt.

Diese Dissertation behandelt die Klassifikation von Volumsdaten basierend auf der Beobachtung von Unterschieden der skalaren Werte in einer lokalen Umgebung. Ein übliche Ansatz solcher Untersuchungen ist es, die Taylor-Entwicklung skalare Datensätze zu untersuchen. Als ein wichtiges Thema stellt sich bei der Analyse in höheren Dimensionen die zu untersuchenden Richtungen heraus. Gewisse Antworten können Eigensysteme von algebraischen Strukturen liefern, welche Ableitungen erster oder zweiter Ordnung beinhalten.

Nach einem kurzen Überblick über, auf Ableitungen basierenden Klassifizierungen in der Volumsvisualisierung und Volumsverarbeitung, präsentieren wir neue Beiträge, welche auf drei verschiedene Probleme anwendbar sind: Spezifizierung von Transferfunktionen, Bestimmung der relevanten Merkmale im Volumendatensatz und merkmalgesteuerte Rekonstruktion durch Interpolation.

Contents

List of figures

List of tables

Chapter 1

Introduction

Volume data is an expression for sampled three-dimensional scalar fields. To understand their content, visualization tools are being developed for more than a decade:

> ... in 10 years all rendering will be volume rendering
> *Jim Kajiya at SIGGRAPH '91* [9]

> Why is real-time volume rendering no longer a year away? Because it is more than two years away!
> *Bill Lorensen at IEEE Visualization '98* [36]

> Why is real-time volume rendering no longer a year away? Because it is a half a year away!
> *Hanspeter Pfister at IEEE Visualization '98* [36]

> ... full screen volume rendering may arrive in 25 years, not 5 years. And full eye in 70 years!
> *David Nadeau at WSCG 2001* [68]

Though the prophecy of Kajyia has not been fulfilled, we nowadays *can* visualize volumes of moderate resolutions, say $256 \times 256 \times 256$, in real-time at a reasonable quality level. The amount of the data, however, tends to increase far beyond of what we are able to interactively display nowadays and, according to Nadeau, much faster than the capabilities of hardware which could process them. It therefore still will be that the hardware solutions can not substitute good algorithms. To achieve easily interpretable results,

the computer assistance in deciding on important features of the data, specification of visualization parameters, and corresponding user interfaces will remain indispensable and will require an ongoing research in 3D-data analysis.

The *visualization* of volume data, however, is only one part of the research interest – we would like to expect more from computers. The situation is more clearly articulated in 2D imaging where, in most cases, we are able to understand the images without computer assistance. In this case the research is essentially triggered by the requirement that we want the *computers* to "have a look" at our images, process them and understand them. The importance of processing and recognition tasks increases with the dimensionality and size of the data.

What this thesis is about

The motivation for work addressed by this thesis comes from 2D image processing. Here, one of several approaches to classify pixels is based on filters which utilize differentiation. This yields tools for, e.g., detection of ridges, corners, T-junctions, and cross-junctions. This thesis addresses a similar problem in 3D – a classification of volume voxels based on observations of how the scalar values vary in their vicinity.

A usual approach for such an investigation is to consider the terms of a Taylor series expansion of the scalar function at the inspected points. In higher dimensions, an important question which arises with such an analysis is in which directions should be the Taylor series examined. A particular answer can be found studying the eigensystems of algebraic structures composed of the derivatives. Chapter 2 provides a concise reference to those theorems of linear algebra which are related to real symmetric matrices and their eigensystems.

We deal with sampled 3D signals which are thus discontinuous and non-differentiable. In section 3.1 of chapter 3 we briefly summarize the concepts which make differentiation on discrete grids possible. In the remainder of this chapter, in sections 3.2 and 3.3 we summarize the areas where derivatives, related structures and their eigensystems are already used in volume visualization and processing. These two sections are intended as the state of the art.

Chapter 4 is based on our work [25] on transfer functions which are an integral part of a volume visualization pipeline. The classifications of vox-

els presented here reflects the magnitudes of the principal curvatures of an underlying surface.

The work [26, 23, 27, 24] assembled in chapter 5 also exemplifies the link between volume visualization and analysis. To understand the content of a volume, it is usually not necessary to display each and every voxel. An advantage of object-based volume-rendering techniques is that they may concentrate only on the salient parts of the data. Our approach to identify content-carrying voxels is based on an analysis of the second derivatives of the scalar field.

High-quality image-based visualization techniques require resampling, i.e., an interpolation within the scalar field. The interpolation methods can broadly be divided into two categories: signal-based and shape-based. In chapter 6 we describe our contribution [22] to the class of shape-based interpolation algorithms which is based on an eigen-analysis of the structure tensors.

As chapters 4–6 deal with distinct problems, we finish them separately with short conclusions, comments, and hints for possible ongoing research. An overall summary of this thesis can be found in chapter 7.

What this thesis is *not* about

The title of this work indicates three well-separated topics which one might expect to be explained in great detail. This, however, is not the purpose of this thesis which rather concentrates on our own contributions. So what is actually missing?

Firstly, this thesis avoids an introduction to volume visualization and processing. We believe that both topics are well covered by books published in recent years [35, 47, 5, 12, 51, 70].

Secondly, the thesis deals with derivatives of volume data. We do not address, however, the filter-design issues which are crucial for an accurate differentiation. Instead, we base our implementations on existing research [71, 60, 61, 62, 63, 87, 89] and concentrate especially on the applicability of the information coming from differentiation.

Finally we only address eigensystems of algebraic structures which are based on derivatives. Thus we do not deal with those structures which are composed of the actual data values as it is, for instance, the case of the inertia tensor [51, Chap. 3],[34, Chap. 15] or the diffusion tensor [40].

We conclude the introductory part of the thesis with a comment to the following two quotations:

> Volume graphics today is where surface graphics was fifteen years ago.
>
> *David Nadeau at WSCG 2001* [68]

> The more dimensions are processed, the more important it is that computer graphics and computer vision come closer together.
>
> *Bernd Jähne* [34]

In our opinion, the time delay between volume *processing* and volume *visualization* is still of a similar order as the difference between volume graphics and surface graphics. This thesis contributes to bridging the gap between these two fields.

Chapter 2

Eigensystems and quadratic forms

In this chapter we recall the definitions and theorems from linear algebra which are related to this thesis. They are intended to highlight the properties of real symmetric matrices and the related quadratic forms as well as their eigensystems. In order to allow the reader to refresh the underlying relationships we repeat the theorems together with proofs. In the last section of this chapter we give several hints on effective implementation.

2.1 Eigensystems

Definition 2.1.1. Let $\mathbf{A}$ be an $n \times n$ matrix.
An *eigenvector* of $\mathbf{A}$ is a nonzero vector $\boldsymbol{v}$ such that

$$\mathbf{A}\ \boldsymbol{v} = \lambda\ \boldsymbol{v} \tag{2.1}$$

for some scalar λ. An *eigenvalue* of $\mathbf{A}$ is a scalar λ with the property that $\mathbf{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ for some nonzero vector $\boldsymbol{v}$. If $\mathbf{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ for some λ and $\boldsymbol{v} \neq \mathbf{0}$ we say that $\boldsymbol{v}$ is the eigenvector corresponding to the eigenvalue λ.

Solving eigenvalues of an $n \times n$ matrix reduces to finding the roots of an n-th degree polynomial (section 2.4). A real $n \times n$ matrix, therefore, will generally have n complex, not necessarily distinct eigenvalues. The eigensystems of *real symmetric* matrices exploited in this thesis, however, feature several useful relationships which are summarized in the next section.

2.2 Real symmetric matrices

Definition 2.2.1 (Symmetric Matrix). An $n \times n$ matrix $\mathbf{A}$ is called *symmetric* if $\mathbf{A} = \mathbf{A}^{\mathbf{T}}$.

Theorem 2.2.1. *Let* $\mathbf{A}$ *be a real symmetric matrix. Then the eigenvalues and eigenvectors of* $\mathbf{A}$ *are real, i.e.,* $\lambda_i \in \mathbb{R}, \boldsymbol{v}_i \in \mathbb{R}^n, i = 1 \dots n$.

Proof. Let $\mathbf{A}$ be a real symmetric matrix, let $\lambda \in \mathbb{C}$ be an eigenvalue of $\mathbf{A}$ and let $\boldsymbol{v}$ be an eigenvector of $\mathbf{A}$ corresponding to this eigenvalue. Conjugating and transposing the equality $\mathbf{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ we get $\boldsymbol{v}^{*T}\mathbf{A}^{*\mathbf{T}} = \lambda^*\boldsymbol{v}^{*T}$ and since $\mathbf{A}$ is real and symmetric we have $\boldsymbol{v}^{*T}\mathbf{A} = \lambda^*\boldsymbol{v}^{*T}$. Multiplying both sides by $\boldsymbol{v}$ on the right we obtain

$$\boldsymbol{v}^{*T}\mathbf{A}\boldsymbol{v} = \lambda^*\boldsymbol{v}^{*T}\boldsymbol{v} \Rightarrow \lambda\boldsymbol{v}^{*T}\boldsymbol{v} = \lambda^*\boldsymbol{v}^{*T}\boldsymbol{v} \Rightarrow (\lambda - \lambda^*)\boldsymbol{v}^{*T}\boldsymbol{v} = \mathbf{0}$$

Since by definition $\boldsymbol{v} \neq \mathbf{0}$, we have $\lambda - \lambda^* = 0 \Rightarrow \lambda \in \mathbb{R}$

As $\mathbf{A}$ is a real matrix, coefficients of $(\lambda\mathbf{I} - \mathbf{A})$ are real and therefore the eigenvectors – the solutions $\boldsymbol{v}$ of $(\lambda\mathbf{I} - \mathbf{A})\boldsymbol{v} = \mathbf{0}$ must be real, too. □

Theorem 2.2.2. *Let* $\mathbf{A}$ *be a real symmetric matrix. Then the eigenvectors of* $\mathbf{A}$ *are mutually orthogonal.*

Proof. Let $\mathbf{A}$ be a real symmetric matrix, let $\lambda_1, \lambda_2 \in \mathbb{R}$ be distinct eigenvalues and $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^n$ be the corresponding eigenvectors of $\mathbf{A}$. Then

$$\mathbf{A}\boldsymbol{v}_1 = \lambda_1\boldsymbol{v}_1 \Rightarrow \boldsymbol{v}_1^T\mathbf{A} = \lambda_1\boldsymbol{v}_1^T \Rightarrow \boldsymbol{v}_1^T\mathbf{A}\boldsymbol{v}_2 = \lambda_1\boldsymbol{v}_1^T\boldsymbol{v}_2 \Rightarrow \lambda_2\boldsymbol{v}_1^T\boldsymbol{v}_2 = \lambda_1\boldsymbol{v}_1^T\boldsymbol{v}_2 \Rightarrow$$

$$(\lambda_1 - \lambda_2)\boldsymbol{v}_1^T\boldsymbol{v}_2 = 0 \Rightarrow \boldsymbol{v}_1^T\boldsymbol{v}_2 = 0$$

□

Definition 2.2.2 (Orthogonal Matrix). An $n \times n$ matrix $\mathbf{A}$ is called *orthogonal* if $\mathbf{A}\mathbf{A}^{\mathbf{T}} = \mathbf{A}^{\mathbf{T}}\mathbf{A} = \mathbf{I}$.

Theorem 2.2.3 (Principal Axes Theorem). *Let* $\mathbf{A}$ *be a real symmetric matrix. Then there exists an orthogonal matrix* $\mathbf{P}$ *such that* $\mathbf{\Lambda} = \mathbf{P}^{\mathbf{T}}\mathbf{A}\mathbf{P}$ *is a diagonal matrix.*

The proof is too lengthy to show it here in all details and we refer the reader to any book on algebra, e.g. [54]. The proof is constructive and from the construction of matrix $\mathbf{P}$ it can be seen that it is assembled from the *unit* eigenvectors $\boldsymbol{e}_i$ of $\mathbf{A}$, and that the diagonal matrix $\mathbf{\Lambda}$ contains the eigenvalues at the diagonal, i.e., $\mathbf{P} = (\boldsymbol{e}_1\ \boldsymbol{e}_2\ \dots\ \boldsymbol{e}_n)$ and $\mathbf{\Lambda} = \text{diag}\ (\lambda_1, \lambda_2, \dots, \lambda_n)$.

2.3 Quadratic forms

Definition 2.3.1 (Quadratic Form). A real quadratic form $Q(x_1, \ldots, x_n)$ is a homogeneous polynomial of degree 2 in variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{R}$.

Remark: Each quadratic form can be expressed as

$$Q(x_1, \ldots, x_n) = Q(\boldsymbol{x}) = \boldsymbol{x}^T \mathbf{A} \boldsymbol{x}$$

where $\mathbf{A}$ is an $n \times n$ real symmetric matrix.

Theorem 2.3.1. *Let Q be a quadratic form associated with an $n \times n$ real symmetric matrix $\mathbf{A}$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be eigenvalues of $\mathbf{A}$ and $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ the corresponding unit eigenvectors.*
The maximum value of $Q(\boldsymbol{u})$ for unit vectors $\boldsymbol{u} \in \mathbb{R}^n$ is λ_1 and is attained at the unit eigenvector $\boldsymbol{e}_1$.
The minimum value of $Q(\boldsymbol{u})$ for unit vectors $\boldsymbol{u} \in \mathbb{R}^n$ is λ_n and is attained at the unit eigenvector $\boldsymbol{e}_n$.

Proof. By the Principal Axes Theorem (2.2.3) we can find an orthogonal matrix $\mathbf{P}$ such that

$$\mathbf{P^T A P} = \mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}.$$

For any vector $\boldsymbol{u} = (u_1, \ldots, u_n)^T \in \mathbb{R}^n$ we write $\boldsymbol{u} = \mathbf{P}\boldsymbol{v}$, where $\boldsymbol{v} = (v_1, \ldots, v_n)^T \in \mathbb{R}^n$. Since $\mathbf{P}$ is orthogonal

$$\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u}^T \boldsymbol{u}} = \sqrt{(\mathbf{P}\boldsymbol{v})^T \mathbf{P}\boldsymbol{v}} = \sqrt{\boldsymbol{v}^T \boldsymbol{v}} = \|\boldsymbol{v}\|$$

hence $\|\boldsymbol{u}\| = 1 \Rightarrow \|\boldsymbol{v}\| = 1$.
Let $\boldsymbol{u}$ be a unit vector. Then

$$\begin{aligned} Q(\boldsymbol{u}) = \boldsymbol{u}^T \mathbf{A} \boldsymbol{u} &= (\mathbf{P}\boldsymbol{v})^T \mathbf{A} \mathbf{P} \boldsymbol{v} = \boldsymbol{v}^T \mathbf{\Lambda} \boldsymbol{v} = \\ &= \lambda_1 v_1^2 + \lambda_2 v_2^2 + \cdots + \lambda_n v_n^2 \leq \\ &\leq \lambda_1 v_1^2 + \lambda_1 v_2^2 + \cdots + \lambda_1 v_n^2 = \\ &= \lambda_1 (v_1^2 + v_2^2 + \cdots + v_n^2) = \\ &= \lambda_1 \end{aligned}$$

On the other hand, if $\boldsymbol{e}_1$ is a unit eigenvector corresponding to λ_1 then, due to Eq. (2.1)

$$Q(\boldsymbol{e}_1) = {\boldsymbol{e}_1}^T \mathbf{A} \boldsymbol{e}_1 = {\boldsymbol{e}_1}^T (\lambda_1 \boldsymbol{e}_1) = \lambda_1.$$

Similar arguments prove the second statement about the minimum. □

2.4 Computation issues

Equation (2.1) is equivalent to the homogeneous system of equations

$$(\lambda \mathbf{I} - \mathbf{A})\boldsymbol{v} = \mathbf{0} \tag{2.2}$$

which has a nontrivial solution only if

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0 \tag{2.3}$$

Equation (2.3) is, after an expansion, an nth degree polynomial in λ whose roots are the eigenvalues. Substituting the roots back to system (2.2) and solving it gives the associated eigenvectors.

In the general case (large dimension n), the numerical root-searching in the *characteristic equation* (2.3) is computationally a poor method for finding eigenvalues and eigenvectors [77]. There are canned *eigen-packages* which provide mechanisms to determine all eigenvalues and, in addition, offer separate paths to find none, some, or all corresponding eigenvectors. Moreover, these packages usually provide separate paths for the following special types of matrices: real symmetric tridiagonal, real symmetric banded, real symmetric, real non symmetric, complex Hermitian, and complex non Hermitian. An overview of the algorithms with implementation in **C** can be found, e.g., in the *Numerical Recipes* [77] and at the related web site [1].

For real symmetric matrices Press et al. [77] recommend the fast converging Jacobi method. The topics addressed in this thesis, however, only deal with 2D or 3D signals and employ 2×2 and 3×3 real symmetric matrices. For these two special cases it is more efficient to use the analytical solution.

Special case: Diagonal matrices

The eigenvalues of a diagonal matrix are the diagonal elements. The eigenvectors can be chosen to form an orthonormal basis. Note that this case also includes the zero matrix.

Special case: 2×2 real symmetric matrix

Equation (2.3) expands to a quadratic equation in λ

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \qquad \det(\lambda \mathbf{I} - \mathbf{A}) = \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}^2) = 0$$

and the eigenvalues are

$$\lambda_{1,2} = \frac{1}{2}\left(a_{11} \pm \sqrt{4a_{12}^2 + (a_{11} - a_{22})^2} + a_{22}\right) \tag{2.4}$$

Since the case $a_{12} = 0$ is caught by the diagonal case above, the (not normalized) eigenvectors equal to

$$\boldsymbol{v}_{1,2} = \left[\frac{1}{2a_{12}}\left(a_{11} \pm \sqrt{4a_{12}^2 + (a_{11} - a_{22})^2} - a_{22}\right), \quad 1\right]$$

Special case: 3×3 **real symmetric matrix**

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

The determinant in Eq. (2.3) expands to a monic cubic polynomial in λ:

$$\begin{aligned} \det(\lambda\mathbf{I} - \mathbf{A}) &= \lambda^3 + b\lambda^2 + c\lambda + d = 0, \quad \text{where} \\ b &= -a_{11} - a_{22} - a_{33} \\ c &= a_{11}a_{22} + a_{11}a_{33} + a_{22}a_{33} - a_{12}^2 - a_{13}^2 - a_{23}^2 \\ d &= a_{11}a_{23}^2 + a_{22}a_{13}^2 + a_{33}a_{12}^2 - a_{11}a_{22}a_{33} - 2a_{12}a_{13}a_{23} \end{aligned} \tag{2.5}$$

Following the Theorem 2.2.1 the roots of the polynomial (2.5) will be real and therefore the quantity

$$p = \frac{3c - b^2}{9}$$

will be nonpositive [3]. Case $p = 0$ indicates a triple root $\lambda_{1,2,3} = -b/3$. In case $p < 0$ the roots can be computed as follows [3]:

$$\begin{aligned} q &= \frac{2b^3 - 9bc + 27d}{54} \\ r &= \begin{cases} +\sqrt{-p} & \text{if } q \geq 0 \\ -\sqrt{-p} & \text{if } q < 0 \end{cases} \\ \varphi &= \arccos\left(\frac{q}{r^3}\right) \end{aligned} \tag{2.6}$$

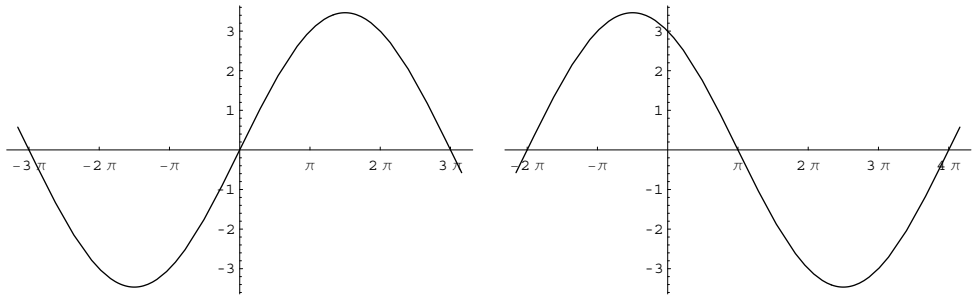


Figure 2.1: The terms $\lambda_2 - \lambda_3$ and $\lambda_1 - \lambda_2$ as functions of argument φ displayed for $r = -1$. For $r < 0$ the terms are nonnegative on the interval $\langle 0, \pi \rangle$.

$$\begin{aligned} \lambda_1 &= -\frac{b}{3} - 2r\cos\left(\frac{\varphi}{3}\right) \\ \lambda_2 &= -\frac{b}{3} + 2r\cos\left(\frac{\varphi + \pi}{3}\right) \\ \lambda_3 &= -\frac{b}{3} + 2r\cos\left(\frac{\varphi - \pi}{3}\right) \end{aligned} \tag{2.7}$$

Due to the theorem 2.3.1, applications usually benefit from the ordering of eigenvalues. While for the 2×2 case the ordering $\lambda_1 \geq \lambda_2$ is trivially given by Equation (2.4), the relationship of eigenvalues of a 3×3 matrix is not so evident.

We point out that the ordering of the eigenvalues from Eq. (2.7) depends on the sign of r as follows:

$$\begin{aligned} r = 0 &\Rightarrow \lambda_3 = \lambda_2 = \lambda_1 \\ r < 0 &\Rightarrow \lambda_3 \leq \lambda_2 \leq \lambda_1 \\ r > 0 &\Rightarrow \lambda_3 \geq \lambda_2 \geq \lambda_1 \end{aligned}$$

Proof. $r = 0 \Leftrightarrow p = 0$ which indicates a triple root.
To prove the inequalities for $r < 0$ we show that the terms $\lambda_2 - \lambda_3$ and $\lambda_1 - \lambda_2$ are nonnegative. Trigonometric simplifications yield $\lambda_2 - \lambda_3 = -2\sqrt{3}\, r \sin(\varphi/3)$ and $\lambda_1 - \lambda_2 = -2\sqrt{3}\, r \cos(\varphi/3 + \pi/6)$. As φ is computed as an arccos(.), the investigations are restricted to the interval $\varphi \in \langle 0, \pi \rangle$ where, for $r < 0$, both terms are nonnegative (see also Fig. 2.1). Formally:
$\varphi \in \langle 0, \pi \rangle \wedge r < 0 \Rightarrow 0 \leq -2\sqrt{3}\, r \sin(\varphi/3) \quad = \lambda_2 - \lambda_3 \Rightarrow \lambda_3 \leq \lambda_2$, and
$\varphi \in \langle 0, \pi \rangle \wedge r < 0 \Rightarrow 0 \leq -2\sqrt{3}\, r \cos(\varphi/3 + \pi/6) = \lambda_1 - \lambda_2 \Rightarrow \lambda_2 \leq \lambda_1$.
Similar arguments prove the statement for $r > 0$. □

The eigenvectors $\boldsymbol{v_i}$ are the solutions of the homogeneous system of equations

$$(\lambda_i \mathbf{I} - \mathbf{A})\boldsymbol{v}_i = \mathbf{0}$$

Chapter 3

Volume data and derivatives

In order to investigate three-dimensional signals it is necessary to mention the nature of the domain and the dimensionality of the range. Depending on the acquisition, the devices may record one- or more-channel physical quantities as a function of 2D position and time (image sequences), 2D position and wavelength (hyperspectral images), and others.

The *volumes* which are the subject of this thesis, refer to monochrome 3D digital images, i.e., to the *sampled* versions of real-world single-channel signals recorded as a function of a 3D position.

Being discretized by sampling, the digital signals are discontinuous and therefore non-differentiable. This chapter addresses two issues. First, in section 3.1 it recalls the concepts and mechanisms which allow to perform differentiation on discrete grids. Second, sections 3.2 and 3.3 give an overview on how the derivatives and the eigensystems of related matrices are involved in volume analysis and visualization.

3.1 Continuous and discrete signals revisited

Mathematically, a 3D single-channel signal g is a three-dimensional scalar function:

$$g : \mathbb{R}^3 \to \mathbb{R} \qquad g = g(\boldsymbol{x}) = g \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{3.1}$$

where the terms x_i denote 1D spatial coordinates.

Physically, a signal records an information about a physical quantity and can be assumed to have some properties which make its mathematical handling easier [34]:

Continuity. Real signals do not show any abrupt changes or discontinuities. Mathematically this means that signals can generally be regarded as arbitrarily often differentiable.

Finite range. The physical nature of both the signal and the imaging sensor ensures that a signal is limited to a finite range. Moreover, some signals are restricted to positive values: $g(x) \geq 0$.

Finite energy. Normally a signal corresponds to the amplitude or to the energy of a physical process. As the energy of the physical system is limited, any signal must be square integrable:

$$\int_{\mathbb{R}^3} |g(\boldsymbol{x})|^2 \, \mathrm{d}V < \infty \tag{3.2}$$

This property ensures the existence of the Fourier transform.

While the continuous representation is useful for a solid mathematical foundation of signal processing, digital computers can only handle discrete data. It is therefore required to represent the signal as a three-dimensional array of values. This process involves two independent steps, i.e., *sampling* and *quantization*.

3.1.1 Sampling and quantization

The task of sampling is to construct a lattice – a set P of positions $\boldsymbol{p}$, in the domain of the function and to take the actual values of the function at these positions. Mathematically, the sampling g_s of the signal g can be expressed as a multiplication of g with replicas of the Dirac impulse δ:

$$g_s(\boldsymbol{x}) = g(\boldsymbol{x}) \sum_{\boldsymbol{p} \in P} \delta(\boldsymbol{x} - \boldsymbol{p}) \tag{3.3}$$

where the (three dimensional) Dirac impulse δ is defined by the two following conditions:

$$\begin{array}{rcll} \delta(\boldsymbol{x}) & = & 0 & x \neq \mathbf{0} \\ \int_{\mathbb{R}^3} \delta(\boldsymbol{x}) \, \mathrm{d}V & = & 1 & \end{array} \tag{3.4}$$

The result g_s of sampling (3.3) is a function which is zero everywhere except for the grid points where the original scalar values are preserved.

There are two principal questions concerning the lattice P:

Firstly the *pattern* of P is addressed. In spite of results from christalography which provide with storage-optimal irregular configurations of the sampling points [88], there are several reasons why *Cartesian* grids predominate. Firstly, most of contemporary scanners deliver the data as 3D arrays. Secondly, computation on rectangular grids is easily extensible to higher dimensions, and the orthogonality plays a crucial role in the design of separable filters. Finally, the cubical setup due to Cartesian grids is the only one which yields a regular partitioning of the 3D Euclidean space. Tiling with any of the other four regular polyhedra (tetrahedron,octahedron, dodecahedron, and icosahedron) is not space filling. In this thesis we only address Cartesian grids.

Secondly, the density of P is important. The problem addressed here is, how densely should the samples be taken such that the original signal can later be reconstructed. For Cartesian grids the answer is given by the *sampling theorem* due to Shannon [83], which states that if the signal is *band-limited* then the distance between samples must be at least half the size of the smallest detail in the signal.

There are two more outcomes from the sampling theorem for the mathematical foundations of signal processing. First, as long as a discrete signal g_s satisfies the sampling theorem, any analytic result valid for continuous functions still remains valid, because the sampled signal g_s is an exact representation of the continuous signal g. Secondly the theorem also makes it possible to distinguish between errors inherent to a chosen algorithm and those errors that are introduced by discretization [34].

Quantization

After being sampled, the values of a signal are still continuous. For handling by a computer, it is necessary to map the continuous range onto a limited number Q of discrete values

$$q : \langle 0, \infty) \rightarrow \{q_0, q_1, \ldots, q_{Q-1}\} \tag{3.5}$$

Again, two principal questions arise here. Firstly, *how many* quantization levels are necessary? It should not be possible for the human visual system to recognize the quantization levels. On the other hand, the number of levels is often determined by properties of acquisition devices (scanners, CCD cameras, ...) and usually equals to 2^8, 2^{12}, and 2^{16}. The second issue is whether

quantization intervals are equidistant or not. Although the equidistant approach is far more often used, it features a low dynamical range leading either to underexposure of dark parts or overexposure of bright parts. In order to correspond to the human visual system which exhibits rather logarithmic than linear response, it is advised to use a non-linear mapping, e.g., logarithmic or exponential.

3.1.2 Signal reconstruction

Reconstruction g_r of a signal from the samples g_s is performed by a suitable interpolation. The appropriate mechanism for interpolation coming from image processing is convolution.

Convolution and its properties

Mathematically, convolution is a functional, i.e., an operator on function spaces. It is defined through an integral which 'expresses the amount of overlap of one function k as it is shifted over another function g' [94]. If $\mathcal{F}$ denotes the set of all square-integrable functions $\mathbb{R}^3 \to \mathbb{R}$, then

$$\otimes : \mathcal{F} \times \mathcal{F} \to \mathcal{F} \qquad (k \otimes g)(\boldsymbol{x}) = \int_{\mathbb{R}^3} g(\boldsymbol{y})\; k(\boldsymbol{x} - \boldsymbol{y}) \;\; \mathrm{d}\boldsymbol{y} \tag{3.6}$$

For computation purposes, we will in this thesis repeatedly recall the following properties which are evident from the *Convolution theorem* [34, p. 46]. Convolution is

$$\text{commutative:} \qquad k \otimes g = g \otimes k \tag{3.7}$$

$$\text{associative:} \qquad (k_1 \otimes k_2) \otimes g = k_1 \otimes (k_2 \otimes g) \tag{3.8}$$

$$\text{distributive over addition:} \qquad (k_1 + k_2) \otimes g = k_1 \otimes g + k_2 \otimes g \tag{3.9}$$

Finally, convolution commutes with differentiation:

$$\frac{\partial}{\partial x_i}(k \otimes g) = \frac{\partial k}{\partial x_i} \otimes g = k \otimes \frac{\partial g}{\partial x_i} \tag{3.10}$$

The ideal reconstruction filter

There ideal reconstruction kernel, sinc, is for 1D signals defined as follows:

$$\mathrm{sinc}_{1D}(x) = \lim_{\xi \to x} \frac{\sin(\pi\xi)}{\pi\xi} = \begin{cases} 1 & \text{if } x = 0 \\ \sin(\pi x)/(\pi x) & \text{otherwise} \end{cases} \tag{3.11}$$

For 3D signals, one of the definitions of the ideal 3D reconstruction filter is the product of three 1D sinc functions:

$$\mathrm{sinc}_{3D}(x_1, x_2, x_3) = \mathrm{sinc}_{1D}(x_1) \cdot \mathrm{sinc}_{1D}(x_2) \cdot \mathrm{sinc}_{1D}(x_3) \tag{3.12}$$

Approximation versus reconstruction

The second part of the sampling theorem states that a band-limited signal g sampled above the Nyquist frequency can exactly be reconstructed by convolving the samples g_s with the sinc_{3D} function:

$$g(\boldsymbol{x}) = (\mathrm{sinc}_{3D} \otimes g_s)(\boldsymbol{x}) \tag{3.13}$$

The above is referred to as the ideal reconstruction. Limitations for a practical implementation arise when either the signal g is not bend-limited, or it was not appropriately sampled. Moreover, the infinite support of the sinc makes the reconstruction (3.13) impractical.

To avoid *aliasing* coming from an improper sampling and to avoid a convolution with an infinitely wide function, other reconstruction kernels k have been investigated which yield only an approximation of the original signal

$$g(\boldsymbol{x}) \approx g_r(\boldsymbol{x}) = (k \otimes g_s)(\boldsymbol{x}) \tag{3.14}$$

The right side of Eq. (3.14) can be interpreted as a finite linear combination of kernels k. Differentiability of the kernel k and of the approximating signal g_r is therefore of the same order: if k is a C^n function then also g_r is a C^n function. This implication provides a framework for differentiation on discrete grids: to perform an analysis of the sampled signal g_s based on n-th order derivatives, it is necessary to reconstruct the signal with (at least) an n-times differentiable kernel.

3.1.3 Reconstruction of derivatives

A common approach to analyze the behavior of the reconstructed signal g_r in a neighborhood of point p is to consider the initial terms of the Taylor series

expansion [13], i.e., to investigate the changes of the signal in directions given by unit vectors $\boldsymbol{v}$:

$$\begin{aligned} g_r(p+\boldsymbol{v}) &= \sum_{n=0}^{\infty} \frac{1}{n!} \frac{d^n}{d\boldsymbol{v}^n} g_r(p) \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{v}\cdot\nabla)^n g_r(p) \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(v_1 \frac{\partial}{\partial x_1} + v_2 \frac{\partial}{\partial x_2} + v_3 \frac{\partial}{\partial x_3} \right)^n g_r(p_1,p_2,p_3) \end{aligned} \tag{3.15}$$

In order to be able to perform an analysis based on Eq. (3.15) a mechanism to compute the partial derivatives of g_r

$$\frac{\partial^n}{\partial x_1^a \partial x_2^b \partial x_3^c} g_r(p) \qquad \text{where} \quad a+b+c=n \tag{3.16}$$

is needed.

Recalling that convolution commutes with differentiation (Eq. 3.10) and that the signal is reconstructed by convolving its samples with the reconstruction kernel k (Eq. 3.14), Equation (3.16) rewrites as

$$\frac{\partial^n}{\partial x_1^a \partial x_2^b \partial x_3^c} g_r(p) = \frac{\partial^n}{\partial x_1^a \partial x_2^b \partial x_3^c} \Big(k \otimes g_s(p) \Big) = \left(\frac{\partial^n}{\partial x_1^a \partial x_2^b \partial x_3^c} k \right) \otimes g_s(p) \tag{3.17}$$

In other words, the (partial) derivatives of a sampled signal can be reconstructed directly from the samples g_s if we convolve them with the corresponding derivatives of the reconstruction kernel.

3.1.4 Computation issues

The equations in the previous section provide a solid *theoretical* background for processing of continuous signals. For computer implementation, the discrete version of convolution is needed:

$$(k_s \otimes g_s)(\boldsymbol{x}) = (g_s \otimes k_s)(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\mathbb{Z}^3} k_s(\boldsymbol{y}) g_s(\boldsymbol{x}-\boldsymbol{y}) \tag{3.18}$$

Due to the *discrete* convolution theorem, commutativity, associativity, and distributivity remain valid also for the discrete convolution [34, p. 56]

From Equation (3.18) it is evident that the actual computation can be restricted only to nonzero points of the kernel and that a convolution with

the infinitely wide *discrete* sinc function can not be computed. The straightforward truncation of sinc has been recognized as a poor solution, because it causes ringing. The research concentrated therefore on reliable approximations of the sinc on a finite support [57, 62, 63, 87].

For discrete derivative kernels, the problem is essentially the same. The derivatives of sinc, the cosc functions, also have an infinite support and their truncation fails. Similarly to the sinc, the research efforts concentrated, both in the signal-processing [71] and the volume-processing communities [60, 61, 62, 63, 87], on an approximation of the continuous cosc on a finite, possibly narrow support.

The research effort on the design of reconstruction filters aims at an accurate reconstruction of the signal *between* the samples g_s. The tasks addressed in this thesis, however, are restricted only to a classification of the grid points. This fact reasonably simplifies the previous discussions. Firstly we do not need to reconstruct the signal at the grid points. Secondly, there are several well-established derivative kernels coming from 2D image processing which are optimized for computations at the grid points. Their 3D extensions can directly be applied for 3D images.

The 2D variants of the derivative kernels used in our work are listed in Appendix A. The most frequent choice in the literature, the central differences (A.1), arises naturally from the definition of continuous derivatives but is only reasonable when the spacing between samples is well below the Nyquist limit [34, p. 411]. As this is generally not assured, the principal problem is that a derivative operator can only be *approximated*. This is the basic reason why there is such a wide variety of derivative kernels. The common property of the Prewitt (A.2), Sobel (A.3), and the optimized Sobel (A.4) kernels is that they simultaneously perform both smoothing and differentiation in mutually orthogonal directions.

In computer vision, many authors use sampled Gaussian derivatives (A.5) for two reasons. Firstly they exhibit better properties than central differences and are less computationally expensive than the truncated or windowed cosc functions. Secondly, later in the 90-ties, the Gaussian filter and its derivatives was found to be a *unique* derivative filter for feature recognition in *scale spaces*.

Efficient convolutions

Higher-dimensional convolution with even moderately-sized kernels is usually a time- and space-expensive process. In the following we summarize several hints leading to an efficient implementation.

Time-complexity reduction To speed up the convolution, hardware features on specific platforms can be used [28, 30]. The approaches to kernel-specific software acceleration include saving multiplications by grouping equal coefficients, selection of an optimal scale (i.e., kernel size) and use of separability. The later one belongs to the most efficient strategies: a convolution with a separable 3D $m \times m \times m$ filter k can be replaced by three subsequent one-dimensional convolutions with axes-aligned filters k_1, k_2, k_3 of size m. This strategy reduces the time complexity from $O(m^3)$ to $O(3m)$.

The separable computation of derivatives due to the right side of Equation (3.17) rewrites as follows:

$$\left(\overbrace{\frac{\partial^{a+b+c}}{\partial x_1^a \partial x_2^b \partial x_3^c} k}^{m\times m\times m}\right) \otimes g_s = \overbrace{\frac{d^a}{dx_1^a} k_1}^{m\times 1\times 1} \otimes \left(\overbrace{\frac{d^b}{dx_2^b} k_2}^{1\times m\times 1} \otimes \left(\overbrace{\frac{d^c}{dx_3^c} k_3}^{1\times 1\times m} \otimes g_s\right)\right) \tag{3.19}$$

The separability of frequently used derivative filters is discussed in more details in Appendix A.

Reducing storage requirements The discrete convolution can be thought as an operator on volumes – it takes the discrete input g_s and the discrete kernel k_s, and produces an output volume which, of course, requires additional space in memory. Sometimes it can be desirable to rewrite the original volume with the result. This, however, is not possible directly on a voxel-to-voxel basis because each voxel in the original volume is required for the computation of m^3 voxels in the output and its value must not be, during their computation, rewritten. In-place convolution for general (nonrecursive) kernels can be achieved with *cyclic buffers* [34, p. 120].

For separable kernels, the in-place convolution can more easily be implemented with *accumulation buffers*. The mechanism for the 2D case and a separable kernel is explained in Fig. 3.1. Separable 2D convolution is a two-pass, row and column, process. Each row in the first pass is read into the line buffer, convolved with the 'row' kernel and written back to the original place. Similarly, in the second pass each column is read to the (same) buffer, convolved with the 'column' kernel and written back. A disadvantage of this

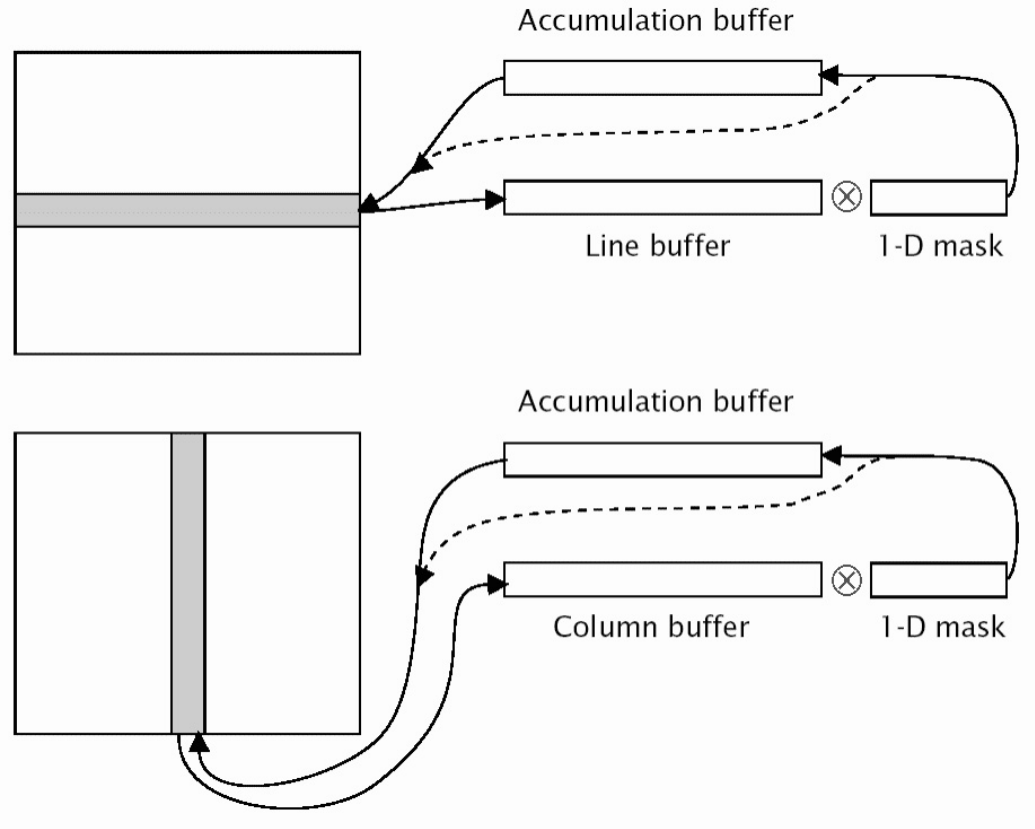


Figure 3.1: In-place computation of a 2D separable convolution. Image by Jähne [34, p. 122].

implementation is that each pixel will be copied twice which increases the data-flow between the main memory and cache. On the other hand, the same algorithm for 1D convolution can be used for both directions, while only different routines for copying are necessary. An extension to three dimensions is straightforward.

In the reminder of this chapter we give an overview of how derivatives (up to second order), related structures, and their eigensystems are used in volume processing.

3.2 1st-order derivatives in volume analysis

3.2.1 Gradient and gradient magnitude

The gradient field of scalar function g is a vector field defined as:

$$\nabla g = \begin{bmatrix} \partial g/\partial x_1 \\ \partial g/\partial x_2 \\ \partial g/\partial x_3 \end{bmatrix} \tag{3.20}$$

With help of the gradient, a derivative of g in a direction given by a *unit* vector $\boldsymbol{v}$ is expressed as:

$$\frac{d}{d\boldsymbol{v}}\, g = \boldsymbol{v}^T \nabla g \tag{3.21}$$

Since the right side of Eq. (3.21) is a scalar product it is evident, that the maximal directional derivative is in the direction of a unit vector $\boldsymbol{n} = \nabla g / \|\nabla g\|$. The rate of change in direction $\boldsymbol{n}$ is equivalent to the magnitude of the gradient:

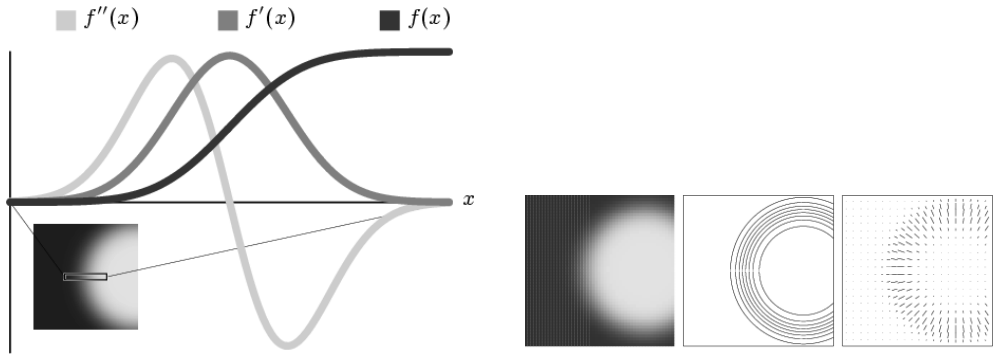
$$\frac{d}{d\boldsymbol{n}}\, g = \boldsymbol{n}^T \nabla g = \frac{\nabla g^T}{\|\nabla g\|} \nabla g = \|\nabla g\| \tag{3.22}$$

In 2D image processing the gradient magnitude provides an isotropic *edge* detector (refer also to Figure 3.2(a)). Similarly, in volume processing the gradient magnitude is used for *boundary* detection and additionally is involved in the classification of isosurfaces [46].

Within a scalar field the vector $\boldsymbol{n}$ is parallel or anti parallel to the normal of an implicitly defined isosurface (refer also to Figure 3.2(b)). Considering an isolevel boundary of an object with higher intensities than the background, vector $\boldsymbol{n}$ corresponds thus to the *inner* normal of the boundary while the vector $-\boldsymbol{n}$ corresponds to the *outer* normal of the boundary. The orientation of inner and outer normals swap if the object intensities are lower than the background. To render isosurfaces without the a priori knowledge on object-versus-background intensities, the terms in the Phong shading model [76] which depend on the direction of the normal can be rewritten with absolute values:

$$I \;=\; k_A + k_D |\boldsymbol{n}^T \boldsymbol{l}| + k_S |\boldsymbol{r}^T \boldsymbol{v}|^r \tag{3.23}$$

where I is the output intensity, $\boldsymbol{r}, \boldsymbol{l}, \boldsymbol{v}$ are the reflection vector, incoming light direction, and viewing direction. k_A, k_S, k_D represent the ambient, specular and diffuse coefficients, respectively and $r > 1$ controls the sharpness of reflections.



(a) **(b)**

Figure 3.2: (a) A 1D profile f and its derivatives f', f'' across an ideal boundary in g. The middle of the intensity transition corresponds to the maximum of the first derivative and to a zero-crossing of the second derivative. (b) The same scene, isocontours, and the gradient field. Images by Kindlmann and Durkin [39].

In non-photorealistic rendering, the contours of surfaces can be emphasized involving a view-dependent component. Parts of surfaces which are perpendicular to the viewing direction are highlighted [8, 7]:

$$Contourness(\boldsymbol{v}) = \left(1 - \left|\boldsymbol{n}^T \boldsymbol{v}\right|\right)^m \tag{3.24}$$

where m is an exponent controlling the sharpness of contours.

In most cases, the rendering results are very sensitive to the accuracy of the gradient direction. For this reason, more sophisticated algorithms are desirable to replace the estimation of partial derivatives, e.g., the linear regression model introduced by Neumann et al. [69].

3.2.2 Structure tensor

For a 3D scalar field, the structure tensor introduced by Knutsson [42] is a 2nd-rank tensor of order 3. It therefore can be represented by a 3×3 matrix.

Roughly said, the structure tensor $\mathcal{J}$ is a weighted average of outer products of gradients over a neighborhood. For a neighborhood $U(p_0)$ of a specific point p_0 and a weight function h it is defined as:

$$\mathcal{J}(p_0) = \int_{U(p_0)} h(p_0 - u) \left(\nabla g(u)\ \nabla g(u)^T\right) \mathrm{d}u \tag{3.25}$$

$$= \int_{U(p_0)} h(p_0 - u) \begin{pmatrix} \left(\frac{\partial g(u)}{\partial x_1}\right)^2 & \frac{\partial g(u)}{\partial x_1}\frac{\partial g(u)}{\partial x_2} & \frac{\partial g(u)}{\partial x_1}\frac{\partial g(u)}{\partial x_3} \\ \frac{\partial g(u)}{\partial x_2}\frac{\partial g(u)}{\partial x_1} & \left(\frac{\partial g(u)}{\partial x_2}\right)^2 & \frac{\partial g(u)}{\partial x_2}\frac{\partial g(u)}{\partial x_3} \\ \frac{\partial g(u)}{\partial x_3}\frac{\partial g(u)}{\partial x_1} & \frac{\partial g(u)}{\partial x_3}\frac{\partial g(u)}{\partial x_2} & \left(\frac{\partial g(u)}{\partial x_3}\right)^2 \end{pmatrix} \mathrm{d}u$$

$\mathcal{J}$ is a real symmetric matrix consisting of the following elements

$$\mathcal{J}_{pq} = \int_{U(p_0)} h(p_0 - u) \left(\frac{\partial g(u)}{\partial x_p}\frac{\partial g(u)}{\partial x_q}\right) \mathrm{d}u \tag{3.26}$$

and so the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are real and the corresponding eigenvectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ can be chosen to form an orthonormal basis (section 2.2).

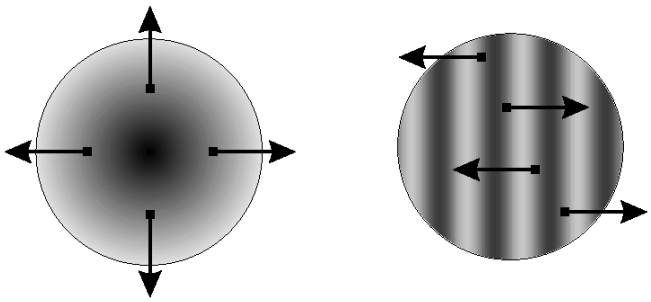
In the neighborhood $U(p_0)$, the line passing through p_0 in the direction of the *principal eigenvector* $\boldsymbol{e}_1$ of the structure tensor exhibits the least deviation from *all* lines in U defined by points $p \in U$ and local gradients $\nabla g(p)$ [33]. The principal eigenvector has therefore been proposed [42] to describe the orientation of the neighborhood U.

There are three significant issues which make a neighborhood-orientation analysis based on structure tensors superior to an analysis based on the gradient field.

1. An integral of the gradient field over a neighborhood results to a zero vector in those cases where the intensity distribution is constant, isotropic (Fig. 3.3(a)), and ideally oriented (Fig. 3.3(b)). A distinction among these three cases based on the gradient integration is therefore not possible.

2. A local classification of intensity distributions is possible due to the rank of the structure tensor. The patterns which can be identified in this way are summarized in Table 3.1.

3. There is more information in the eigensystem of the structure tensor: the eigenvector associated with the smallest eigenvalue gives the direction of the *minimal* intensity change.



(a) (b)

Figure 3.3: An isotropic neighborhood (a) and a neighborhood with an ideal orientation (b). The integral of the gradient field over neighborhood vanishes in both cases. Images by Haußecker and Jähne [19].

rank $(\mathcal{J})$	3D object	3D texture
0	constant value	constant value
1	boundary	layered
2	edge	extruded
3	corner	isotropic noise

Table 3.1: Structure-tensor-based analysis of intensity distributions.

We recall these properties in more details at a more appropriate place – in Chapter 6.

Practical applications of the 3×3 structure-tensor analysis cover segmentation [95] and 2D-motion analysis in spatio-temporal images [19, 33, 34].

3.3 2nd-order derivatives in volume analysis

In image processing, higher-order derivatives are used to design filters for ridges (2nd-order), corners (2nd-order), T-junctions (3rd-order) and cross-junctions (4th-order) [48, 49, 86]. Although a volume can be considered as a direct generalization of a 2D gray-level image, there are only few methods introduced to volume analysis which exploit second-order derivatives. To our knowledge, there are currently no applications which use 3rd-order or higher-order derivatives.

In the following we summarize the volume analysis techniques which deal with 2nd-order derivatives. The applications, as grouped by applica-

bility, aim on boundary detection (section 3.3.1), structure-filtering (section 3.3.2) and isosurface emphasis (sections 3.3.3 and 3.3.4) and illustration (section 3.3.5).

3.3.1 Laplacian operator

Whenever an intensity change occurs in one direction, there will be a corresponding peak in the first directional derivative or equivalently, a zero-crossing in the second directional derivative. The task of detecting boundaries can thus be reduced to the task of finding the zero-crossings of the second derivative in an "appropriate direction" [12, p. 81].

To find such a direction it would be necessary to compute a number of directional derivatives which would involve several convolutions (see section 3.3.2). For performance reasons, it would be convenient if the number of convolutions could be reduced to one orientation-independent operator. This immediately points towards the Laplacian:

$$\nabla^2 g = \nabla \cdot (\nabla g) = \text{div}\ (\nabla g) = \sum_{i=1}^{3} \partial^2 g / \partial x_i^2 \tag{3.27}$$

The zero responses to the Laplacian operator correspond to the zero-crossings, i.e., to the boundaries and historically have received a lot of attention due to the work of Marr [56].

Although in volume visualization boundaries are most-frequently detected due to the gradient magnitude (section 3.2.1) there are several facts which make the Laplacian-based detection interesting.

Firstly, it is the computation cost. The computation of the gradient magnitude requires three convolution-passes to compute the three partial derivatives, followed by three multiplications, two additions and one square root. The term of Laplacian, on the other hand, only involves linear operations. That means, only one convolution pass is necessary to compute the responses to the Laplacian operator. The disadvantage of the Laplacian operator, however, arises if the input data set is too noisy. Since the response to kernels of a small width (e.g., 3 voxels) is too sensitive to noise, a pre-smoothing with the Gaussian filter is strongly recommended, leading to a larger kernel – the Laplacian of Gaussian.

Secondly it is the way how the responses are further processed. Problems with the gradient-based detection arise when the boundary of an object features variations in gradient magnitude. Thresholding in such a case may,

depending on the actual threshold, lead to broken contours of the object. The more costly search for the zero-crossings in responses to the Laplacian operator, on the other hand, solves this problem and leaves the contours closed.

3.3.2 Hessian matrix

The Hessian matrix is a matrix comprised of the 2nd-order partial derivatives:

$$\mathbf{H}g = \begin{bmatrix} \partial g^2/\partial x_1^2 & \partial g^2/\partial x_1 \partial x_2 & \partial g^2/\partial x_1 \partial x_3 \\ \partial g^2/\partial x_2 \partial x_1 & \partial g^2/\partial x_2^2 & \partial g^2/\partial x_2 \partial x_3 \\ \partial g^2/\partial x_3 \partial x_1 & \partial g^2/\partial x_3 \partial x_2 & \partial g^2/\partial x_3^2 \end{bmatrix} \tag{3.28}$$

The identity

$$\nabla^2 g = \text{trace}\,(\mathbf{H}g) \tag{3.29}$$

means that the diagonal elements of the Hessian matrix can be used for filtering of boundaries as explained in the previous section.

Since the order of differentiation is interchangeable, i.e., $\partial g^2/\partial x_i \partial x_j = \partial g^2/\partial x_j \partial x_i$, the Hessian matrix is symmetric and has real eigenvalues and real orthogonal eigenvectors (see section 2.2). The associated quadratic form yields the second derivative in a direction represented by unit vector $\boldsymbol{v}$

$$\frac{d^2 g}{d\boldsymbol{v}^2} = \boldsymbol{v}^T \mathbf{H}g\ \boldsymbol{v} \tag{3.30}$$

In the following we show that the diagonalization of the Hessian matrix yields more information.

Eigensystem of the Hessian matrix

The Principal Axes Theorem (2.2.3) provides the following decomposition of the Hessian matrix:

$$\mathbf{H}g = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{\mathbf{T}} = \begin{pmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{pmatrix} \tag{3.31}$$

where the $\boldsymbol{e_i}$ are the orthonormal eigenvectors and $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the corresponding eigenvalues.

With this decomposition computation of the Laplacian and the second directional derivatives is still possible. Since the trace of a matrix is an invariant under a similarity transformation $\mathbf{P}$, Equation (3.29) rewrites as

$$\nabla^2 g = \text{trace}\left(\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{\mathbf{T}}\right) = \text{trace}\left(\mathbf{\Lambda}\right) = \lambda_1 + \lambda_2 + \lambda_3 \tag{3.32}$$

that is, having only the eigenvalues stored, we can use their sum for boundary detection (section 3.3.1).

From Theorem (2.3.1) it follows, that at a fixed point x_0 the second derivatives in all possible unit directions v are bounded by eigenvalues of the Hessian matrix

$$\lambda_3 = \frac{d^2 g(x)}{d\boldsymbol{e}_3^2} \leq \frac{d^2 g(x)}{d\boldsymbol{v}^2} \leq \frac{d^2 g(x)}{d\boldsymbol{e}_1^2} = \lambda_1 \tag{3.33}$$

Equation (3.30) for computation of a directional derivative rewrites as

$$\frac{d^2 g}{d\boldsymbol{v}^2} = (\boldsymbol{v}^T\mathbf{P})\mathbf{\Lambda}(\mathbf{P}^{\mathbf{T}}\boldsymbol{v}) = \sum_{i=1}^{3} \lambda_i u_i^2 \qquad \text{where } \boldsymbol{u} = \mathbf{P}^{\mathbf{T}}\boldsymbol{v} \tag{3.34}$$

In the following we give an overview of further usages of Hessian's eigensystem.

In order to segment *bright* curvilinear structures in volume data, Sato et al. [81] introduce the similarity-to-line concept – a weighting function of eigenvalues which emphasizes the areas where

$$\lambda_3 \approx \lambda_2 \ll \lambda_1 \approx 0 \tag{3.35}$$

Frangi et al. [13] extend this condition in two ways. Firstly they also deal with darker-than-background objects. Secondly they generalize the concept for tubular structures to enable the detection of blob- and sheet-like structures. Table 3.2 lists the conditions corresponding to the respective patterns.

The conditions corresponding to bright structures have been adopted in the continued work of Sato et al. [82] which addresses the problem of tissue classification in medical data. A 5-dimensional transfer function is introduced including the data value g, the gradient magnitude $\|\nabla g\|$, and the three similarity-to-a-structure functions of eigenvalues:

$$\begin{aligned} \mathcal{S}_{line} &= |\lambda_3| \cdot \psi(\lambda_2, \lambda_3) \cdot \omega(\lambda_1, \lambda_2) \\ \mathcal{S}_{blob} &= |\lambda_3| \cdot \psi(\lambda_2, \lambda_3) \cdot \psi(\lambda_1, \lambda_2) \\ \mathcal{S}_{sheet} &= |\lambda_3| \cdot \omega(\lambda_2, \lambda_3) \cdot \omega(\lambda_1, \lambda_2) \end{aligned} \tag{3.36}$$

	$\lambda_3 \simeq \lambda_2 \simeq \lambda_1$	$\ll 0$	bright blob-like structure
	$\lambda_3 \simeq \lambda_2 \ll \lambda_1$	$\simeq 0$	bright tubular-like structure
	$\lambda_3 \ll \lambda_2 \simeq \lambda_1$	$\simeq 0$	bright sheet-like structure
$0 \simeq$	$\lambda_3 \simeq \lambda_2 \simeq \lambda_1$	$\simeq 0$	noisy or homogeneous area
$0 \simeq$	$\lambda_3 \simeq \lambda_2 \ll \lambda_1$		dark sheet-like structure
$0 \simeq$	$\lambda_3 \ll \lambda_2 \simeq \lambda_1$		dark tubular-like structure
$0 \ll$	$\lambda_3 \simeq \lambda_2 \simeq \lambda_1$		dark blob-like structure

Table 3.2: Classification of the blob-, tubular-, and sheet-like structures.

where the real functions $\psi(\lambda_s, \lambda_t), \omega(\lambda_s, \lambda_t)$ map pairs of eigenvalues onto the interval $\langle 0, 1 \rangle$ and model the inequalities $\lambda_s \simeq \lambda_t$ and $\lambda_s \ll \lambda_t \simeq 0$ occurring in Table 3.2. The usability of such a kind of filtering is apparent when compared to a single channel (i.e., opacity) classification (see Figure 3.4). The drawback of the method is that the user has to specify a transfer function in five dimensions.

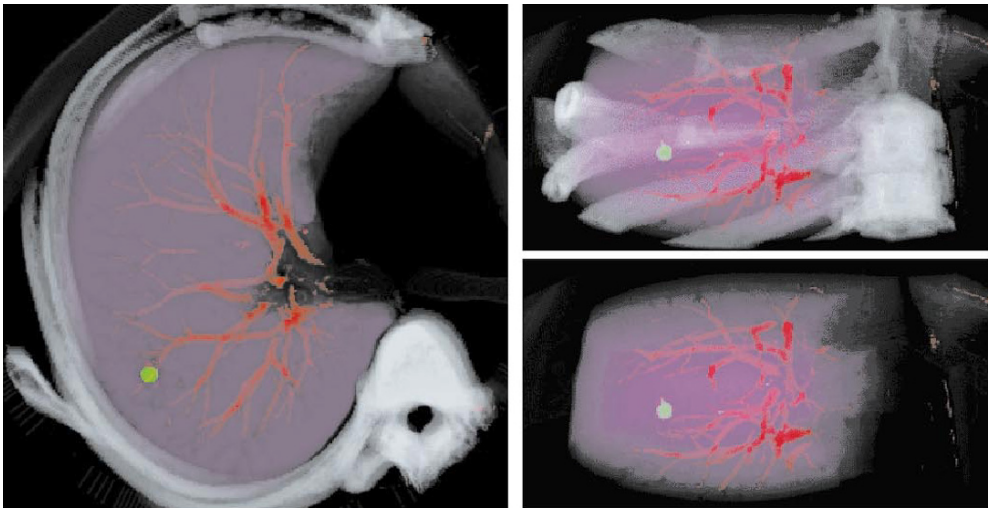
Our contributions [26, 27, 23, 24] to applying the eigenvalues of Hessian for volume analysis are discussed in Chapter 5.

3.3.3 Relative position across the boundary

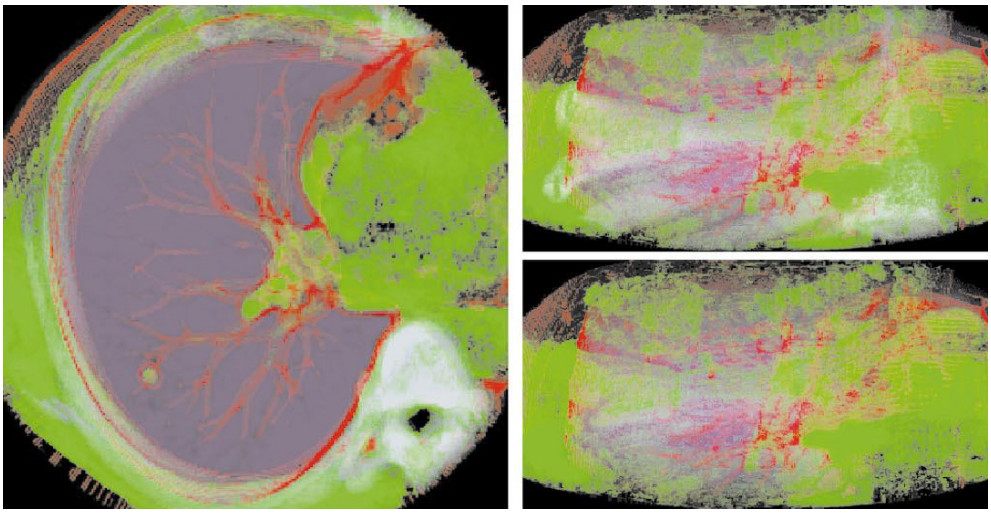
Boundary detectors locate boundaries in the spatial domain. In contrast, volume rendering applications often require to emphasize boundary regions and isosurfaces only in dependence on the data values, that is, irrespective of the spatial position. This problem is referred to as specification of an *opacity transfer function*. With respect to the large number of possible configurations the problem of transfer function specification is, in spite of ongoing research (revisited in Chapter 4), known to be very complex.

An outstanding approach is the analytical method by Kindlmann and Durkin [39]. The algorithm they proposed gained great popularity in recent years, and for that reason we will present it in more details.

To provide the user with a convenient tool for transfer-function specification, Kindlmann and Durkin [39, 38] introduce, for each scalar value g_i, the relative *position across the boundary* p. The signed value of $p(g_i)$ tells on which side and how far from the nearest boundary the scalar value g_i on average tends to fall. This information is further passed to an easy-to-define *boundary emphasis function*, b. The function b allows the user to control whether rendered boundaries will appear thick or thin, sharp or fuzzy, and



(a)

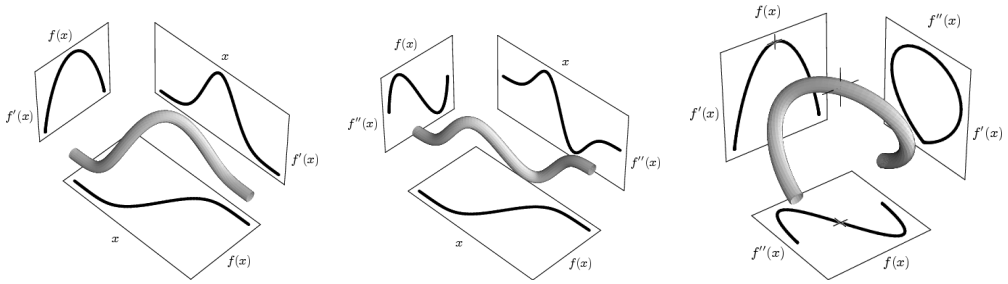


(b)

Figure 3.4: Classification due to 3D local structures (a) and due to only original scalar values (b). Images show a CT-slice of a human lung in the left part and direct volume rendering from two viewpoints in the right part. Images by Sato et al. [82].

to control the proximity of the rendered boundary to the object interior. Concatenation $\alpha = b \circ p$ gives the desired opacity transfer function.

The computation of the function p is based on an analysis of how the data values change in the direction of the isosurface normal using the first and the second directional derivatives. Along the normal, these changes can be expressed as the derivatives $f'(x)$ and $f''(x)$ of a real function f of one variable x. The relationship between position, data value, first and second derivative is shown in Figure 3.5.



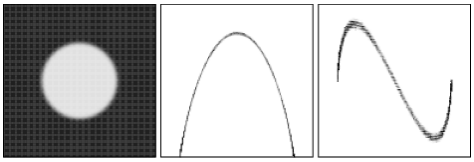
(a) **(b)** **(c)**

Figure 3.5: The relationship of the derivatives across a boundary, i.e., in the direction of the gradient. The plots in (a) and (b) include the spatial coordinate x. The plot in (c) excludes it and demonstrates the relationship between zero-, first-, and second-derivative of the scalar data in the gradient direction. Images by Kindlmann and Durkin [39].

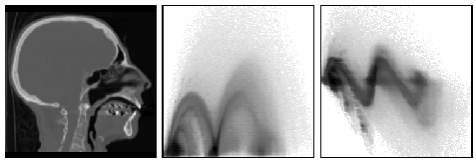
The basis for detection of boundaries resides in the position-independent spatial parametric curve in Figure 3.5(c): if a 3D record of the relationship between f, f', and f'' for a given data set exhibits curves of this type, it can be assumed that these curves are manifestations of boundaries in the volume.

To track this record, Kindlmann and Durkin [39] compute a 3D histogram with axes corresponding to f, f', and f''. Orthogonal projections of this 3D histogram yield scatterplots as those in Figure 3.6. If there was a boundary in the volume, the shapes of the scatterplots conform to the orthogonal projections of the parametric curve in Figure 3.5(c).

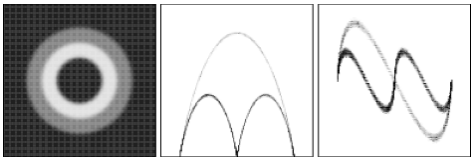
The final step is an analysis of the histogram for each of the scalar values g_i. The relative position p across boundaries given by isovalues g_i is defined as $p(g_i) = -\sigma^2\gamma_2(g_i)/\gamma_1(g_i)$ where γ_1, γ_2 are the *average* first and second directional derivatives measured for g_i over the entire data set, and σ is the average boundary thickness, specific to the acquisition device. The quantities γ_1, γ_2, and σ are easily read from the 3D histogram [39].



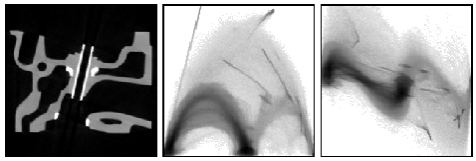
(a) Cylinder



(b) CT Head



(c) Two Nested Cylinders



(d) Engine Block

Figure 3.6: Data set slice (left images), and the orthogonal projections of the 3D histogram volume: f' versus f (middle images) and f'' versus f (right images). Images by Kindlmann and Durkin [39].

The method requires the computation of the directional derivatives in the direction of the surface normal $\boldsymbol{n}$. The first derivative is computed due to Equation (3.21): $f'(\boldsymbol{x}) = dg(\boldsymbol{x})/d\boldsymbol{n} = \|\nabla g(x)\|$. For the second derivative Kindlmann and Durkin discuss three possibilities, listed here in increasing order of numerical accuracy: 1. an approximation by the Laplacian: $f''(\boldsymbol{x}) \approx \sum_{i=1}^{3} \partial^2 g(\boldsymbol{x})/\partial x_i^2$, 2. the expression for the 1st derivative applied on a scalar field of gradient magnitudes yielding $f''(\boldsymbol{x}) = \nabla(\|\nabla g(\boldsymbol{x})\|).\nabla g(\boldsymbol{x}) \;/\; \|\nabla g(\boldsymbol{x})\|$, and 3. a computation of the quadratic form associated with the Hessian matrix (Eq. 3.30) yielding $f''(\boldsymbol{x}) = d^2 g(\boldsymbol{x})/d\boldsymbol{n}^2 = \boldsymbol{n}^T \mathbf{H} g(x) \boldsymbol{n}$.

3.3.4 The total gradient

In order to emphasize the isosurfaces S_i given by scalars g_i which not only exhibit sharp boundaries (as measured by the gradient magnitude) but also large areas, Bajaj et al. [2] proposed to emphasize those scalar values g_i with large *total gradient* (see also Fig. 3.7):

$$F(g_i) = \int_{S_i} \|\nabla g\| \, \mathrm{d}S_i \tag{3.37}$$

To compute the values of $F(.)$ efficiently for all scalar values, Pekar et al. [72] exploit the divergence theorem which rewrites the previous surface integral to a volume integral of the Laplacian:

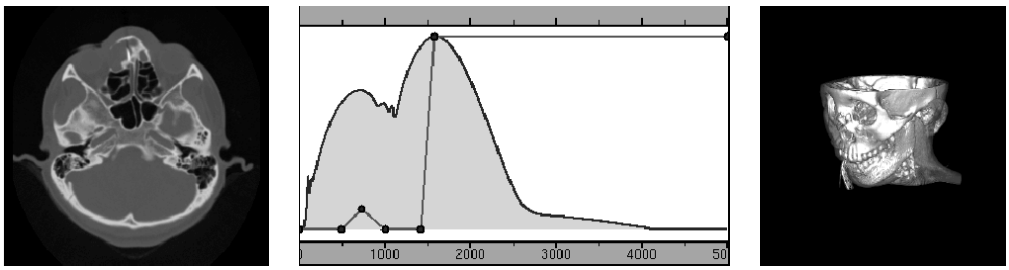
$$F(g_i) = \int_{S_i} \|\nabla g\| \, \mathrm{d}S_i = \int_{S_i} \nabla g^T \boldsymbol{n} \, \mathrm{d}S_i = -\int_{V_i} \mathrm{div}\,(\nabla g) \, \mathrm{d}V_i = -\int_{V_i} \nabla^2 g \, \mathrm{d}V_i \tag{3.38}$$

The negative sign in front of the volume integral is due to the direction of $\boldsymbol{n}$ which is opposite to the *outer* normal involved in the divergence theorem.

In discrete notation the previous equation rewrites as

$$F(g_i) = -\sum_{g(x) \geq g_i} \nabla^2 g(x) \tag{3.39}$$

and can effectively be computed using cumulative histograms [72].



(a) **(b)** **(c)**

Figure 3.7: Detection of meaningful isosurfaces due to the total gradient: One slice of a CT head data set (a), total gradient feature curve combined with the opacity transfer function (b) used for direct volume rendering of the data set (c). Images by Pekar et al. [72].

3.3.5 Curvature of an isosurface

Another descriptor of an isosurface comes from differential geometry. It is known that a regular patch can be, at a specific point P, locally described by two orthogonal tangent vectors $\boldsymbol{s_1}$, $\boldsymbol{s_2}$ referred to as the *principal directions* and by two real numbers κ_1,κ_2 referred to as the *principal curvatures*.

Principal directions and principal curvatures

The principal directions $\boldsymbol{s_1}$, $\boldsymbol{s_2}$ in point P are the directions of the maximal and minimal bending of a surface in P and the (signed!) values of κ_1, κ_2 give the corresponding *quantitative* curvature measures.

They are determined as the eigensystem of the *Second Fundamental Form*, II_P, of the surface in P. The second fundamental form is a quadratic form of two variables which maps *unit* tangents $\boldsymbol{t}_i$, expressed in the tangent plane as 2D vectors, to the corresponding *normal curvatures*:

$$\mathrm{II}_P(\boldsymbol{t}_i) = \boldsymbol{t_i}^T \begin{bmatrix} L & M \\ M & N \end{bmatrix} \boldsymbol{t_i} \tag{3.40}$$

where the coefficients can be computed, with respect to any orthogonal frame $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{n}$ as follows [11, p. 352]:

$$L = -\boldsymbol{e}_1 \cdot (\partial \boldsymbol{n} / \partial \boldsymbol{e}_1) \tag{3.41}$$

$$M = -\tfrac{1}{2} \left[\boldsymbol{e}_1 \cdot (\partial \boldsymbol{n} / \partial \boldsymbol{e}_2) + \boldsymbol{e}_2 \cdot (\partial \boldsymbol{n} / \partial \boldsymbol{e}_1) \right] \tag{3.42}$$

$$N = -\boldsymbol{e}_2 \cdot (\partial \boldsymbol{n} / \partial \boldsymbol{e}_2) \tag{3.43}$$

Probably the most impressive application of the above concepts to volume rendering is illustration of overlapping isosurfaces by Interrante [32, 31]. To increase comprehensiveness of overlapping isosurfaces coming either from *marching cubes* [52] or from Levoy's definition [46] Interrante illustrates them with 3D texture mapping. The 3D texture consists of strokes representing a flow defined by the first principal directions $\boldsymbol{s_1}$ (Fig. 3.8). The strokes are generated by distributing a randomly generated spot noise through line integral convolution (LIC) [4]. Interrante further proposes to refine the texture adjusting the lengths and/or widths of the strokes in accordance to the magnitude of the first principal curvature $|\kappa_1|$.

We propose [25] transfer functions defined in the domain of the principal curvatures. This work is discussed in Chapter 4.

Gaussian curvature and mean curvature

Gaussian curvature

$$K = \kappa_1 \kappa_2 \tag{3.44}$$

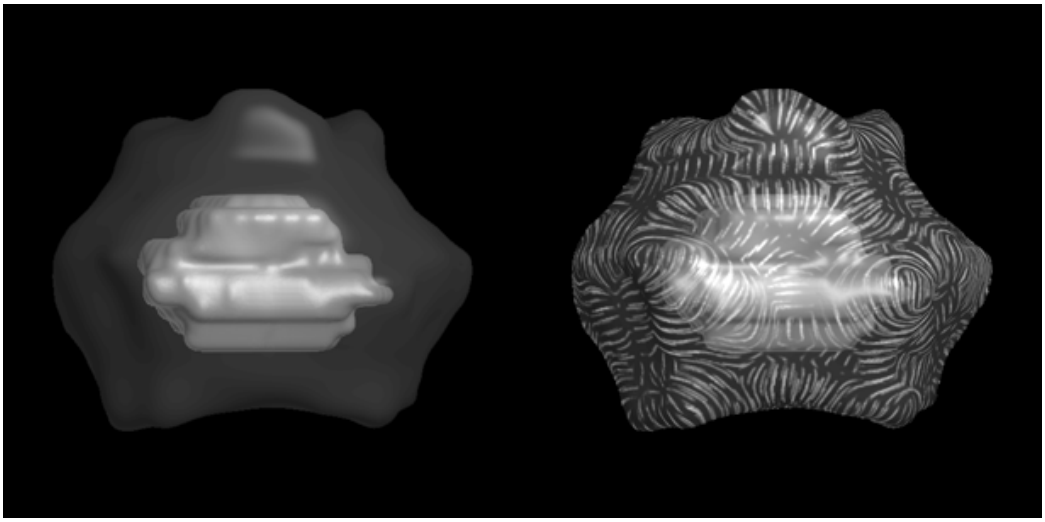


Figure 3.8: Illustrating isosurfaces using principal directions. Images by Interrante [31].

and the mean curvature

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) \tag{3.45}$$

are an alternative way of a coordinate–independent description of a surface. At a specific point, a surface can be locally approximated, up to the second order, by a quadratic patch. The signs of the Gaussian curvature and the mean curvature indicate whether this patch will (locally) be a plane ($H = K = 0$), parabolic cylinder ($H \neq K = 0$), paraboloid ($K > 0$), or a hyperbolic paraboloid ($K < 0$).

In visualization, the information on the local surface type is usually encoded in color [50, 85]. Tools for isosurface emphasis may consider the curvatures to enhance those isosurfaces, for instance, which show up only a little bending [72]. We introduced a novel concept of transfer functions which involves the curvature magnitudes [25].

The Gaussian and the mean curvatures can be computed according to the definitions (3.44) and (3.45) respectively, after computing the the eigenvalues of the second fundamental form (3.40). Alternatively, they can be computed directly from the scalar field due to the components of the gradient and

Hessian matrix [51, Chap. 7]. Abbreviating the terms $\partial g/\partial x_i$ with g_i and $\partial^2 g/\partial x_i \partial x_j$ with g_{ij} yields:

$$\begin{aligned} \|\nabla g\|^4 K &= g_1^2 (g_{22}g_{33} - g_{23}^2) + 2g_2 g_3 (g_{13}g_{12} - g_{11}g_{23}) + \\ &\quad g_2^2 (g_{11}g_{33} - g_{13}^2) + 2g_1 g_3 (g_{23}g_{12} - g_{22}g_{13}) + \\ &\quad g_3^2 (g_{11}g_{22} - g_{12}^2) + 2g_1 g_2 (g_{13}g_{23} - g_{33}g_{12}) \end{aligned} \tag{3.46}$$

$$\begin{aligned} 2\|\nabla g\|^3 H &= g_1^2 (g_{22} + g_{33}) - 2g_2 g_3 g_{23} + \\ &\quad g_2^2 (g_{11} + g_{33}) - 2g_1 g_3 g_{13} + \\ &\quad g_3^2 (g_{11} + g_{22}) - 2g_1 g_2 g_{12} \end{aligned} \tag{3.47}$$

Since the principal curvatures are also solutions to the quadratic equation $\kappa^2 - 2H\kappa + K = 0$ (compare also to Eqs. (3.44) and (3.45)), they can be computed after enumerating the Gaussian and mean curvatures.

$$\kappa_{1,2} = H \pm \sqrt{H^2 - K} \tag{3.48}$$

All of the terms require the first-, and the second-order derivatives of the scalar field g. The numerical accuracy of the curvature-related quantities thus crucially depends on the underlying differentiation filters. Indeed, Truco and Fisher [92] report that qualitative curvature properties, i.e. the *sign* of Gaussian and the mean curvature can be more reliably estimated than the quantitative ones, i.e. curvature *magnitude*.

An alternative way of computing the curvature properties is due to approximation of the isosurface by a patch with well-known curvature characteristics. In our early work [25] we proposed an algorithm which only requires a reconstruction of eight isosurface points from the vicinity of the inspected point P. Together with the work on curvature-based transfer functions, it is dealt with in the next chapter.

Chapter 4

Curvature-based transfer functions for direct volume rendering

In this chapter we give an updated version of our early work on transfer functions [25]. In contrast to the usual concepts, we defined the transfer functions in the domain of principal-curvature magnitudes. Such a definition is intended to provide the user with a tool for enhancement or suppression of specific shape-classes and for specification of smooth color/opacity transitions within thick surfaces and solid objects. As compared to the density-based transfer functions, the domain of principal curvatures has a unique interpretation and the transfer functions therefore require less specification and user interaction.

4.1 Introduction

In direct volume rendering, the transfer function is responsible for the classification of the data set. Its task is to assign optical properties to values the data set consists of. During the rendering process, the sampled and/or reconstructed data values are passed through the transfer function to determine their contribution to the final image.

Generally, we can think of a transfer function as a mapping from a Cartesian product of scalar fields F to a Cartesian product of optical properties O (Fig. 4.1):

$$\tau : F_1 \times F_2 \times \cdots \times F_n \longrightarrow O_1 \times O_2 \times \cdots \times O_m$$

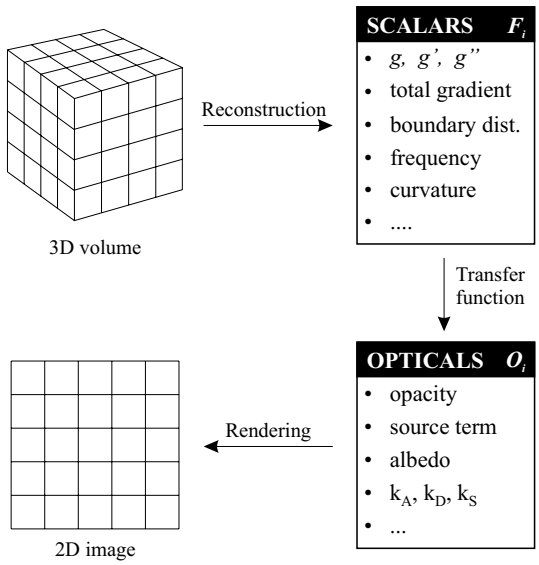


Figure 4.1: Transfer function: its task, domain and range.

Since a specification of transfer functions is a challenging task, the values of n and m are usually kept small in practice. Typically, a transfer function maps density values ($n = 1$) to opacity and color ($m = 2$), while other optical properties are determined by an illumination model. More sophisticated transfer functions also include the gradient magnitude ($n = 2$) in the domain of the transfer function [46, 39]. From a user's point of view, even in this restrictive case ($n, m \leq 2$), it is a problem to specify an appropriate transfer function. The panel discussion "The Transfer Function Bake-Off" at the IEEE Visualization 2000 conference underlined this fact again (for written material see [74]).

There are systems which analyze the input data [2, 39, 14, 72] or output images [10] to provide the user with an initial, easy to customize transfer-function setup. Alternative systems generate an initial set of transfer functions and pass it to an evolution mechanism [20] or arrange pre-rendered results to provide the user with an overview of possibilities, hence an easier choice of an appropriate transfer function [55, 43]. Such an approach requires much computational time to provide a preview image for each generated transfer function, which made these methods non-interactive for a long time. With current rendering hardware [73], however, the usefulness of these interfaces has increased and the specification problem is facilitated. This progress made us thinking of alternative transfer-function types.

According to Lichtenbelt et al. [47], the more general ($m > 2$) transfer functions are those that assign opacity, color, and emittance. Other possibilities to extend the *range* of transfer functions can be found by studying optical models [58] (see also Fig. 4.1). Our approach attempts to extend the *domain* of a transfer function. As already mentioned, the typical mapping is defined over densities and gradient magnitudes. The possible, yet not complete list of other choices, can be found in Fig. 4.1. To emphasize surfaces, for instance, Kindlmann and Durkin [39] define the transfer functions with the help of the first and the second derivatives in the direction of the gradient (see also section 3.3.3). Three years later, Kniss et al. [41] present a powerful user interface for this concept. Although not introduced as a transfer-function approach, Lürig and Ertl [53] involve frequency information to visualize the thickness of objects.

The domain of transfer functions presented in this paper is defined by the magnitudes of the principal curvatures.

From differential geometry it is known, that the vicinity of any point on a regular surface can be described by two tangent vectors - *principal directions* and two corresponding real numbers - *principal curvatures*. This description yields a unique, view-independent characterization.

Although originally developed for smooth analytic surfaces, in recent years curvature information is also used in a variety of applications in the field of volume visualization. An obvious application is to use *Gaussian curvature* to distinguish among cylindrical, parabolic, and hyperbolic parts of surfaces. Interrante [31, 32] has used the *principal directions* to define a flow field over a surface to accentuate its shape. Trucco and Fisher [92] segment the sampled data with the help of both Gaussian and *mean curvatures*. Tang and Medioni [85] extend the tensor voting mechanism by *curvature sign* information to get better densification (i.e. reconstruction) of sparse input data.

The rest of this chapter is organized as follows. In the next section we present our definition of transfer functions which involves the principal curvatures. Computation of curvatures due to derivatives has been discussed in section 3.3.5 Here, in section 4.3, we present another mechanism for their computation. The results section 4.4 demonstrates an application of the initial, automatically generated transfer function. Conclusions and hints for future research are given in section 4.5.

4.2 Curvature-based transfer functions

At a specific point P on a regular surface, the principal directions $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ provide information on where the surface bends the most and the least, respectively. The corresponding quantitative measure, i.e., *how much* the normal vector changes in these directions is expressed by two real numbers $\kappa_1 \geq \kappa_2$ known as the principal curvatures.

With the help of principal directions and curvatures, the surface can be locally approximated, up to the second order, by a quadratic patch. The type of the patch is indicated by the signs of κ_1 and κ_2 as follows:

plane if $\kappa_1 = \kappa_2 = 0$.
The point P is referred to as *planar*.

parabolic cylinder if $\kappa_1 > \kappa_2 = 0$ or $0 = \kappa_1 > \kappa_2$.
The point P is referred to as *parabolic*.

paraboloid if $\kappa_1.\kappa_2 > 0$.
The point P is referred to as *elliptic*.

hyperbolic paraboloid if $\kappa_1.\kappa_2 < 0$.
The point P is referred to as *hyperbolic*.

The transfer function we are going to design will map pairs of principal curvatures to optical properties, e.g. color and opacity in the RGBα model:

$$\tau : \kappa_1 \times \kappa_2 \longrightarrow R \times G \times B \times \alpha$$

We expect such a definition to be useful to

distinguish among shapes. In specific applications, it is useful to visualize surfaces with respect to their shape. This does not include only different properties of the four different cases introduced above. Our approach allows also to distinguish shapes of the same class employing curvature magnitudes. In engineering for instance, it can be distinguished between planar ($\kappa_1 = \kappa_2 = 0$) and tubular ($\kappa_1 > \kappa_2 = 0$) structures. In addition, within the class of tubular structures, different properties can be specified with respect to the magnitude of κ_1 (i.e with respect to the cylinder's radius). Using the same principle, a medical application may be able to suppress registration markers which are typically of tubular (chord) or planar (landmarker) shape. Surgical tools can also be selected, if they have specific shape properties. Finally, several human organs, like bones, vessels or colon polyps might be segmented because of their specific shape.

set smooth transitions. Thick boundaries, i.e., surfaces featuring a slow density transition along the gradient can be understood as a set of coherent layers. Within such a surface it might be useful to see how the curvature (hence shape) changes inside. To convey this, Interrante [31] exploits principal directions to define a texture. In her approach, however, a limited number of layers can be visualized simultaneously without loss of comprehensiveness. Apart of that, this method could be hardly applicable for small structures, e.g. blood vessels. Involving curvature magnitudes, smooth color transition within even small solid objects (say five or six voxels in diameter) can be set, allowing to understand what happens inside. A possible application in medicine is the identification of stenoses.

set the transfer function (semi)automatically. To enhance or suppress a specific shape class, it is evident what combination of values κ_1, κ_2 has to be chosen. In order to provide flexibility with respect to shape-by-magnitude distinction and color transition for application specific tasks, however, a simple user interface is necessary. The setup issues will be briefly discussed in section 4.4.

Being dependent on two real numbers it would be necessary to specify the proposed transfer function in the entire plane $\mathbb{R}^2$. Fortunately, the following facts gradually allow us to restrict the domain:

1. For analytically defined patches, curvatures are by definition given such that $\kappa_1 \geq \kappa_2$ (see also section 4.3.1). The convexity or concavity of paraboloids and parabolic cylinders is determined by the curvature sign(s).

2. For surfaces defined implicitly, the orientation of the normal vector is ambiguous. At a boundary, for instance, we only can decide on whether the gradient corresponds to the outer or the inner normal if we know whether the objects feature higher scalar values then background or vice versa. For the same reason there is ambiguity in the convexity/concavity of paraboloids and parabolic cylinders. Without the a priori information on object-versus-background intensities it is impossible to distinguish the cases $\kappa_1 > \kappa_2 \geq 0$ and $0 \geq \kappa_1 > \kappa_2$. The signs of κ_1, κ_2 surely identify only the planar (both are zero) and hyperbolic (they straddle zero) cases. Therefore we can rearrange the principal curvatures such that κ_1 is nonnegative and reflects the faster bending of the surface. This is simply done by a sign change and a

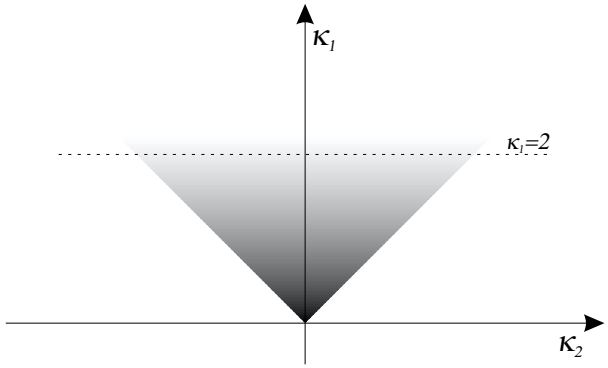


Figure 4.2: The domain of curvature-based transfer functions.

swap of curvatures in points where $|\kappa_1| < \kappa_2$. Such a rearrangement ensures, for all possible cases, that $\kappa_1 \geq |\kappa_2|$.

3. The principal curvature, as a curvature of a planar curve, is defined as the reciprocal of the radius of its osculating circle. Since in unit-distance Cartesian grids we can surely assume that circles with radius smaller than 1/2 do not exist, the curvature magnitudes will be always less than two.

Due to 2) and 3) the domain of transfer functions shrinks from $\mathbb{R}^2$ to

$$|\kappa_2| \leq \kappa_1 < 2 \tag{4.1}$$

Referring to Fig. 4.2, the origin corresponds to planar points, the positive κ_1 axis to parabolic points, and the areas to the right and the left correspond to elliptic and hyperbolic points, respectively. This layout will serve as a base for specification discussed in section 4.4.

For analytically defined patches, we could start a discussion on interface setup. An accurate estimation of the curvature magnitudes (our concept relies on) from digital scenes, however, is known to be a hard problem. The troubles in equations presented in section 3.3.5 come from numerical instabilities in the estimation of the normal and the second derivatives.

In the following section we describe an approach which avoids computing derivatives. The algorithm reduces the problem to a precise reconstruction of surface points in a local neighborhood, fitting osculating circles to these points, and fitting a central conic in the tangent plane.

4.3 Computation of principal curvatures

Despite big efforts in research on recovery of curvature information from sampled data, the results are still at least disputable. Particular success can be seen in the estimation of qualitative properties (the principal directions and the sign of the Gaussian curvature) rather than quantitative (the principal curvature magnitudes) [85]. The difficulties of magnitude estimation arise from the properties of digital scenes, mainly noise, anisotropy and related directional dependencies.

Methods which estimate the curvature of a surface can basically be divided into two groups. There are algorithms which estimate derivatives and apply the fundamental forms (section 3.3.5). These methods strongly rely on an accurate derivative reconstruction, are sensible to noise and require low-pass prefiltering with a large kernel. An alternative approach is the local fitting of a patch, from which the curvatures can be analytically computed. McIvor and Valkenburg [59] conclude that fitting of a quadratic patch gives better results than fitting of a surface of any other type.

In our implementation we have, for several reasons, adopted an algorithm for triangular meshes introduced by Todd and McLeod [91]. Firstly, the authors come up with a concept which gradually reduces the surface-curvature estimation to finding a set of *planar* curves, to estimation of their tangents and curvatures, and to fitting of a central conic. Neither derivative estimation in 3D nor patch fitting are therefore necessary. Due to the two-dimensionality of all these steps one can expect not only an easier implementation but also more reliable results. Secondly, the authors present results which are superior to those achieved by fitting of a quadratic interpolant [59]. Finally, in contrast to the methods which rely on the estimation of the surface normal, the normal is computed as a side product from the estimated curve tangents.

4.3.1 From surface to planar curves

At a fixed point P of a regular surface S, an arbitrary unit tangent vector $\boldsymbol{t}$ together with the surface normal $\boldsymbol{n}$ define a plane which in the vicinity of P meets S in a curve of intersection. The curvature $\kappa_n(\boldsymbol{t})$ of this curve is referred to as *normal curvature* in point P and direction $\boldsymbol{t}$. The normal curvature as a real function of $\boldsymbol{t}$ being defined on the compact set of unit tangent vectors reaches a maximum and a minimum. Directions in which this happens are known as the *principal directions* $\boldsymbol{s_1}$, $\boldsymbol{s_2}$. The corresponding curvatures $\kappa_1 = \kappa_n(\boldsymbol{s}_1)$, $\kappa_2 = \kappa_n(\boldsymbol{s}_2)$ are known as the *principal curvatures*.

The principal curvatures κ_1 and κ_2, we need to estimate, can also be found in the definition of the *Dupin indicatrix*. The Dupin indicatrix is either one or a pair of conics in the tangent plane defined, assuming an arbitrary orthonormal coordinate system in the tangent plane with origin at point P, by the following equation:

$$Lx^2 + 2Mxy + Ny^2 = \pm 1 \tag{4.2}$$

Changing the coordinate system such that the eigenvectors of the quadratic form (4.2) become its axes, however, the Dupin indicatrix will be expressed in a more convenient form:

$$\kappa_1 x^2 + \kappa_2 y^2 = \pm 1 \tag{4.3}$$

Therefore, if we know the Dupin indicatrix we also know the principal curvatures κ_1, κ_2.

In order to reconstruct the Dupin indicatrix, it is necessary to know at least three of its points. To compute them we define the following map:

$$D(\boldsymbol{t}) = \boldsymbol{t} \;/\; \sqrt{|\kappa_n(\boldsymbol{t})|} \tag{4.4}$$

This map scales each given unit tangent vector $\boldsymbol{t}$, in which the normal curvature $\kappa_n(\boldsymbol{t})$ is nonzero, to a positional vector of a point on the Dupin indicatrix. This can be proven with the help of the *Euler theorem*, which establishes a relation between principal curvatures and a normal curvature in an arbitrary direction $\boldsymbol{t}$:

$$\kappa_n(\boldsymbol{t}) = \kappa_1 \cos^2\varphi + \kappa_2 \sin^2\varphi$$

where $\varphi = \varphi(\boldsymbol{t})$ is the angle between $\boldsymbol{t}$ and $\boldsymbol{s_1}$. Taking the principal frame, unit vector $\boldsymbol{t}$ becomes $(\cos\varphi, \sin\varphi)$ and for its image $D(\boldsymbol{t}) = (D_x, D_y)$ holds:

$$\begin{aligned} \kappa_1 {D_x}^2 + \kappa_2 {D_y}^2 &= \frac{\kappa_1 \cos^2\varphi + \kappa_2 \sin^2\varphi}{|\kappa_n(\boldsymbol{t})|} = \frac{\kappa_n(\boldsymbol{t})}{|\kappa_n(\boldsymbol{t})|} = \\ &= sign\ \kappa_n(\boldsymbol{t}) = \pm 1 \end{aligned}$$

which corresponds to definition (4.3).

Taking k ($k \geq 3$) nonzero normal curvature estimates $\kappa_n(\boldsymbol{t}_i)$ in k distinct unit tangent directions $\boldsymbol{t_i}$, we can therefore reconstruct k points (x_i, y_i) on the Dupin indicatrix and set up a system of k equations

$$L{x_i}^2 + 2Mx_iy_i + N{y_i}^2 = sign\ \kappa_n(\boldsymbol{t}_i) \qquad i = 1 \ldots k \tag{4.5}$$

The coefficients L, M, N are found as a solution of a linear equations system ($k = 3$) or a least square fitting algorithm ($k > 3$). Principal curvatures

κ_1, κ_2 and principal directions $\boldsymbol{s_1}, \boldsymbol{s_2}$ are eigenvalues and eigenvectors of the quadratic form (4.2), i.e., of the matrix

$$\begin{bmatrix} L & M \\ M & N \end{bmatrix}$$

The estimation of the normal curvature in a given tangent vector would require firstly a knowledge of the surface normal $\boldsymbol{n}$ in P and secondly a reconstruction of a curve in the normal-section.

Instead, due to another result from differential geometry, we just need to reconstruct k arbitrary planar (i.e. not necessarily normal-section) curves γ_i passing through P and estimate their tangent vectors $\boldsymbol{t_i}$ and curvatures $\kappa(\boldsymbol{t_i})$. Averaging the cross products $\boldsymbol{t_i} \times \boldsymbol{t_j}$ of tangent vectors allows to compute the surface normal $\boldsymbol{n}$. Normal curvatures $\kappa_n(\boldsymbol{t}_i)$ can then be enumerated due to the *Meusnier theorem*:

$$\kappa_n(\boldsymbol{t}_i) = \kappa(\boldsymbol{t_i}) \cos\psi \tag{4.6}$$

where ψ denotes the angle between the plane of the normal section (given by vectors $\boldsymbol{n}, \boldsymbol{t_i}$) and the plane of curve γ_i.

In this section we have shown how to reduce the problem of principal-curvature estimation to curvature estimating of planar curves The entire procedure is summarized in Fig. 4.3.

4.3.2 Curvature of planar curves

The computation of the curvature of a digitized curve is a non-trivial task which should be considered with care [97]. In research on shape analysis of digital curves, Worring and Smeulders [98] identify five essentially different methods for measuring curvatures of digital curves. These methods are based on three different formulations of curvature: tangent orientation change, second derivative of the curve considered as a path, and osculating circle touching the curve. In their work the authors conclude, that none of the presented methods is robust and applicable for all curve types. They advice, however, which method outperforms the others for a specific application.

The very first aim of this work was to come up with a transfer function for visualization of tubular structures, which feature nearly constant and large radii. For this case, Worring and Smeulders [98] recommend to formulate the curvature with the help of osculating circles. In the following we describe our implementation for grid data sets.

To approximate curvatures κ_i of planar curves γ_i passing through P, we reconstruct points in the neighborhood of P from the isosurface defined

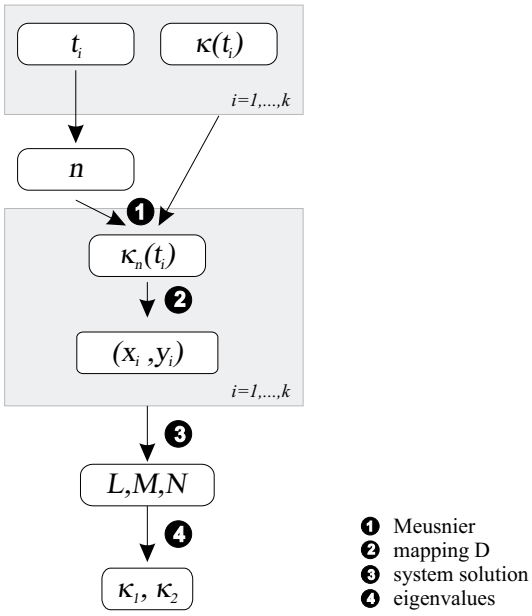


Figure 4.3: Calculating principal curvatures.

by the density value of P. The triplets consisting of P and two isosurface points will approximate the osculating circles. To avoid anisotropy typical for rectilinear data sets, these points should lie on a unit sphere with the center in P. In order to reduce reconstruction errors we advice to reconstruct these points via bilinear interpolation in four planes passing through adjacent grid points of P (Fig. 4.4). As a result we have eight surface points $P_1, \ldots, P_8$ in a small neighborhood of P. For $u \neq v$, each triplet of points P, P_u, P_v lie on some planar curve passing through P. There are two cases:

A) P, P_u, P_v are collinear and define a tangent vector $\boldsymbol{t_i}$ to the surface at P. The corresponding normal curvature $\kappa_n(\boldsymbol{t}_i)$ is zero and can therefore not be used to compute a point on the Dupin indicatrix according to the map defined by equation (4.4). This case provides, however, with a tangent vector and contributes as such to a better estimation of normal $\boldsymbol{n}$ which is necessary for the use of equation (4.6).

B) P, P_u, P_v approximate an osculating circle with the center C. In order to use the Meusnier theorem (4.6) we need to compute, in addition, a tangent vector $\boldsymbol{t_i}$ as a cross product $((P_u - P) \times (P_v - P)) \times (C - P)$ and the curvature $\kappa(\boldsymbol{t_i})$ as a reciprocal of the circle's radius, i.e., $1/\|C - P\|$.

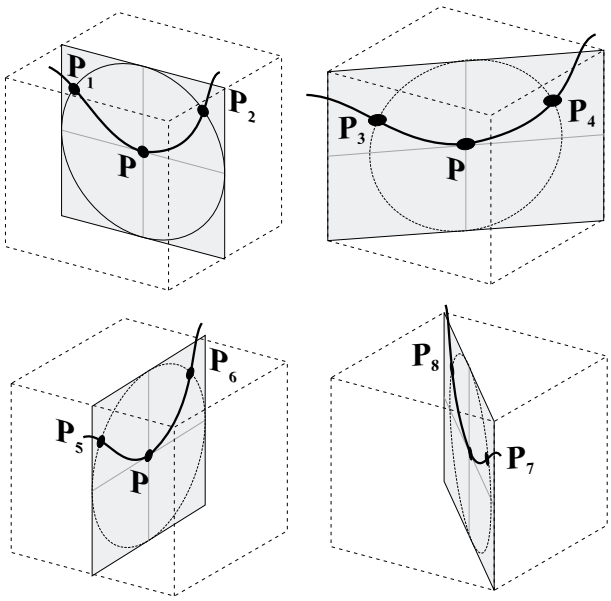


Figure 4.4: Reconstruction of neighborhood isosurface points.

4.3.3 Implementation issues

The definition of map (4.4) presumes a nonzero normal curvature κ_n. This is not fulfilled, however, for case A mentioned in the previous section. Here, such a triplet of points is of no use for the computation of a point on the Dupin indicatrix, and consequently does not contribute to the total number k of equations in system (4.5). In the worst case, e.g., for planar points where all the possible triplets are collinear, the total number k of equations can be less than three, which is not sufficient for finding the coefficients L, M, N. To circumvent this difficulty and, at the same time, to handle all the cases uniformly, we reassign the zero curvature κ_n to some small constant ε. In order to avoid numerical problems, this constant should not be too small. On the other hand it should sufficiently reflect the planarity of the neighborhood of P. For low resolution (e.g. 128^3) volumes we have successfully (Fig. 4.6(d)) set $\varepsilon = 10^{-4}$ which corresponds to curves of sufficiently large radii of 10000 voxels.

The algorithm described in this section is computationally expensive. A possible part for acceleration seems to be the reconstruction of neighborhood points. As we show, the curvature estimation is very sensitive to how this reconstruction is done, therefore this should be considered with care.

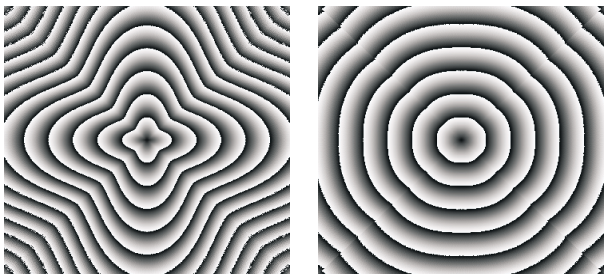


Figure 4.5: Influence of interpolation on curvature estimation.

Having a 3×3 density matrix with the center in P in the reconstruction plane, a first simplification might be achieved interpolating just from the 4-neighbors. The second one would be to find the isovalues on a diamond ($|x| + |y| = 1$) rather than on the unit circle with center in P.

To demonstrate the influence of improper interpolation on curvature estimation, we present a middle slice of the first principal curvatures κ_1 reconstructed from a 361×361×3 volume of concentric cylinders (Fig. 4.5). In order to see the curvature isolines we depict the intensities of $1/\kappa_1$ *mod* 32. Where concentric circles are expected, the left image exhibits an anisotropy with the maximum in diagonal directions. This is a consequence of interpolating just from four neighbors. The situation improves considerably using all eight neighbors for bilinear interpolation. The remaining artifacts appearing in the diagonal direction of the right image have been caused by an approximation of the circle by a diamond.

4.4 Results

To demonstrate our new concept we refer to Figures 4.6 and 4.7. Fig. 4.6(a) depicts an example of a transfer-function specification scheme with respect to the definition (4.1) of its domain (see also Fig. 4.2).

Recalling the distinction of the four shape classes introduced in section 4.2 one would expect an exact segmentation of the transfer-function domain. For practical applications, however, we find it useful to provide the user with a certain degree of tolerance. Consequently, the sharp borders between shape classes change to *transition areas*.

The green area in the vicinity of the origin corresponds to planar points. The blue-yellow transition area specifies the curvature change inside

parabolic structures and is intended to reflect the diameter change within solid cylinders present in the data sets. The red area corresponds to elliptic points.

Slight alterations of this specification are used to render the following density volumes, generated by the *vxt* library [84]:

A wire frame cube (Fig. 4.6(b)). This is a demonstration of the curvature change inside solid cylinders ($\kappa_1 > \kappa_2 \approx 0$) of a 38×38×38 cube. Note, that the diameter of cylinders in the data set is less than six voxels. In order to attract the user's attention, the high values of κ_1 (i.e. small diameters) have been mapped to bright yellow. The smooth transition to blue towards lower values of κ_1 corresponds to diameter increase. As the axes of cylinders do not define a surface, they have been excluded from curvature computation and therefore do not affect the final image. The red parts correspond to elliptic points ($\kappa_1 \geq \kappa_2 > 0$).

A wire frame octahedron (Fig. 4.6(c)). The transfer function has been specified in the same way as for Fig. 4.6(b), with more emphasis on smaller cylinders (depicted in yellow). A staircase effect in diagonal directions can be noticed. The resolution of the data set is 59×59×59 voxels.

A facet cube (Fig. 4.6(d)). A 38×38×38 data set similar to that used in Fig. 4.6(b) in this case with attached faces is shown. The transfer function maps the corresponding (i.e. zero) curvatures to transparent green. The joint of faces with cylinders was not smooth and exhibits therefore high curvature depicted in yellow. Similarly as in Fig. 4.6(b), the red areas correspond to elliptic points.

Transfer functions used for rendering of figures 4.7(a) and 4.7(b) additionally require a specification also in the area of hyperbolic points ($\kappa_2 < 0$):

A torus (Fig. 4.7(a)). The transfer function has been set to distinguish among elliptic (red), parabolic (green) and hyperbolic (blue) points of a 59×59×20 torus. The green points on the outer side are identified as planar due to a volume crop.

The Möbius strip (Fig. 4.7(b)). Visualization of low (green) and high curvature (red) points of a 50×52×16 thickened Möbius strip.

The reconstruction of curvature using the method described in section 4.3 involves several steps. Its time complexity depends mainly on how many

plane curvatures (i.e. equations of system (4.5)) are reconstructed, and how the points on the isosurface are reconstructed. In the discussion in section 4.3.3 we gave arguments why the reconstruction of curve points should be done as accurately as possible. Therefore we do not encourage to save time there. Instead, time should be saved adapting the number k of equations in the system (4.5). To demonstrate the timing we have used the minimal and maximal possible values of k. For $k = 3$, the curvature has been reconstructed in approximately 4500 voxels in a second while for $k = 28$ the speed was about 2000 voxels per second. The times have been measured on a PC with a 400 MHz PentiumII CPU and 512 MB of RAM.

4.5 Concluding remarks

We have proposed [25] a new class of transfer functions which assign optical properties to principal curvatures reconstructed from the input data. Such transfer functions allow to assign optical properties to objects with respect to their shape. Moreover, within one shape class, the objects can be distinguished by curvature magnitudes. As opposed to density transfer functions, curvature-based transfer functions allow to see the structural changes inside solid objects even if the density changes are small. Moreover, both the domain and the significance of its parts are expected to be acquisition-independent. This yields an automatic initial setup and easy specification by the user.

On the other hand, there are several facts which make the implementation of the presented concept difficult. Firstly, a robust algorithm for estimation of principal curvature magnitudes is missing. The algorithm we have used reduces this problem to curvature estimation of planar curves and is thus only dependent on the accuracy of methods which deal with this two-dimensional subproblem. These methods, however, are not in all cases robust either [97] and a specific algorithm should be chosen with care for a particular application. Our implementation of the osculating circle method, for instance, tends to exhibit staircase artifacts in areas where the principal directions of the surface are not aligned to the grid axis of the input volume. Secondly, the curvature estimation is time demanding, which makes the concept currently unsuitable for online rendering. The curvatures can, however, be computed in a preprocessing step and stored in separate volumes.

Further research based on work presented in this chapter should primarily concentrate on a better estimation of curvature. For the method presented in section 4.3, for instance, the use of a larger neighborhood or better re-

construction filters for the description of planar curves can be taken into consideration. A quantitative error analysis and a comparative study with other algorithms are still necessary to establish the usability of the technique for visualizing real data.

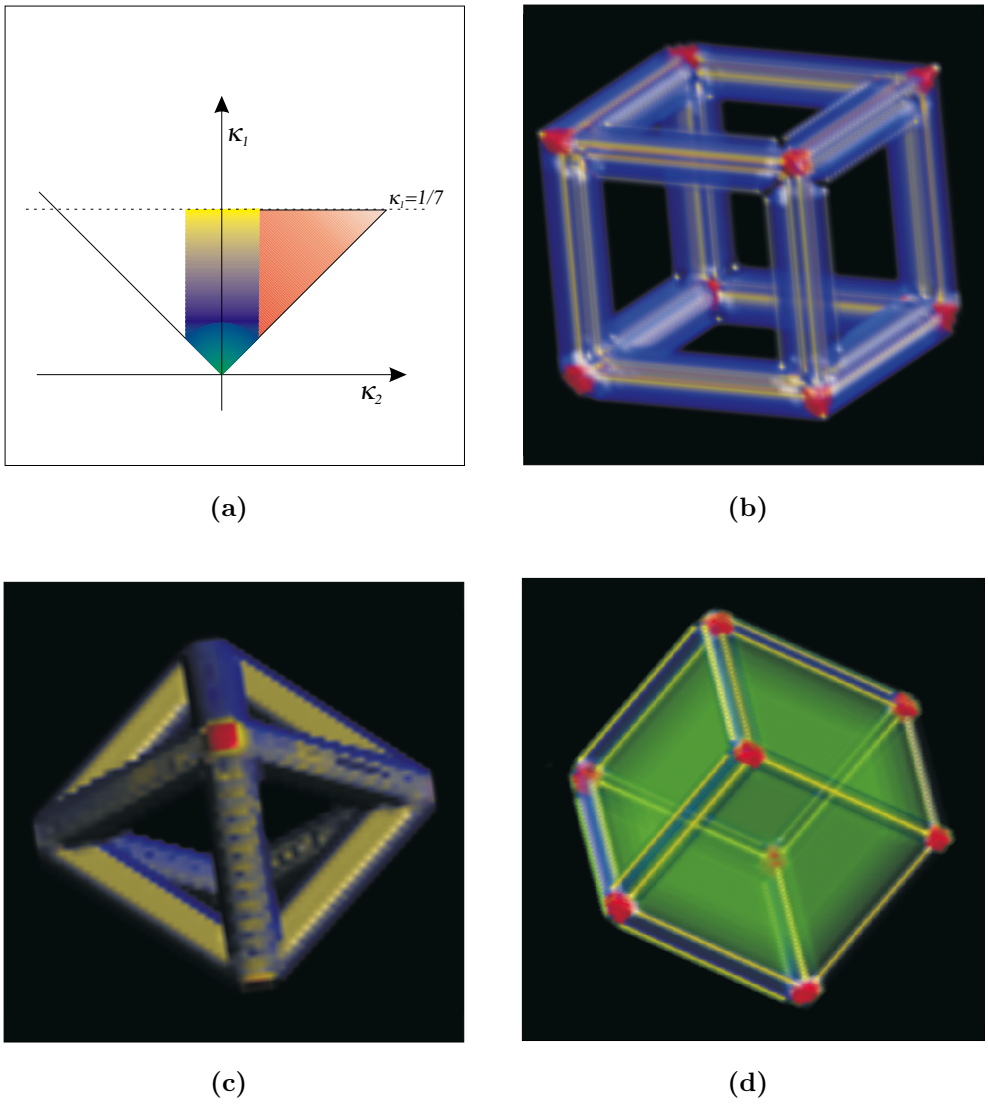
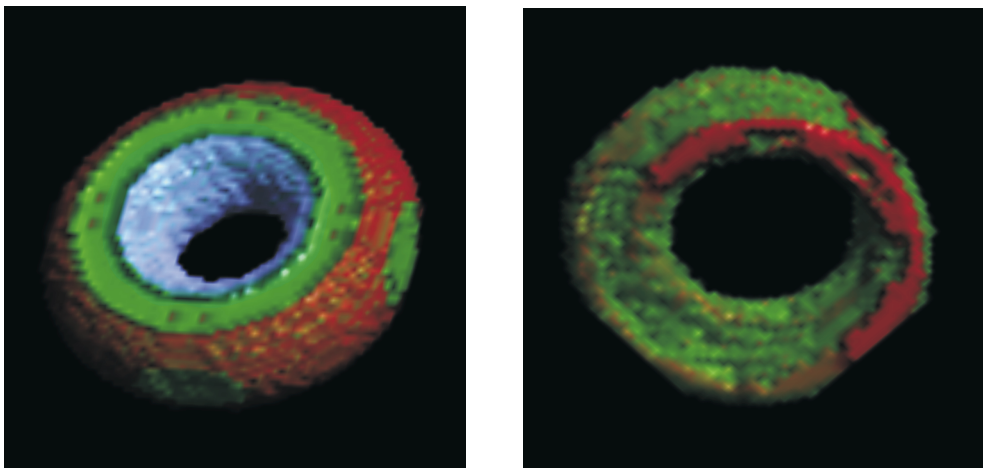


Figure 4.6: Transfer-function specification (a) for the images of a wire-frame cube (b), a wire-frame octahedron (c), and a facet-cube (d).



(a) **(b)**

Figure 4.7: Classification of the parabolic (red), planar (green), and hyperbolic (blue) parts of a torus (a) and a thickened Möbius strip (b).

Chapter 5

Salient representation of volume data

In this chapter we present an algorithm for content-based retrieval of representative subsets of volume data. Our technique [26, 27] is based on thresholding of the eigenvalues of the Hessian matrix. We compare our approach to feature detection based on the gradient magnitude and observe that our method allows to represent volumes by smaller amounts of voxels. Practical applications of our method include fast volume display due to object-space oriented techniques, progressive visualization over the network and the related generation of preview data sets for web-based repositories. For these applications, the size of the representative subset can be estimated automatically with respect to the bottleneck of the visualization system or a network bandwidth.

5.1 Introduction

With the VolumePro board [73] one might have gotten an impression that *the technology* of volume graphics already matured to become *a tool*. "Volume graphics today is where surface graphics was fifteen years ago" says David Nadeau at WSCG 2001 [68], however, and emphasizes two reasons: the lack of authoring tools which would help spreading volume graphics, and the limited display capabilities. The VolumePro board is able to display at 30 fps just volumes of rather small resolutions, i.e., 256^3. Without introducing blur, rendering such volumes can fill only moderately-sized output images, i.e., up to 256×256 pixels. Constrained by a 21-inch monitor with a resolution of 1280×1024 pixels, Nadeau compares a projection of a 256^3 volume to as we

were legally blind and of a 1024^3 volume to as we were visually impaired. In order to approach the human visual capabilities determined by the density of cones at the fovea, we will need to render volumes of much higher resolutions. The 21-inch screen usage would require a projection of a 2400^3 volume. Considering a 180° visual field in virtual/augmented reality applications, the volume resolutions necessary to satisfy the human eye increase to 21600^3 voxels.

The current technology is far from able to display such resolutions at interactive frame rates. The main advantage of volume graphics over surface graphics, i.e., the display of *all* volume elements in a data set, is also its main disadvantage – the size of data to be visualized limits the practical use. In his talk [68], Nadeau surveys on how to cope with this phenomenon and concludes a part of the keynote with a question: "How about changing our data?".

In this chapter we give one of the possible answers to this question. We report on a newly-developed technique aiming at a content-based retrieval of the most important areas from a volume data set. Our approach utilizes convolution, computation of eigensystems, and thresholding. It enables representation of volumes by crucial features contained in much smaller subsets.

5.2 Motivation and related work

A representation of volume data by just a small subset of *content-carrying* voxels is desirable for and addressed by many applications.

Appropriately reorganized sparse volumes can be rendered using object-space display techniques at interactive frame rates [65, 6]. Saito [80] introduces a non-realistic previewing. Each voxel from a sparse subset of the volume is represented by a simple 3D entity like a point, line, or a polar cross. A list of these entities is passed to a conventional rendering pipeline achieving real-time results. Splatting introduced by Westover [96] and enhanced over the years [67] usually yields high-quality display results. Researchers further optimize [37] this technique to achieve interactive frame rates for volumes of moderate resolutions. Recently, point-based visualization of large data sets became popular due to the work of Rusinkiewicz et al. [79] because of both simplicity and speed. The discussions on quality aspects of this approach are triggered by Pfister et al. [75].

Interactive volume visualization over the Internet based on a client/server architecture profits from elaborated strategies for progressive data transmis-

sion. Here it is desirable that the content of a volume is visually interpretable already in the early stages of transmission to and visualization by a client. To achieve this, the server may start transmitting salient features earlier than the rest of the data.

Non-distributed visualization may benefit from storing a small, representative subset of the data to disk. Such a representation can be reused later for a quick preview.

There are many techniques aiming at identifying important subsets of a volume data set. Isosurface extraction algorithms belong to the most often used techniques in indirect volume rendering. The choice on the iso-value(s) can be done interactively [52] or, after a previous analysis, semi-automatically [39], or automatically [15]. Saito [80] employs a non-uniform stochastic Poisson sampling. Mroz et al. [66] reduce the size of the data with respect to the applied visualization technique, i.e., maximum intensity projection (MIP). Gradient magnitude of the scalar field can also be considered as a priority function because it emphasizes boundaries which mostly attract the human attention.

Similarly to the gradient-magnitude techniques, our concept is based on filtering. The quantity being filtered is the second-order derivative of the scalar field. It is, after solving its eigensystem, extracted from the Hessian matrix.

In the following we introduce an easy-to-use framework for using eigenvalues of the Hessian matrix for identification of a useful subset of a given volume.

5.3 Symmetric thresholding of eigenvalues of the Hessian

Given the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ of the Hessian matrix (section 3.3.2), Equation (3.32) shows that it is possible to combine them into the Laplacian operator and use them for boundary detection (see also section 3.3.1).

Experimenting with the eigenvalue images, however, we have found that treating the eigenvalues separately rather than adding them is suitable for thresholding. Figures 5.1 and 5.2 demonstrate this property on 2D examples.

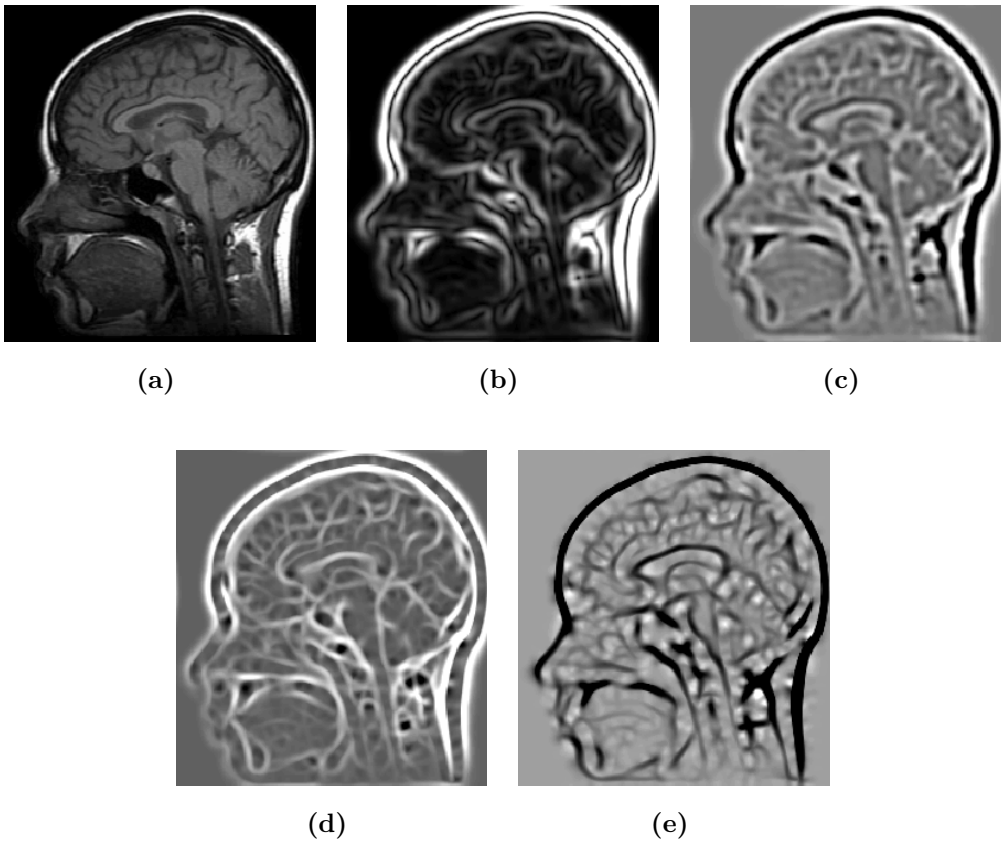


Figure 5.1: An example of a 2D MRI image. (a) original image g, (b) gradient magnitude $\|\nabla g\|$, (c) response to the Laplacian operator, (d) λ_1-image, and (e) λ_2-image. The Laplacian image corresponds to the sum of the eigenvalue images, which are well suited for thresholding.

To extract features from a 2D image we propose a two-fold thresholding as follows:

$$g_{new}[x_1, x_2] = \begin{cases} g[x_1, x_2] & \text{if } \lambda_2[x_1, x_2] \leq T_2 \ \vee \ T_1 \leq \lambda_1[x_1, x_2] \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where x_1, x_2 denote the spatial image-coordinates and T_1, T_2 are thresholds specified by the user.

For each voxel of a 3D image there are three eigenvalues, $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Fig. 5.3 shows axial slices of the eigenvalue volumes computed from a CT head data set. Since, in our experience, images corresponding to λ_2 lack good

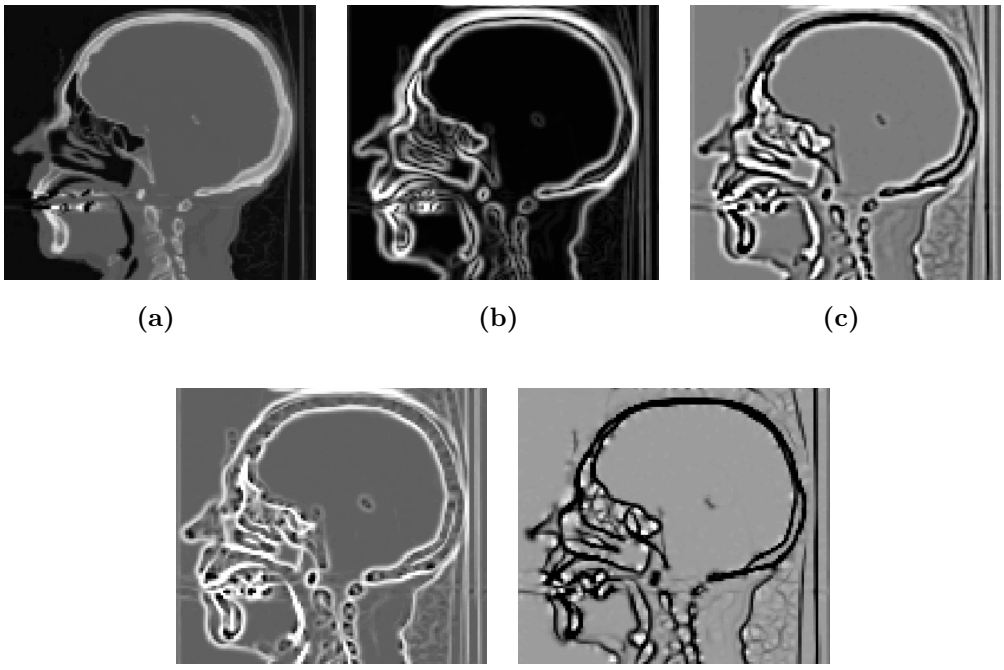


Figure 5.2: An example of a 2D CT image. (a) original image g, (b) gradient magnitude $\|\nabla g\|$, (c) response to the Laplacian operator, (d) λ_1-image, and (e) λ_2-image. The Laplacian image corresponds to the sum of the eigenvalue images, which are well suited for thresholding.

contrast, we propose a two-fold (instead of three-fold) thresholding. It only takes into account the two eigenvalues λ_1 and λ_3:

$$g_{new}[x] = \begin{cases} g[x_1, x_2, x_3] & \text{if } \lambda_3[x_1, x_2, x_3] \leq T_3 \ \vee \ T_1 \leq \lambda_1[x_1, x_2, x_3] \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where x_1, x_2, x_3 denote the spatial volume-coordinates and T_1, T_3 are thresholds specified by the user. The task of the user is to specify the two thresholds T_1, T_3 (Fig. 5.3).

5.3.1 Results

Table 5.1 and Figure 5.4 indicate that for visualization purposes Equation (5.2) allows to represent volumes by approximately 10% of the voxels. The thresholds T_1 and T_3 allow the user to interactively control the trade-off between display quality and amount of displayed information.

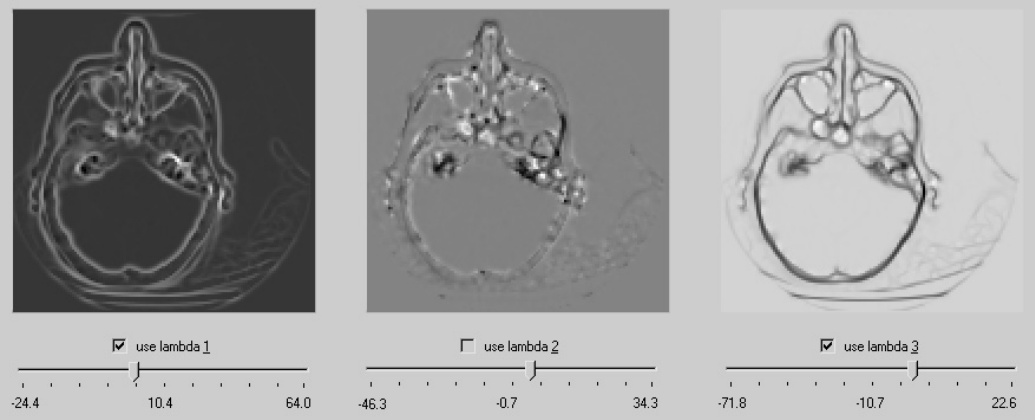


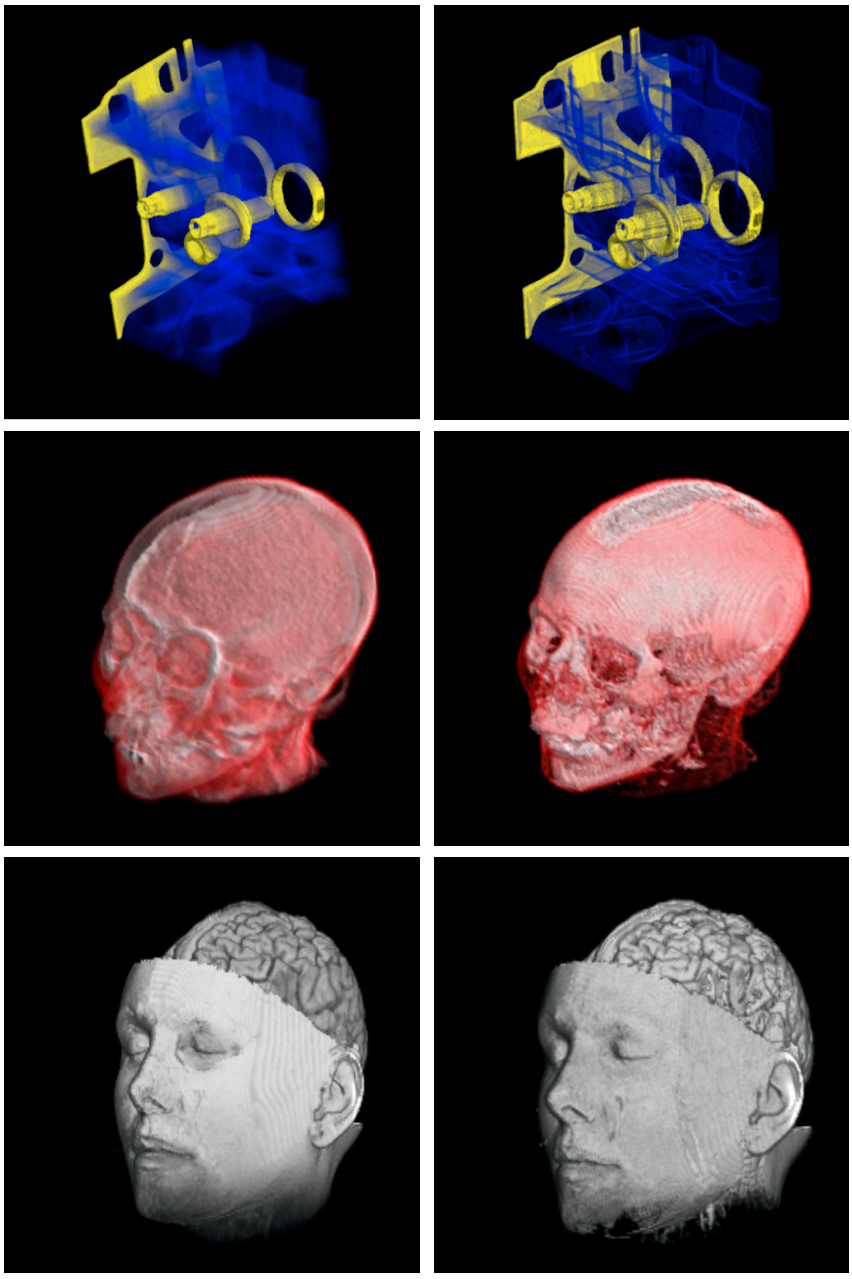
Figure 5.3: Part of an interface and illustration that eigenvalues λ_1 and λ_3 are suitable for thresholding. The λ_2-image lacks contrast information and is therefore from thresholding excluded.

Input Volume			Sparse Volume			
Data set	Resolution	KB	T_1	T_3	KB	%
Engine Block	256 × 256 × 110	7 040	21.953	-23.381	658	9.36
CT Head	128 × 128 × 113	1 808	16.981	-19.182	183	10.12
MRI Head	256 × 256 × 109	6 976	7.822	-6.364	871	12.48

Table 5.1: Results for some typical volume data sets. Columns from left to right: name of the data set; its resolution; number of voxels in KB; threshold values T_1, T_3 used in Equation (5.2) for the generation of a sparse volume; number of nonzero voxels in KB in the sparse volume; and the relative size of the sparse volume to the input volume.

Equation (5.2) defines a variant of a 2nd-order boundary detectors which are important in bioperception [56]. This is mostly noticeable by comparing the rendered images of the engine block data sets (top row of Fig. 5.4). In the sparse volume, areas corresponding to boundaries are emphasized and provide the observer with better information on structures found in the data set.

All tested volumes have been quantized to 256 gray levels. To compare the visual appearance, both the input and the sparse volumes have been displayed with the shear-warp algorithm [44] implemented in the VolumePro board [73].



(a) Input volumes **(b)** Sparse volumes

Figure 5.4: Ray casting [73] of the Engine Block, CT Head, and MRI Head data sets (a) and of the corresponding sparse volumes (b) generated due to Equation (5.2) with threshold parameters set as in Table 5.1.

5.4 Further subset reduction

The previously introduced concept can be further extended [27] assuming that the objects are of a higher intensity than the background.

Two statements can be made under this assumption. First, the *bright narrow* structures can be identified due to responses to the 2nd-order derivative filter. Second, a 2nd-order operator responds by negative and positive values at the inner and outer side of a boundary, respectively. We put this fact into contrast to the gradient-based boundary detection, which yields an equal response on both sides of a boundary and exploit this fact to represent the objects' boundaries only by their internal side. Compared to the gradient method such a selection requires a smaller amount of voxels for boundary representation.

5.4.1 Motivation

The importance of boundary information for machine vision is usually motivated from the observation that under rather general assumptions about the image formation process, a discontinuity in image brightness can be assumed to correspond to a discontinuity in either depth, surface orientation, reflectance, or illumination [48]. A line as a different type of discontinuity is also a structure of particular interest. While in a 2D image the representatives of narrow solid structures are spots and lines, in volume data this is more general – blobs, cylinder-like, and sheet-like structures play a crucial role, e.g., in medical visualization [82].

Gradient magnitude is an indicator to identify object boundaries. The identification of narrow solid structures, however, requires the use of either special filters or 2nd-order derivative filters.

In the reminder of this chapter we propose a filtering technique for the identification of both boundaries and narrow structures. Our algorithm is based on the identification of areas with large-in-magnitude negative second derivatives, and handles both of the cases in a uniform way. Defining a salience function based on this quantity allows to identify those voxels of the input volume which constitute the most significant content of the data set.

5.4.2 Edge detectors and line detectors revisited

Two types of filters have been designed for edge detection – those based on looking for maxima of the first derivative (section 3.2.1) and those based on looking for zero-crossings of the second derivative (section 3.3.1).

While these concepts are intuitive for 1D signals, the situation in higher dimensions gets more complicated. To use the extrema of the 1st derivatives we need to know the directions in which they occur. From calculus it is known that for the first derivative this direction is the gradient vector ∇g and the derivative in this direction is the magnitude of the gradient: $g'_{max} = g'_{\nabla g} = \|\nabla g\| = (\sum_{i=1}^{3}(\partial g/\partial x_i)^2)^{1/2}$. Looking for maxima of gradient magnitudes yields an isotropic boundary detector which equally responds to the outer *and* the inner side of the object (Fig. 5.5(a)). For the second derivative approach it is necessary to check the neighborhood of a voxel for zero-crossings, i.e., for areas where the 2nd derivative changes its sign. The 2nd derivative is usually estimated by the rotationally invariant Laplacian $\nabla^2 g = \sum_{i=1}^{3} \partial^2 g/\partial x_i^2$. A less referred feature of the Laplacian operator is that it responds with negative values at the inner part and by positive values at the outer part of the object's boundary (Fig. 5.5(b)). We put this into contrast to the gradient-based boundary detection, which yields an equal response on both sides of a boundary and exploit this fact to represent the objects' boundaries only by their internal side. Compared to the gradient method, such a representation requires a smaller amount of voxels. From the viewpoint of a client/server visualization system running with low bandwidth, such an identification would yield a better distribution of voxels over boundaries especially in early stages of the progressive transmission.

Considering the density profile, it is evident that the concepts of 1st derivative maxima can not be directly applied for spot and line detection (or, more generally speaking, for detection of narrow regions which in 3D correspond to blobs, lines, and sheet-like structures). The response of a 1st-order derivative filter to a line, for instance, results in two lines, which would require a special, nontrivial mechanism for detection of the in-between area (Fig. 5.5(a)). A 2nd-order derivative filter, on the other hand, responds to lines by negative values in the interior (Fig. 5.5(b)).

As a result we get a twofold interpretation of areas where the 2nd-order derivative operator responds with negative values. Firstly such areas correspond to internal parts of a boundary and secondly they identify narrow structures. To make the search for negative areas more feasible for separation by thresholding, we are interested in the directions where the 2nd-order derivatives are minimal.

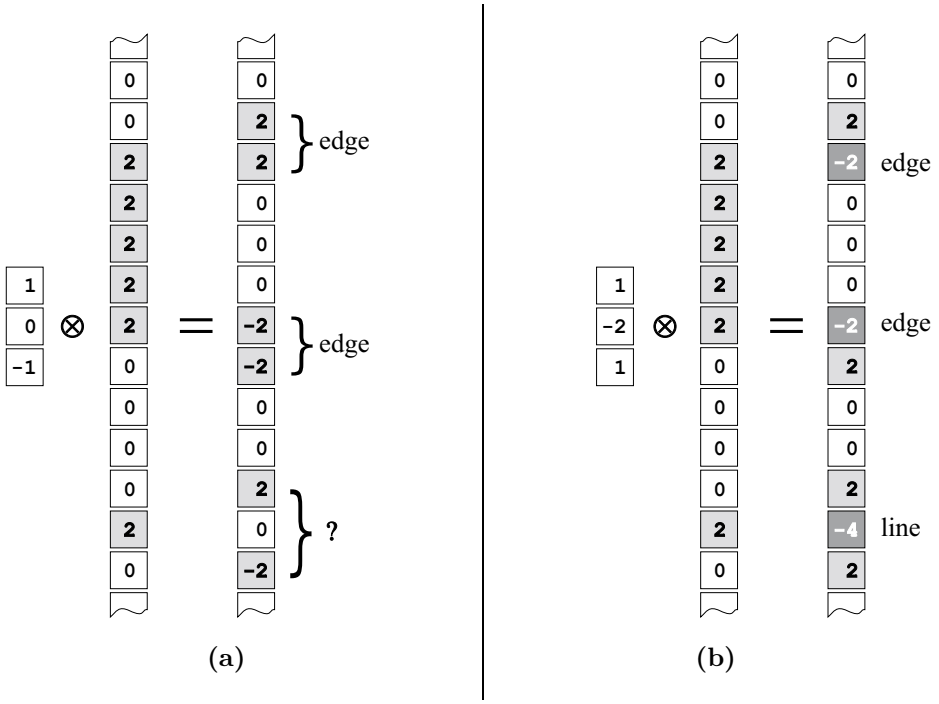


Figure 5.5: Examples of a 1D density profile. Responses of a 1st-order derivative filter (a) and a 2nd-order derivative filter (b) to an edge and to a line.

5.4.3 The smallest 2nd derivative

At a specific point, the eigenvalues λ_i of the Hessian give the second derivatives in the directions of the associated eigenvectors $\boldsymbol{e_i}$: $g''_{\boldsymbol{e}_i} = d^2g/de_i^2 = \boldsymbol{e_i}^T \mathbf{H}g\, \boldsymbol{e_i} = \lambda_i$ (section 3.3.2). Since $\mathbf{H}g$ in this context represents a quadratic form, computing the smallest eigenvalue directly yields the minimal directional derivative, i.e., $g''_{min} = d^2g/d\boldsymbol{e_3}^2 = \lambda_3$.

5.4.4 Salience by the smallest eigenvalue

So far we described a mechanism to find the smallest 2nd-order directional derivative at a given grid point of a volume. As λ_3 is just a special case of a 2nd-order directional derivative, the interpretation from section 5.4.2 remains the same:

1. Areas featuring very low λ_3 represent an inner part of an object's boundary. Unlike in the case of gradient magnitude where looking

for the maxima yields a representation of the boundary from both the outer and the inner side, our approach restricts the representation of boundaries just to the inner side. Compared to the gradient magnitude operator, the boundary can therefore be represented by a smaller amount of voxels.

2. Areas with very low λ_3 correspond to the blobs, lines, and sheet-like structures. Their detection with a first derivative operator would be impossible.

3. Having the minimal second derivative is more suitable for separation of the structures by thresholding instead of having a second derivative in an arbitrary direction.

Due to these reasons, areas featuring low negative eigenvalues λ_3 yield a better representation of a volume than those with high positive values of the gradient magnitude.

In order to provide a comparative study between these two approaches, we define the two following salience functions S_Γ, S_Λ of a voxel v and the two corresponding $p\%$-subsets of the input volume g they determine:

$$S_\Gamma[v] = \|\nabla g\|[v] \qquad \Gamma[p\,\%] = \{p\,\%\text{ of } g \text{ with the highest } S_\Gamma\} \quad (5.3)$$

$$S_\Lambda[v] = -\lambda_3[v] \qquad \Lambda[p\,\%] = \{p\,\%\text{ of } g \text{ with the highest } S_\Lambda\} \quad (5.4)$$

For a given percentage p, functions S_Γ, S_Λ determine the $p\,\%$ of 'top salient' voxels which will represent the volume. For a progressive transmission of data through a network, these functions determine the priority of transmission: the voxels with higher salience will be transmitted earlier.

Obviously, there are also other candidates which might succeed well in the task of volume representation by a fraction of the data. In the following we discuss several possible competitors and argue why we do not compare them to our method:

Isosurface methods require user input to specify the density which determines an isosurface. The result is dependent on and yields only the structures defined by this choice. Our method processes data automatically and delivers surfaces of more than just one isolevel.

The gradient integral (section 3.3.4) proposed by Bajaj et al. [2] is intended to emphasize those isosurfaces which feature both long gradients and large area. The integral is efficiently calculated with cumulative

histograms of the Laplacian [72], i.e., the trace of the Hessian matrix. The saliency provided by the gradient integral is, however, based on densities and is essentially inconsistent with our position-based approach.

The "position across boundary" (section 3.3.3) introduced by Kindlmann and Durkin [39] yields an opacity transfer function for boundary emphasis. The algorithm performs a statistical analysis of the zero, 1st, and 2nd-order derivatives in the direction of the gradient, and finds those densities which primarily contribute to the boundaries. Similarly to the previous case, this analysis is density-based and therefore not comparable with our approach.

Density distribution analysis based on *all* eigenvalues of the Hessian (section 3.3.2) as proposed by Frangi et al. [13] or Sato et al. [81, 82] restricts the search space just to structures of a particular shape and a certain scale, and excludes boundaries of objects. In contrast, our approach handles both boundaries and structures in a uniform way.

5.5 Implementation and complexity

5.5.1 Hessian matrix versus gradient vector

Computation of both the gradient vector and the Hessian matrix at grid points involves an approximation of the first and the second partial derivatives, respectively. For this task, the data is convolved with kernels which are designed for a particular derivative in a specific direction (section 3.1.4).

For first derivatives, kernels of size up to three are usually found in the textbooks: Prewitt and Sobel filters (Appendix A) are feasible for fast computation [51].

Calculating the Hessian matrix requires an estimation of 2nd-order derivatives which is, especially for small kernels, much more sensitive to noise. The usual practice is to pre-smooth the input data with a Gaussian filter. Due to the associativity of convolution, the smoothing and the differentiation steps can be combined, resulting in a convolution of the data with a derivative of the Gaussian filter of a bigger size.

To remain consistent for comparison of both the quality of results and the computational costs we used filters of the same size both for 1st and 2nd derivatives. Using the Gaussian filter requires that its size k is proportional

to the standard deviation, so the kernels usually involved are 5, 7, or 9 voxels wide. Convolution with even moderately-sized kernels is usually a computationally expensive process. To speed it up, we exploit the separability of the Gaussian kernel:

$$\left(\overbrace{\frac{\partial^{a+b+c}}{\partial x_1^a \partial x_2^b \partial x_3^c} G_\sigma(\boldsymbol{x})}^{k\times k\times k}\right) \otimes g = \overbrace{\frac{d^a}{dx_1^a} G_\sigma(x_1)}^{k\times 1\times 1} \otimes \left(\overbrace{\frac{d^b}{dx_2^b} G_\sigma(x_2)}^{1\times k\times 1} \otimes \left(\overbrace{\frac{d^c}{dx_3^c} G_\sigma(x_3)}^{1\times 1\times k} \otimes g\right)\right) \tag{5.5}$$

where the sum $a+b+c \in \{1,2\}$ of nonnegative integers a, b, c determines the order of differentiation, and σ is the standard deviation of the Gaussian filter $G_\sigma(x) = \exp(-\frac{x^2}{2\sigma^2})/\sqrt{2\pi}\sigma$ (see also Appendix A.5). The decomposition according to Equation (5.5) reduces the cost for calculating a partial derivative at a grid point from convolution with a 3D kernel (complexity $O(k^3)$) to three convolutions with a 1D kernel (complexity $O(3k)$).

A direct application of Equation (5.5) would require $6 \times 3 = 18$ 1D convolutions for calculating the 6 distinct Hessian elements and $3 \times 3 = 9$ 1D convolutions for calculating the gradient vector. Further speed-up can be achieved by appropriate reorganization and caching. Three 1D convolution passes can be saved for the computation of the Hessian matrix, (e.g., $G_\sigma(x_3) \otimes g$ can be reused three times and $G'_\sigma(x_3) \otimes g$ twice) and one convolution pass can be saved for the gradient (e.g., $G_\sigma(x_3) \otimes g$ can be reused twice). This reduces the number of required 1D convolutions to 15 for the Hessian and to 8 for the gradient (see the corresponding entries in Table 5.2).

5.5.2 Eigenvalues of the Hessian versus magnitude of the gradient

While computing the Euclidean norm of a 3D vector requires only three multiplications, two additions and one square root, the computation of eigenvalues of a 3×3 matrix is generally more time demanding. Fortunately, since the Hessian matrix is real symmetric, the roots of the associated characteristic polynomial are real and the analytical solution introduced in section 2.4 can be used.

Table 5.2 summarizes the overall time costs concluding that the computation of eigenvalues is on average 1.87 times more expensive as compared to the computation of the gradient magnitude.

Input Volume		Cost of $\|\nabla g\|$		Cost of λ_i		Factor
Data set	Resolution	$8 \otimes$	+norm	$15 \otimes$	+roots	
Lobster	$120 \times 120 \times 34$	6.36	6.44	11.90	12.02	1.87
Vertebra 1	$128 \times 128 \times 74$	17.46	17.68	31.48	31.94	1.81
CT Head	$128 \times 128 \times 113$	26.14	26.46	48.11	48.77	1.84
MRI Head	$256 \times 256 \times 109$	110.93	112.25	208.43	210.96	1.88
Engine Block	$256 \times 256 \times 110$	112.20	113.46	210.98	213.36	1.88
Tooth	$256 \times 256 \times 161$	164.62	166.24	308.45	311.33	1.87
Vertebra 2	$256 \times 256 \times 241$	247.31	250.12	497.02	499.93	2.00

Table 5.2: Time in seconds for computing the gradient magnitude and the eigenvalues of the Hessian matrix as measured on a Pentium II, 400 MHz. 1D cyclic convolutions (Eq. 5.5) with kernel of size $k = 7$ have been used. The meaning of columns from left to right: name of the data set and its resolution; time for the computation of all partial derivatives for the gradient vector, and after calculating the Euclidean norm; time for the computation of all partial derivatives for the Hessian, and after computation of the cubic-polynomial roots; the ratio of overall times for the eigenvalues and the gradient magnitude calculation.

5.5.3 Construction of representative subsets

To build the subsets $\Gamma(p\%)$ and $\Lambda(p\%)$ defined by Equations (5.3) and (5.4), we first construct cumulative histograms of the quantities $\|\nabla g\|$ and $-\lambda_3$, respectively. This is done in one pass through the volume data in linear time. The percentage p controls the number of voxels to be included into the respective subset. The search for adjacent histogram bins straddling this number is logarithmic. The indices of bins correspond to a threshold which serves for the final decision.

5.6 Results

To compare the quality of volume representations given by Equations (5.3) and (5.4), we generate sparse volumes where the density of voxels not present in either of the subsets Γ and Λ have been set to zero. Such volumes have been rendered eith the OpenSplat package [29] (Figs. 5.6, 5.7) and by the shear-warp algorithm [44] implemented in the VolumePro board [73] (Figs. 5.8, 5.9), respectively.

Lobster: (Fig. 5.6) The representation by the $\Lambda(1\%)$ subset provides a better idea about the data set than the same amount of voxels in the

representation by $\Gamma(1\%)$. While the legs of the lobster are visible and well recognizable already in the $\Lambda(3\%)$ subset, they just start to appear in the $\Gamma(7\%)$ subset. In contrast, the $\Lambda(7\%)$ is already close to describe the entire topology of the data set.

Vertebra 1: (Fig. 5.7) Neither of the 1%-representations provides enough information, though there is more content visible in the $\Lambda(1\%)$ subset. At 3%, 5%, and 7% we observe that the contours in the $\Lambda(.)$ subsets close much faster than in the corresponding $\Gamma(.)$ subsets. Moreover, we estimate that the $\Lambda(1\%)$, $\Lambda(3\%)$, and $\Lambda(5\%)$ subsets provide approximately the same level of information as the $\Gamma(3\%)$, $\Gamma(5\%)$, and $\Gamma(7\%)$ subsets, respectively.

Vertebra 2: (Fig. 5.8) Subset $\Gamma(2\%)$ features only high density screws. While the ribs only begin to appear in $\Gamma(4\%)$, they are better visible already in $\Lambda(2\%)$ due to a more even distribution of boundary voxels. $\Gamma(6\%)$ and $\Gamma(8\%)$ provide less information than $\Lambda(4\%)$. In contrast, $\Lambda(6\%)$ contains the most relevant features and $\Lambda(8\%)$ seems to provide all the features plus aditional voxels from the noisy area surrounding the vertebra.

Tooth: (Fig. 5.9) There are two significant features identified in this data set: the tooth and the cylindrical shell of the medium (not displayed) in which the tooth was set prior to scanning. The area of the shell is comparable to the area of the tooth – about 53% of the $\Lambda(.)$ and the $\Gamma(.)$ voxels contribute to the shell, on average. In spite of this fact, the differences between the two representations are still apparent. Since there are no narrow structures present in the input data set, this example shows that the $\Lambda(.)$ representations yield a better distribution of the voxels over the surfaces.

5.7 Concluding remarks and future work

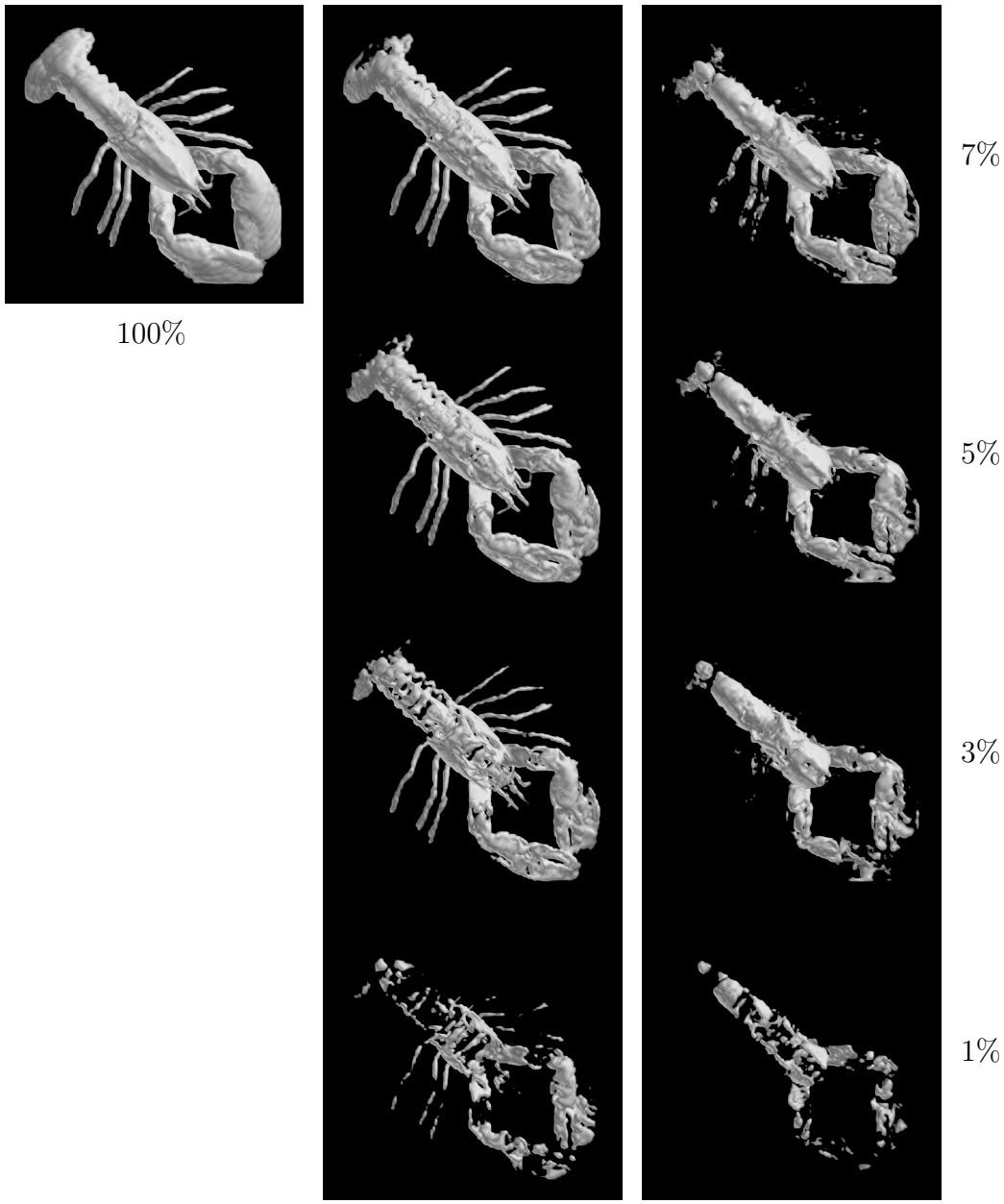
We proposed an easy-to-use framework for exploiting eigenvalues of the Hessian matrix to represent volume data by small subsets. We showed the suitability of thresholding eigenvalue volumes, and defined a two-fold threshold operation to generate sparse data sets.

For data where it can be assumed that objects exhibit higher intensities than background, we further improved this framework taking into account only the smallest eigenvalue. This resulted into further reduction of the

representative subsets. These subsets convey, in an easy and uniform way, information contained in two essentially different modalities, the object's boundaries and the narrow structures.

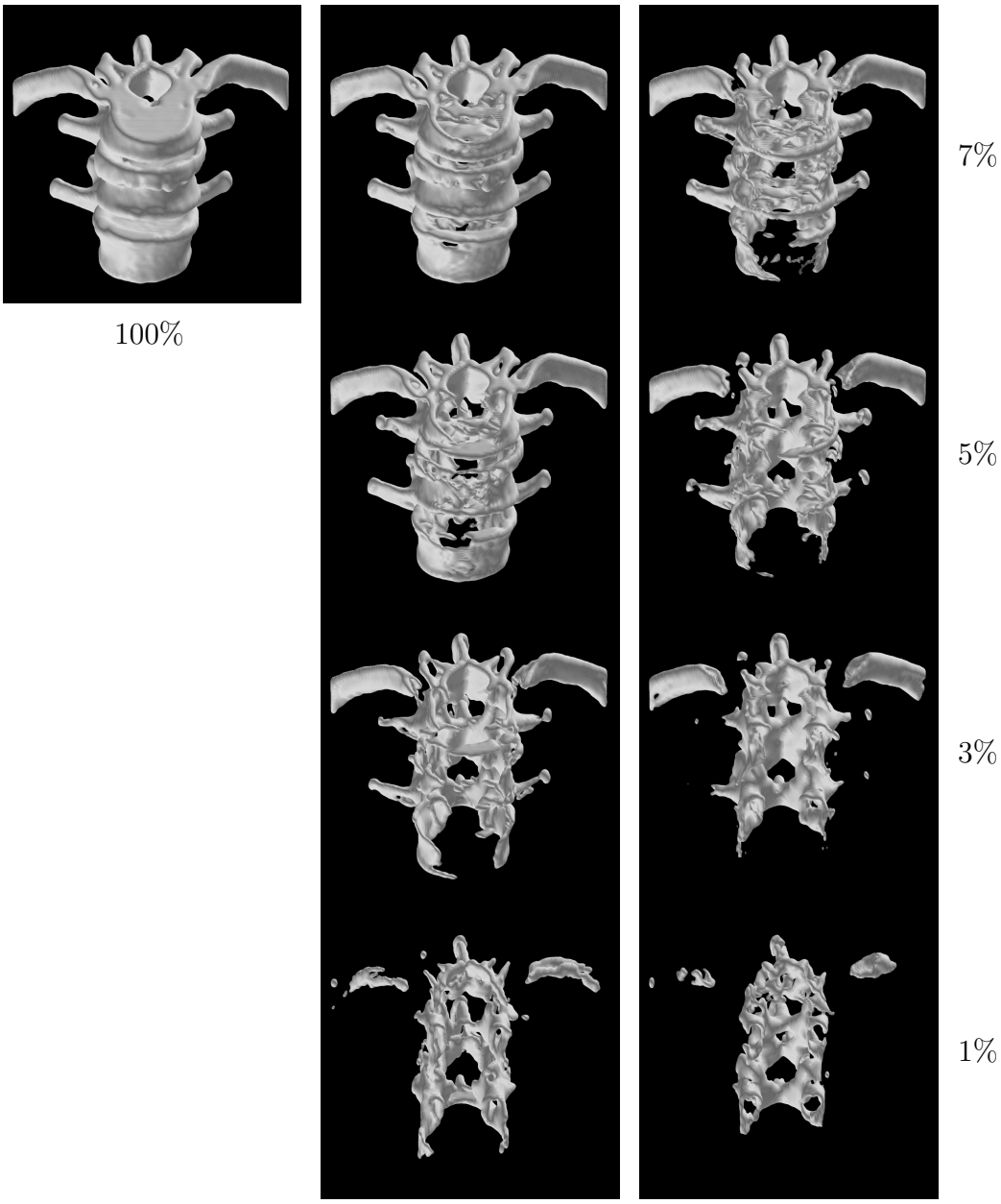
We evaluated our method with several data sets from different modalities and compared it to the feature-detection based on thresholding of the gradient magnitude. We conclude that, for the same level of perception, our method allows to represent data sets by reasonably smaller subsets.

The possible applications of such a compact representation are, e.g., fast rendering due to object-space display techniques, progressive transmission over the internet and the generation of preview data sets. The drawback of our method is a higher computational cost. Computation of the Hessian's eigenvalues is on average 1.87 times more expensive than the computation of the gradient magnitude. Speed-ups might be achieved, for instance, through better caching strategies and exploiting hardware capabilities. The performance issues should be addressed in future work.



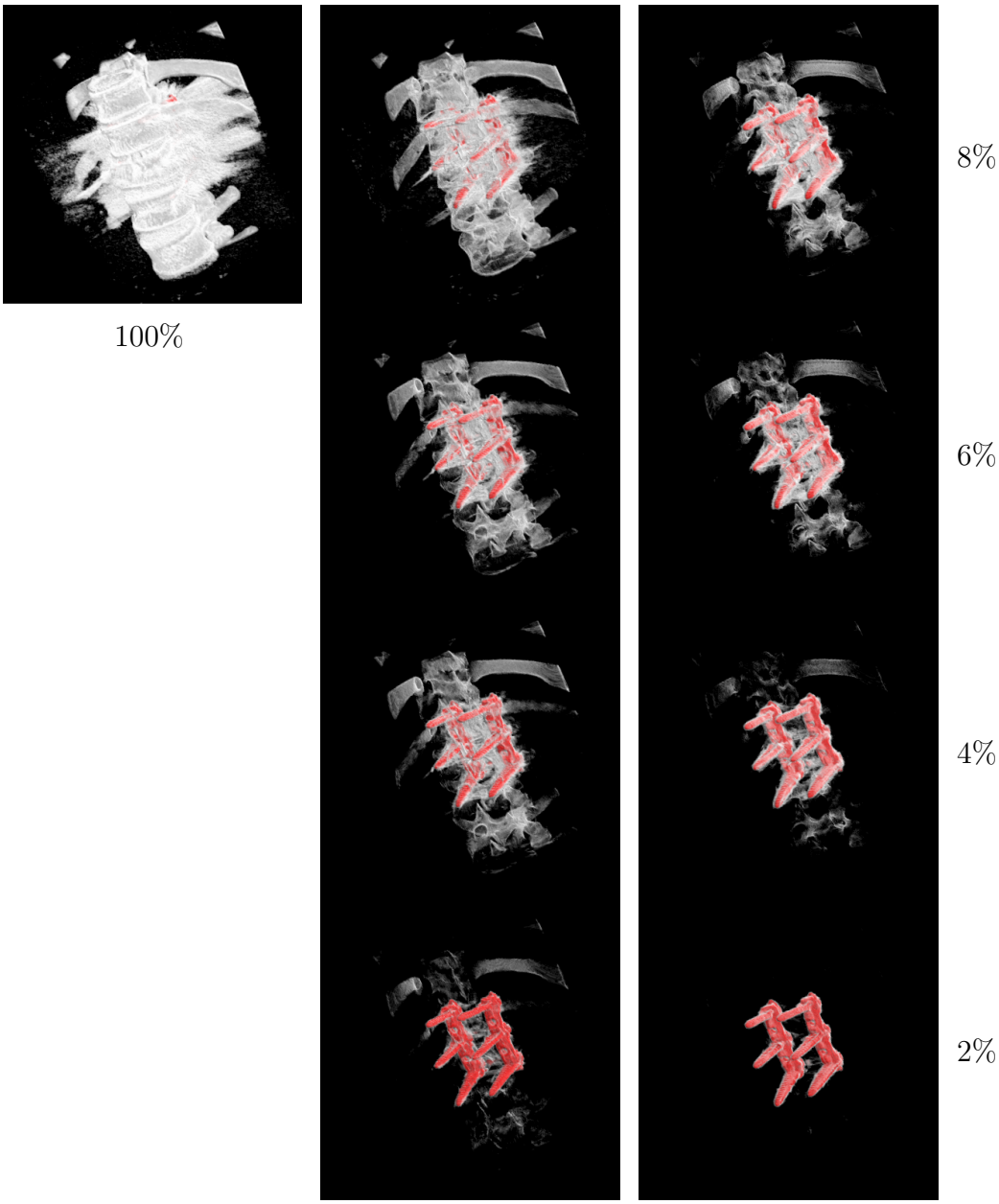
(a) original data set **(b)** Λ-subsets **(c)** Γ-subsets

Figure 5.6: Splatting [29] of the Lobster data set (a) and its representations due to the salience provided by eigenvalue λ_3 (b) and by detection due to gradient magnitude (c). From top to bottom the subsets comprise 7.0 %, 5.0 %, 3.0 %, and 1.0 % of the voxels of the original data set.



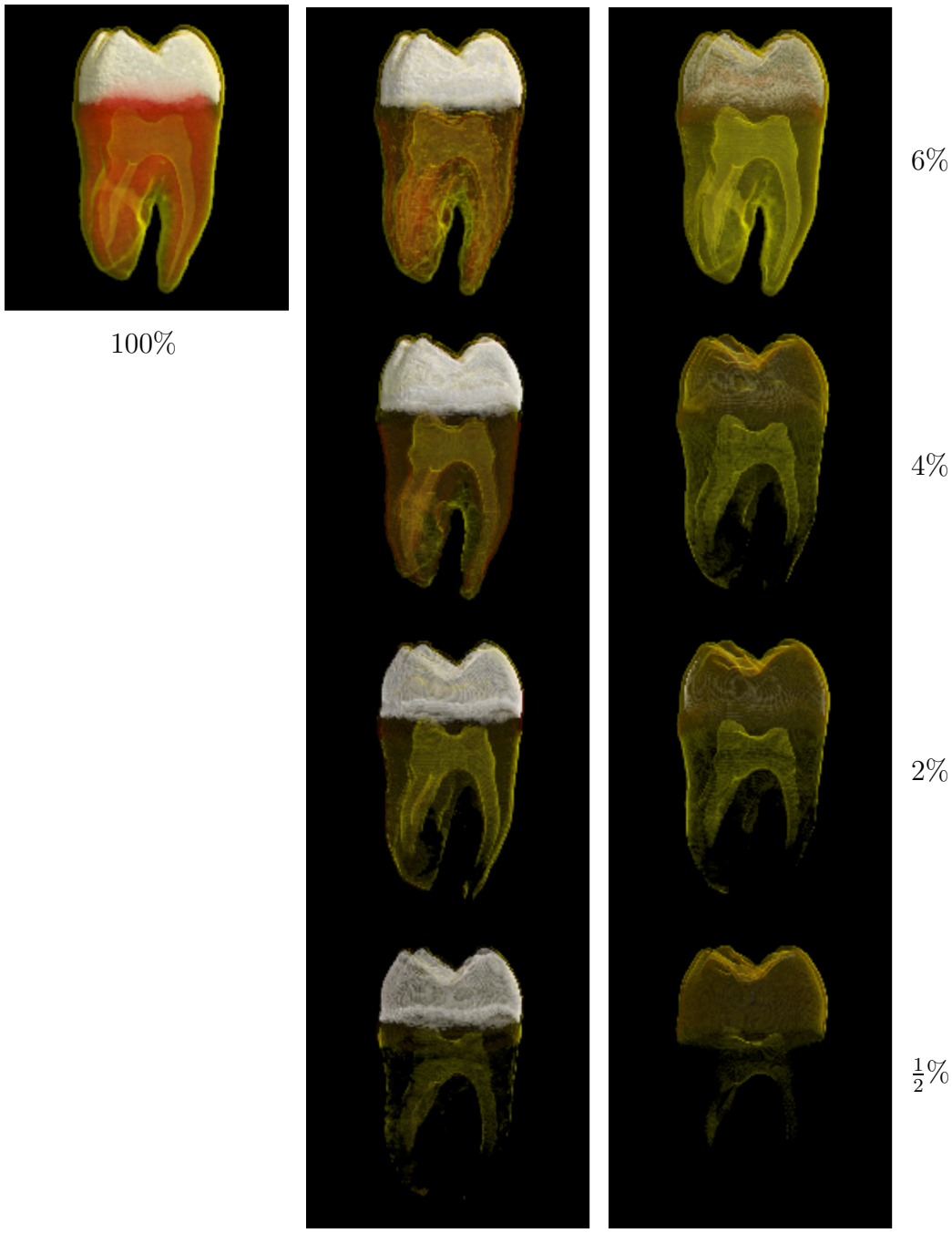
(a) original data set **(b)** Λ-subsets **(c)** Γ-subsets

Figure 5.7: Splatting [29] of the Vertebra 1 data set (a) and its representations due to the salience provided by eigenvalue λ_3 (b) and by detection due to gradient magnitude (c). From top to bottom the subsets comprise 7.0 %, 5.0 %, 3.0 %, and 1.0 % of the voxels of the original data set.



(a) original data set **(b)** Λ-subsets **(c)** Γ-subsets

Figure 5.8: Ray casting [73] of the Vertebra 2 data set (a) and its representations due to the salience provided by eigenvalue λ_3 (b) and by detection due to gradient magnitude (c). From top to bottom the subsets comprise 8.0 %, 6.0 %, 4.0 %, and 2.0 % of the voxels of the original data set.



(a) original data set **(b)** Λ-subsets **(c)** Γ-subsets

Figure 5.9: Ray casting [73] of the Tooth data set (a) and its representations due to the salience provided by eigenvalue λ_3 (b) and by detection due to gradient magnitude (c). From top to bottom the subsets comprise 6.0 %, 4.0 %, 2.0 %, and 0.5 % of the voxels of the original data set.

Chapter 6

Orientation-driven shape-based interpolation of volume data

In this chapter we recall our novel approach [22] to shape-based interpolation of gray-level volume data. In contrast to the segmentation-based techniques our method directly processes the scalar volume requiring no user interaction. The key idea is to perform the interpolation in the directions given by analysis of the eigensystem of the *structure tensor*. Our method processes a 256×256 slice within a couple of seconds yielding satisfactory results. We give a quantitative and a visual comparison to the linear inter-slice interpolation. Analysis of the results lead us to the conclusion that our technique has a strong potential to compete with well-established shape-based interpolation algorithms.

6.1 Motivation

In general, interpolation is required whenever the acquired data are not at the same level of discretization as the level that is desired [16]. In volume visualization by ray casting, for instance, it is necessary to acquire samples at regular distances on a ray. The interpolation routine has to provide the ray caster with values at any position between the grid points. Algorithms for volume analysis are often based on filtering techniques which presume isotropy. The input data has to be resampled in most cases to an isotropic discretization. Medical imaging systems usually acquire the data in slice-by-slice order resulting in different sampling rates in x_1, x_2, and x_3 directions. For an appropriate medical treatment it is necessary that the data at requested locations are reconstructed as precisely as possible taking into

account not only the characteristics of the data as a 3D signal but also the topological properties of the explored structures.

6.2 Related work

Interpolation algorithms can broadly be divided into two categories – scene-based (image-based) and object-based (shape-based). Although scene-based filters received a lot of attention in the visualization community in recent years [62, 45, 90, 87], there is repeated evidence in the literature of the superior performance of object-based over scene-based interpolation techniques [17].

From the beginning, shape-based algorithms required a segmented volume, i.e., an identification of objects to be interpolated. The first approaches are based on contour interpolation.

Raya and Udupa [78] proposed an interpolation of distance fields computed from binary slices. Higgins et al. [21] also focus on the problem of interpolating binary objects rather then high-resolution gray-scale images. They extend the approach of Raya and Udupa [78] employing the original density information to avoid inconsistencies in the transition of objects' cross sections and their centroids. Turk and O'Brien [93] exploit both contour descriptions and distance fields encoded as implicit functions. They come up with a powerful framework for interpolation of *analytically* given objects.

Moshfegi [64] proposes a technique aiming at removing staircase effects in a manner that is consistent with maximum-intensity projection (MIP). The direction of interpolation is aligned adaptively with the axes of the vessels. The proposed template matching considers a pair of scan-lines iteratively looking in reference windows for the best match due to mean square error and a correlation coefficient. The interpolation method is, however, applied after the projection providing thus a solution only for 2D scenes.

The first purely gray-level-based approach yielding reasonable results seems to be the three-pass algorithm proposed by Grevera and Udupa [16]. In order to adopt the previously introduced technique [78], the n-dimensional gray scene is *lifted* to a $[n+1]$-dimensional binary scene in the first step. Then binary shape-based interpolation [78] is applied. Finally, the newly interpolated $[n+1]$-binary scene is *collapsed* back to n dimensions. A drawback of this method is a high time and space complexity. Two years later this method and two variants thereof are compared [17] to the five most referred-to interpolation techniques. The paper emphasizes the superior performance

of object-based over scene-based interpolation techniques, too. To see how far this is evident in a specific application, i.e., detection of brain lesions, Grevera and Udupa [18] statistically compare 100 data sets resulting from 10 patients, 2 modalities and 5 interpolation methods.

In contrast to the frequently used techniques we propose a direction-driven interpolation. The interpolation is driven by the eigenvectors of the *structure tensor*. Structure tensors carry information on the density distribution in *simple neighborhoods*. As they are computed directly from scalar data, our method does not require any segmentation or user interaction. This makes a reasonable difference between our method and most shape-based interpolation techniques.

An introduction to simple neighborhoods and the related tensor analysis as well as the concept of our method are presented in section 6.3. Section 6.4 discusses implementation issues. To see the performance, qualitative and quantitative evaluation of our method is given in section 6.5.

6.3 Pattern-driven interpolation

Usual strategies to study local neighborhoods are based on analyzing discontinuities in the intensity [48].

The human visual system, however, can easily recognize objects that do not differ from a background by their mean gray value but only by the orientation or scale of patterns. To perform this recognition task with a digital image processing system, operators which determine the orientation of patterns are needed [33].

6.3.1 Simple neighborhoods and structure tensor

For an appropriate representation of the emplacement of an object or a texture it is necessary to distinguish between *direction* given by an angle from the interval $\langle 0°, 360° \rangle$ and *orientation* $\langle 0°, 180° \rangle$. As we are locally not able to distinguish between patterns that are rotated by 180 degrees, the operators are also required to make no distinction.

A representation of the orientation by one scalar value representing the angle turns out to be not appropriate, because a measure for certainty which describes the neighborhood independently from the absolute density is not involved. This means that an at least two-component vectorial description is required.

An attempt to describe a neighborhood by the gradient vectors fails because it does not allow to distinguish between neighborhoods with constant intensity, with isotropic intensity distribution, and with ideally oriented patterns (section 3.2.2 and Fig. 3.3).

The following optimization method has been proposed [33] to find an operator which encodes the angle of orientation, provides a certainty measure and distinguishes between constant and isotropic distributions.

A neighborhood $U(p_0)$ of the point of interest p_0 will be described by a unit vector $\boldsymbol{s}$ which exhibits the least deviation from the *orientations* of all gradients ∇g from $U(p_0)$. The quadratic form $(\boldsymbol{s}^T \nabla g)^2 = \boldsymbol{s}^T (\nabla g \nabla g^T) \boldsymbol{s}$ meets a criterion for evaluating this deviation – the more the vectors $\boldsymbol{s}$, ∇g tend to be collinear, the higher values the quadratic form produces. A search for $\boldsymbol{s}$ therefore leads to maximization of the following integral:

$$\int_{U(p_0)} h(p_0 - u)\ (\boldsymbol{s}^T \nabla g(u))^2\ \mathrm{d}u \tag{6.1}$$

where h is a positive weighting function defined on $U(p_0)$.

The maximization of the integral (6.1) can more conveniently be rewritten using the matrix $\mathcal{J}$ of the quadratic form of the integrand. Matrix

$$\mathcal{J} = \int_{U(p_0)} h(p_0 - u)\ (\nabla g(u) \nabla g(u)^T)\ \mathrm{d}u \tag{6.2}$$

referred to as the *structure tensor*, is a real symmetric, positive semidefinite 3×3 matrix consisting of the following elements:

$$\mathcal{J}_{pq} = \int_{U(p_0)} h(p_0 - u) \left(\frac{\partial g(u)}{\partial x_p} \frac{\partial g(u)}{\partial x_q} \right)\ \mathrm{d}u \tag{6.3}$$

With $\mathcal{J}$, maximization of the integral (6.1) rewrites to $\boldsymbol{s}^T \mathcal{J} \boldsymbol{s} \rightarrow \textit{maximum}$. Following Theorem 2.3.1 the solution to $\boldsymbol{s}$ is the principal eigenvector of $\mathcal{J}$.

Since the matrix $\mathcal{J}$ is positive semidefinite, its eigenvalues λ_i are non-negative. In the following we assume the ordering $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and, in accordance with Theorem 2.2.3, we choose the corresponding orthogonal eigenvectors $\boldsymbol{e}_i$ to be of a unit length.

6.3.2 Eigenvectors-aligned kernel

The aim of our interpolation approach is to preserve, to the most possible extent, the boundaries of objects as well as lines, edges, and ridges. To

achieve this we propose to perform the interpolation in the directions given by these structures. The presumption here is that the change of the density values in these directions is minimal.

Although the structure tensor has originally been proposed to describe the orientation of a neighborhood by its principal eigenvector $\boldsymbol{e}_1$, there is more information from its analysis. For a unit vector $\boldsymbol{u}$, the term $\boldsymbol{u}^T \mathcal{J} \boldsymbol{u}$ gives the square of the density change in the direction of $\boldsymbol{u}$. Following the Theorem 2.3.1, the minimum of the squared density change and therefore also the minimum of the density change are reached in the direction of the eigenvector $\boldsymbol{e}_3$.

According to the rank of tensor $\mathcal{J}$ the following four cases can be distinguished in a 3D image (refer also to Table 6.1):

rank $(\mathcal{J})$	Conditions	Neighborhood description
0	$\lambda_1 \simeq \lambda_2 \simeq \lambda_3 \simeq 0$	constant neighborhood
1	$\lambda_1 > \lambda_2 \simeq \lambda_3 \simeq 0$	boundary or layered texture
2	$\lambda_1 \simeq \lambda_2 > \lambda_3 \simeq 0$	edge or extruded texture
3	$\lambda_1 \simeq \lambda_2 \simeq \lambda_3 > 0$	corner or isotropy

Table 6.1: The cases of density distribution in a 3D scene

rank $(\mathcal{J}) = 0$: The scalar values do not change in any direction. The represented neighborhood features constant gray values. Interpolation can simply be done with the nearest-neighborhood interpolation method.

rank $(\mathcal{J}) = 1$: The scalar values significantly change in the direction of the principal eigenvector $\boldsymbol{e_1}$, while they remain (nearly) constant in the remaining perpendicular directions, $\boldsymbol{e_2}$ and $\boldsymbol{e_3}$. The corresponding neighborhood contains either a layered texture or a boundary between two objects. The interpolation will be driven by a disc spanned by the eigenvectors $\boldsymbol{e_2}$ and $\boldsymbol{e_3}$.

rank $(\mathcal{J}) = 2$: The scalar values significantly change in the directions $\boldsymbol{e_1}$ and $\boldsymbol{e_2}$ while remain (nearly) constant in $\boldsymbol{e_3}$. The corresponding neighborhood features either an edge of an object or an extruded texture. The interpolation will be driven by a stick determined by the eigenvector $\boldsymbol{e_3}$.

rank $(\mathcal{J}) = 3$: The scalar values change in all directions. There is either a corner of an object or a distributed 3D texture in the neighborhood. The interpolation therefore shall consider all of the eigenvectors.

The second row of Figure 6.2 gives examples of color coding of the rank of the structure tensor. The areas with $\operatorname{rank}(\mathcal{J}) = 1$, 2, and 3 are encoded by green, red, and blue color, respectively. Homogeneous areas are excluded from the rendering.

To perform a directional interpolation at a point p_0 we will weight the density contributions $g(y)$ of voxels y from its neighborhood according to their relative position to an ellipsoid $E(p_0)$. The main axes of the ellipsoid will be given by scaled eigenvectors $\boldsymbol{a}_i$ as follows:

$\operatorname{rank}(\mathcal{J})$	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	$\boldsymbol{a}_3$
1	$\varepsilon\ \boldsymbol{e}_1$	$\delta\ \boldsymbol{e}_2$	$\delta\ \boldsymbol{e}_3$
2	$\varepsilon\ \boldsymbol{e}_1$	$\varepsilon\ \boldsymbol{e}_2$	$\delta\ \boldsymbol{e}_3$
3	$\delta\ \boldsymbol{e}_1$	$\delta\ \boldsymbol{e}_2$	$\delta\ \boldsymbol{e}_3$

(6.4)

where $\delta > 1 \gg \varepsilon > 0$. The purpose of scaling (6.4) is to suppress the directions of a large density change and to emphasize the directions of a small density change.

We define the weight of a point y from the interior of ellipsoid $E(p_0)$ as:

$$w(y) = 1 - \sum_{i=1}^{3} \left(\frac{(y - p_0)^T \boldsymbol{a_i}}{\boldsymbol{a_i}^T \boldsymbol{a_i}} \right)^2 \tag{6.5}$$

and set it to zero for exterior points and points on the boundary of $E(p_0)$. This defines a smooth fade-of of weights from the center of the ellipsoid to its borders. Fig. 6.1 demonstrates how the situation may look like in a plane given by the eigenvectors $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$. The gray-level coding of the contributing grid points corresponds to their weights.

Finally, we define the interpolated density value as a weighted average:

$$g(p_0) = \frac{\sum_{y \in U(p_0)} w(y)\ g(y)}{\sum_{y \in U(p_0)} w(y)} \tag{6.6}$$

6.3.3 Tensor propagation

With the framework for calculating derivatives assumed in this thesis, the computation of the structure tensor due to Equation (6.3) only applies to grid points. To calculate the directional information also at off-grid points a mechanism to compute the structure tensor there is needed.

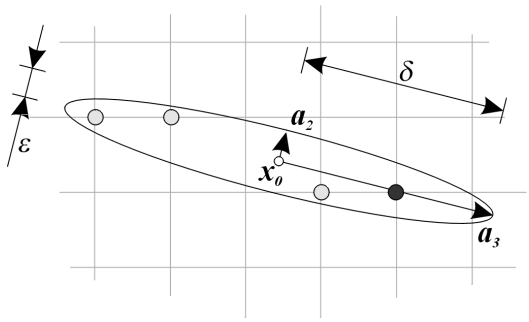


Figure 6.1: The ellipsoid given by the scaled eigenvectors and the weighting of the contributing grid points. The similarity in weighting of the three light-gray grid points is due to Equation (6.5).

A straightforward solution would be to generalize Equation (6.3) involving derivative filters for computing partial derivatives between grid points [62, 87].

An indirect approach can benefit from the already computed information at the surrounding grid points employing an interpolation of the corresponding eigensystems. Using quaternions for this task is the first choice for a high quality interpolation. On the other hand, since there is a strong coherence both in orientation and magnitude of the structure tensor, it is not necessary to sample the tensor field densely [33]. Therefore nearest-neighbor interpolation provides a stable and much simpler solution.

6.4 Implementation

As the interpolation usually is to be performed for a larger amount of voxels, the implementation can be divided into two steps. In preprocessing, the eigenvalue analysis of the structure tensor is performed for a set of voxels which surround the positions to be interpolated.

P1. computation of the structure tensor $\mathcal{J}$: Identifying a convolution in Equation (6.3), the elements $\mathcal{J}_{pq}$ of the structure tensor $\mathcal{J}$ can be computed in two convolution passes. In the first pass the differential operators $\partial/\partial x_i$ are applied resulting in the partial derivatives $\partial g/\partial x_i$. Then the products $\partial g/\partial x_p \cdot \partial g/\partial x_q$ are computed. In the second convolution pass the products are smoothed with a filter corresponding to the weighting function h. We have used the optimized Sobel filter

(Appendix A.4) for the differentiation step and the Gaussian filter (Appendix A.5) defined on a $7 \times 7 \times 7$ neighborhood for the averaging step. To reduce the computational overhead we exploited the separability of both filters.

P2. eigen-analysis of $\mathcal{J}$: Since the structure tensor $\mathcal{J}$ is represented by a real symmetric matrix, the calculation schema of section 2.4 can be applied.

After the preprocessing step the interpolation involves the following steps for each requested position p_0.

I1. inheriting the tensor information: In accordance with section 6.3.3 we have used nearest neighbor interpolation for the tensor transfer.

I2. interpolation: For all voxels y_i from the neighborhood of p_0 determined by the scaling factor δ, weights $w(y_i)$ are computed according to Equations (6.4) and (6.5) and the weighted average according to Equation (6.6) is taken as the result of the interpolation. The values of δ and ε have been set empirically, after an error analysis from several experiments to $\varepsilon = 0.1$ and $\delta = 2.2$.

6.5 Evaluation

The purpose of this section is to provide a three-fold analysis of the performance of the proposed method. Firstly, in section 6.5.1 we give a quantitative error analysis and a comparison to the linear filtering in z direction. Secondly, a visual evaluation of error volumes is presented in section 6.5.2. Thirdly, the time and space complexity of our method is discussed in section 6.5.3.

For the following analysis and comparison we have used 11 data sets covering a broad spectrum in terms of modality, resolution, noise characteristics and object detail. The first group consists of the small-resolution, noise-free artificial data sets generated with the *vxt library* [84]. The second group comprises data acquired from CT and MRI scanners. All data sets were quantized to 256 levels.

6.5.1 Quantitative evaluation

The approach to the comparison is to pretend there is a slice missing in the volume, to estimate this slice and compare it to the original. For the comparison, we reuse the framework for error analysis by Grevera and Udupa [17]. The authors introduced three *figures-of-merit* (FOM). In the following expression of FOMs, m denotes an interpolation method (linear in z and directional), S is the data set being interpolated, $g(v)$ represents the original density value in the voxel given by coordinates v, and $g_m(v)$ represents the density estimation due to the method m at the voxel given by coordinates v. Interpolation runs over all slices of S, where slice V^k is defined by its z-coordinate k.

1. Mean-squared difference:

$$\mathrm{FOM}_m^1(S) = \frac{1}{N} \sum_k \sum_{v \in V^k} (g(v) - g_m(v))^2 \tag{6.7}$$

   where N is the total number of voxels involved in the comparison.

2. Number of sites of disagreement

$$\mathrm{FOM}_m^2(S) = \sum_k \sum_{v \in V^k} \tau(|g(v) - g_m(v)|) \tag{6.8}$$

   where

$$\tau(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.9}$$

3. Largest difference

$$\mathrm{FOM}_m^3(S) = \max_{\substack{v \in V^k \\ k}} \{|g(v) - g_m(v)|\} \tag{6.10}$$

The measurements due to the Equations (6.7)–(6.10) for the linear interpolation and the proposed directional interpolation are summarized in Table 6.2.

We make the following three observations regarding the performance of our technique.

1. Our technique yields on average better estimates of the original density values than the linear interpolation. This is especially remarkable for the *vxt* data. As the Engine data set features many z-axis aligned objects the directional interpolation does not outperform the linear interpolation considerably.

Data set	FOM[1] z-lin.	FOM[1] dir.	FOM[2] z-lin.	FOM[2] dir.	FOM[3] z-lin.	FOM[3] dir.
vxt Wire Tetrahedron	13.1	4.5	1 380	1 156	30	16
vxt Wire Octahedron	41.0	0.1	9 672	1 428	24	3
vxt Wire Dodecahedron	14.3	1.8	1 300	963	37	17
vxt Facet Tetrahedron	14.9	4.7	7 877	6 846	24	23
vxt Facet Octahedron	77.7	0.1	22 895	5 644	35	5
vxt Facet Dodecahedron	35.9	2.5	19 612	17 891	31	16
Lobster	51.0	19.8	6 815	6 567	103	61
Engine Block	7.1	6.3	705 850	609 911	36	34
CT Head with Markers	105.3	25.1	133 312	131 901	124	114
MRI Head	84.7	41.3	570 784	556 033	116	90
Vertebra 1	7.0	4.8	189 584	114 351	32	26

Table 6.2: Error measurements due to the Equations (6.7)–(6.10) for the linear interpolation in z direction and the newly proposed directional interpolation.

2. An exact reconstruction of the original density values happens at more sites in the case of directional interpolation. Again, it is more obvious in the case of the *vxt* data and gets almost negligible for the scanned data sets. In our opinion, however, these measurements are of lower importance than the mean-squared error.

3. The maximal error is generally smaller in the case of directional interpolation.

6.5.2 Visual evaluation

To evaluate the performance of both interpolation methods visually, we create two error volumes consisting of voxels with densities set to $\mid g_m(v) - g(v) \mid$, where m denotes the interpolation method. The error volumes are then displayed with the OpenSplat package [29]. The 3rd and the 4th rows of Figure 6.2 show examples of rendering the error volume introduced by the linear and the directional interpolation, respectively.

Since the directional interpolation yields very small differences in the case of the vxt Facet Octahedron data set, almost every pixel is assigned to the background color in the corresponding image. In case of the CT head with markers the improvement by our interpolation is mostly apparent at the positions featuring objects, e.g, the wires in the markers. As the error

Input Volume		Eigensystem of $\mathcal{J}$			Interpo	Total
Data set	Resolution	Sobel	Gauss	$\{\lambda_i, \boldsymbol{e}_i\}$	lation	
vxt Wire Tetrahedron	50× 44× 42	1.2	3.4	0.03	1.6	6.2
vxt Wire Octahedron	59× 59× 59	2.3	6.2	0.05	3.3	11.9
vxt Wire Dodecahedron	58× 56× 48	1.8	4.9	0.05	3.2	10.0
vxt Facet Tetrahedron	50× 45× 42	1.1	3.0	0.03	2.7	6.8
vxt Facet Octahedron	59× 59× 59	2.4	6.7	0.05	5.2	14.3
vxt Facet Dodecahedron	59× 56× 49	2.0	5.3	0.06	6.8	14.2
Lobster	120×120× 34	5.7	15.6	0.16	5.8	27.3
Engine Block	256×256×110	88.8	383.2	2.43	140.0	614.4
CT Head with Markers	256×256× 44	33.4	110.6	0.78	39.4	184.2
MRI Head	256×256×109	74.7	249.1	2.07	127.3	453.2
Vertebra 1	128×128× 74	12.7	50.4	0.39	20.4	83.9

Table 6.3: Time in seconds for reconstruction of all volume slices with the directional interpolation as measured on a 400 MHz Pentium II.

measurements for the Engine data set do not differ significantly a visual evaluation is almost impossible in this case.

6.5.3 Time and space complexity

Referring to Table 6.3, the performance of our method is clearly dominated by the smoothing with the Gaussian filter (section 6.4, P1). The total timing therefore strongly depends on the size of the support of the weighting function h. Our implementation processes in average 11000 voxels per second on a 400 MHz Pentium II. This corresponds to 5.12 seconds for reconstruction of one 256×256 slice on average.

To store the eigenvectors of the structure tensor, our implementation requires a space of $O(3n)$ where n is the size of the data to be interpolated.

6.6 Concluding remarks

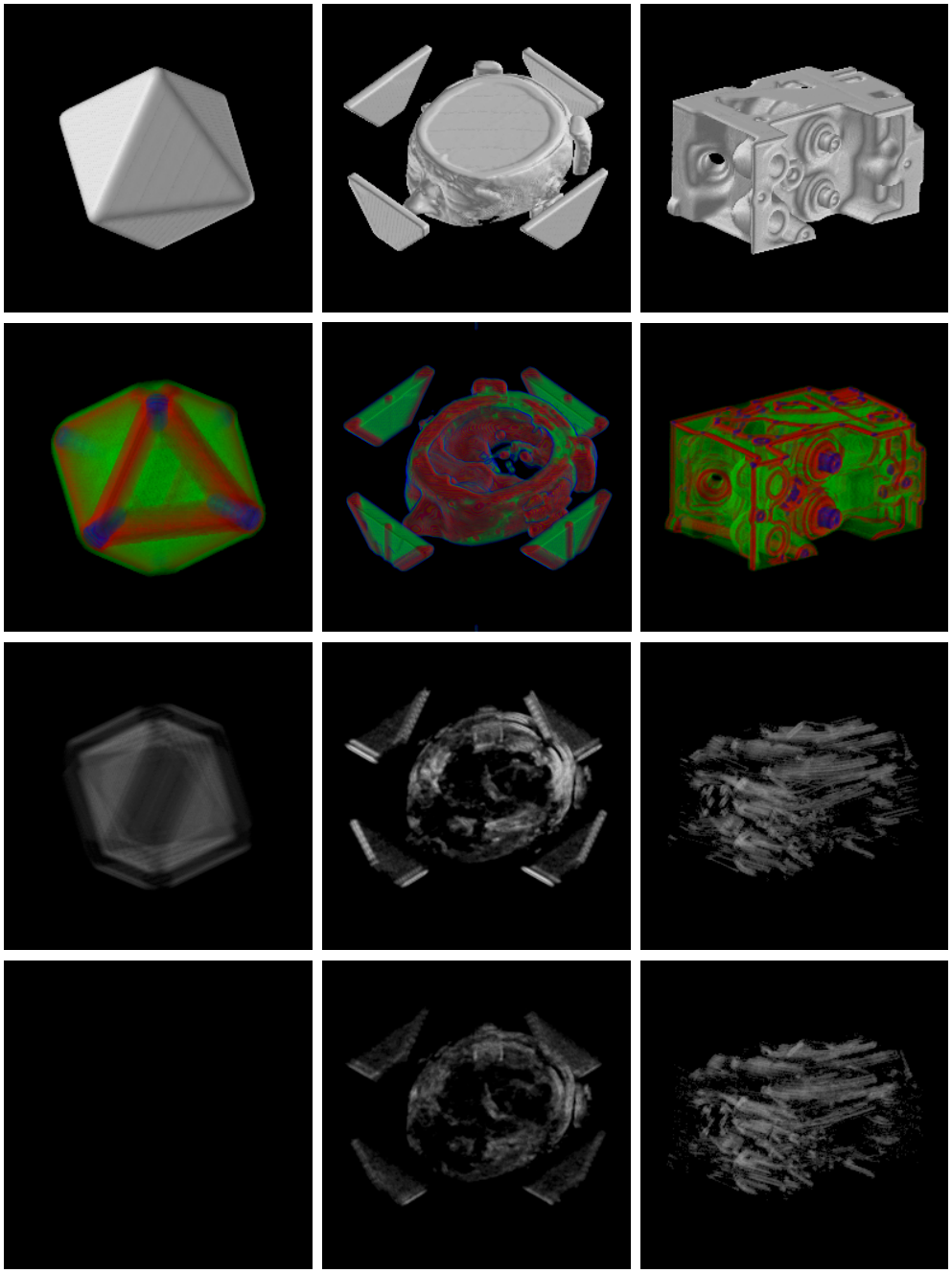
We presented a new method for interpolating gray-level volume data taking into account topological properties of the structures. Unlike the methods which are based on an a priori information from segmentation, our method processes the density data directly requiring no user interaction.

In order to preserve the boundaries of objects, lines, and ridges as far as possible, we proposed to perform the interpolation along the directions of these structures. The information on the direction of interpolation is inherited from the textural description of the objects through structure tensors.

For 11 data sets, we compared the performance of our method to the most frequently used [17] linear interpolation in the direction of the z axis. Our method outperforms linear interpolation in the mean error, maximal error and in the number of sites with exact reconstruction.

There are two main issues not addressed in this work. Firstly, the size of the oriented filter is given by two constants which have been set empirically after many experiments. In our opinion this is not appropriate if the method is required to perform robustly for all kind of data. Instead of using fixed constants we think of using parameters which reflect the *magnitude* of the density change separately in each neighborhood. A natural choice for the scaling factors seem to be the *eigenvalues* of the structure tensor. In preliminary tests, however, the performance of such a scaling was inferior as compared to the constants we introduced.

Secondly, we find it necessary to compare our method to other techniques, especially to the state-of-the-art interpolation introduced by Grevera and Udupa [16]. We would like to conclude with remarks on possible results of this comparison. Clearly, our method requires far less space and time. To interpolate one slice of n voxels, the algorithm by Grevera and Udupa requires space of $O(Qn)$ where Q denotes the quantization level and time in the order of tens of minutes (as measured on a Sparc 2). In contrast, our method interpolates one slice in a couple of seconds requiring $O(3n)$ space. Finally, since we have used the same error measurements and a statistical comparison to the same interpolation method as Grevera and Udupda [16], we were able to indirectly compare the error performance of both methods. Even though this preliminary comparison indicates that our method will perform better, further work is required in this respect. We are convinced that such a comparison has to be done directly and for the same data sets.



vxt Facet Octahedron CT Head with Markers Engine Block

Figure 6.2: Volume rendering of the original data sets (1st row), color coding of the rank of the structure tensor (2nd row), error volume of linear interpolation (3rd row), and error volume of directional interpolation (4th row).

Chapter 7

Summary and conclusion

In the context of volume visualization, discussions on derivatives usually lead towards reconstruction and use of the gradient vector and its magnitude. Discussions on higher-order derivatives are rather seldom.

One contribution of this thesis is the state-of-the-art chapter 3 which in addition discussed the structure tensor and a variety of combinations of 2nd-order information included in the Hessian matrix. Our contribution to the field is reflected in the rest of the thesis:

In Chapter 4 we introduced a novel definition of transfer functions which are suggested to reflect shape properties of isosurfaces. Since such transfer functions are independent of the actual data values they are supposed to be acquisition independent.

In chapter 5 we presented an approach to identify volume-data features based on the eigenvalues of the Hessian matrix. We showed that such a kind of identification yields better results as compared to those based on the gradient-magnitude and is well suited for progressive transmission and data representation.

In chapter 6 we introduced a new shape-based interpolation method. For the interpolation we used kernels which are aligned to the eigensystem of the structure tensor.

As the chapters 4– 6 deal with distinct-enough topics, we concluded them separately. Nevertheless, more general conclusions which are based on the acquired experience during the development of concepts and implementation, can be pointed out.

- The motivation for derivative-based analysis of 2D and 3D images arises from the observation that the discontinuities in depth, surface orienta-

tion, reflectance, or illumination (the human visual system is sensitive to) correspond to changes in image intensity.

- The information in 3D imaging carried by 9 distinct numbers, i.e., 3 gradient components plus 6 elements of the Hessian matrix, is a source for a spectrum of powerful combinations.
- The barrier which makes difficulties in extending the concepts from 2D imaging to higher dimensions resides in the additional freedom in choosing the direction to be examined.
- Eigensystems of symmetrically composed matrices of derivatives give a particular answer to the above problem. This is due to the correspondence between eigensystems of the real symmetric matrices and the minimum and maximum values of the associated quadratic forms.
- Computation of eigensystems of real symmetric matrices is relatively cheap as compared to general matrices. This fact has to be considered during implementation.

In this thesis we only researched derivatives up to the second order. Being motivated by results coming from 2D image processing we believe that there is a strong potential in investigating also derivatives of order larger than two.

Appendix A

Separable derivative kernels

This appendix lists the derivative kernels used in the thesis. Starting from the most frequently used central differences, we recall the separable Prewitt and Sobel kernels for 1st-order differentiation, and the Gaussian kernels and its derivatives for 1st- and 2nd-order differentiation.

To be more illustrative we also give the corresponding 2D counterparts as they appear in the image processing literature.

The following notation is used: $X_a^{\rightarrow}$ denotes a 1D kernel oriented in the direction of the x_a-axis. X_{ab} denotes a 2D kernel X which performs an ath-order differentiation in direction of x_1 and an bth-order differentiation in direction of x_2. Similarly, X_{abc} denotes a 3D kernel X which performs an ath-order differentiation in direction of x_1, a bth-order differentiation in direction of x_2, and a cth-order differentiation in direction of x_3.

A.1 Central differences

Central differences arise naturally from the definition of continuous derivatives. Since no smoothing in the cross-direction(s) is performed, their use is only reasonable when the spacing between the signal-samples is well below the Nyquist limit [34, p. 411]. In spite of that they are, for performance reasons, a very frequent choice.

$$C_1^{\rightarrow} = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad C_2^{\rightarrow} = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

and $C_3^{\rightarrow}$ is an analogous kernel performing the differentiation in the direction of the x_3 axis.

A 2D (nD) separable derivative kernel with the same support size in every coordinate direction can be composed of a 1D derivative kernel and 1D smoothing kernel(s) in the orthogonal direction(s) [34, p. 146].

A.2 Prewitt kernels

Prewitt kernels involve central differences in directions of differentiation and box filters, $\sqcap = \frac{1}{3}\,[1\ 1\ 1]$, for smoothing.

$$P_{10} = \frac{1}{6}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \otimes \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = C_1^{\leftrightarrow} \otimes \sqcap_2^{\rightarrow}$$

$$P_{01} = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \sqcap_1^{\rightarrow} \otimes C_2^{\leftrightarrow}$$

The 3D kernels are computed as follows:

$$P_{100} = C_1^{\leftrightarrow} \otimes \sqcap_2^{\rightarrow} \otimes \sqcap_3^{\rightarrow}$$
$$P_{010} = \sqcap_1^{\rightarrow} \otimes C_2^{\leftrightarrow} \otimes \sqcap_3^{\rightarrow}$$
$$P_{001} = \sqcap_1^{\rightarrow} \otimes \sqcap_2^{\rightarrow} \otimes C_3^{\leftrightarrow}$$

A.3 Sobel kernels

Sobel kernels involve central differences in directions of differentiation and triangle filters, $\wedge = \frac{1}{4}\,[1\ 2\ 1]$, for smoothing.

$$S_{10} = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = C_1^{\leftrightarrow} \otimes \wedge_2^{\rightarrow}$$

$$S_{01} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \wedge_1^{\rightarrow} \otimes C_2^{\leftrightarrow}$$

The 3D kernels are computed as follows:

$$S_{100} = C_1^{\leftrightarrow} \otimes \wedge_2^{\rightarrow} \otimes \wedge_3^{\rightarrow}$$
$$S_{010} = \wedge_1^{\rightarrow} \otimes C_2^{\leftrightarrow} \otimes \wedge_3^{\rightarrow}$$
$$S_{001} = \wedge_1^{\rightarrow} \otimes \wedge_2^{\rightarrow} \otimes C_3^{\leftrightarrow}$$

A.4 Optimized Sobel kernels

Optimized Sobel kernels involve central differences in directions of differentiation and a variant of the triangle filters, $\dot{\Lambda} = \frac{1}{16}$ [3 10 3], for smoothing. The filter is optimized for gradient direction [33].

$$\dot{S}_{10} = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 3 \\ 10 \\ 3 \end{bmatrix} = C_1^{\rightarrow} \otimes \dot{\Lambda}_2^{\rightarrow}$$

$$\dot{S}_{01} = \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 3 & 10 & 3 \end{bmatrix} \otimes \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \dot{\Lambda}_1^{\rightarrow} \otimes C_2^{\rightarrow}$$

The 3D kernels are computed as follows:

$$\begin{aligned} \dot{S}_{100} &= C_1^{\rightarrow} \otimes \dot{\Lambda}_2^{\rightarrow} \otimes \dot{\Lambda}_3^{\rightarrow} \\ \dot{S}_{010} &= \dot{\Lambda}_1^{\rightarrow} \otimes C_2^{\rightarrow} \otimes \dot{\Lambda}_3^{\rightarrow} \\ \dot{S}_{001} &= \dot{\Lambda}_1^{\rightarrow} \otimes \dot{\Lambda}_2^{\rightarrow} \otimes C_3^{\rightarrow} \end{aligned}$$

A.5 Gaussian kernels

Gaussian kernels involve the Gaussian function and its derivatives for smoothing and differentiation, respectively. For a general dimension n the Gaussian function is defined as follows:

$$n\mathrm{D}: \quad G(\boldsymbol{x};\sigma) = \frac{1}{(\sqrt{2\pi\sigma^2})^n} \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2\sigma^2}\right) \tag{A.1}$$

where σ is the standard deviation. The plots of the 1D and 2D Gaussian and its derivatives are shown in Figs. A.1– A.3.

Since both the Gaussian function and its derivatives are separable it is only necessary to sample the 1D versions

$$G(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{A.2}$$

$$G'(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \frac{-x}{\sigma^2} \tag{A.3}$$

$$G''(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \frac{x^2-\sigma^2}{\sigma^4} \tag{A.4}$$

and combine them into the first-order derivative kernels:

$$G_{100} = G_1^{\leftrightarrow\prime} \otimes G_2^{\leftrightarrow} \otimes G_3^{\leftrightarrow}$$
$$G_{010} = G_1^{\leftrightarrow} \otimes G_2^{\leftrightarrow\prime} \otimes G_3^{\leftrightarrow}$$
$$G_{001} = G_1^{\leftrightarrow} \otimes G_2^{\leftrightarrow} \otimes G_3^{\leftrightarrow\prime}$$

and into the second-order derivative kernels:

$$G_{200} = G_1^{\leftrightarrow\prime\prime} \otimes G_2^{\leftrightarrow} \otimes G_3^{\leftrightarrow} \quad G_{110} = G_1^{\leftrightarrow\prime} \otimes G_2^{\leftrightarrow\prime} \otimes G_3^{\leftrightarrow} \quad G_{101} = G_1^{\leftrightarrow\prime} \otimes G_2^{\leftrightarrow} \otimes G_3^{\leftrightarrow\prime}$$
$$G_{020} = G_1^{\leftrightarrow} \otimes G_2^{\leftrightarrow\prime\prime} \otimes G_3^{\leftrightarrow} \quad G_{011} = G_1^{\leftrightarrow} \otimes G_2^{\leftrightarrow\prime} \otimes G_3^{\leftrightarrow\prime}$$
$$G_{002} = G_1^{\leftrightarrow} \otimes G_2^{\leftrightarrow} \otimes G_3^{\leftrightarrow\prime\prime}$$

An important question to be answered is the relationship between the width of the kernel and the standard deviation σ. To avoid an early truncation of the Gaussian function, the width of the kernel has to be proportional to the standard deviation σ (see also Fig. A.1). Although there is quite a large variance in the literature concerning this trade-off between performance and accuracy, many authors [81, 82, 86] agree that the *radius* of the kernel should be at least 4σ.

In the following we give the sampled 1D kernels for computation of derivatives in $5 \times 5 \times 5$ and $7 \times 7 \times 7$ neighborhoods.

directly sampled: width $= 5$; $\sigma = 0.5$

$$\begin{array}{lrrrrrr} G & = & [\ 0.07 & 27.64 & 204.26 & 27.64 & 0.07\ ] \ / \ 256 \\ G' & = & [\ 0.55 & 110.57 & 0. & -110.57 & -0.55\ ] \ / \ 256 \\ G'' & = & [\ 4.11 & 331.72 & -817.03 & 331.72 & 4.11\ ] \ / \ 256 \end{array}$$

optimized for gradient direction [34, p. 147]: width $= 5$; $\sigma \approx 0.85$

$$\begin{array}{lrrrrrr} G & = & [\ 5.96 & 61.81 & 120.46 & 61.81 & 5.96\ ] \ / \ 256 \\ G' & = & [\ 21.38 & 85.24 & 0. & -85.24 & -21.38\ ] \ / \ 256 \\ G'' & = & [\ 47.35 & 32.70 & -167.58 & 32.70 & 47.35\ ] \ / \ 256 \end{array}$$

directly sampled: width $= 7$; $\sigma = 0.75$

$$\begin{array}{lrrrrrrrr} G & = & [\ 0.05 & 3.89 & 55.98 & 136.17 & 55.98 & 3.89 & 0.05\ ] \ / \ 256 \\ G' & = & [\ 0.24 & 13.83 & 99.52 & 0. & -99.52 & -13.83 & -0.24\ ] \ / \ 256 \\ G'' & = & [\ 1.22 & 42.26 & 77.41 & -242.08 & 77.41 & 42.26 & 1.22\ ] \ / \ 256 \end{array}$$

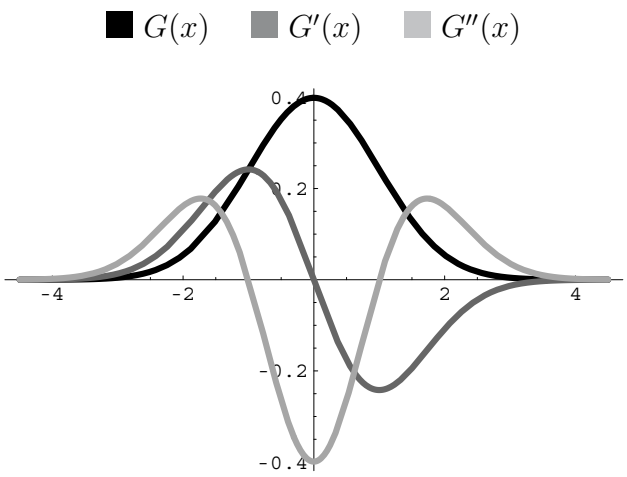
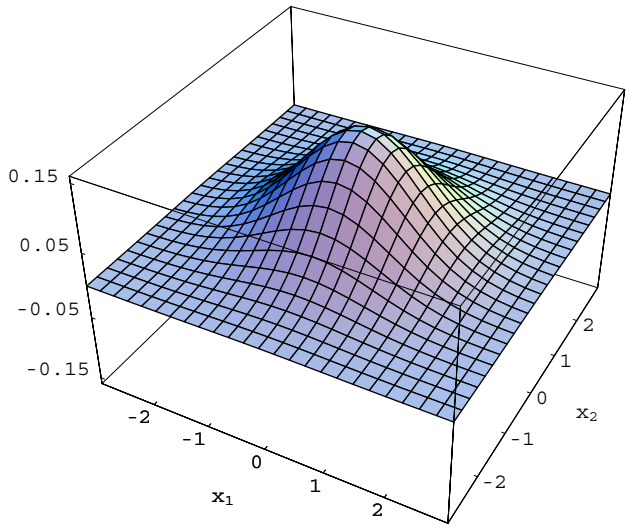
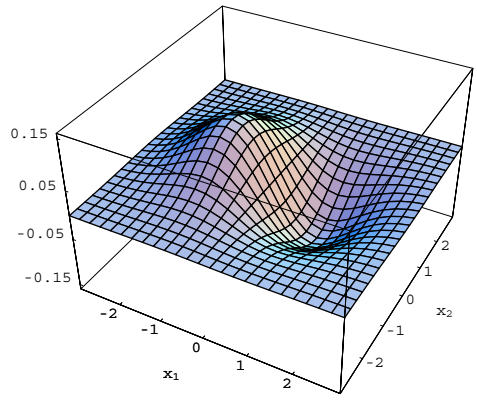


Figure A.1: 1D Gaussian function and its derivatives. $\sigma = 1$.

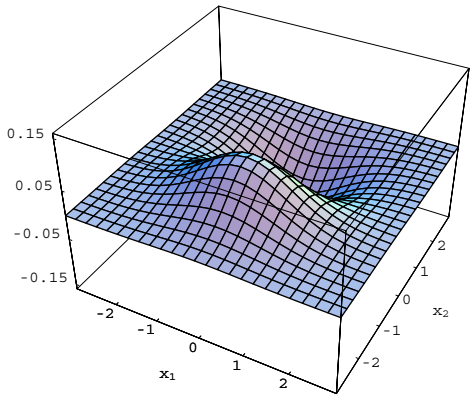


$G(x_1, x_2)$

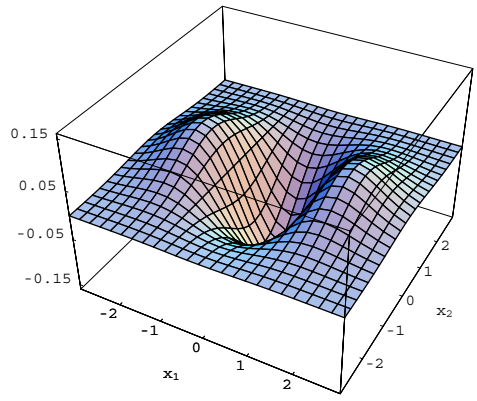
Figure A.2: 2D Gaussian function with $\sigma = 1$.



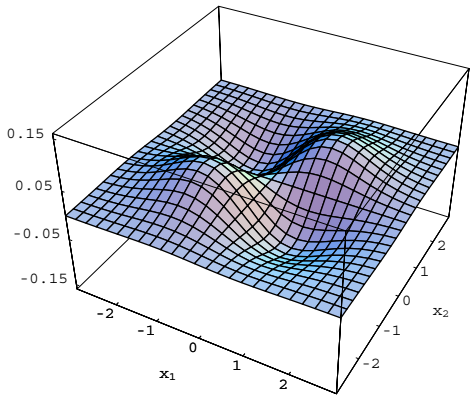
$\partial G(x_1, x_2)/\partial x_1$



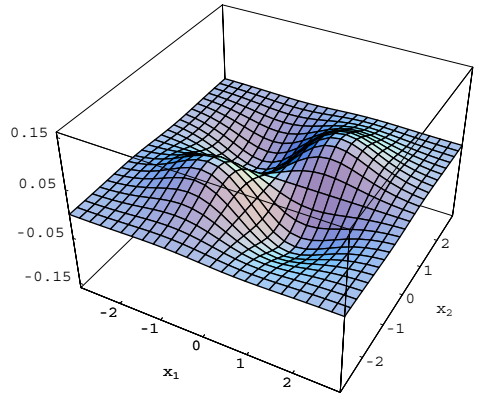
$\partial G(x_1, x_2)/\partial x_2$



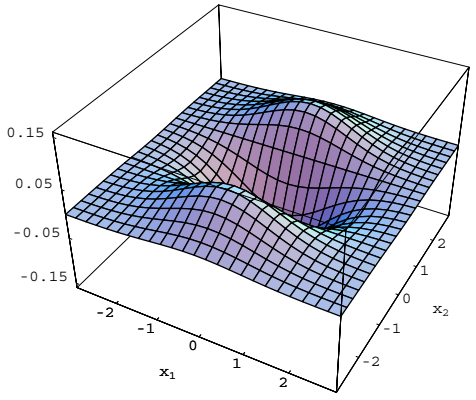
$\partial^2 G(x_1, x_2)/\partial x_1^2$



$\partial^2 G(x_1, x_2)/\partial x_1 \partial x_2$



$\partial^2 G(x_1, x_2)/\partial x_2 \partial x_1$



$\partial^2 G(x_1, x_2)/\partial x_2^2$

Figure A.3: The 1st- and 2nd-order derivatives of the 2D Gaussian function with $\sigma = 1$.

Bibliography

[1] Numerical Recipes Home Page. http://www.nr.com/. Referenced on page(s) 18

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proceedings of IEEE Visualization*, pages 167–175, 1997. Referenced on page(s) 41, 47, 72

[3] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*, chapter 1.6.2: Gleichungen 1. bis 4. Grades, pages 40–41. Verlag Harri Deutsch, 4 edition, 1993. Referenced on page(s) 19

[4] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 263–272, 1993. Referenced on page(s) 43

[5] M. Chen, A. E. Kaufman, and R. Yagel, editors. *Volume Graphics*. Springer Verlag, 2000. Referenced on page(s) 13

[6] B. Csébfalvi. *Interactive Volume-Rendering Techniques for Medical Data Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, June 2001. Referenced on page(s) 63

[7] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. In *Proceedings of EUROGRAPHICS*, pages 452–460, 2001. Referenced on page(s) 32

[8] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization*, pages 195–202, 2000. Referenced on page(s) 32

[9] T. T. Elvins. A survey of algorithms for volume visualization. In *Proceedings of ACM SIGGRAPH*, pages 194–201, 1992. Referenced on page(s) 11

[10] S. Fang, T. Biddlecom, and M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *Proceedings of IEEE Visualization*, pages 319–326, 1998. Referenced on page(s) 47

[11] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Academic Press, 4. edition, 1996. Referenced on page(s) 43

[12] O. Faugeras. *Three-Dimensional Computer Vision (A Geometric Viewpoint)*. MIT Press, 1993. Referenced on page(s) 13, 35

[13] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. *Lecture Notes in Computer Science*, 1496:130–137, 1998. Referenced on page(s) 27, 37, 73

[14] I. Fujishiro, T. Azuma, and Y. Takeshima. Automating transfer function design for comprehensible volume rendering based on 3D field topology analysis. In *Proceedings of IEEE Visualization*, pages 467–470, 1999. Referenced on page(s) 47

[15] T. Gerstner. Fast multiresolution extraction of multiple transparent isosurfaces. In *Data Visualization 2001, Proceedings of the Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization*, pages 35–44,336, 2001. Referenced on page(s) 64

[16] G. Grevera and J. Udupa. Shape-based interpolation of multidimensional grey–level images. *IEEE Transactions on Medical Imaging*, 15(6):881–892, 1996. Referenced on page(s) 82, 83, 93

[17] G. Grevera and J. Udupa. An objective comparison of 3-D image interpolation methods. *IEEE Transactions on Medical Imaging*, 17(4):642–652, 1998. Referenced on page(s) 83, 90, 93

[18] G. Grevera, J. Udupa, and Y. Miki. A task-specific evaluation of three-dimensional image interpolation techniques. *IEEE Transactions on Medical Imaging*, 18(2):137–143, 1999. Referenced on page(s) 84

[19] H. Haußecker and B. Jähne. A tensor approach for local structure analysis in multi-dimensional images. In *Proceedings of 3D Image Analysis and Synthesis*, 1996. Referenced on page(s) 34

[20] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proceedings of IEEE Visualization*, pages 227–234, 1996. Referenced on page(s) 47

[21] W. Higgins, C. Morice, and E. Ritman. Shape-based interpolation of tree-like structures in three-dimensional images. *IEEE Transactions on Medical Imaging*, 12(3):439–450, 1993. Referenced on page(s) 83

[22] J. Hladůvka and E. Gröller. Direction-driven shape-based interpolation of volume data. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, pages 113–120,521, 2001. Referenced on page(s) 13, 82, 113

[23] J. Hladůvka and E. Gröller. Exploiting the Hessian matrix for content-based retrieval of volume-data features. *To appear in Visual Computer*, 18(2), 2002. Referenced on page(s) 13, 38, 113

[24] J. Hladůvka and E. Gröller. Smallest 2nd-order derivatives for efficient volume-data representation. *To appear in Computers and Graphics*, 26(2), 2002. Referenced on page(s) 13, 38, 113

[25] J. Hladůvka, A. König, and E. Gröller. Curvature-based transfer functions for direct volume rendering. In *Proceedings of Spring Conference on Computer Graphics (SCCG)*, pages 58–65, 2000. Referenced on page(s) 12, 43, 44, 45, 46, 59, 113

[26] J. Hladůvka, A. König, and E. Gröller. Exploiting eigenvalues of the Hessian matrix for volume decimation. In *The 9th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 124–129, 2001. Referenced on page(s) 13, 38, 62, 113

[27] J. Hladůvka, A. König, and E. Gröller. Salient representation of volume data. In *Data Visualization 2001, Proceedings of the Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization*, pages 203–211,351, 2001. Referenced on page(s) 13, 38, 62, 69, 113

[28] M. Hopf and T. Ertl. Accelerating 3D convolution using graphics hardware. In *Proceedings of IEEE Visualization*, pages 471–474, 1999. Referenced on page(s) 29

[29] J. Huang, K. Mueller, N. Shareef, and R. Crawfis. OpenSplat: Software package for optimized splatting on rectilinear grids, ver. 2.0.2. http://www.cis.ohio-state.edu/graphics/research/FastSplats/, Aug. 2000. Referenced on page(s) 75, 78, 79, 91

[30] Intel Corporation. IPL–Intel Image Processing Library, v2.5, 2000. Referenced on page(s) 29

[31] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 109–116, 1997. Referenced on page(s) 43, 44, 48, 50

[32] V. Interrante, H. Fuchs, and S. Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *Proceedings of IEEE Visualization*, pages 211–218, 1996. Referenced on page(s) 43, 48

[33] B. Jähne. *Digital Image Processing*, chapter 11.3 First-order tensor representation, pages 348–364. Springer–Verlag Berlin Heidelberg, 4. edition, 1997. Referenced on page(s) 33, 34, 84, 85, 88, 99

[34] B. Jähne, H. Haußecker, and P. Geißler, editors. *Handbook of Computer Vision and Applications*, volume 2: Signal Processing and Pattern Recognition. Academic Press, 1st edition, 1999. Referenced on page(s) 13, 14, 23, 24, 25, 27, 28, 29, 30, 34, 97, 98, 100

[35] A. Kaufman, editor. *Volume Visualization*. IEEE Computer Society Press, 1991. Referenced on page(s) 13

[36] A. Kaufman, M. Brady, B. Lorensen, F. Kitson, and H. Pfister. Why is real-time volume rendering no longer a year away? Panel discussion at IEEE Visualization, 1998. Referenced on page(s) 11

[37] S. Kilthau and T. Möller. Splatting optimizations. Technical Report TR 2001-02, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, 2001. Referenced on page(s) 63

[38] G. Kindlmann. Semi-automatic generation of transfer functions for direct volume rendering. Master's thesis, Cornell University, 1999. Referenced on page(s) 38

[39] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 79–86, 1998. Referenced on page(s) 32, 38, 40, 41, 47, 48, 64, 73

[40] G. Kindlmann, D. Weinstein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):124–138, 2000. Referenced on page(s) 13

[41] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization*, pages 255–262, 2001. Referenced on page(s) 48

[42] H. Knutsson. Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, 1989. Referenced on page(s) 32, 33

[43] A. König and E. Gröller. Mastering transfer function specification by using VolumePro technology. In *Proceedings of Spring Conference on Computer Graphics (SCCG)*, pages 279–286, 2001. Referenced on page(s) 47

[44] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of ACM SIGGRAPH*, pages 451–458, 1994. Referenced on page(s) 67, 75

[45] T. Lehmann, C. Gonner, and K. Spitzer. Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, 1999. Referenced on page(s) 83

[46] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988. Referenced on page(s) 31, 43, 47

[47] B. Lichtenbelt, R. Crane, and S. Naqvi. *Introduction to Volume Rendering*. Prentice Hall, 1998. Referenced on page(s) 13, 48

[48] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 465–472, 1996. Referenced on page(s) 34, 69, 84

[49] T. Lindeberg. Scale-space: A framework for handling image structures at multiple scales. In *Proceedings of CERN School of Computing*, pages 8–21, 1996. Referenced on page(s) 34

[50] H. Löffelmann, T. Theußl, A. König, and E. Gröller. SMURF: a SMart SURFace model for advanced visualization techniques. In *The 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG)*, pages 156 – 164, 1999. Referenced on page(s) 44

[51] G. Lohmann. *Volumetric Image Analysis*. Chichester Wiley, 1998. Referenced on page(s) 13, 45, 73

[52] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH*, pages 163–170, 1987. Referenced on page(s) 43, 64

[53] C. Lürig and T. Ertl. Hierarchical volume analysis and visualization based on morphological operators. In *Proceedings of IEEE Visualization*, pages 335–342, 1998. Referenced on page(s) 48

[54] S. MacLane and G. Birkhoff. *Algebra*. Macmillan Publishers, 2nd edition, 1968. Referenced on page(s) 16

[55] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH*, pages 389–400, 1997. Referenced on page(s) 47

[56] D. Marr. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman Publishers, 1982. Referenced on page(s) 35, 67

[57] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization*, pages 100–107. IEEE, 1994. Referenced on page(s) 28

[58] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. Referenced on page(s) 48

[59] A. M. McIvor and R. J. Valkenburg. A comparison of local surface geometry estimation methods. *Machine Vision and Applications*, 10(1):17–26, 1997. Referenced on page(s) 52

[60] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Classification and local error estimation of interpolation and derivative filters for volume rendering. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 71–78, 1996. Referenced on page(s) 13, 28

[61] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. A comparison of normal estimation schemes. In *Proceedings of IEEE Visualization*, 1997. Referenced on page(s) 13, 28

[62] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and Design of Filters Using a Taylor Series Expansion. *IEEE Transactions on*

Visualization and Computer Graphics, 3(2):184–199, 1997. Referenced on page(s) 13, 28, 83, 88

[63] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 143–151, 1998. Referenced on page(s) 13, 28

[64] M. Moshfeghi. Directional interpolation for magnetic resonance angiography data. *IEEE Transactions on Medical Imaging*, 12(2):366–379, 1993. Referenced on page(s) 83

[65] L. Mroz. *Real-Time Volume Visualization on Low-End Hardware*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, May 2001. Referenced on page(s) 63

[66] L. Mroz, H. Hauser, and E. Gröller. Interactive high-quality maximum intensity projection. In *Proceedings of EUROGRAPHICS*, pages 341–350, 2000. Referenced on page(s) 64

[67] K. Mueller, T. Möller, and R. Crawfis. Splatting without the blur. In *Proceedings of IEEE Visualization*, pages 363–370, 1999. Referenced on page(s) 63

[68] D. R. Nadeau. Volume visualization: Technology to tools. Key note speach at The 9th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG), 2001. Referenced on page(s) 11, 14, 62, 63

[69] L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4D linear regression. In *Proceedings of EUROGRAPHICS*, pages 351–358, 2000. Referenced on page(s) 32

[70] N. Nikolaidis and I. Pitas. *3-D Image Processing Algorithms*. Wiley-Interscience, 2000. Referenced on page(s) 13

[71] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, 1975. Referenced on page(s) 13, 28

[72] V. Pekar, R. Wiemker, and D. Hempel. Fast detection of meaningful isosurfaces for volume data visualization. In *Proceedings of IEEE Visualization*, pages 223–230, 2001. Referenced on page(s) 42, 44, 47, 73

[73] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. In *Proceedings of ACM SIGGRAPH*, pages 251–260, 1999. Referenced on page(s) 47, 62, 67, 68, 75, 80, 81

[74] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. Visualization viewpoints: The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22, 2001. Referenced on page(s) 47

[75] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH*, pages 335–342, 2000. Referenced on page(s) 63

[76] B.-T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975. Referenced on page(s) 31

[77] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*, chapter 11: Eigensystems, pages 456–469. Cambridge University Press, 2 edition, 1992. Referenced on page(s) 18

[78] S. Raya and J. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, 1990. Referenced on page(s) 83

[79] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH*, pages 343–352, 2000. Referenced on page(s) 63

[80] T. Saito. Real-time previewing for volume visualization. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 99–106, 1994. Referenced on page(s) 63, 64

[81] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis. 3D multi–scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, 2(2):143–168, 1998. Referenced on page(s) 37, 73, 100

[82] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3D local intensity structures for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000. Referenced on page(s) 37, 39, 69, 73, 100

[83] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37:10–21, 1949. Referenced on page(s) 24

[84] M. Šrámek. vxt: A class library for voxelization of geometric objects. http://www.cs.sunysb.edu/ vislab/vxtools/. Referenced on page(s) 58, 89

[85] C.-K. Tang and G. Medioni. Robust estimation of curvature information from noisy 3D data for shape description. In *Proceedings of International Conference on Computer Vision*, pages 426–433, 1999. Referenced on page(s) 44, 48, 52

[86] B. M. ter Haar Romeny and L. M. J. Florack. Front-end vision, a multi-scale geometry engine (lecture notes in computer science). In *Proceedings of IEEE International Workshop on Biologically Motivated Computer Vision*, pages 1–35, 2000. Referenced on page(s) 34, 100

[87] T. Theußl, H. Hauser, and E. Gröller. Mastering windows: Improving reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 101–108, 2000. Referenced on page(s) 13, 28, 83, 88

[88] T. Theußl, T. Möller, and E. Gröller. Optimal regular volume sampling. In *Proceedings of IEEE Visualization*, pages 91–98, 2001. Referenced on page(s) 24

[89] T. Theußl, T. Möller, J. Hladůvka, and E. Gröller. Reconstruction issues in volume visualization. *To appear in Proceedings of Dagstuhl seminar on Scientific Visualization 2000*. Referenced on page(s) 13, 113

[90] P. Thevenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000. Referenced on page(s) 83

[91] P. Todd and R. McLeod. Numerical estimation of the curvature of surfaces. *Computer-Aided Design*, 18(1):33–37, 1986. Referenced on page(s) 52

[92] E. Trucco and R. B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):177–182, 1995. Referenced on page(s) 45, 48

[93] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH*, pages 335–342, 1999. Referenced on page(s) 83

[94] E. W. Weisstein. *The CRC Concise Encyclopedia of Mathematics*. CRC Press, 1998. Referenced on page(s) 25

[95] C. Westin, A. Bhalerao, H. Knutsson, and R. Kikinis. Using local 3D structure for segmentation of bone from computer tomography images. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, 1997. Referenced on page(s) 34

[96] L. Westover. Footprint evaluation for volume rendering. In *Proceedings of ACM SIGGRAPH*, pages 367–376, 1990. Referenced on page(s) 63

[97] M. Worring. *Shape Analysis of Digital Curves*. PhD thesis, Department of Computer Science, Faculty of Science, University of Amsterdam, 1993. Referenced on page(s) 54, 59

[98] M. Worring and A. W. M. Smeulders. Digital curvature estimation. *CVGIP: Image Understanding*, 58(3):366–382, 1993. Referenced on page(s) 54

Related publications

This thesis is based on the following publications:

Chapter 3

[89] T. Theußl, T. Möller, J. Hladůvka, and E. Gröller Reconstruction issues in volume visualization To appear in *Proceedings of Dagstuhl seminar on Scientific Visualization 2000*.

Chapter 4

[25] J. Hladůvka, A. König, and E. Gröller. Curvature-based transfer functions for direct volume rendering. In *Proceedings of Spring Conference on Computer Graphics (SCCG)*, pages 58–65, 2000.

Chapter 5

[26] J. Hladůvka, A. König, and E. Gröller. Exploiting eigenvalues of the Hessian matrix for volume decimation. In *The 9th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 124–129, 2001.
An extended and revised version accepted for [23]:

[23] J. Hladůvka and E. Gröller. Exploiting the Hessian matrix for content-based retrieval of volume-data features. *To appear in Visual Computer*, 18(2), 2002.

[27] J. Hladůvka, A. König, and E. Gröller. Salient representation of volume data. In *Data Visualization 2001, Proceedings of the Joint Eurographics – IEEE TCVG Symposium on Visualization*, pages 203–211,351, 2001.
An extended and revised version accepted for [24]:

[24] J. Hladůvka and E. Gröller. Smallest 2nd-order derivatives for efficient volume-data representation. To appear in *Computers and Graphics*, 26(2), 2002.

Chapter 6

[22] J. Hladůvka and E. Gröller. Direction-driven shape-based interpolation of volume data. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, pages 113–120,521, 2001.

Acknowledgements

The last lines of the text are an expression of gratitude to all those who helped to make this thesis possible.

The work presented in this thesis has been supported by the VisMed and the BandViz projects. VisMed is supported by Tiani Medgraph, Vienna, and the FFF, Austria. BandViz is supported by the FWF, Austria.

I'd like to thank to Meister Eduard Gröller for a patient supervision, scientific discussions and valuable criticism. I'd like to express the highest gratitude to my father who actually brought me to computer science many years ago, who bought the indispensable household devices – a PMD 85 and an ATARI 800 XE, and who supported me through the duration of my studies. Last but not least, I am very thankful to the VisGroup people – Anna, Tom, Armin, Balázs, Lukas, and Helwig – for scientific feedback, barrels of beer and a really nice working atmosphere.

Talking about the working environment brings me to the TEXnical side of the work. I'd like to thank the sick concepts of M$ Windows and M$ Word for convincing me to go for an alternative. Credits for Linux, TEX, and LATEX, go to Linus Torvalds, Donald E. Knuth, and Leslie Lamport.

Finally, I feel a strong need to thank those people who contributed indirectly, nevertheless intensively: To Gibs, the "Kaiser des Imperiums", for a place with bed, shower, piano and, above all, with deep insights. To *all* my friends (a list would be too long) for keeping the bridge between Vienna, Bratislava, and Trenčín alive. To all the potential girl-friends for not disturbing me ;) during the long nights when I was working and listening to Dave Brubeck, Frédérik Chopin, Andrei Gavrilov, George Gershwin, David Helfgott, Antonio Carlos Jobim, Andrea Lucchesini, Pat Metheny, Arthur Rubinstein, Frank Sinatra, and many others.

Thank to you all. Danke. Gràcies. Köszönöm. Ďakujem.

Curriculum vitae

Personal data

Jiří Hladůvka
Born on April 19th, 1972, in Bratislava, Czechoslovakia.

e-mail: jiri@cg.tuwien.ac.at
phone: +43 (1) 58801 186 44

Rötzergasse 20/5
A-1170 Wien
Austria

Education

09/1978–06/1986	Basic school. Trenčín, Czechoslovakia.
09/1986–06/1990	Secondary school. Trenčín, Czechoslovakia.
10/1991–06/1996	Studies at the Faculty of Mathematics and Physics, Comenius University, Bratislava, Slovakia. M.Sc. in mathematics.
10/1998–01/2002	Ph.D. studies at the Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria.

Employment

06/1995–08/1996	TatraMed Ltd., Bratislava, Slovakia. Implementation of fundamental image-processing algorithms, development of fast volume-rendering algorithms, diploma-thesis research.
09/1996–01/1998	Leibinger Ltd., Freiburg, Germany. Ray casting based dose calculation for CT data, fast reconstruction of X-ray images, mesh based dose calculation for MRI data.
02/1998–05/1998	Comenius University, Bratislava, Slovakia. Assistant at the Department of Computer Graphics and Image Processing. Partial job until 2000: a lecture "Image Processing".
06/1998–01/2002	Vienna University of Technology, Vienna, Austria. Research assistant at the Institute of Computer Graphics and Algorithms. Participation in the VisMed project.

Publications

See section "Related publications" in this thesis for the refereed proceedings and journal publications.

Languages

Slovak	*native*
Czech	*native*
Russian	*fair*
English	*fluent*
German	*fluent*

IT Skills

Specialization	Computer graphics, image processing, volume visualization and processing.
OS	UNIX (IRIX, Linux), VMS, MS Windows.
Programming	Python, Java, C++, C, PASCAL, BASIC, assembler.
DTP	LaTeX, Corel-DRAW!, Adobe-Photoshop, MS-Office, ...

Hobbies

Piano, guitar, squash, skiing, books.

Vienna, December 18th, 2001